Title:
Design of LEGO assembly transformation

Authors:
Wei Pan, vpan@foxmail.com, Singapore University of Technology and Design, Singapore
Lujie Chen, chenlujie@sutd.edu.sg, Singapore University of Technology and Design, Singapore

Keywords:
Lego Assembly Transformation, Brick Layout, Rasterized Structure, Pick and Place, Brick Reuse

DOI: 10.14733/cadconfP.2017.122-127

Introduction:

LEGO bricks are world-widely popular not only as edutainment toys, but also idea-inspiring research tools in disciplines such as software engineering [2], rapid prototyping [8], material science [1] and nucleic acid nanoengineering [3, 6]. This lies in the fact that they are versatile construction blocks with significant possibility to be rearranged: there are many different shapes available; the exponential combinations of these bricks enable diverse shapes. Therefore, it is challenging to design and assemble complex 3D structures without instructions [9], which requires a significant amount of trials-and-errors [10]. This arises the LEGO construction problem stated as "given any 3D body, how can it be built from LEGO bricks" [11].

Researchers have been working on the LEGO construction problem in terms of optimization towards the interlayer connectivity of bricks as well as the stability of the whole structure. There are plenty of heuristic-based strategies proposed since 1998 [4,7]. So far, most works are focusing on the LEGO construction problem.

In this paper, we are investigating the problem of constructing one structure directly from a pre-built structure, rather than build a structure solely from the LEGO block boxes, which we call it transformation, or re-fabrication. Since the bricks can be accurately located and transported, the transformation process is done by a process called pick-and-place (PnP), which initially refers to the picking and placement of electronic components onto circuit boards. PnP is also applied to assemble 3D structures from parts such as LEGO bricks layer by layer. In this paper, some strategies are investigated to automatically generate the set of instructions to transform one LEGO model to another.

Principle:

General pipeline

The input of our method consists of two models. As Fig. 1 shows, the first model is given as a pre-built LEGO assembly of arbitrary shape, which is regarded as LEGO bricks source; the second model is provided with a polygonal mesh file, which is then rasterized into voxel representation and thereby, can be merged into LEGO bricks so as to generate the brick layouts (Fig. 1(b)). Finally, users could build the second model layer by layer by following the bricks transporting sequences that transforming the source shape and stock into the goal shape (Fig. 2(c)).

Voxelization

3D shapes are generally expressed as mesh data stored in computer. Rasterize these mesh data is well studied in the open literature [5]. Figure 1(b) shows the voxelization result of Figure 1(a). To save LEGO bricks, the inner voxels can be removed in this process.

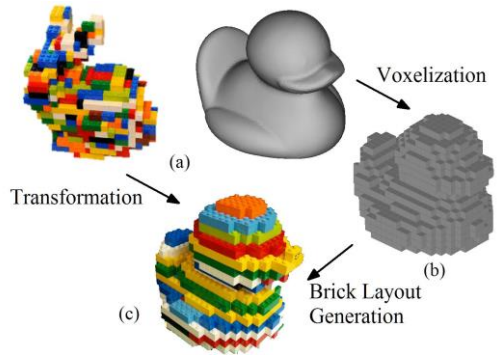

Fig. 1: The flow of the proposed Lego shape assembly transformation: (a) The first model is a prebuilt Lego bunny; the second model is given as a 3D mesh duck shape, (b) Voxelized duck shape, (c) Generated Lego brick layout of the duck.

LEGO bricks layout generation

The brick layout assembly is generated layer by layer. In each layer, the first step is to select the voxel that has fewer neighbors to merge into a brick, as a voxel with fewer neighbors is more likely to form weak brick connections. The possibility whether a voxel is to be selected is determined by Eqn. 1.

$$P = \begin{cases} 1 & \text{if } N_{Neibour} < 3 \\ 1/\exp(N_{Neibour}) & \text{otherwise} \end{cases} \quad (1)$$

The second step is to merge the voxel we selected above into a brick with its merge-able neighbors. It's proved that bricks with more perpendicular connections result in better stability of the sculpture [4]. Therefore, the brick layout of the bottom layer will be merged based simply on the bricks size. For the other layers, the brick merging will be determined by both connections with bricks in the previous layer and its brick size. A brick value integrated the two factors is computed by Eqn. 2, the brick with the best brick value will be selected thus merged. Fig. 2 illustrates the brick layout result of a mesh model with shape as a cow.

$$V_{Brick} = \alpha C_{Connection} + (1-\alpha) C_{BrickSize} \quad (2)$$

The value of α ranging from 0 to 1 determines the weight of $C_{Connection}$ and $C_{BrickSize}$. Higher α strengthens the connections while increases the demand for big bricks. It should be noted that some structures include parts that are inevitably pendant when it's assembled bottom up layer by layer until a type of larger brick, e.g. 2×10 brick is introduced, another alternative solution is to apply the method proposed by Zhang [12]. This is illustrated in Fig. 3(d).

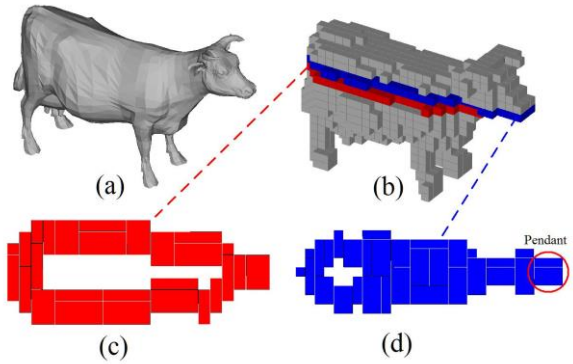

Fig. 2: The layout result of a cow model: (a) A mesh cow, (b) A voxelized cow shape, (c) The brick layout of 12th layer, (d) The brick layout of 13th layer.

Evaluation merits

Before introducing the transporting strategies, it's necessary to define several merits and basic concepts for evaluation.

Source refers to the first Lego shape which is to be disassembled and the bricks from Source are to be reused for re-fabrication.

Goal refers to the second Lego shape which is to be assembled.

Stock refers to the box containing unlimited basic Lego bricks, and bricks used from here are not counted as reused bricks.

Buffer is an empty box which is used to place the bricks disassembled from Source, and bricks used from here are counted as reused bricks.

Transform Pair is a pairing of a brick from either Source or Buffer which has same property (e.g. brick type, color, etc.) as a brick slot in the Goal. They are regarded as one Transform Pair so that a transportation process can be carried out.

Cost (C) is defined as the number of times to transport a brick. For example, cost will plus one when transporting a brick from Source to Buffer. Any transportation involved in the construction of Goal by bricks from Source, Buffer and Stock, is counted as cost.

The Reuse Ratio (R) is the number of reused bricks from Source divided by the total number of bricks in Source. Higher R means a better strategy.

Top Surface Constraint (TSC) means that the transportation of LEGO bricks is only applicable to those at the top surface of a structure. After accepting the bricks layout of the first model, we build the graph representations so that only the top bricks checked by TSC can be formed as Transform pairs.

Bricks transporting strategies

We proposed two categories of strategies to achieve bricks transformation, namely layout first strategy (LFS) and transport first strategy (TFS).

Layout First Strategy (LFS): The bricks layout of Goal is generated first so as to obtain the transporting sequence of the Lego bricks bottom up from Source. In some real scenarios, e.g. LEGO set bought from LEGO store, the Goal is pre-provided, so we directly look into strategies that transform the Source into the Goal.

Transport First Strategy (TFS): An alternative consideration is to generate the LEGO layout of the Goal based on the transportable LEGO bricks of the Source, since all the transportable bricks are labeled. A binary variable $\beta \in [0,1]$ will be used to check whether there is a transportable brick to generate the brick layout of the second model. In this case, to calculate the value of the merging bricks, Eqn. 2 should be modified, shown as Eqn. 3.

$$V_{Brick} = \beta \times (\alpha C_{Connection} + (1-\alpha) C_{BrickSize}) \tag{3}$$

In each category, three strategies are implemented and have shown different performance according to their reused rate and transporting cost.

First Strategy (LFS1/TFS1) is what normally happens for re-fabrication, as shown in Fig. 3(a). Firstly, the Source is completely disassembled and all the bricks are transported to Buffer. Next, construct the second model with all the bricks in Buffer, if one type of brick is used up, use the bricks in Stock. Considering the TSC, the construction of Goal will start from the bottom up layer by layer.

Second Strategy (LFS2/TFS2) is developed that the system traverses all the transportable bricks from Source and Buffer and pairs them with empty slots of Goal and then transports bricks according to these pairings until there is no more pairs can be found. Then some transportable bricks will be picked and transported to buffer whenever the system gets stuck. This strategy is displayed in Fig. 3(b).

Third Strategy (LFS3/TFS3) is to do without Buffer. As Fig. 3(c) shows, the system will traverse all the top transportable bricks of the first model and the constructing layer of the second model, then pair and transport each of the transportable brick to its paired empty slot until there is no more pairs

(The pairing system will be stuck). Then bricks from Stock will be used to finish current constructing layer.

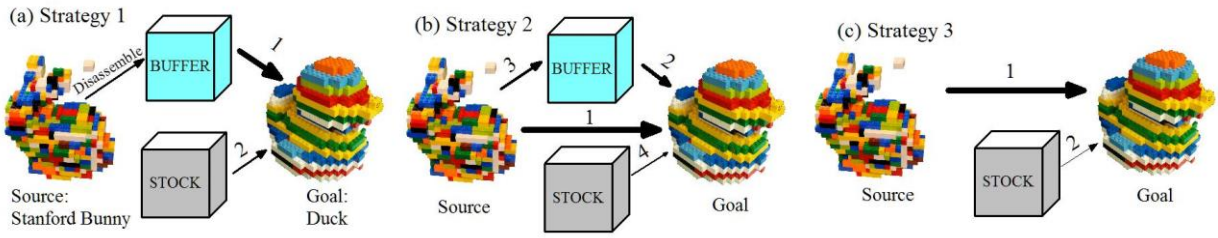

Fig. 3: The three different strategies elicits from and emulates human re-fabrication behavior, (a-c) display Strategy 1-3 respectively. The width of arrow and the number on it represent the priority of transportation.

Experiments and Results:

Fig. 4 describes the differences of each strategy across 10 different resolutions. Within each category, the first and second strategy perform better than the third strategy in terms of reused rate. However, in the case that transporting cost is the only concern or if the Buffer space is limited. The third strategy outperforms the rest without trivial transportation of brick from Source to Buffer. Compared with LFS, TFS shows a significant enhancement in the light of reused ratio, especially TFS2. TFS2 is even better than TFS1, which is because of the difference in the number of bricks in Goal (different layout). TFS is suitable for scenarios where reused rate is the first concern. Fig. 5 shows several frames of LEGO shape transformation from a hollowed Stanford Bunny to a hollowed Duck in which the sequences are obtained from the three strategies of LFS.

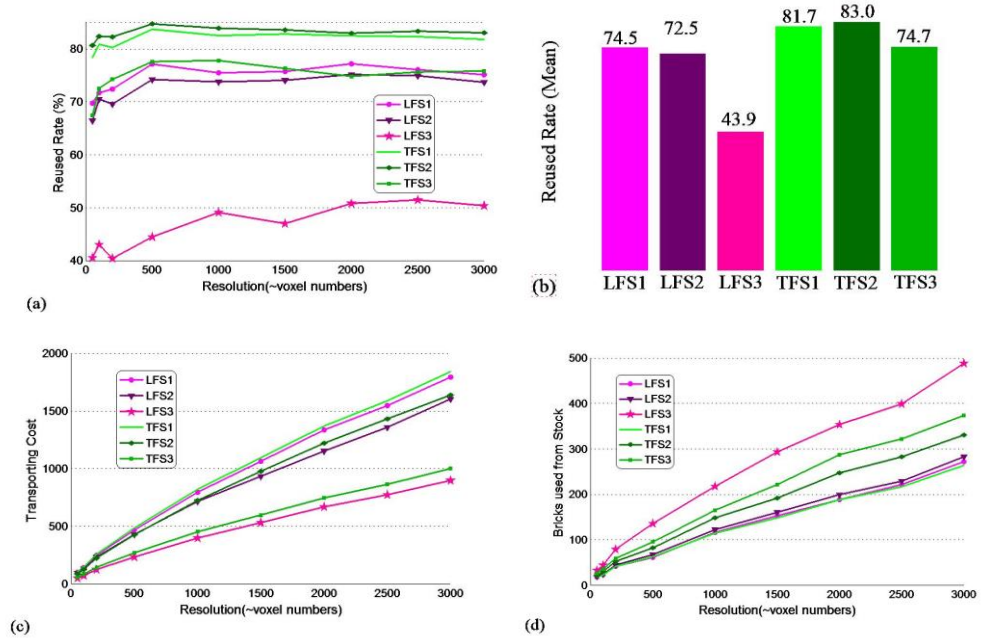

Fig. 4: Lego shape transformation method comparison in terms of (a) the average reused ratio, (b) mean value of the reused rate in (a), (c) the average cost value, (d) the average number of bricks used from Stock.

Based on current LEGO layout algorithms, this paper is aimed to solve the problem to transform one LEGO shape into another. The transporting instruction sequences are given as the solution so that it could be applied by both human or machines.

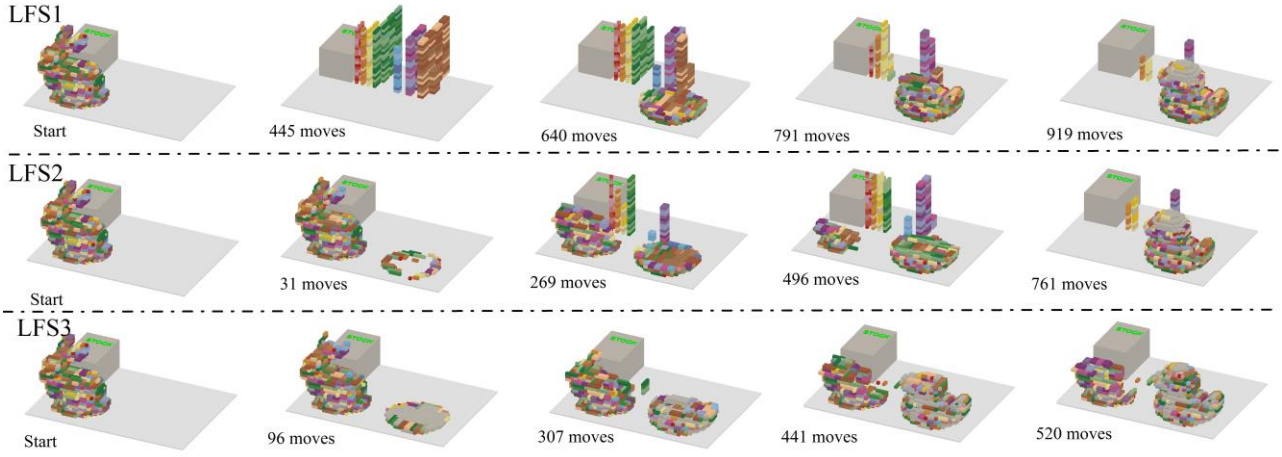

Fig. 5: A sequence of LEGO shape transformation from a hollowed Stanford Bunny model (2437 voxels, 445 bricks) to a hollowed Duck shape (2355voxels, 474 bricks) obtained from the three strategies of LFS. The seven basic bricks are labeled with different colors: 1 * 1 - red; 1 * 2 - orange; 1 * 3 - yellow; 1 * 4 - green; 2 * 2 - blue; 2 * 3 - purple; 2 * 4 - brown. The bricks labeled with light grey are from Stock.

References:

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

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

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

[4] 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. https://doi.org/10.1007/s00371015-1072-4

[5] Huang, J.; Yagel, R.; Filippov, V.; Kurzion, Y.: An accurate method for voxelizing polygon meshes, Volume Visualization IEEE Symposium, 1998, 119–126. https://doi.org/10.1145/288126.288181

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

[7] 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. https://doi.org/10.1145/2816795.2818091

[8] 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, 3827–3834. https://doi.org/10.1145/2559206.2574779

[9] Ono, S.; Alexis, A.; Chang, Y.: Automatic generation of Lego from the polygonal data, International workshop on advanced image technology, 2013, 262–267.

[10] Testuz, R.; Schwartzburg, Y.; Pauly. M.: Automatic generation of construct-able brick sculptures, in: Eurographics (Short Papers), 2013, 81–84.

[11] Timcenko, O.: Lego: How to build with Lego, 32nd European Study Group with Industry Final Report, 1998, 81-94.

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