COMPUTER-AIDED DESIGN & APPLICATIONS
2018, VOL. 15, NO. 6, 796–806
https://doi.org/10.1080/16864360.2018.1477670

Transformation of LEGO Models

Wei Pan and Lujie Chen

Singapore University of Technology and Design, Singapore

ABSTRACT

Container stacking is a common activity in various industries. At a large scale, the shipping industry stacks cargo containers, while at a smaller scale the home and office storage providers stack paper or plastic boxes. An interesting question behind container stacking is how to efficiently move containers from one stack to another. In this paper, we investigate a remotely related question: shape transformation between LEGO models. While many computational algorithms have been developed to construct LEGO models, the problem of constructing a LEGO model using bricks from an existing model has not been explored in open literature. We propose two objectives to optimize the transformation: the movement cost and the reuse rate of LEGO bricks. Two basic approaches coupled with three strategies are implemented to achieve shape transformation. Experiments and analysis reveal the efficiency of different algorithms under several metrics of evaluation.

KEYWORDS

Container stacking; LEGO construction; shape transformation; rasterized structure; pick and place

1. Introduction

Efficiency is crucial to all kinds of physical construction and in different scenarios, the meaning of efficiency varies. In cargo container stacking, the efficiency is strongly related to the number of movements because each move involves loading and unloading of a heavy container using a crane. The cost comes from crane usage and time taken to complete a job. While minimizing the number of movements is a central objective, the efficiency is also related to the use of space. A movement sequence that requires less space is often preferred as it increases the capacity of a ship yard. In the scenario of providing home and office storage service, the efficiency might be more related to space usage than the number of movements because a service provider always tries to maximize the usage of his warehouse, while the movement of boxes is largely done by his employees (not machines), who are paid for a fixed wage.

An interesting question behind container stacking is the generation of motion sequences that optimize certain goals. In this paper, we investigate a remotely related question based on LEGO models. We propose algorithms that generate motion sequences of LEGO bricks so that a new LEGO model can be constructed fully or partially using bricks from an existing model. In this scenario, we may consider a LEGO brick as a container and a LEGO model as a stack of containers. Our investigation is an attempt to address motion-sequence-generation issues under a multi-objective framework; however, the results of this paper may not be directly applicable to a specific industrial application, such as cargo container stacking. Using LEGO models as the basis of our study has the advantage that the scenario is well defined and is familiar to a reader. The high-level strategies proposed in this paper is potentially useful as a reference to generate solutions to a specific container stacking problem.

LEGO is a type of popular edutainment toy. Its application has been found in research ideas relating to software engineering [4], rapid prototyping [14], material science [1], and nucleic acid nanoengineering [6,9]. LEGO bricks are inherently suitable for constructing 3D shapes: they enable diverse part combinations; they are in various colors; and they can be reused; however, the complexity of LEGO construction increases significantly with the size of a 3D shape. Although many ways of part combination are feasible, it is difficult to build an accurate LEGO model without step-by-step instructions.

A LEGO construction problem was proposed in 1998 [7]: Given any 3D body, how can it be built from LEGO bricks? Nowadays, the problem is largely solved owing to several computational algorithms [10]. These algorithms mostly fall into two categories: one focuses on interlayer connectivity and the other on structural stability. For example, the algorithm proposed in [7] evaluated the connectivity between bricks by heuristics, in which a cost function was defined as the evaluation metric. The algorithm aimed to optimize the cost and automatically generate an assembly. Later, alternative optimization

CONTACT Wei Pan vpan@foxmail.com; Lujie Chen chenlujie@sutd.edu.sg

methods, such as an evolutionary algorithm [20] and a cellular automata algorithm [22], were introduced to improve to the computational speed and the generated LEGO assembly.

More recent work focused on the balance and stability of LEGO models [8, 13, 16, 23, 24, 29, 30]. In [16] and [24], an initial LEGO structure was represented by a graph; then a graph algorithm was applied to identify structurally weak points corresponding to weak LEGO bricks; the vicinity of these bricks were replaced by an alternative layout to achieve stronger assembly. The process was iteratively applied till the overall structural strength was up to certain criteria. Luo et al. [13] used a force-based metric to evaluate the balance condition of LEGO bricks in terms of the force and torque they were subject to. Weak and unstable bricks were modified to avoid collapsing. Several real-sized objects were constructed to verify the force-based method. Hong et al. [8] proposed a centroid adjustment method that can optionally hollow the interior of a LEGO model. Structures produced by the method can stand in a particular pose steadily. Zhang et al. [30] proposed a divide-and-conquer method through a concept called "pseudo floor", which divides a structure into components without floating bricks. The components can be assembled separately and then be connected to each other to produce the final assembly.

All the above methods address the problem of LEGO construction, while Pasek and Yip-Hoi [19] proposed an interesting computer integrated manufacturing system based on LEGO bricks. They described various components of the system aimed at educational purposes, and ways to assemble and disassemble LEGO models. They even showed conceptual ideas of a gripper designed to pick and place LEGO bricks automatically. Their work touches upon a research topic has not been explored to the best of our knowledge. That is how to achieve shape transformation from one LEGO model to another. This involves optimization strategies to minimize the steps of transformation and to maximize the reuse rate of LEGO bricks.

2. Related work

Shape transformation was investigated in building construction as large-scale motion control systems that applied robots to construct modular houses [12]. It was studied as optimization strategies to control bricklaying robots [28]. It was tightly interwoven with a relatively new type of construction: aerial robotic construction [5, 26], where unmanned aerial vehicles (UAVs) carried bricks from palettes to designated positions to build structures based on predetermined sequences. It was also touched upon in research in swarm intelligence [25], where a swarm of robots assembled structures by mimicking social animals such as termites. In these studies, shape transformation was limited to producing a particular shape from material stock. Efficiency in motion control is the main concern, while reusability is not.

Shape transformation was also investigated in depth in the field of modular self-reconfigurable robots (MSR) [3, 15, 27]. A MSR system often consists of many equal-sized modular robots, which can be reconfigured into different 86 shapes to achieve various functions. Shape transformation is a key factor for the evaluation of versatility of a system. During transformation, a governing algorithm determines how and where each module should move. The algorithm may be hosted on a central station, or be embedded in each module's firmware. Motion sequence planned by the algorithm is affected by sensor data, e.g. positions, collected from each module.

An important idea to achieve shape transformation in MSR was reported by Pamecha et al. [17] and Chiang and Chirikjian [2]. Firstly, they defined a few basic moves that could be easily achieved by each module. Secondly, they recursively simplified a complex transition from a start to an end configurations by many intermediate ones. Their recursive algorithm came to a stop if the transition between each two consecutive configurations can be realized by the basic moves. Then, the Hungarian method [11] was used to obtain optimally matched module pairs between two configurations. This approach is generally applicable to transforming many kinds of shapes, and has been applied to 2D hexagon [17] and 2D lattice [2].

Pan et al. [18] applied the Hungarian method in a large-scale shape transformation system based on pick-and-place (PnP) realization. The time in completing a task in the above assignment problem was associated with the cost of moving a cube, the basic building block of a rasterized structure. Physical constraints were introduced to the system to ensure that cubes can only be picked from and placed on the top surface of a structure. The constraints affected how the Hungarian method was applied in generating a feasible PnP motion sequence. The authors compared several strategies of motion planning and proposed a heuristic framework in obtaining a near-optimal solution under arbitrary physical constraints. As in studies of MSR, the reusability issue was not fully investigated because both robot modules and cubes are treated as identical units; hence, the reuse rate is simply the volume ratio between two different shapes.

When it comes to shape transformation of LEGO models, the problem exhibits multiple objectives. Minimizing the number of steps of motion remains important, while maximizing the reuse rate becomes a new meaningful metric. This paper presents optimization

strategies that consider both objectives in achieving shape transformation.

3. Principle

A LEGO package has bricks in various shapes and colors. To facilitate the presentation of the principle, we used seven different shapes, as shown in Figure 1, and one hundred and fourteen colors. PnP of a LEGO brick is subject to physical constraints. A brick fully or partly under other bricks cannot be picked; a brick disconnected from a partially built model cannot be placed as it would be hanging in the air. Figure 2 shows an additional constraint. Although the 2-by-2 yellow brick is on the top surface, it is completely surrounded by green bricks; while it is not impossible to pick the yellow brick without moving the green one, it is almost certain that additional PnP actions are needed; for example, to place another 2-by-2 brick on top of the yellow one and try to pull out the two bricks together. To simplify the calculation of PnP steps, bricks that cannot be directly picked are assumed to be not movable under their current status. Hereafter, we refer to bricks that satisfy the physical constraints as movable bricks.

3.1. Basic approaches

Two basic approaches are proposed. The first approach assumes that a source and a target LEGO model are available. A PnP motion sequence to transform the source to target is generated by an algorithm. The second approach assumes that the source LEGO model is available while the LEGO representation of the target is not fixed but is created from an interim voxelized model during transformation. Figure 3 depicts the difference between the two approaches: rigid transformation (RT) and flexible transformation (FT). In both approaches, the source model is disassembled from top to bottom and the target model is built from bottom to top, layer by layer. In RT, without loss of generality a variant of the algorithm described in [8] is applied to generate both LEGO models; at each layer, large LEGO bricks are used with priority. In FT, movable LEGO bricks on the source model are used with priority. When required bricks are unavailable, they are obtained from a stock that is assumed to have unlimited supply of all types of bricks; then large bricks are use with priority just as in RT. The motivation behind FT is to increase the reuse rate; however, the side effect is that the target model generated by FT has larger number of LEGO bricks and is less stable than that by RT. While we are not concerned with the stability issue in this study, we will evaluate the cost of shape transformation associated with the number of LEGO bricks.

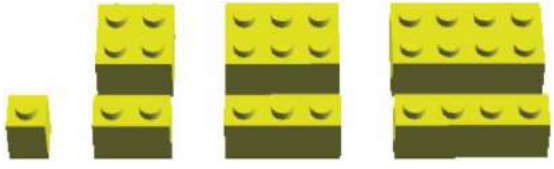

Figure 1. Seven different-shaped LEGO bricks used in this study.

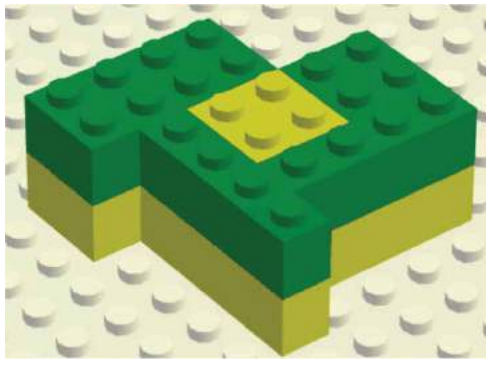

Figure 2. A fully surrounded brick, the 2-by-2 yellow brick, is not movable.

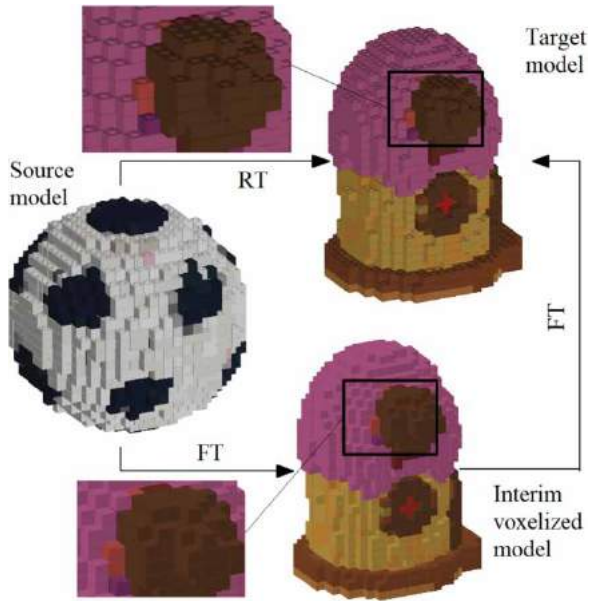

Figure 3. Two basic approaches: rigid transformation (RT) and flexible transformation (FT).

3.2. LEGO generation

The most common format of a 3D object is a triangle mesh model. When a LEGO representation is to be generated from a triangle mesh, at least two steps are needed. Firstly, the triangle mesh is converted to a rasterized model made of identical voxels. Then, the voxels

are grouped into LEGO bricks, which is the key step in LEGO generation. The algorithm described in [8] generates various possible voxel groups (i.e. LEGO bricks) at each layer by heuristics, and applies a scoring system to determine the final bricks. The scoring system is not fixed and can be changed according to different situations. We propose the following scoring system to generate LEGO models.

In the RT approach, the score of a brick b is calculated by

$$S_1 = \begin{cases} A_b, \text{ Bottom layers} \\ N_c, \text{ Other layers} \end{cases} \tag{3.1}$$

where A_b is the area of the brick (e.g. $A_b = 4$ for a 2-by-2 brick), and N_c is the number of bricks at a lower layer that b is connected to. LEGO generation goes from bottom to top. A bottom layer does not have a layer beneath it; this includes a base layer and the lowest layer of an overhung section. On a bottom layer, a large brick gets a high score; on upper layers, a brick with many connections gets a high score. If there is a tie (e.g. a 2-by-2 or a 1-by-4 brick on the bottom layer), one of the bricks is selected randomly to be generated.

In the FT approach, the source model is generated in the same way as in RT. For the target model, the score of a brick b is calculated by

$$S_2 = \begin{cases} N_b S_1, \ N_b > 0 \\ N_c / V, \ N_b = 0 \end{cases} \tag{3.2}$$

where N_b is the number of movable b type bricks currently available from the source, and V is the number of voxels in the target model. N_b works as a weighting factor that biases towards a type of brick that has abundant supply from the source. If the b type is unavailable from movable bricks ($N_b = 0$), its score is S_1/V , effectively the same as switching back to Eqn 3.1. Dividing V ensures that movable bricks always get a higher score than unavailable bricks.

Based on the calculated scores, voxel groups are converted to LEGO bricks in a score-descending order. In general, LEGO models generated by Eqn. 3.1 is more robust, whereas the target models generated by Eqn 3.2 would make use of more bricks from the source model.

3.3. Strategies

Under each approach, three strategies were investigated, as illustrated in Figure 4. In Strategy 1, the source model is disassembled completely and the LEGO bricks are put to a buffer; then the target model is built using bricks from the buffer and a stock with the buffer used by default. If certain brick cannot be found in the buffer, it is obtained from the stock, which can supply of all types and colors of bricks.

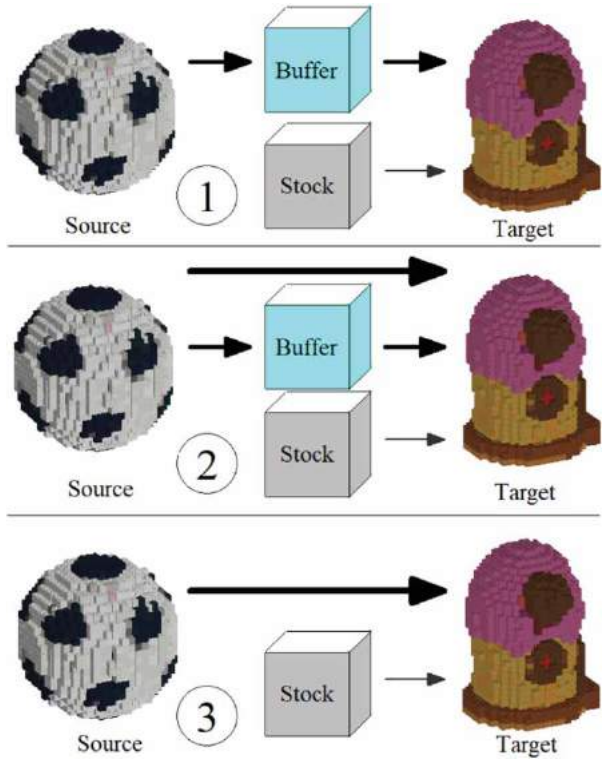

Figure 4. Three strategies under each approach. Strategy 1: Disassemble the source model completely to a buffer; then build the target from the buffer and a stock. Strategy 2: Build the target from the source, a buffer (disassembled bricks from the source), and a stock. Strategy 3: Build the target from the source and a stock. The size of arrow represents its priority to move.

In Strategy 2, movable bricks from the source model are directly picked and placed on the target model. If some required bricks on a target layer do not have matching movable bricks on the source, an initially empty buffer is searched for such bricks. If they exist, PnP continues from the buffer to target; otherwise, one layer of bricks is disassembled from the source and placed to the buffer so that a new layer can be uncovered on the source model. If movable bricks on the new layer satisfy the requirement of the target layer, PnP can resume from the source to target; otherwise, another layer is disassembled to the buffer; so on and so forth till there is no brick on the source. If that happens, a stock will be used to provide a required brick for the target. Then, bricks are moved from the buffer and stock to the target, just as in Strategy 1. In other words, Strategy 2 tries to move bricks from the source to target directly; if stuck, it disassembles some bricks hoping that newly uncovered bricks can be used for transformation; but if no suitable one is found even when completely disassembling the source, the stock will be used.

In Strategy 3, no buffer is involved. Movable bricks are transported from the source to target; if no suitable brick is available from the source, the stock will provide one. So on and so forth till the target model is finished.

3.4. Evaluation metrics

Coupling the strategies with the basic approaches, we have six algorithms for shape transformation of LEGO models: FT1, FT2, FT3, RT1, RT2, and RT3. To evaluate the algorithms, three metrics are applied: cost (C), reuse rate (R), and overall performance (P). The cost is defined as the number of PnP needed to build the target model. Each PnP action is associated with a cost value of 1. (The cost defined here is not the value of a brick. Associating the cost to a PnP action reflects the motivation of this study: to simulate PnP of a container in container stacking.) This includes PnP actions on one LEGO brick from source to buffer, from source to target, from buffer to target, and from stock to target. Low cost suggests high efficiency.

The reuse rate is defined as

$$R = \frac{N_r}{N_r} \tag{3.3}$$

where N_r is the number of bricks on the target model that are originally from the source, and N_s is the number of bricks on the source model. N_r includes the bricks moved directly from the source and those moved indirectly through a buffer.

The overall performance is defined as

$$P = V\frac{R}{C} \tag{3.4}$$

where V is the number of voxels in the voxelized target model. P increases with the reuse rate and decreases with increasing cost. V puts the P metric on a relatively normalized scale against model size. It is not ideal to replace V with the number of LEGO bricks because different FT algorithms do not produce the same number of bricks on the target model, while V is same in all algorithms given a fixed model size.

4. Results and discussions

Eight triangle mesh models, as shown in Figure 5, were converted to LEGO models and used to test the proposed algorithms. Three types of tests were conducted. In the first two tests, performances of the algorithms were evaluated by the cost, reuse rate, and overall performance metrics without considering the color information. In the third test, the color information was incorporated, and a sequence of transformation was visually examined.

4.1. Test 1: Transform model A to other models

Model A in Figure 5 was transformed to the other seven models at a particular resolution which produced an average of 2500 voxels on each model. (A typical voxelization process cannot guarantee an exact number of voxels on a triangle mesh model but this does not affect the test.) LEGO models were generated using methods described in [8]. Figure 6 shows the results of the algorithms under different performance metrics. The three strategies exhibit a descending order of cost as expected. Strategy 1 always disassembles the source model to a buffer, thereby having the highest cost. Strategy 3 does not use the buffer; hence, its cost is exactly the number

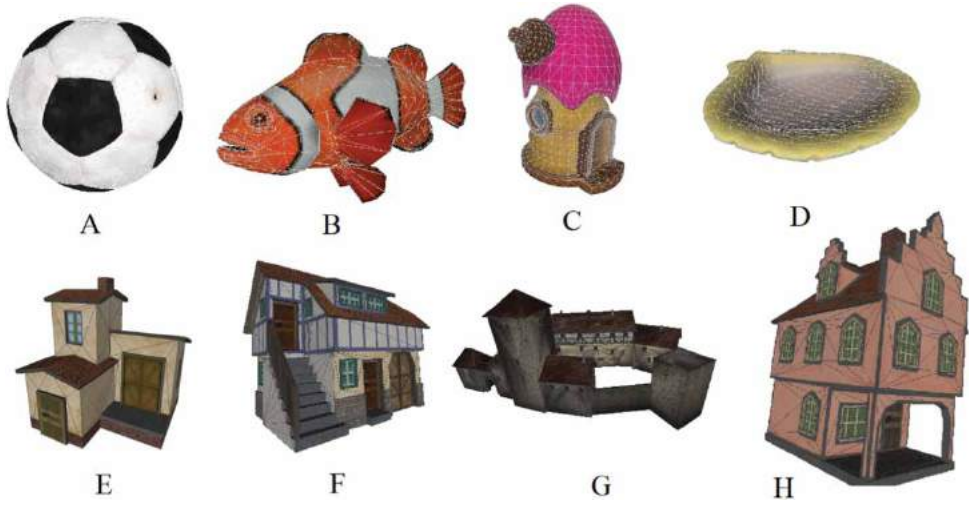

Figure 5. Triangle mesh models used to test the algorithms. The models were obtained from Reiner "Tiles" Prokein (http://www.reinerstilesets.de/graphics/lizenz) [21].

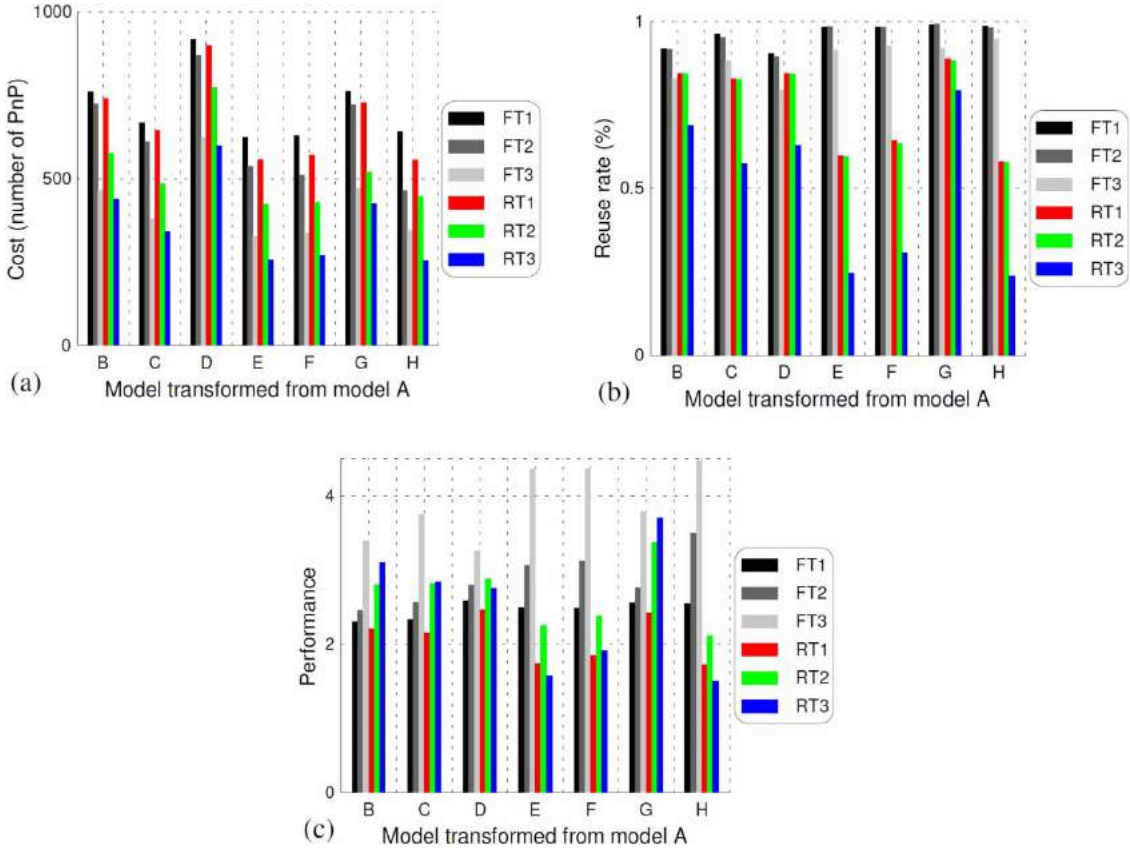

Figure 6. Performances of the algorithms in transforming model A to other models. (a) Cost, (b) reuse rate, and (c) overall performance of transformation.

of LEGO bricks on the target model. Comparing FT3 and RT3, we see that the number of bricks on the target generated by the two algorithms is different. In RT, the LEGO target was generated before transformation, and large bricks were used with priority. In FT, the LEGO target was generated on the fly, and available bricks from the source and buffer were used before resorting to the stock; therefore, each FT algorithm produces different number of bricks on the target, and they use more bricks than the RT algorithms.

Intuitively, Strategy 1 and 3 should produce the highest and lowest reuse rate respectively. While this is indeed reflected in the trend of FT and RT respectively in Figure 6(b), we see exceptions in model A-to-E and A-to-G transformations by FT. The reuse rate of FT2 is slightly higher than that of FT1. This is caused by the different number of bricks in the target model generated by the different FT algorithms. FT1 has the source completely disassembled to the buffer; so many large bricks are available when the target is generated on the fly. FT2 exposes movable bricks layer by layer; hence, the available large bricks at any point of time are not that many; on-the-fly target generation is forced to use smaller bricks. Consequently, more bricks are used in FT2 than those in FT1, which makes the reuse rate of the former marginally higher than that of the latter. We also see that the reuse rate of a FT algorithm is higher than that of the corresponding RT algorithm. Being flexible in target model generation, FT naturally makes use of more bricks from the source than RT; however, these bricks tend to be smaller than the best-fit large bricks, resulting in weaker LEGO structures that have internal sliding planes. Overall, FT increases the reuse rate by compromising the structural strength.

The overall performance, shown in Figure 6(c), measures efficiency of the algorithms. In the FT series, FT3 stands out as it achieves similar reuse rate as FT1 and FT2 at a much lower cost. This suggests that for on-the-fly target generation, it is not efficient to involve a buffer because a brick is very likely to be reused given only seven types of bricks without considering color (Figure 1). In the RT series, RT2 is better than RT3 on model A-to-D, A-to-E, A-to-F, and A-to-H transformations. This suggests that if the target bricks are fixed types, making moderate use of a buffer produces a good balance between the cost and reuse rate.

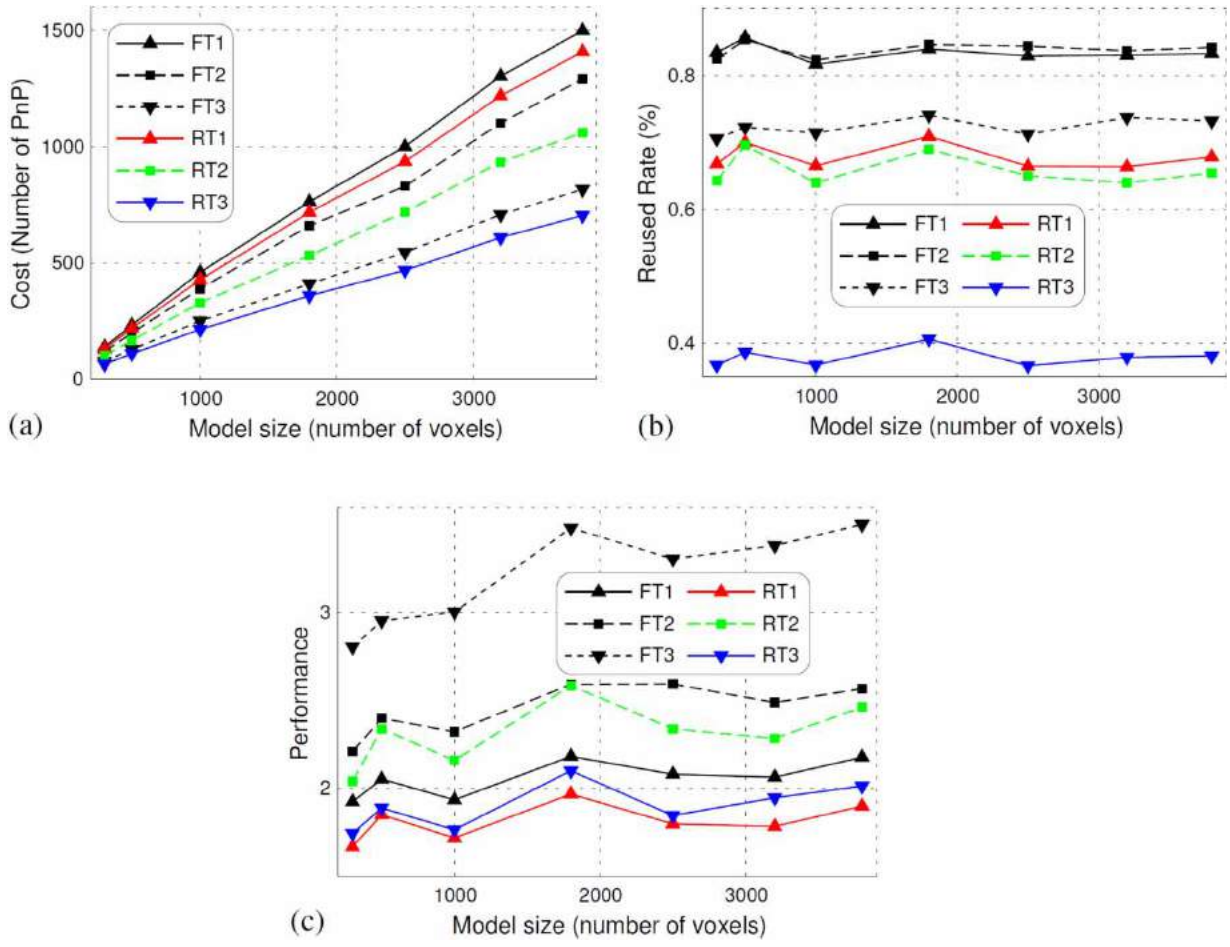

Figure 7. Average performances of the algorithms in transforming eight models to each other. (a) Cost, (b) reuse rate, and (c) overall performance of transformation.

4.2. Test 2: Transform eight models to each other

This test was conducted at seven resolutions, i.e. seven average model sizes, from 300 to 4500 voxels per model. The eight models were transformed to each other and there were $7 \times 8 = 56$ transformations at each model size. Figure 7 shows the average results over the 56 transformations. It is obvious from Figure 7(a) that the cost increases linearly with the model size. Strategies 1, 2, and 3 have a consistent descending cost at a particular model size. The average of these transformations reveals the inherent cost associated with each strategy. A FT algorithm has a higher cost than its corresponding RT algorithm because FT produces a target model made of more bricks.

Figure 7(b) shows that Strategies 1 and 2 produce very similar reuse rate, suggesting that completely disassembling the source upfront (Strategy 1) is not worthwhile; it is better to do partial disassembly to save some cost while maintaining a high reuse rate. When Strategy 3 is coupled with the FT approach (i.e. FT3), the reuse rate is actually quite high, comparable to the best of RT algorithms; however, the reuse rate of RT3 is significantly lower than that of RT1 and RT2. This implies that when the bricks that will be used to construct the target model have been determined previously, disabling the buffer makes it difficult to find matching bricks from the source, and on average more than half of the target bricks would come from the stock. It is also seen from the figure that the reuse rate is not that sensitive to the change of model size above 1700 voxels per model. Smaller model size (fewer than 1700 voxels) does reduce the reuse rate because the reduced brick samples in smaller models make it harder to find matching bricks.

In Figure 7(c), statistics of the overall performance shows that FT3 is much better than other algorithms. Within RT algorithms, RT2 is statistically the best. Note that the overall performance is only an empirical metric. Different scenarios may emphasize different evaluation criteria. It may be possible that in certain scenario the cost metric must be redefined, and the relative performance of the algorithms will vary from our results; nevertheless, the contribution of this study is in framing the

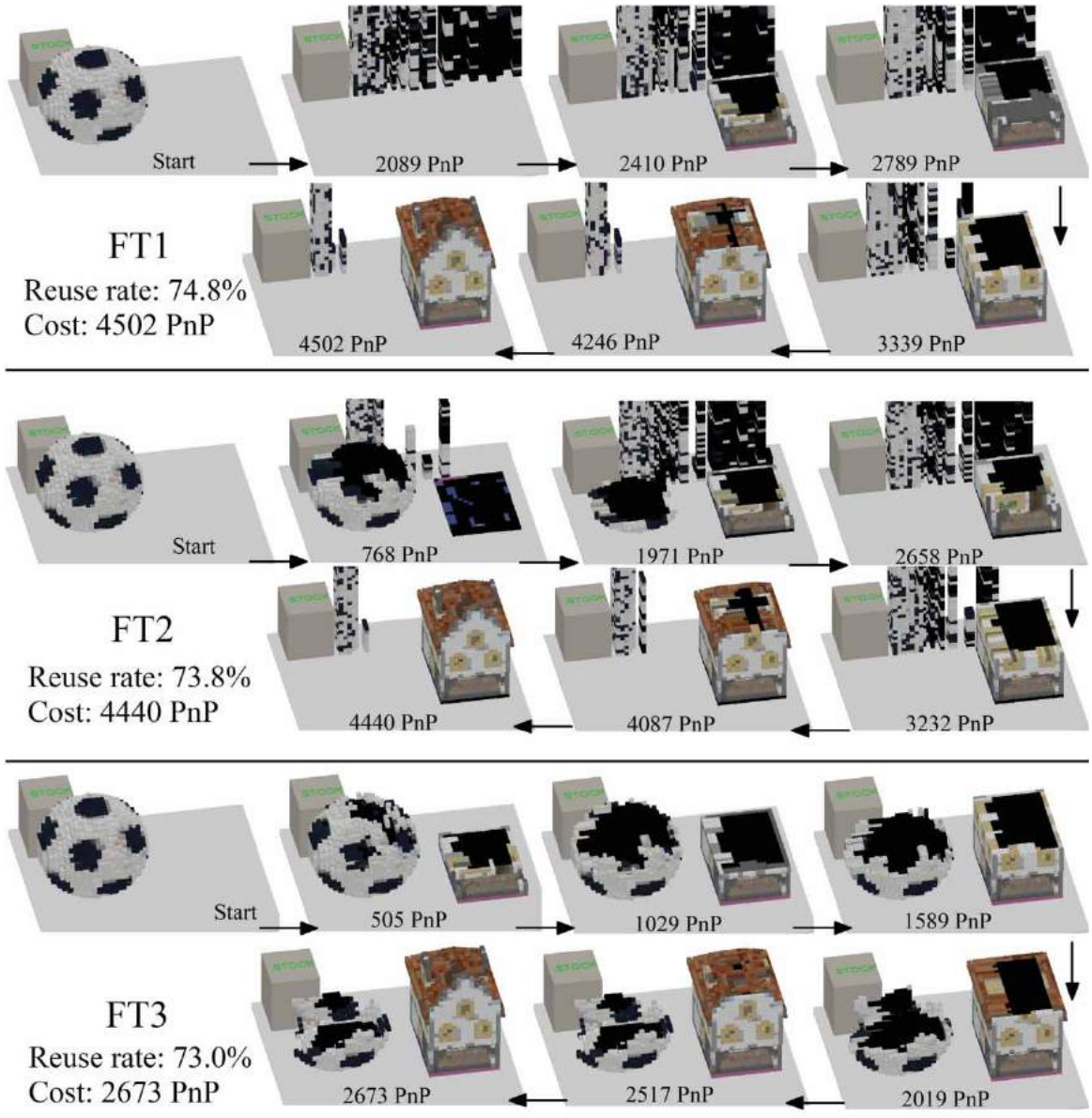

Fig. 8: Typical FT sequences between model A and H.

Figure 8. Typical FT sequences between model A and H.

LEGO transformation problem as a multi-objective optimization process, in which different strategies should be explored to achieve some balance between objectives.

4.3. Test 3: Transformation with color information

This test incorporated the color information of each triangle mesh model when building the corresponding LEGO model. A legitimate match between a source and a target brick should be the same color as well as the same shape. Due to the fact that more than one hundred colors were used, finding a match was much less likely than in Tests 1 and 2 where the color information was ignored; however, in our implementation of the algorithms, an interior brick can take on any color; only exterior bricks must find exact color match. Consequently, although the average reuse rate in this test is lower than that in Tests 1 and 2, it is not down to zero.

Figure 8 shows typical FT sequences between two models. FT1 first disassembles the source to a buffer, displayed next to the stock, at a cost of 4502 PnP. Then it builds the target using bricks from the buffer and stock. Interior bricks of the target model can be of any color, while exterior bricks must be the surface color. FT2 moves the source bricks to the buffer only if there are no suitable ones. The reuse rate of FT2 are about the same

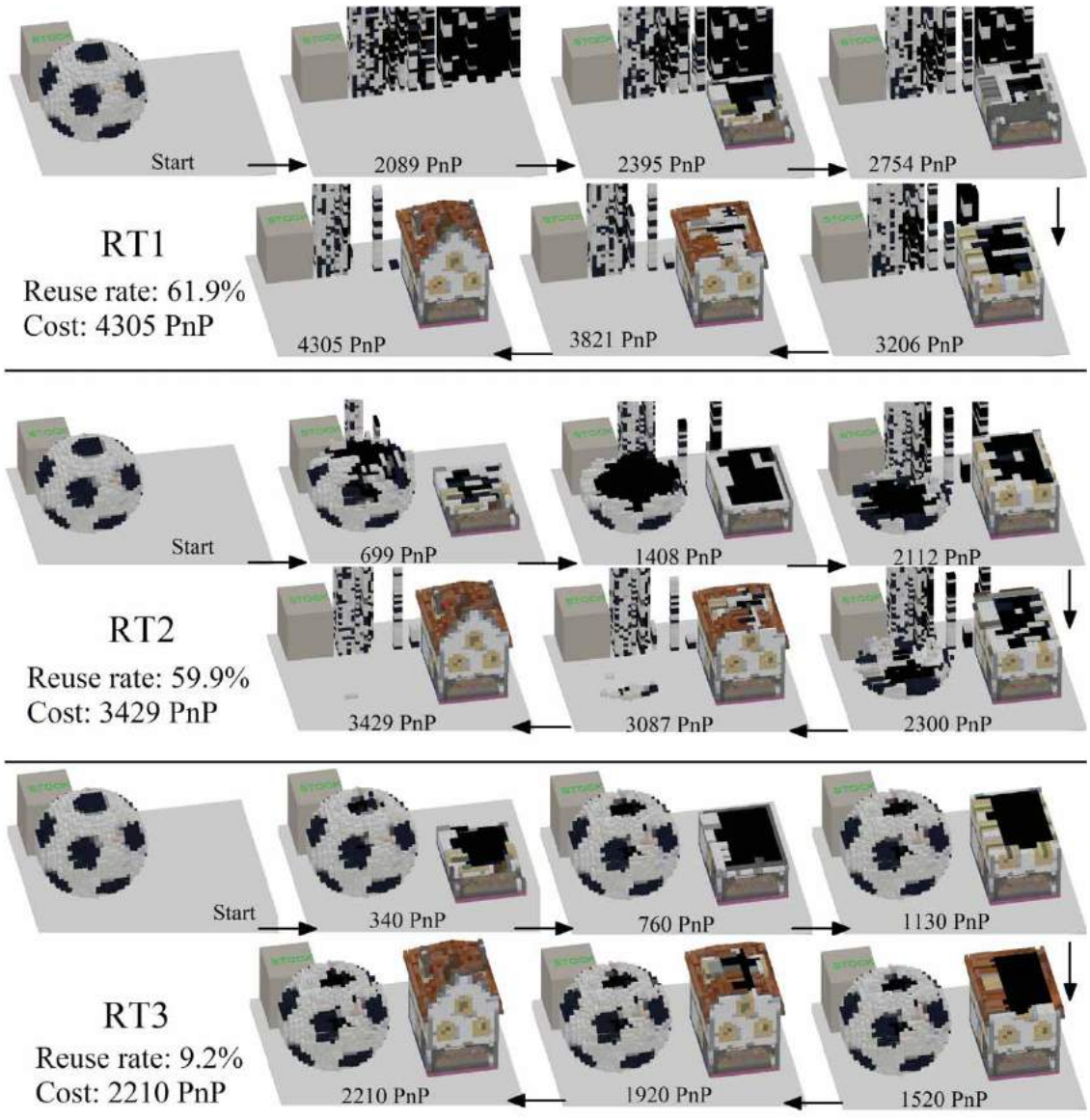

Fig. 9: Typical RT sequences between model A and H.

Figure 9. Typical RT sequences between model A and H.

as that of FT1. FT3 does not use the buffer. The source is only partially disassembled when the target is fully built. FT3 produces the lowest cost and reuse rate.

Figure 9 shows RT sequences between the same two models. As both models contain many white surface bricks, the reuse rates of RT1 and RT2 are relatively high, above 59%. In RT3, the interior of the target contains many black bricks. We have used them to indicate bricks obtained from the stock and placed on the interior of the target. (They do not have to be black; any color is fine.) Comparing FT3 with RT3, both do not use the buffer, while FT3 has the liberty to build the target based on available bricks; therefore, it does not use the stock to provide many interior target bricks and its reuse rate is much higher than RT3.

5. Conclusion and future work

Shape transformation of LEGO models has been investigated as a multi-objective optimization problem. Two basic approaches have been proposed. The first approach, FT, allows for on-the-fly generation of a target model based on available LEGO bricks from a source model. The second approach, RT, requires both the source and target models to be generated before transformation. Three strategies are adopted under each approach, which

leads to six algorithms. A cost, a reuse rate, and an overall performance metrics have been applied to evaluate the algorithms. Strategy 1 produces the highest cost and reuse rate on average. Strategy 3 produces the lowest lost and reuse rate. Strategy 2 achieves a balance between Strategies 1 and 3. It gives the best overall performance in RT algorithms. In FT, owing to on-the-fly target generation, Strategy 3 obtains a reasonably high reuse rate at a low cost; hence, it outperforms the other strategies in overall performance.

FT achieves better results than RT due to flexible model generation; however, this has the disadvantage of producing relatively weak LEGO structures, which have internal sliding planes. FT also tends to produce a model made of more bricks than RT.

In this study, we only used one brick as a unit of PnP. In practice, sometimes a group of LEGO bricks can be reused as a whole. Future work along this direction may look into novel LEGO model generation methods that tend to produce modularized LEGO models. Then, shape transformation may be achieved at a modular level, not at a single brick level. Successful implementation of such modular-oriented methods would lead to highly efficient LEGO transformation algorithms.

ORCID

Wei Pan http://orcid.org/0000-0002-0933-2453
Lujie Chen http://orcid.org/0000-0002-9153-9348

References

[1] Celli, P.; Gonella, S.: Manipulating waves with Lego bricks: A versatile experimental platform for metamaterial architectures, Applied Physics Letters 107 (8), 2015, 081901. http://doi.org/10.1063/1.4929566.

[2] Chiang, C.-J.; Chirikjian, G.-S.: Modular robot motion planning using similarity metrics, Autonomous Robots 10 (1) (2001) 91–106. http://doi.org/10.1023/A:1026552720914.

[3] Feczko, J.; Manka, M.; Krol, P.; Giergiel, M.; Uhl, T.; Pietrzyk, A.: Review of the modular self reconfigurable robotic systems, in: Robot Motion and Control (RoMoCo), 2015 10th International Workshop on, IEEE, 2015, pp. 182–187. http://doi.org/10.1109/RoMoCo.2015.7219733.

[4] Feijs, L.; Jong, R.-D.: 3d visualization of software architectures, Communications of the ACM 41 (12) 1998, 73–78. http://doi.org/10.1145/290133.290151.

[5] Fink, J.; Michael, N.; Kim, S.; Kumar, V.: Planning and control for cooperative manipulation and transportation with aerial robots, The International Journal of Robotics Research 30 (3), 2011, 324–334. http://doi.org/10.1007/978-3-642-19457-3.

[6] Gothelf, K.-V.: Lego-like DNA structures, Science 338 (6111), 2012, 1159–1160. http://doi.org/10.1126/science.1229960.

[7] Gower, R.; Heydtmann, A.; Petersen, H.: Lego: automated model construction, in: 32nd European Study Group with Industry Final Report, 1998, pp. 81–94.

[8] Hong, J.-Y.; Way, D.-L.; Shih, Z.-C.; Tai, W.-K.; Chang, C.-C.: Inner engraving for the creation of a balanced Lego sculpture, The Visual Computer 32 (5), 2016, 569–578. http://doi.org/10.1007/s00371-015-1072-4.

[9] Ke, Y.; Ong L.-L.; Shih, W.-M.; Yin, P.: Three-dimensional structures self-assembled from DNA bricks, Science 338 (6111) (2012) 1177–1183. http://doi.org/10.1126/science.1227268.

[10] Kim, J.-W.; Kang, K.-K.; Lee, J. H.: Survey on automated Lego assembly construction, in: WSCG 2014 Conference on Computer Graphics, Visualization and Computer Vision, V´aclav Skala-UNION Agency, 2014.

[11] Kuhn, H.-W.: The Hungarian method for the assignment problem, Naval research logistics quarterly 2 (1-2), 1955, 83–97. http://doi.org/10.1002/nav.3800020109.

[12] Leyh, W.: Experiences with the construction of a building assembly robot, Automation in construction 4 (1), 1995, 45–60. http://doi.org/10.1016/0926-5805(94)00034-K.

[13] Luo, S.-J.; Yue, Y.; Huang, C.-K.; Chung, Y.-H.; Imai, S.; Nishita, T.; Chen, B.-Y.: Legolization: optimizing Lego designs, ACM Transactions on Graphics (TOG) 34 (6), 2015, 222. http://doi.org/10.1145/2816795.2818091.

[14] Mueller, S.; Mohr T.; Guenther, K.; Frohnhofen, J.; Baudisch, P.: Fabrickation: fast 3D printing of functional objects by integrating construction kit building blocks, in: Proceedings of the 32nd annual ACM conference on Human factors in computing systems, ACM, 2014, pp. 3827–3834. http://doi.org/10.1145/2559206.2574779.

[15] Murata, S.; Kurokawa, H.: Self-reconfigurable robots, Robotics & Automation Magazine, IEEE 14 (1), 2007, 71–78. http://doi.org/10.1109/MRA.2007.339607.

[16] Ono, S.; Alexis, A.; Chang, Y.: Automatic generation of Lego from the polygonal data, in: IWCIT2013 in Nagoya, 2013, pp. 262–267.

[17] Pamecha, A.; Ebert-Uphoff, I.; Chirikjian, G.-S.: Useful metrics for modular robot motion planning, Robotics and Automation, IEEE Transactions on 13 (4), 1997, 531–545. http://doi.org/10.1109/ROBOT.1996.503816.

[18] Pan, W.; Chen, L.; Dritsas, S.: Pick-and-place process sequencing for transformation of rasterized 3d structures, Automation in Construction 75, 2017, 56–64. http://doi.org/10.1016/j.autcon.2016.12.007

[19] Pasek, Z. J.; Yip-Hoi D: Lego factory: An educational cim environment for assembly, in: Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition, Vol. 10, 2005, p. 1.

[20] Petrovic, P.: Solving lego brick layout problem using evolutionary algorithms, in: Proceedings to Norwegian Conference on Computer Science, 2001. http://doi.org/10.1.1.16.4146.

[21] Reiner's Tilesets: http://www.reinerstilesets.de/graphics/lizenz

[22] Smal, E.: Automated brick sculpture construction, Ph.D. thesis (2008).

[23] Stephenson, B.: A multi-phase search approach to the LEGO construction problem, in: Proceedings of the Ninth International Symposium on Combinatorial Search.

[24] Testuz, R.; Schwartzburg, Y.; Pauly, M.: Automatic Generation of Constructable Brick Sculptures, in: M.-A. Otaduy, O. Sorkine (Eds.), Eurographics 2013 - Short Papers, The Eurographics Association, 2013. http://doi.org/10.2312/conf/EG2013/short/081-084.

[25] Werfel, J.; Petersen, K.; Nagpal, R.: Designing collective behavior in a termite-inspired robot construction team, Science 343 (6172), 2014, 754–758. http://doi.org/10.1126/science.1245842.

[26] Willmann, J.; Augugliaro, F.; Cadalbert, T.; D'Andrea, R.; Gramazio, F.; Kohler, M.: Aerial robotic construction towards a new field of architectural research, International journal of architectural computing 10 (3), 2012, 439–460. http://doi.org/10.1260/1478-5170771.10.3.439.

[27] Yim, M.; Shen, W.-M.; Salemi, B.; Rus, D.; Moll, M.; Lipson, H.; Klavins, E.; Chirikjian, G.-S.: Modular self-reconfigurable robot systems: grand challenges of robotics, IEEE Robotics & Automation Magazine 14 (1), 2007, 43–52. http://doi.org/10.1109/MRA.2007.339623.

[28] Yu, S.-N.; Ryu, B.-G.; Lim, S.-J.; Kim, C.-J.; Kang, M.-K.; Han, C.-S.: Feasibility verification of brick-laying robot using manipulation trajectory and the laying pattern optimization, Automation in Construction 18 (5), 2009, 644–655. http://doi.org/10.1016/j.autcon.2008.12.008.

[29] Yun, G.; Park, C.; Yang, H.; Min, K.: Legorization with multi-height bricks from silhouette-fitted voxelization, in Proceedings of the Computer Graphics International Conference, http://doi.org/10.1145/3095140.3095180.

[30] Zhang, M.; Igarashi, Y.; Kanamori, Y.; Mitani, J.: Compo nent-based building instructions for block assembly, Computer-Aided Design and Applications, 2016, 1–8. http://doi.org/10.1080/16864360.2016.1240450.