faBrickation: Fast 3D Printing of Functional Objects by Integrating Construction Kit Building Blocks

Stefanie Mueller, Tobias Mohr, Kerstin Guenther, Johannes Frohnhofen, Patrick Baudisch
Hasso Plattner Institute, Potsdam, Germany
{firstname.lastname}@hpi.uni-potsdam.de

ABSTRACT
We present a new approach to rapid prototyping of functional objects, such as the body of a head-mounted display. The key idea is to save 3D printing time by automatically substituting sub-volumes with standard building blocks—in our case Lego bricks. When making the body for a head-mounted display, for example, getting the optical path right is paramount. Users thus mark the lens mounts as "high-resolution" to indicate that these should later be 3D printed. faBrickator then 3D prints these parts. It also generates instructions that show users how to create everything else from Lego bricks. If users iterate on the design later, faBrickator offers even greater benefit as it allows re-printing only the elements that changed. We validated our system at the example of three 3D models of functional objects. On average, our system fabricates objects 2.44 times faster than traditional 3D printing while requiring only 14 minutes of manual assembly.

Author Keywords: rapid prototyping; 3D printing; design iteration; building blocks; physical prototyping.

ACM Classification Keywords: H.5.2 [Information interfaces and presentation]: User Interfaces.

General Terms: Design; Human Factors.

INTRODUCTION
The recent development in rapid prototyping tools, such as laser cutters, milling machines, and 3D printers [8], allows users to prototype one-off objects and to iterate over designs. These tools offer sufficient shape complexity and resolution to allow prototyping functional objects, such as the body of a head-mounted display with its bosses and mounts for holding the lenses, the head-strap, etc.

While 3D printers can create objects with the necessary level of detail, they are also very slow. Since printing time is largely proportional to the volume of the printed object, fabricating the body of the head-mounted display shown in Figure 1d takes hours on current 3D printers (14:30h on our Dimension SST 1200es). Even printing with multiple heads [6] achieves only a constant speed-up.

CHI 2014, April 26–May 1, 2014, Toronto, Ontario, Canada.

http://dx.doi.org/10.1145/2556288.2557005

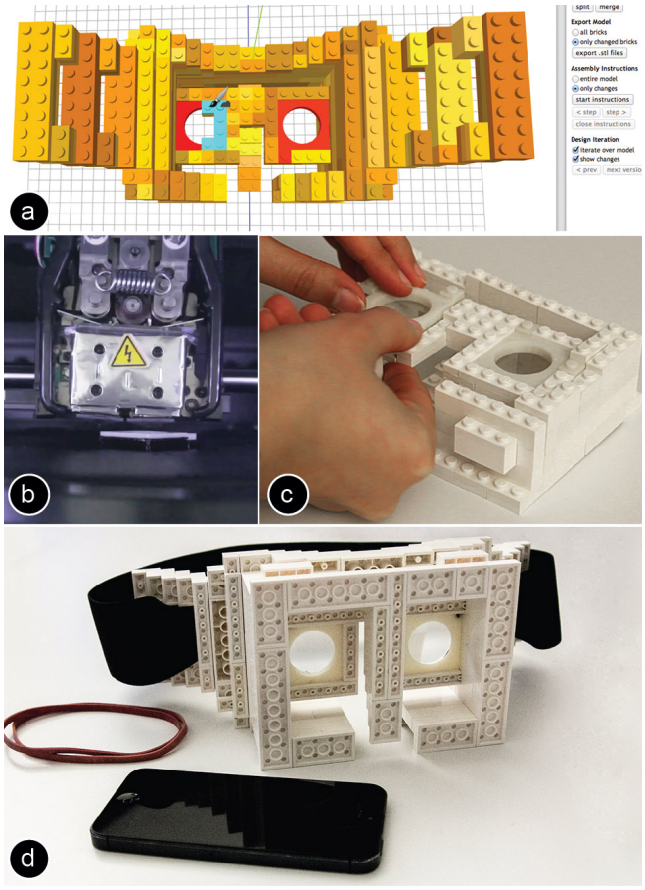

Figure 1: Let's fabricate this head-mounted display body quickly: (a) The exact shape of the lens mounts matters; the user thus marks them as "high-resolution" in faBrickator and (b) prints them. (c) faBrickator shows the user how to create everything else from Lego bricks and how to insert the 3D printed part. (d) Done in 67 minutes instead of the 14:30h for 3D printing.

Yet speed is mission critical: while printing can be done over night, this still limits designs to a single iteration per day. Thus a typical iteration process easily adds up to a week—even though the actual design work may not have taken longer than a day. As a result, in today's common design workflow, the 3D printer forms the bottleneck.

Experienced designers tend to address the problem by mixing fabrication techniques and weaving in prefabricated objects into the fabrication process. However, this requires a good amount of ingenuity and experience.

We propose addressing the problem by limiting 3D printing to where it is necessary—we use a much more rapid technique, the assembling of Lego bricks, everywhere else.

FABRICKATION: FAST 3D PRINTING WITH BRICKS

Figure 1 illustrates the rapid prototyping workflow we enable with our system faBrickator. faBrickator is a system that substitutes parts of 3D models with Lego bricks. Users specify, which areas they want to have executed in full detail, i.e., by the 3D printer. faBrickator then substitutes everything else with Lego bricks. Based on building instructions generated by faBrickator, users assemble the Lego part while the 3D printer is printing. Since it reduces the volume that needs to be 3D printed, faBrickator speeds up the fabrication process. While faBrickator already speeds up the fabrication of the initial prototype, it saves even more time when subsequently iterating over segments since users only have to reprint the parts that actually changed rather than reprinting the entire model.

Walkthrough: printing a head-mounted display body

In the following example session, the user rapidly prototypes the body of a head-mounted display using faBrickator. In order to get the optical properties of the display right, the exact shape and position of the lens mounts is crucial; the user thus decides to 3D print the lens mounts while fabricating the rest of the design in Lego.

The user starts by modeling the body of the head-mounted display in a 3D editor, here *Blender* (Figure 2). Alternatively, many users will choose to download a model from the web. The shown model would normally take 14:30h to 3D print. By using faBrickator to limit what gets 3D printed to the lens mounts, the user will be able to print the model in 67 minutes instead.

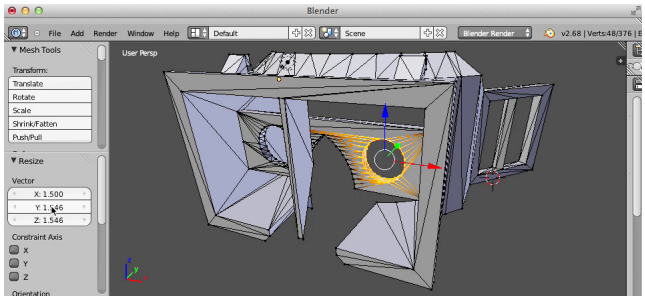

Figure 2: Walkthrough: The user has created a model of a head-mounted display body in *Blender*.

#1 Loading and legofication: The user is currently editing the model in Blender (Figure 2). As the user exits the *edit* mode, the 3D model is automatically loaded into faBrickator, initiated by the *Blender* plug-in we provide. In faBrickator (Figure 3), the user hits the *legofy*-button, which causes faBrickator to display a naïve legofication of the model, i.e., one that consists only of Lego's smallest building blocks—the 1x1 plates (Figure 3b). The user now hits the *layout*-button (Figure 3c). faBrickator responds by grouping the 1x1 plates into larger plates and bricks, so as to minimize the overall number of bricks. faBrickator also

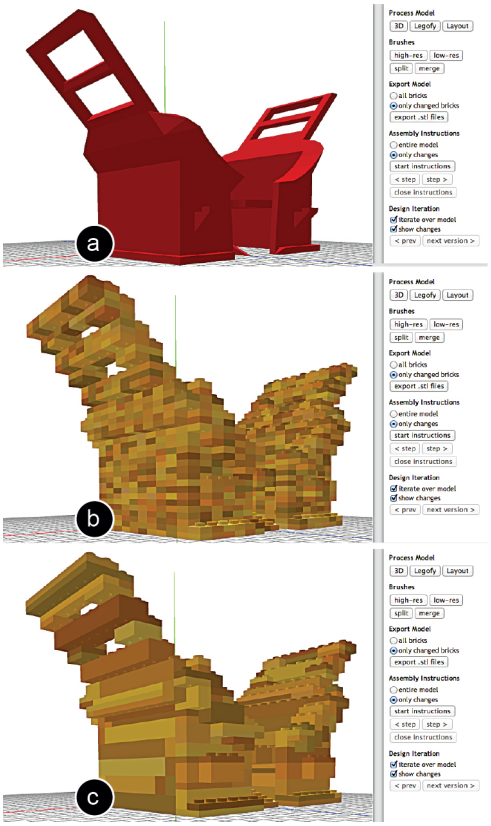

Figure 3: (a) The model is loaded into faBrickator, (b) legofied into 1x1 Lego plates, and (c) layouted.

assures a stable design in this step by iterating over the brick layout until all components are connected.

#2 Marking the high-res areas of the model: Figure 4 demonstrates how the user defines the regions that need to be 3D printed. (a) The user brushes the region around the lens mounts with the *high-res* brush. (b) This tells faBrickator to execute this area in full detail. The system responds by changing the preview accordingly. By default, faBrickator merges the marked area to a single 3D-printed block. The user repeats the process for the other lens opening.

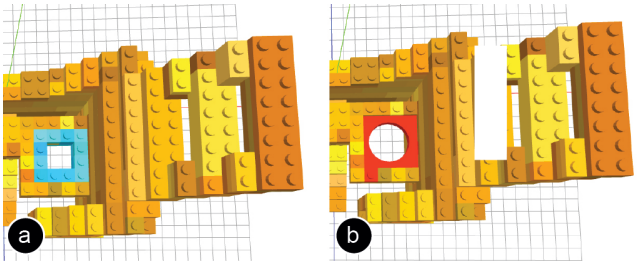

Figure 4: Using the *high-res* brush the user defines the region that will be 3D printed in full detail.

#3 Printing and assembling: To print, the user *exports* the model. This creates a 3D printable .stl file for each of the two lens mounts. The user then sends the files to the printer using the 3D printer software (in our case *CatalystEX*).

As soon as the 3D model has been sent to the printer, the user starts assembling the Lego part of the model. As shown in Figure 5, faBrickator provides animated assembly instructions that show how to assemble bottom-up, layer-by-layer. This helps to minimize the time needed for assembly.

67 minutes after the design was sent to the printer, the 3D printer is done fabricating the two lens mounts. Each brick bears a unique ID embossed into one of its sides (Figure 5b). This helps the user to quickly place them into the partially assembled Lego model.

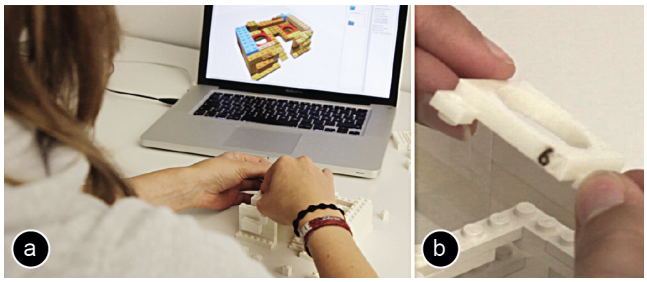

Figure 5: (a) Assembly instructions highlight the next brick in blue. (b) Each 3D printed part has a unique ID.

Figure 6 shows the assembled body of the head-mounted display with the accurately fitted lenses, which took 25 minutes to assemble. The overall fabrication time was thus still bound by the 3D printer, so that the 67 min is the final time for the entire fabrication. In this case, a speed-up of a factor of 12.99.

The main benefit of our approach is that it can change the fabrication process from overnight printing to something that can be carried out multiple times a day. This allows users to perform several design iterations in a day that would otherwise have stretched out over the course of a week.

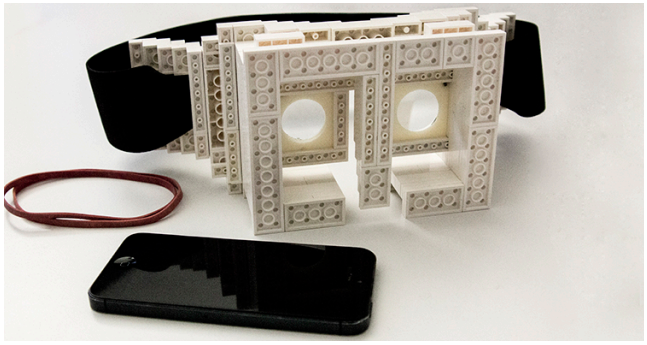

Figure 6: The final faBrickated model.

Switching from low-res to high-res parts later on

faBrickator allows users to change their mind later on in that it allows replacing additional Lego bricks with 3D printed parts. Figure 7 illustrates this, continuing our walkthrough: the optics of the head-mounted display work great, but after wearing the device for a while, the user notices that it does not sit comfortably on nose and fore-

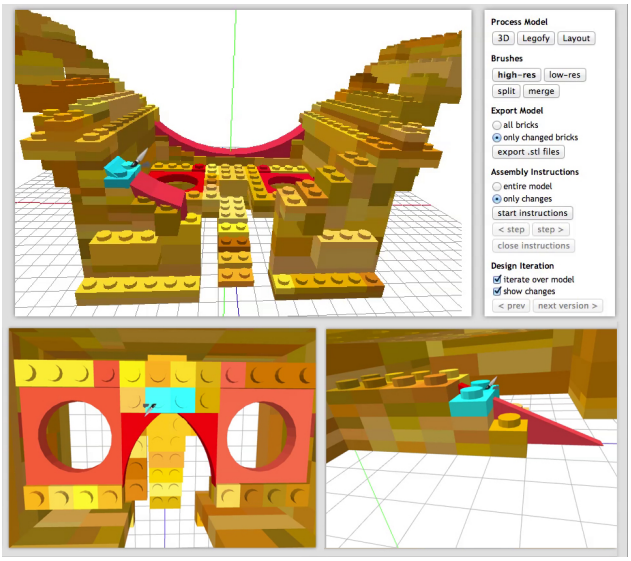

Figure 7: Making the forehead and nosepiece high-res for more comfort.

head. The user goes back into faBrickator to create more ergonomic forehead and nose pieces.

As shown in Figure 7, the user marks the forehead and nose regions as *high-res,* then exports and 3D prints them. Figure 8 shows how the user removes the corresponding Lego bricks from the faBrickated head-mounted display body and replaces them with the printed parts. This iteration took 3:55 hours including assembly.

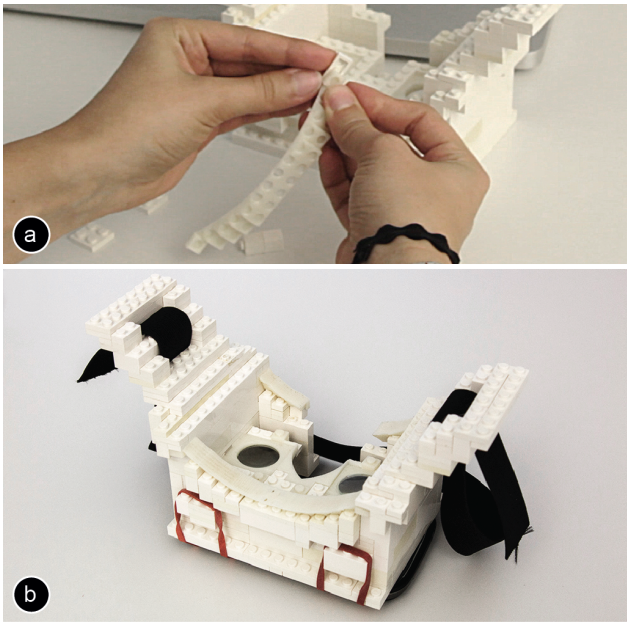

Figure 8: (a) Replacing bricks with the 3D printed parts for the head. (b) The assembled model.

FASTER ITERATION THROUGH LOCALIZED CHANGES

The second key benefit of faBrickator is that helps users iterate more quickly. Rather than reprinting the entire object, faBrickator allows users to make *local* changes to

an already printed object. As this allows for faster design iteration, it allows users to iterate more and thus ultimately leads to better designs [1].

Back in the 3D editor *Blender,* the user extends the nosepiece so as to perfectly fit the user's nose. After the user exits the edit mode in *Blender,* faBrickator aligns the current version of the 3D model with the previously printed one to minimize the number of bricks that need to be re-fabricated. Figure 9a shows how faBrickator highlights all bricks that changed compared to the last version. (b) The user now rebuilds the nose piece and (c) after printing exchanges it with the one that didn't fit perfectly.

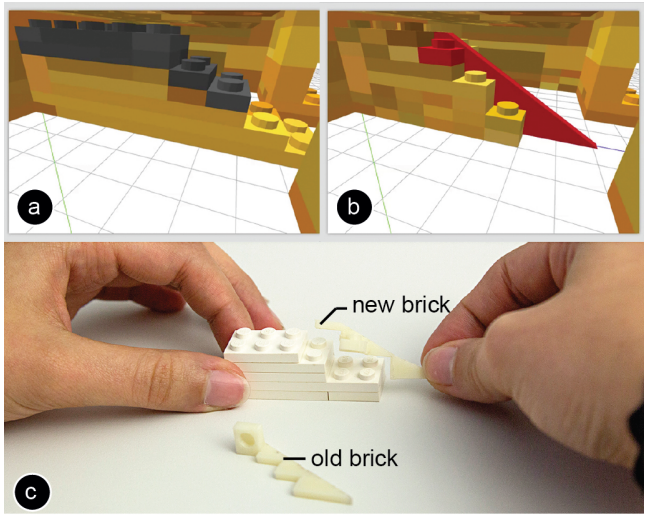

Figure 9: (a,b) faBrickator highlights what changed between two model versions. (c) The user then reprints only the part that changed.

CONTRIBUTION, BENEFITS, AND LIMITATIONS

The main contribution of this paper is a novel approach to fast 3D fabrication and a system called faBrickator that implements this approach. The main idea is to complement 3D printing with construction kit building blocks. The main benefit of our approach is that it allows for faster fabrication—2.44 times faster than traditional 3D printing on average (see section *Validation*). This allows users, such as industrial designers, to achieve several design iterations in a day that would otherwise extend across multiple days. faBrickator's approach is especially beneficial in situations that underlay external constraints such as changing customer needs and thus require many subsequent iterations.

faBrickator implements additional speed-ups by (1) printing in low-detail where possible, i.e., using Lego bricks, (2) allowing for fast iteration by limiting reprinting to those regions that have actually changed, (3) minimizing the effort for manual assembly by using different sized Lego bricks. As a side effect, it also allows repairing objects that are partially broken. Unlike approaches that are entirely based on Legos, our approach allows creating precise parts for regions that require additional accuracy.

On the flipside, faBrickator trades in detail for speed. While suitable for iteration, the partially legofied objects do require a full 3D printing process eventually, i.e., before being deployed. There is also a modest amount of manual effort required, since users need to assemble the Lego part of the object manually.

Finally, as discussed in the introduction, faBrickator applies to *functional* objects, i.e., objects that have shape constraints only in specific areas, which is commonly the case for objects that fulfill a functional purpose. Most of the speed-up faBrickator achieves results from the fact that a given functional purpose can be implemented in many different forms, allowing faBrickator to choose the one that can be fabricated fastest. This approach, however, does not lead to benefits when designing objects where the entire shape matters, such as artistic or decorative shapes.

IMPLEMENTATION

faBrickator is built in *CoffeeScript* using the *constructive solid geometry* library for its geometry operations. We also provide a *Blender* Plugin written in *Python* for easier file exchange between *Blender* and *faBrickator*.

#1 File Exchange between Blender and faBrickator

Our *Blender* plugin automatically exports the 3D model as an .stl file into a predefined folder as soon as the user exits the *edit* mode. faBrickator automatically detects if a new .stl file appears in the folder and imports it accordingly. The *Blender* plugin runs in a separate thread to prevent interference with *Blender's* editing functionality.

#2 Legofying: converting the 3D model into Legos

Figure 10 shows how faBrickator converts the 3D model into Lego's smallest building blocks—the 1x1 Lego plates. faBrickator first determines the bounding box of the model and extends it to the full multitude of a 1x1 Lego plate (8x8x3.2mm). Afterwards, faBrickator fills the bounding box with Lego plates in those positions that are either inside the model or share a portion of the volume with it. The remaining locations are left empty.

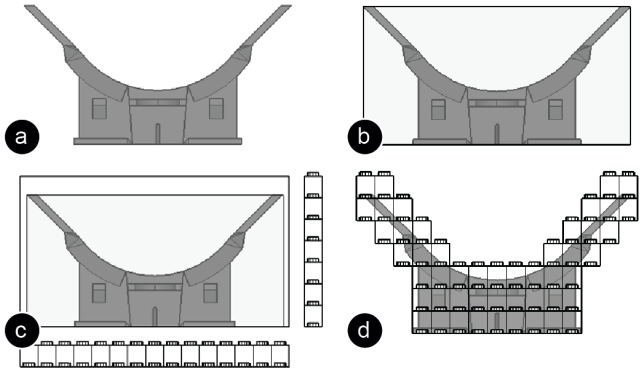

Figure 10: Legofying: (a) model, (b) bounding box, (c) extended bounding box, (d) legofied model.

#3 Layouting Lego Bricks

faBrickator's layout algorithm optimizes for two things: it minimizes the number of bricks used for assembly while

maximizing stability. For layouting, we use the layout algorithm proposed by Testuz et al. [22]. The algorithm works as following: (1) it merges the 1x1 plates into larger Lego bricks, (2) it creates a connectivity graph representing the brick layout, (3) it identifies weak points in the graph (unstable connections), and (4) tries to relayout the bricks at these positions in a different way to erase the weak points.

#4 Converting Lego bricks into 3D printable bricks

When users select a Lego brick with the *high-res* brush, it is converted into a 3D printable brick. In order to generate the 3D printable brick, we use a Boolean *intersection* of the Lego brick with the original 3D model geometry (see Figure 11).

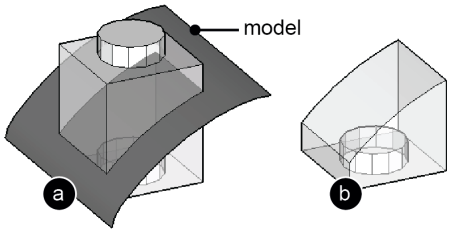

Figure 11: The 3D printable brick is a result of a Boolean intersection of the model and the Lego brick.

#5 Merging and splitting Lego bricks

When users select several bricks using the *merge* brush, they are merged into a single compound 3D printable brick. In our initial approach, we used the Boolean *union* operation to merge the bricks, which turned out to be very expensive. The reason is that the side faces of the two Lego bricks that should be merged are always co-planar, which is the most expensive case for a *union* operation.

Instead faBrickator uses a *tagging* approach for merging. As can be seen in Figure 12a, faBrickator tags each side of a Lego brick according to its orientation in space (e.g., side-y, side+y). In order to merge two bricks, we simply identify the matching neighbor sides by their tag and remove the sides since they would be inside the compound brick (Figure 12b). If later on users apply the *split* brush, we just enable the sides again.

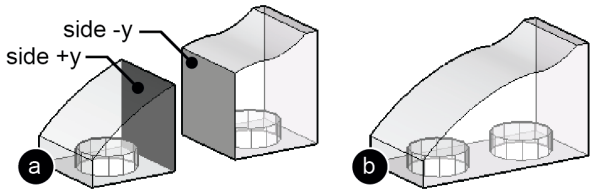

Figure 12: Merging: (a) The sides of each brick are tagged according to their orientation. (b) We use this to identify, which sides need to be removed for merging.

#6 Detecting the completeness of knobs and tubes

In order to ensure a stable brick layout, faBrickator needs to detect working knobs on a 3D printable brick. Only complete knobs can be used to make a connection with another brick. A knob is only complete if all of the knob's faces were inside the model before the intersection with the model (see Figure 13). If a part of the knob lays outside the model, it is cut off during the intersection and the knob is no longer functional.

A regular Boolean *intersect* determines inside and outside faces in the process of calculating the intersection. However, as the result the *intersect* operation only returns a geometrical volume from which it is difficult to tell if the knob is complete or not. faBrickator therefore has its own *intersect* operation that stores the meta-information about inside and outside faces in *tags*. Before the intersection, all the faces of a knob are tagged with the *knob-tag* (Figure 13a). During the *intersect,* faBrickator tags, which faces are *inside/outside* of the model (Figure 13b). Afterwards, we check if there is any face that was with the *knob-tag* that also has the *outside-tag*. If there is none the knob is complete. We use the same mechanism for the tubes on the bottom side of the bricks.

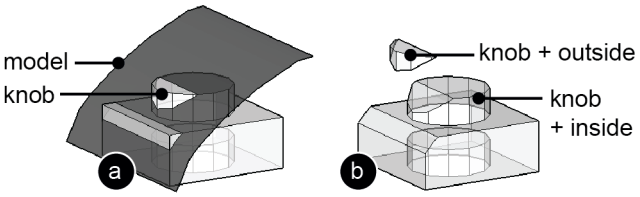

Figure 13: Detecting the completeness of knobs: (a) During intersection, we (b) tag if a part of the knob lies outside the model.

#7 Calculating changes between design iterations

When users iterate over a model, faBrickator displays which part of the model changed. For this, we first align the new 3D model with the old 3D model because faBrickator allows users to rotate the 3D model in *Blender* and to move it around freely. In order to align the two models, faBrickator calculates a transformation matrix based on the distance of specific model features to a reference point—we use the point of origin. The triangles of a model are not suited as model features because even small changes of the model result in a completely different tessellation during the export from a 3D modeling program (e.g., *Blender*).

In order to create a representation of the model that is comparable, we merge the triangles into larger sides based on their orientation. First we merge neighbor triangles that have the same normal into larger convex polygons. In a second step, we merge all convex polygons with the same normal into a side. We use these sides for comparison (i.e. their outline) and after finding a match, calculate the transformation matrix and align the models.

Now that the models are aligned, we legofy the new model. Afterwards, we determine which parts changed by calculating the differences based on the 1x1 Lego plates. If a plate was added or removed, the part changed. For bricks that were marked as *high-res* 3D printed bricks in the old model, we use a volume comparison to determine changes. For bricks that did not change, we copy the brick layout of

the old model to minimize the number of bricks that need to be reassembled.

#8 Eliminating Non-Manifold Geometry

When downloading models from the online platform *Thingiverse*, we experienced that many 3D models have invalid geometry, i.e. are not manifold/watertight. Therefore, we identified and cleaned the faulty geometry in the program *meshlab* before we further processed it with faBrickator.

VALIDATION

Beside the head-mounted display body, we *faBrickated* two additional example objects we downloaded from *Thingiverse*. faBrickator fabricates the three objects 3.11 times, 2.75 times, and 1.45 times faster (avg. 2.44 times) than traditional 3D printing while requiring only 14 minutes of manual assembly on average. How much faBrickator speeds up the prototyping process depends on how much volume can be substituted with bricks.

Figure 14 shows the first model: a soap bar dispenser that dispenses small flakes of soap when the sled is moved back and forth. It consists of two pieces: (1) the body that holds the soap and that the user moves with his dominant hand in the sled, and (2) the mount for the razor blades and the sled.

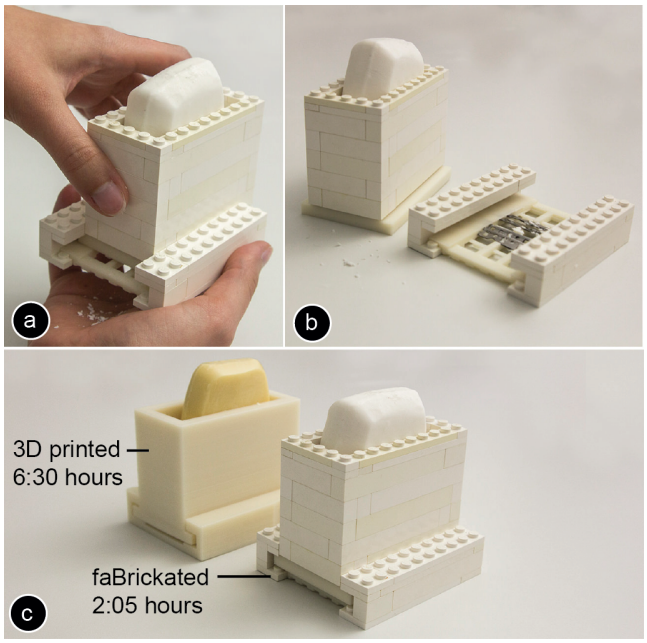

Figure 14: (a,b) This soap dispenser only takes (c) 2:05h and 5 minutes assembly compared to the 6:30h of traditional printing.

Producing this object with traditional 3D printing takes 6:30h compared to only 2:05h printing and 5 minutes assembly when using faBrickator (casing: 0:51h printing and 3 minutes assembly, razor blade: 1:14h printing and 2 minutes assembly). This is an improvement of a factor 3.11 for 3D printing time.

Figure 15 shows the second model: a "penny ballista" that allows ejecting a penny with a rubber band. The ballista consists of a mount for the penny and the ballista body with a sled.

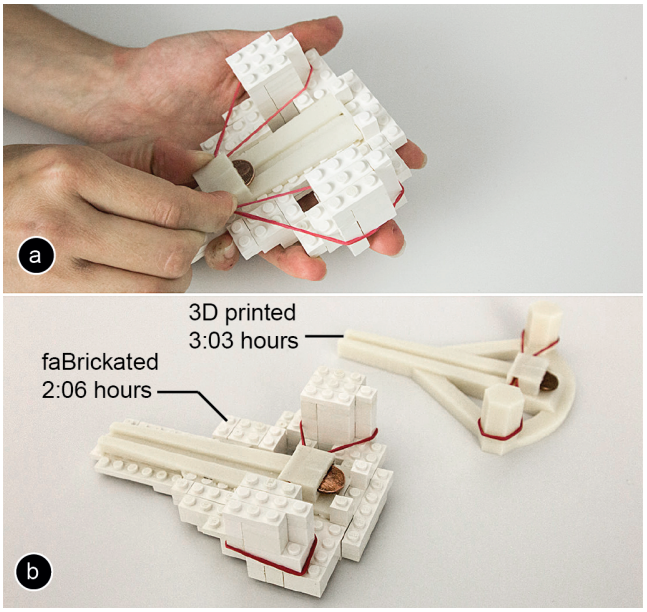

Figure 15: (a) A faBrickated penny ballista takes (b) only 2:06h for printing and 11 minutes for assembly compared to 3:03h when 3D printing entirely.

Using 3D printing, this model takes 3:03h compared to 2:06h printing and 11 minutes assembly when using faBrickator (penny mount: 0:59h printing, ballista body: 1:07h printing and 11 minutes for assembly). This is an improvement of a factor 1.45 for 3D printing time.

The third model is the already presented head-mounted display body with the lens mounts and the head and nose parts. When using faBrickator, it took 5:17 hours for 3D printing and only 25 minutes for assembly (lens mounts: 0:33h and 0:34h, forehead piece top: 1:26h, forehead pieces bottom: 0:41h and 0:44h, nose bridge: 0:25h, nose: 0:54h). This is an improvement of a factor 2.75 for 3D printing time, which for the completely 3D printed model is 14:30h.

While these examples give first insights into potential time-savings when using faBrickator, we think that faBrickator will provide an even greater speed-up when creating very large objects. We hope this will encourage users to start designing and fabricating objects that exceed the volume of current 3D printers.

RELATED WORK

The work presented in this paper builds on personal fabrication, fast fabrication of three-dimensional objects, and brick-based assembly.

Personal Fabrication

Personal fabrication devices such as 3D printers, milling machines, and laser cutters allow users to rapidly fabricate one-off physical objects [8]. The traditional workflow in

personal fabrication requires users to first create a digital model in a CAD program. A range of projects in HCI lower the entry barrier to 3D modeling by restricting the space of possible objects to chairs (*SketchChair* [19]), plush animals (*Plushi* [10]), furniture (*FurnitureFactory* [13]), lamps (*SpatialSketch* [26]), and dresses (*Dress-up* [24]). Recently, researchers proposed interfaces that do not only allow the design of passive objects, but also the design of interactive prototypes. *Midas* [17] enables users to prototype capacitive touch layouts with a vinyl cutter. *Printed Optics* [25] and *Papillon* [1] create optical sensors and display elements. Finally, *Sauron* [18] allows creating interactive controllers via optical marker tracking.

Interactive Fabrication

Interactive fabrication systems [27] do not require the user to model the entire object before fabrication, but offer an alternation between user and system control. Since users need to wait for the machine after each input, fast fabrication is a key element of interactive fabrication systems to keep users in the flow [3]. *ModelCraft* [20] allows users to annotate on a paper model and then implements the changes by reprinting the model. *CopyCAD* [4] enables users to copy geometry from existing objects using a milling machine. *constructable* [11] re-introduces speed and precision using a fast laser cutter and a system of constraints. Other systems combined traditional fabrication with interactive fabrication: *Position-correcting router* [16] automatically corrects the user's routing path according to a loaded 3D model. *FreeD* [30] is a milling machine that can stop its spindle to prevent users from making mistakes.

Fast Fabrication of Three-Dimensional Objects

While 3D printers can create the most complex shapes, they are also very slow. Different approaches try to reduce printing time by either massively parallelizing the printing process using multiple heads [6] or by assembling objects layer-wise from prefabricated voxels of equal size [7]. However, as printing resolution increases, fabrication gets slower since more voxels or layers need to be fabricated. Laser cutters on the other side are fast, but can only cut 2D pieces. *LaserOrigami* [12] showed how to use a laser cutter to create 3D objects by cutting and folding an acrylic sheet in a single integrated process.

Complementing Objects with 3D Printed Parts

Several projects used 3D printed parts to repair or to complement existing physical objects. In *Hybrid reAssemblage* [29] the authors repair broken vases by first scanning their geometry and then replacing the part with a 3D printed structure. This process is also common in medicine for replacing parts of broken bones [20]. Petchkowvsky [14] finally enhanced existing physical objects with additional 3D printed features that perfectly blend into the shape and texture of the existing object. faBrickator, in contrast, combines existing bricks with custom 3D printed parts to speed up the rapid prototyping process while assuring accuracy in the areas that need more detail.

Making Objects From Bricks

The hardest problem in building any 3D object from bricks is the automatic generation of a constructable brick layout. Researchers proposed different algorithms, such as using a cost function for simulated annealing [5], evolutionary algorithms [15], beam search [28], and cellular automata [23]. Besides aiming at minimizing the overall number of bricks, an additional focus is on stability, i.e. to prevent bricks from falling off [22]. faBrickator extends these layout algorithms for its 3D printed bricks, which have irregular shapes and therefore additional constraints that need to be taken into account.

CONCLUSION

In this paper, we presented a new approach to rapid fabrication and our system—faBrickator—that implements this approach. The key idea is to substitute sub-volumes of 3D models with standard building blocks. We demonstrated how the modular approach of faBrickator is especially useful when iterating over a design since only the parts that changed are reprinted – not the entire model.

As future work, we plan to extend faBrickator so as to work with a wider range of building blocks and objects in order to further decrease printing time. We also plan to improve the assembly instructions according to the Lego specification. Finally, we aim at automating the assembly process by building on existing tools that are able to assemble standard Lego bricks [9].

ACKNOWLEDGEMENT

We thank the 3D printing lab of the Berlin Institute of Technology for sharing their 3D printing resources, Alexandra Ion for helping with the 3D prints, and Serafima Gurevich for feedback.

REFERENCES

1. Brockmeyer, E., Poupyrev, I., Hudson, S. PAPILLON: Designing Curved Display Surfaces With Printed Optics. *Proc. UIST'13*, 457-462.
2. Buxton, B. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, '07.
3. Csikszentmihalyi, M. *Flow: The Psychology of Optimal Experience*. Harper Perennial Modern Classics, '08.
4. Follmer, S., Carr, D., Lovell, E., Hiroshi, I. CopyCAD: remixing physical objects with copy and paste from the real world. *Adjunct Proc. UIST '10,* 381-382.
5. Gower, R., Heydtmann, A., Petersen, H. LEGO: Automated Model Construction. *European Study Group with Industry. Study Report '98,* 81–94.
6. Hansen, C. J., Saksena, R., Kolesky, D. B., Vericella, J. J., Kranz, S. J., Muldowney, G. P., Christensen, K. T., Lewis, J. A. High-Throughput Printing via Microvascular Multinozzle Arrays. *Advanced Materials '13, Vol. 25, Issue 1*.
7. Hiller, J., Lipson, H. Methods of Parallel Voxel Manipulation for 3D Digital Printing. *Proc. SFF Symposium '07,* 200-211.

8. Gershenfeld, N. *Fab: The Coming Revolution on Your Desktop--From Personal Computers to Personal Fabrication.* Basic Books, '07.
9. Lego Assembler, 2010. http://mikeshouts.com/Lego-made-3D-printer-that-builds-Lego-out-of-Lego/
10. Mori, Y., Igarashi, T. Plushie: an interactive design system for plush toys. *SIGGRAPH '07, No. 45.*
11. Mueller, S., Lopes, P., Baudisch, P. Interactive Construction: Interactive Fabrication of Functional Mechanical Devices. *Proc. UIST '12,* 599-606.
12. Mueller, S., Kruck, B., Baudisch, P. LaserOrigami: Laser-Cutting 3D Objects. *Proc. CHI'13*, 2585-2592.
13. Oh, Y., Johnson, G., Gross, M., Do, E.Y. The Designosaur and the Furniture Factory. *Proc. DCC'06*, 123-140.
14. Petchkovsky, G. Mixing Digital Sculpture with Real Objects. '12.
15. Petrovic, P. Solving the LEGO brick layout problem using evolutionary algorithms. *Tech. rep., Norwegian University of Science and Technology '01.*
16. Rivers, A., Moyer, I.E., Durand, F. Position-correcting tools for 2D digital fabrication. *Proc. SIGGRAPH'12 (TOG), Vol. 31, Issue 4, No. 88.*
17. Savage, V., Zhang, X., Hartmann, B. Midas: Fabricating Custom Capacitive Touch Sensors to Prototype Interactive Objects. *Proc. UIST '12,* 579-588.
18. Savage, V., Chang, C., Hartmann, B. Sauron: Embedded Single-Camera Sensing of Printed Physical User Interfaces. *Proc. UIST'13*, 447-456.
19. Saul, G., Lau, M., Mitani, J., Igarashi, T. SketchChair: an all-in-one chair design system for end users. *Proc. TEI '11*, 73-80.
20. Science News, skull implant, 2013. https://www.sciencenews.org/article/plastic-implant-replaces-three-quarters-mans-skull
21. Song, H., Guimbretière, F., Lipson, H., Hu, C. ModelCraft: Capturing Freehand Annotations and Edits on Physical 3D Models. *Proc. UIST '06*, 13-22.
22. Testuz, R., Schwartzburg, Y., Pauly, M. Automatic Generation of Constructable Brick Sculptures. *Proc. Eurographics '13,* 81-84.
23. Van Zijl, L., Smal, E. Cellular automata with cell clustering. *Proc. Automata '08,* 425-441.
24. Wibowo, A., Sakamoto, D., Mitani, J., Igarashi, T. DressUp: a 3D interface for clothing design with a physical mannequin. *Proc. TEI '12*, 99-102.
25. Willis, K.D.D., Brockmeyer, E., Hudson, S.E., Poupyrev, I. Printed Optics: 3D Printing of Embedded Optical Elements for Interactive Devices. *Proc. UIST '12,* 589-598.
26. Willis, K.D.D., Lin, J., Mitani, J., Igarashi, T. Spatial sketch: bridging between movement & fabrication. *Proc. TEI '10*, 5-12.
27. Willis, K.D.D., Xu, C., Wu, J.K., Levin, G., Gross, M.D. Interactive fabrication: new interfaces for digital fabrication. *Proc. TEI '11*, 69-72.
28. Winkler, D.: Automated brick layout. BrickFest, 2005.
29. Zoran, A., Buechley, L. Hybrid reAssemblage: An Exploration of Craft, Digital Fabrication and Artifact Uniqueness. *Leonardo Journal '13, Vol. 46, Issue 1*.
30. Zoran, A., Paradiso, J.A. FreeD: a freehand digital sculpting tool. *Proc. CHI'13,* 2613-2616.