


Approximating shapes with standard and custom 3D printed LEGO bricks

F. A. Fanni¹ , A. Dal Bello¹, S. Sbardellini¹ and A. Giachetti¹

¹Department of Computer Science, University of Verona, Italy



Figure 1: From left to right: the original model, the LEGO bricks placed according to the layout, and the resulting custom bricks, an exploded view of the custom bricks.

Abstract

In this paper, we present a work-in-progress aimed at developing a pipeline for the fabrication of shapes reproducing digital models with a combination of standard LEGO bricks and 3D printed custom elements. The pipeline starts searching for the ideal alignment of the 3D model with the brick grid. It then employs a novel approach for shape "legolization" using an outside-in heuristic to limit critical configuration, and separates an external shell and an internal part. Finally, it exploits shape booleans to create the external custom parts to be 3D printed.

CCS Concepts

• *Computing methodologies* → *Shape modeling*; • *Applied computing* → *Computer-aided design*;

1. Introduction

LEGO bricks are a well-known toy used by kids and adults to represent any kind of shape. More complex shapes are often built with more bricks, to facilitate the design of a feasible layout. Indeed, creating such a layout is not a trivial task, as the final result needs to be stable, physically robust, and still resemble the original shape. This has led many researchers to try and tackle the problem computationally, creating complex algorithms to generate and optimize the LEGO layout.

In particular, the idea of approximating 3D models with the assembly of LEGO blocks has been proposed in many recent papers. Typically, the idea is to voxelize shapes and aggregate voxels into larger bricks trying to make the resulting assembly connected and stable. The problem is quite hard, and no method able to guarantee

the requirements in the general case has been found. Actually, it is not difficult to prove that there are configurations that can not be built with standard LEGOs.

Although most techniques aim at optimizing the physical properties of the layout, using only the standard rectangular parallelepiped LEGOs, others face the issue of finding a better surface approximation, using sloped and curved bricks.

In our work in progress, we aim to combine standard LEGOs with custom 3D printed ones to obtain a perfect shape while retaining the classic experience of LEGO building.

In order to do so, we are developing a pipeline that generates the layouts with two objectives in mind:

- Brick connectivity, to ensure the whole shape can be constructed as one single piece, and improve stability.
- External shell thickness, to limit how much any external block can pierce inside the shape, and thus increasing the number of standard LEGOs that will not have to be 3D printed.

The pipeline uses a preprocessing step to better adapt the shape to the voxelization grid and exploits a novel "outside-in" approach for the aggregation of 1x1 bricks (voxels) in larger ones with graph-based priority rules.

2. Related Work

Many techniques have been proposed to generate compositions of LEGO bricks well approximating existing shapes. A survey discussing the first attempts found in the literature is presented in [KKL14]. The survey presents a list of papers trying to approximate shapes with LEGO bricks, typically starting from an object voxelization and trying to build a connected and stable assembly.

The basic idea is typically to define local cost functions maximizing bricks attachment and stability as proposed in [GHP98]. Based on such functions different strategies try to optimize the global properties of the assembly.

Testuz et al. [TSP13] aggregate 1x1 bricks iteratively, starting from a random voxel and merging it with the one that locally maximizes connections (randomly if the connections are the same). After the first step, the layout is iteratively refined trying to obtain a single connected component with few weak points, but with no guarantee of success. The method also outputs building instructions.

Recent works tried to improve the effectiveness of the optimization steps while maintaining a similar approach.

Lee et al. [LKKM15] propose a genetic algorithm creating mutations of layers configurations and maximizing connectivity while minimizing the number of bricks. In a later work [LKM18], the same authors propose a split-and-merge-based genetic algorithm (SM-GA) designed to always generate a feasible brick layout given a voxelized model considering the stability and connectivity between bricks.

Luo et al. [LYH*15] propose an iterative optimization of random voxel aggregation based on local and stochastic reconfigurations, performing a complex stability analysis based on the estimation of the forces connecting the bricks.

In [KS21] the authors use an adaptive large neighborhood search metaheuristic, in combination with a mixed-integer programming model neighborhood search procedure and a quadratic programming model to ensure connectivity and static equilibrium.

In [ZCX19] the authors explore the idea of using a combination of standard cuboid pieces and the less common sloped and circular bricks, to obtain a more faithful representation of the shape. However, the reproducible features are limited by the four extra pieces considered and have to match strict constraints on spacing and placing.

The idea of decomposing shapes in a mixture of universal building blocks and custom printed elements has been proposed in [CLF*18], even if not using LEGO bricks.

Extending the scope of the state of the art analysis we can find some resemblance with shape decomposition works.

In [SFLF15] and [TSW*19], the authors propose solutions to decompose the shape in interlocking pieces starting from the objects voxelizations. The stability of the result is given by the interlacing of the pieces, and a similar interlacing might be beneficial for the LEGO layouts as well.

Other works focus more on the fabrication purpose of shape decomposition. In [FCM*18] the shapes are decomposed in a set of pieces to be glued after the fabrication, and [MLS*18] introduces several constraints in the decomposition process to ensure fabricability with both 3D printers and milling machines. To avoid the gluing process, a different approach is taken in [LBRM12] and connectors are placed in the contact surfaces to wedge the pieces together.

One of the steps we perform in the pipeline is the search for the model optimal rotation. This step is often performed in the polycube maps ([THCM04]) creations, to increase the final polycube quality. In [GSZ11] a rotation-driven deformation approach is used to pre-align the mesh, while simultaneously initialize the polycube structure. Similarly, in [HJS*14] the L_1 norm is used to measure its deviation from an axis alignment during the optimization. These algorithms are however extremely complex and slower than the simple technique we propose, which is sufficient for our use case.

3. Preliminary Work

The pipeline we are working on comprises two steps. The first one is the design of a feasible LEGO layout from a suitable voxelization, and the second one is the creation of fabricable custom bricks. However, we believe an additional preprocessing step may deeply affect the voxelization quality, and thus the final results' quality.

3.1. Pre-processing: optimal rotation

We propose to start the pipeline looking for an optimal z rotation of the triangle mesh, thus maintain the vertical axis unchanged. Indeed, changing the vertical axis may be detrimental, as it might dramatically change the object's pose and its balance. A desirable orientation is one that aligns the flat surfaces of the model with the voxelization grid axis so that they do not suffer from aliasing.

To do it, we let each triangle cast a weighted vote. Each triangle vote the angle between the x axis and the projection of their normal on the xy plane. The same projected normal is then used to generate a plane, perpendicular to the xy plane by construction, against which the triangle is projected. The area of this newly projected triangle is then computed and used as the weight of the vote. This procedure allows us to weigh more the vote of large vertical triangles, and decrease the importance of almost horizontal ones. After collecting the votes we can use them in a kernel density estimation to extract a continuous function, taking care of modeling the boundary domain as cyclic. Then we can simply extract the maximum of this function and use it to rotate the mesh. The result is shown in Figure 2.

While this preprocessing can have a profound positive effect on

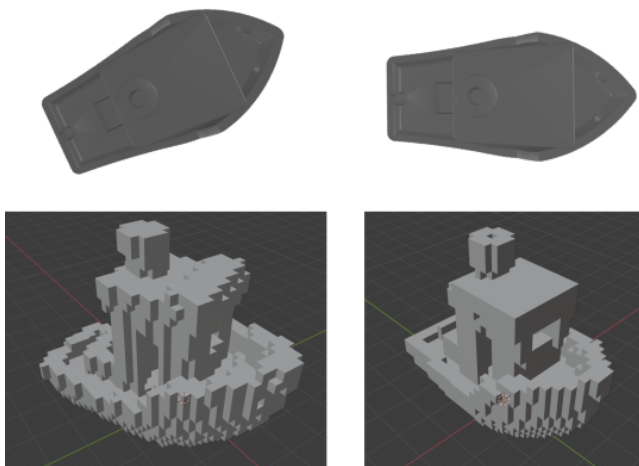


Figure 2: On the left, the original orientation of the boat and its voxelization. On the right, the optimized rotation of the same model creates a dramatically better voxelization

the voxelization, more experiments are needed to see whether it maintains its importance when combined with the fabrication step. Indeed, it is possible that the creation of the custom bricks renders it unnecessary. In this case, we also plan on testing a similar approach to find the orientation that creates a voxelization with the minimum amount of surface voxels. This could potentially be useful to reduce the number of custom bricks and decrease the fabrication times.

Once we have correctly rotated the mesh, we can create the voxelization. The most important aspect to consider is that it must fully contain the original mesh so that the custom bricks can be computed with simple boolean intersections.

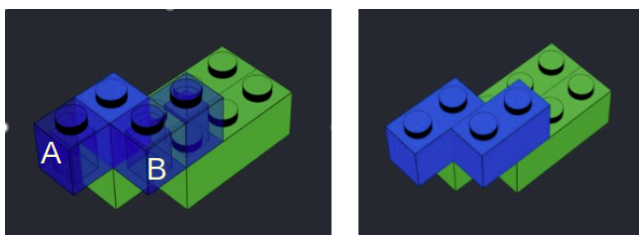


Figure 3: The aggregation of 1x1 blocks in the blue layer works as follows: starting from external elements, we select those that are not supported by the previous layer (A, B) then we apply a priority based on the number of neighboring non-merged 1x1 blocks so that A is the one with the highest priority.

3.2. Priority-based aggregation

After the voxelization of the shape, all the methods proposed in the literature aggregate single voxels corresponding to 1x1 bricks trying to create connected LEGO assemblies. A typical approach is to do this randomly in the first iteration and then iteratively modify the configuration. We are testing a different approach by sequentially

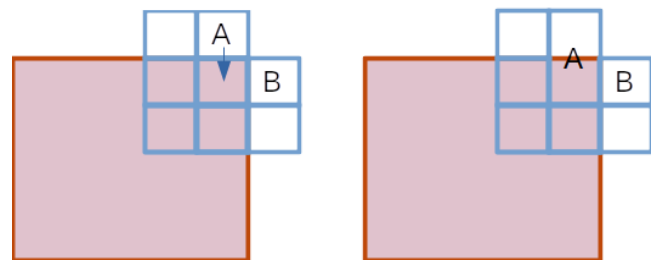


Figure 4: Example of voxel configuration for which it is not possible to find a connected Lego assembly. Red lines and the pink area represent a bottom layer; blue lines are the borders of the 1x1 bricks (voxels) of a top layer to be aggregated in larger bricks. It is clear that if voxel A is merged into a 1x2 brick (right), the 1x1 block B cannot be attached to the assembly with standard blocks.

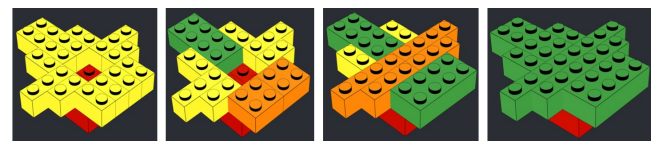


Figure 5: From left to right: external/non supported 1x1 bricks are first connected to the previous layer; 1xN bricks are merged into 2xN bricks; if the internal and external parts need not to be separated, bricks are then extended to fill the layer and finally merged where possible.

processing the layers and performing aggregations of 1x1 blocks using an outside-in approach and giving priorities based on the existence of supporting blocks in the previous layer and the number of side blocks that could be merged to the selected one. A further priority is set to favor the merging of bricks along alternate directions in consecutive layers, i.e. if layer n favors merging along the x-axis, then layer $n + 1$ will favor merging along the y-axis. When two border voxels of the layer have the same priority level the procedure is started randomly from one of them.

The method implemented highly increases the possibility of getting an immediately connected result compared to the classical random methods. In the example in Figure 3, we see an example of the effect of the neighbor-based priority: given the lower layer in green and the 1x1 blocks to be merged in the higher layer in blue (left), we see that A and B have the highest priority w.r.t. the others due to the lack of support, and A has a higher priority than B due to the lower number of side neighbors. This results in the aggregation shown on the right.

It is clear that this cannot guarantee connected results for each voxel configuration, also because it is easy to find examples where the aggregation of 1x1 elements into connected LEGO assemblies is not possible. This is an issue not discussed in the literature and may require investigation.

It is actually not possible, in some cases, to have all the external 1x1 elements connected with the rest of the assembly.

It is, however, worth noting that the occurrence of 1x1 blocks

of this kind is minimized by our priority-based aggregation and, as our pipeline includes 3D printing of custom elements, the issue can be solved by manufacturing a custom aggregation of 1x1 elements not included in the standard set (e.g. a L-shaped element).

The outside-in voxel aggregation is also good for the subsequent generation of custom bricks to improve the quality of the original shape approximation. In fact, in the case we want to replace external bricks with custom points the pipeline is only modified as follows: the merging of 1x1 external blocks into larger ones is stopped as soon a block is connected with the other layers. The following merging steps are performed only between these elements. When all the external elements have been merged, the remaining 1x1 blocks are separately merged. The first group of blocks is then post-processed by subtracting the part outside the original shape while maintaining brick connectivity as described in the next subsection.

After the processing of the external 1x1 blocks of the layer is completed, a further aggregation is done by merging adjacent 1xN blocks into 2xN blocks, merging adjacent 2x2 blocks into 2x4 blocks, increasing the length of 1xN blocks when possible. An example of the iterative steps is shown in Figure 5

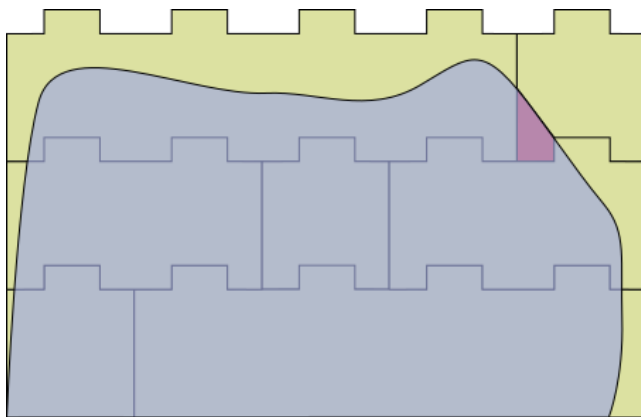


Figure 6: The highlighted red area represents a custom brick that can not be joined to the one beneath it. A simple solution could be merging it with its left neighbor.

3.3. Custom bricks generation

After the layout is computed we move to the generation of the custom bricks. As we mentioned before, the voxelization is built with the guarantee of completely containing the mesh. This allows us to scan through the LEGO design, detect which bricks are on the boundary of the layout, and compute their intersection with the mesh.

This is enough to obtain a collection of pieces that completely reconstruct the surface, but not to guarantee their assemblability. Indeed, the custom bricks might have multiple connected components or a reduced number of joints, as shown in Figure 6. We plan to face this issue by detecting these bricks and merging them with one of their neighbors.



Figure 7: From left to right, top to bottom, the layout of the boat in Figure 1.

4. Preliminary Results

The implementation of the pre-processing step to guarantee optimal orientation with respect to the discretization grid provided good results in our preliminary tests. The outside-in brick merging strategy provided connected results for many tested shapes, even in not trivial cases, like the boat model shown in Figure 1. Figure 7 shows the standard brick layers obtained on that shape, without any iterative optimization of the layout. This model is rather challenging, even if it doesn't show clearly the idea of mixing standard and custom bricks, as with the chosen discretization grid it is mostly composed of custom bricks at the end of the pipeline. An example of the opposite case with many standard blocks is shown in Figure 8. It must be noted that the current output still does not merge non-attachable blocks with the neighbors, but this feature will be added in the next future, as it is not particularly complex.

In future work, we will investigate the quality of the layout compared with the existing optimization-based approaches also introducing novel measurements to assess assembly connection (and

possibly stability) and to develop ad hoc optimization-based post-processing specifically designed for our method.

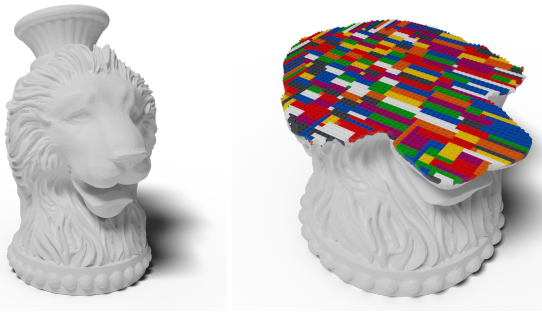


Figure 8: On the left the original model, on the right a section of the LEGO model

5. Discussion

The goal of our work is to develop a simple pipeline that can be used to easily create LEGO layouts, comprising of standard and custom bricks. We believe that our approach has some advantages over existing ones: the preprocessing allows the use of standard bricks, while the outside-in approach allows a faster convergence to a connected result (if existing) and easy separation of the custom shell from the internal standard blocks. Preliminary results are encouraging, demonstrating the ability of our legalization approach to often provide connected assembly without the need for iterative refinement. However, a relevant amount of research effort has still to be performed, for example, to improve the priority criteria for the voxel aggregation, to add a layout optimization scheme, and to remove non-attachable blocks in the custom set. We also plan to investigate on optimal solutions for creating the custom bricks using additive or subtractive fabrication techniques available in our labs (stereolithographic or multi-material inkjet 3D printers, CNC milling machines).

Acknowledgments This work was partially supported by the MIUR Excellence Departments project 2018-2022.

References

- [CLF*18] CHEN X., LI H., FU C.-W., ZHANG H., COHEN-OR D., CHEN B.: 3d fabrication with universal building blocks and pyramidal shells. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–15. [2](#)
- [FCM*18] FANNI F. A., CHERCHI G., MUNTONI A., TOLA A., SCATENI R.: Fabrication oriented shape decomposition using polycube mapping. *Computers & Graphics* 77 (2018), 183–193. [2](#)
- [GHP98] GOWER R. A., HEYDTMANN A. E., PETERSEN H. G.: Lego: Automated model construction. In *32nd European Study Group with Industry* (1998), Jens Gravesen and Poul Hjorth, Department of Mathematics, DTU, pp. 81–94. [2](#)
- [GSZ11] GREGSON J., SHEFFER A., ZHANG E.: All-hex mesh generation via volumetric polycube deformation. *Computer Graphics Forum* 30, 5 (2011), 1407–1416. [2](#)
- [HJS*14] HUANG J., JIANG T., SHI Z., TONG Y., BAO H., DESBRUN M.: $\langle 1 \rangle$ -based construction of polycube maps from complex shapes. *ACM Trans. Graph.* 33, 3 (June 2014). [2](#)
- [KKL14] KIM J. W., KANG K. K., LEE J. H.: Survey on automated lego assembly construction. [2](#)
- [KS21] KOLLSKER T., STIDSEN T. J.: Optimisation and static equilibrium of three-dimensional lego constructions. In *Operations Research Forum* (2021), vol. 2, Springer, pp. 1–52. [2](#)
- [LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: Partitioning models into 3d-printable parts. *ACM Trans. Graph.* 31, 6 (Nov. 2012). [2](#)
- [LKKM15] LEE S., KIM J., KIM J. W., MOON B.-R.: Finding an optimal lego® brick layout of voxelized 3d object using a genetic algorithm. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (2015), pp. 1215–1222. [2](#)
- [LKM18] LEE S.-M., KIM J. W., MYUNG H.: Split-and-merge-based genetic algorithm (sm-ga) for lego brick sculpture optimization. *IEEE Access* 6 (2018), 40429–40438. [2](#)
- [LYH*15] 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. [2](#)
- [MLS*18] MUNTONI A., LIVESU M., SCATENI R., SHEFFER A., PANOZZO D.: Axis-aligned height-field block decomposition of 3d shapes. *ACM Trans. Graph.* 37, 5 (Oct. 2018). [2](#)
- [SFLF15] SONG P., FU Z., LIU L., FU C.-W.: Printing 3d objects with interlocking parts. *Computer Aided Geometric Design* 35-36 (2015), 137–148. Geometric Modeling and Processing 2015. [2](#)
- [THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: Polycube-maps. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 853–860. [2](#)
- [TSP13] TESTUZ R., SCHWARTZBURG Y., PAULY M.: Automatic Generation of Constructable Brick Sculptures. In *Eurographics 2013 - Short Papers* (2013), Otaduy M.-A., Sorkine O., (Eds.), The Eurographics Association. doi:10.2312/conf/EG2013/short/081-084. [2](#)
- [TSW*19] TANG K., SONG P., WANG X., DENG B., FU C.-W., LIU L.: Computational design of steady 3d dissection puzzles. *Computer Graphics Forum* 38, 2 (2019), 291–303. [2](#)
- [ZCX19] ZHOU J., CHEN X., XU Y.: Automatic generation of vivid lego architectural sculptures. *Computer Graphics Forum* 38, 6 (2019), 31–42. [2](#)