

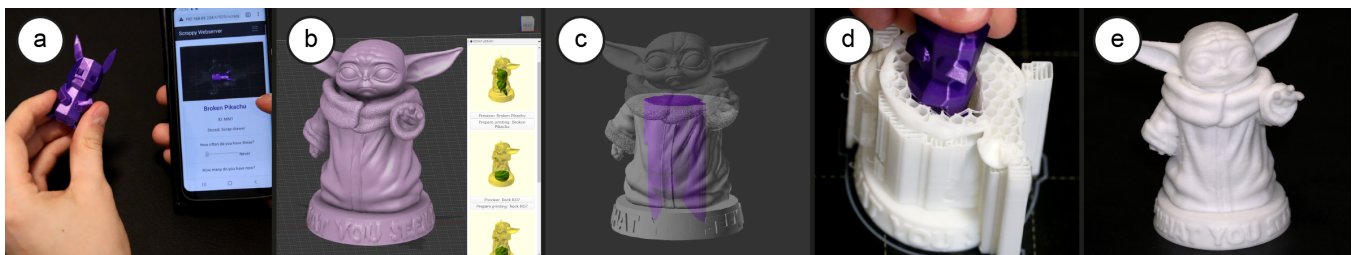
# Scrappy: Using Scrap Material as Infill to Make Fabrication More Sustainable

Ludwig Wilhelm Wall  
lwall@uwaterloo.ca  
University of Waterloo  
Waterloo, Canada

Daniel Vogel  
dvogel@uwaterloo.ca  
University of Waterloo  
Waterloo, Canada

Alec Jacobson  
jacobson@cs.toronto.edu  
University of Toronto  
Toronto, Canada

Oliver Schneider  
oliver.schneider@uwaterloo.ca  
University of Waterloo  
Waterloo, Canada



**Figure 1:** Scrappy is a system that lets users use scrap objects as infill to reduce time, cost, and material: (a) a library tracks scrap objects, like an old print, as well as other common objects; (b) an add-in for Fusion 360 suggests scraps from the library that can fit into a model, sorted by saved printing time; (c) an algorithm finds the best scrap placement and generates a modified model; (d) a custom slicer optimizes infill and adds machine commands to pause the printer with instructions how to insert the scrap into the object; (e) the final printed object has the scrap inside, reducing the material, time, and energy needed to print.

## ABSTRACT

We present a software system for fused deposition modelling 3D printing that replaces infill material with scrap to reduce material and energy consumption. Example scrap objects include unused 3D prints from prototyping and calibration, household waste like coffee cups, and off-cuts from other fabrication projects. To achieve this, our system integrates into an existing CAD workflow and manages a database of common items, previous prints, and manually entered objects. While modelling in a standard CAD application, the system suggests objects to insert, ranked by how much infill material they could replace. This computation extends an existing nesting algorithm to determine which objects fit, optimize their alignment, and adjust the enclosing mesh geometry. While printing, the system uses custom tool-paths and animated instructions to enable anyone nearby to manually insert the scrap material.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

CHI '21, May 8–13, 2021, Yokohama, Japan

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-8096-6/21/05...\$15.00

<https://doi.org/10.1145/3411764.3445187>

## CCS CONCEPTS

• Human-centered computing → Interaction tech.

## KEYWORDS

fabrication, geometry processing, sustainability

### ACM Reference Format:

Ludwig Wilhelm Wall, Alec Jacobson, Daniel Vogel, and Oliver Schneider. 2021. Scrappy: Using Scrap Material as Infill to Make Fabrication More Sustainable. In *CHI Conference on Human Factors in Computing Systems (CHI'21)*, May 8–13, 2021, Yokohama, Japan. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3411764.3445187>

## 1 INTRODUCTION

Personal fabrication methods naturally generate scrap. Our formative study showed that failed prints, tests, and offcuts occur naturally in the iterative, creative practice of designing and testing physical objects. While experts might strive to use material efficiently, there will still be waste; with novices, there will be even more. These scrap objects are often thrown out, but some end up cluttering work spaces with the hope they can be used in the future. In some fabrication environments, bags of waste objects are stored for years with the hope of finding an eventual use. The process of re-melting waste filament to create new spools is commonly known, but almost never implemented, since it requires specialized machines and reduces filament quality.

Meanwhile, fused deposition modelling (FDM) 3D printers, the most consumer friendly and widespread computer aided fabrication tool, use time, energy, and material to create infill for critical internal support structures. This infill sparsely fills the internal volume, most commonly using a repeating pattern such as a honeycomb. We propose to repurpose scrap by integrating it into FDM prints, reducing the material consumption of infill while also finding a new use for scrap objects. Compared to recycling methods like re-melting old prints into new filament [1], our approach requires no additional equipment, does not degrade printed material, and can reuse many other types of scraps, objects, and materials. With the exception of food and other materials that can decay quickly, most materials are suitable as scrap in our system. Examples include single-use packaging, such as jars, lids, and boxes, discarded hardware, such as CDs, floppy discs, and tools that come with construction kits, or consumable items, such as markers, chopsticks, glue sticks, and empty tape spools. Since our system works on a wide variety of scrap materials, even users who already have the means to re-melt their old prints into new filament can benefit from our system. During testing, we were able to continue printing on all tested materials, which were stone, wood, metal and plastics.

Recycling is a conscious choice that many Do It Yourself (DIY) supporters have already made. It takes some additional time and effort for example to wash out an empty jar so that it can be recycled, instead of just putting it in the trash. Recycling requires an object to be reduced to its raw materials so that it can be re-injected into the manufacturing cycle, which requires additional energy. Similar to recycling, using our system requires some additional effort. Unlike recycling objects that have no more use though, our system provides a way to directly repurpose them in their current form. This repurposing effort in turn provides direct cost savings to the user, as well as contributing to a more sustainable lifestyle.

We introduce Scrapy, a system that automatically searches a library of scrap objects for ones that fit into a CAD model, ordered by how much infill it would replace. The implemented system shown consists of a Fusion 360 plugin to monitor the modelling process, search for fitting scrap objects in the background, and compute a hollowed 3D model and printing instruction files to fabricate the model with the selected scrap inside (Figure 1). Our insertion engine that handles the search for possible scrap inserts is based on an existing 3D nesting algorithm [13]. A web-based scrap library keeps track of scrap objects, helps users find them in the workshop, and allows them to add new scrap. A customized slicer software generates printer code optimized for scrap insertions. We demonstrate how Scrapy handles common scrap, including previous outdated or broken prints, single use packaging, broken hardware, and many more, showing that scrap-filled objects require less material, time, and energy to fabricate.

The costs for this approach are a single short user interaction during fabrication, additional computation, and maintaining a scrap library. Most importantly, scrap has to be inserted into the model while it is being printed. Determining which scraps can fit inside a model and adjusting the model accordingly requires additional computation. And finally, the system requires knowledge about which scrap objects are available for insertion, and this set of scraps must be filled. Our system design minimizes these costs. It supports the user in maintaining a scrap library, providing simple

ways to add new scrap, automatically erasing integrated scrap from the database, and providing guidance for identification and location of binned scrap objects. This effort in return provides cost savings to the user, as well as contributing to more sustainable fabrication.

We contribute the idea of re-using many types of scraps as 3D printed infill, and a working system enabled by a novel combination of a custom add-in, slicer modification, a purpose-built end-to-end database, and a novel extension of a geometry processing algorithm.

## 2 BACKGROUND AND RELATED WORK

Prior work has established several techniques that make fabrication more sustainable, though often as a by-product of trying to speed up iteration. We discuss prior work through the lens of sustainability, even though most of these works were motivated by speeding up iteration or increasing print sizes.

### 2.1 Optimizing prints to reduce material usage

Developing ways to reduce the amount of material necessary to print a model saves time and money, and indirectly also increases sustainability. For example, systems from Mueller et al. [19] and Peng et al. [22] approximate the objects' surface, by printing a wireframe version of the geometry, but this reduces fidelity and is therefore only applicable for certain kinds of rapid prototyping. If object detail is important, material consumption can be reduced by optimizing the required infill to support all exterior walls of the object [14], or to optimize only for expected loads [11, 26]. Our system follows a different approach to reduce infill and can be applied in conjunction with infill optimization for even greater material savings. It furthermore works not only for prototypes that approximate the geometry, but also for production quality prints.

### 2.2 Recycling filament material

A different approach to sustainable fabrication is to directly recycle the material of prints that are no longer in use. It has, however, been shown that the mechanical material properties of recycled materials degrade with each recycling iteration [21], and the recycling process itself is energy intensive. Systems that are deployed in changing environments can be made more sustainable by employing a modular system design that allows reusing most of the parts while only fabricating and switching modules when circumstances change [24].

### 2.3 Printing onto objects

Attempting to fit a fabricated part to an existing object often requires multiple iterations before achieving the desired fit. This process can be sped up by incorporating the target object directly in the fabrication process. Reducing the amount of iteration also makes the process more sustainable. Integrating existing objects into the design and fabrication of 3D models has been explored in various ways. Encore [7] is a method to print on top of existing objects to enhance them, and to print through loops of existing objects, such as the handles of scissors. To do so, the loop is placed inside a printed trough during fabrication, before printing on top of the trough which closes a second loop through the first one. Other

projects aim to customize the fabrication process to specific objects, for example by integrating a touch screen into the fabrication machine that allows capturing the objects outline and modelling around it directly in Fitter [2], or by pushing an object into resin actively being cured to create a perfect mold or fixture around it [29].

## 2.4 Construction systems

One way to make fabrication more sustainable is to create or incorporate a construction system that approximates 3D models. Prior work has already incorporated ready-made objects, such as specific bottles or cans, in 3D models to replace parts of large structures instead of printing them. TrussFab [17] for example uses certain types of bottles to create sturdy human scale truss structures, and ReFabricator [27] approximates large arbitrary shapes using everyday objects. This enables the creation of structures that surpass the printing volume of the printer that helped create them. By printing only connectors and approximating the overall shape of the 3D model, these projects also reduce the amount of printing material that would otherwise be needed to fabricate them. Fabrication [20] instead uses Lego bricks to replace most of the models' geometry while only 3D printing the parts of the object in high fidelity that are crucial to its function. While mainly intended to reduce print times, this system allows reusing parts of the models once they are no longer needed. In contrast to these works, our system instead aims to improve the sustainability of prints without approximating the desired original shape.

## 2.5 Inserting objects into prints

Printy3D [28], HotFlex [12] and other systems [8, 9, 25] incorporate electronic components and physical widgets to enhance the functionality of fabricated objects. Similar to our system, these components are inserted during printing. Medley [6] creates fittings during the modelling of the object that allow adding parts such as screws, metal wire, or sandpaper after fabrication to enhance the objects mechanical properties. Our system also involves integrating foreign objects, but for a different goal. To investigate the design of nesting objects, Jacobson created a generalized way of approximating the maximum scale at which one model fits within another [13]. Our inclusion engine is based on their algorithm, but solves a different problem. Both will be discussed further in section 5.2.

## 3 FORMATIVE INTERVIEWS

To help guide our work, we interviewed two experts who run makerspaces (P1, P2) and two people with makerspace experience who now work in industrial fabrication (P3, P4). We discussed challenges in fabrication and prototyping, and attitudes and practices regarding sustainability in fabrication. The interviews were 1-hour long on average. Three participants were male, and one was female.

*Sources of Waste.* We found that makerspaces are regularly visited by people who do not have a lot of experience making things and who do not own fabrication equipment themselves. A lack of experience makes it more likely to encounter problems. They may require more iteration before successfully fabricating their intended design. Makerspaces thus generate lots of waste (P1, P2).

Expert users, especially those who run makerspaces, care about being sustainable (P1-4). Makerspace managers (P1, P2) recycle what they can, and store the rest. One of the makerspaces (P1) essentially stores all PLA waste indefinitely, since it can not be recycled easily by their municipality. Re-melting waste into filament to create new spools would be beneficial, but it is difficult. Waste material first has to be ground up into small shreds and its thickness has to conform to tight tolerances. The machinery required to create quality filament would require an investment comparable in size to a small printer farm, and none of the makerspaces frequented by our participants had such equipment. One participant (P4) heats up and presses old filament from support material into plates for laser cutting, but these plates are smaller than desired.

Much of the stored material has a “recognizable shape” (P1), so repurposing some of it seems feasible. Makers often hold on to failed prints and early prototypes (P1, P3, P4), sometimes until they move (P4). They keep these objects as a reminder to avoid specific printing issues (P1, P4), out of sentimental reasons (P4), or in the hope that they can one day reuse them somehow (P1). All four participants stored some kinds of waste material like failed prints or off-cuts at least temporarily, and when asked, all were willing to store failed prints and other scrap for longer if they could benefit from them in the future.

*Print Time is Valuable.* Since there are usually more users than machines in a makerspace, makers often have limited access to machines. Printing time is as much of a valuable resource as the filament itself. The managers of the makerspaces (P1, P2) provide guidance, training, and proven printing settings to those with less experience. They try to optimize these printing settings for speed and to minimize material usage, but increasing the chance of successful prints usually still takes priority. They encourage makers to use other prototyping methods, such as making sketches and paper prototypes, to make low fidelity prototypes, and to print models at a smaller scale first. Makers however rarely make use of these methods, as “they need permission to make it low fidelity.” (P1). Makers still require two to twelve iterations before finishing a project, with more experienced makers requiring fewer iterations than novice makers.

*Collaboration.* Makers often collaborate on their projects, for example, by sharing printing tips and by brainstorming on how to realize challenging designs. They prefer watching the printing process or at least being in the same room to be able to intervene quickly in case of a printing issue (P1, P3, P4). They even collaborate with strangers, for example by asking someone else to watch their print while they are away (P4). Most would even intervene when they notice an issue without being asked to watch over the process, or empty the build plate and start a queued up print of another maker.

*Available Software and Hardware.* Our participants have experience in other types of fabrication beyond FDM printing, for example with stereolithography resin printing (P2, P3), using CNC routers or laser cutters (P1, P2, P4), or designing clothes digitally (P3). They use a variety of software tools to support them in their fabrication efforts, and all of them were experienced in Autodesk Fusion 360 [4]. Their experience was not limited to digital fabrication either,

and many had experience in woodworking, machining, electronic repairs, and so on.

*Summary.* Our interviews showed that makers and makerspaces store a variety of essentially useless objects. They desire faster processes and being more sustainable, both in using less material and in finding ways to reuse their waste materials. They also regularly observe fabrication processes and are willing to interact with the processes of other makerspace users within the same space. Fusion 360 is a commonly used 3D modelling software in makerspace environments. These results were encouraging, and we tried to incorporate ways to repurpose common waste materials with recognizable shapes, such as old and failed prints and off-cuts, as we designed our system.

## 4 WORKFLOW AND EXAMPLES

We walk through how our system works, using the example of designing and printing a Tardis (the phonebooth model in Fig. 2). Before beginning the project, this user or others have populated a library of 3D models of scrap objects. These could be past prints or objects that might otherwise be thrown out. In our example, we assume the model of an unwanted salt-shaker was entered using a procedurally generated *jar* geometry.

The user begins the project in Autodesk Fusion 360 by modelling or loading the the Tardis. Our Fusion 360 add-in examines the Tardis geometry and the geometries of all available scraps and recommends feasible scrap objects that could be inserted within the Tardis model. This recommendation can either occur in the background as the user updates the Tardis geometry, or on demand by clicking a button. The add-in provides previews for how different fitting scrap objects could be integrated within the Tardis, such as the unwanted salt-shaker. These different options are sorted in descending order by their volume to make it easy to identify the most beneficial insertion objects. The previews also allow users to identify and avoid inserts with potential printing issues, such as gaps around the inserted scrap that are too wide, or to make changes to the model that could accommodate scrap alignments that are easier to print. Our algorithm for determining insertion feasibility ensures that the scrap material does not stick out of the model and collide with the moving print head.

The user can then either load a preview scrap object directly into the scene to inspect its alignment further, or request the hollowed geometry: the original exterior of the Tardis with an embedded inner cavity snugly fitting the scrap. Both identifying possible scrap inserts and generating the hollowed meshes are performed on a dedicated server. This separation of tasks ensures that the 3D modelling process runs smoothly, even while potentially computationally intensive procedures are performed in the background. The user decides to print the Tardis using the first scrap shown in Fig. 2 (b).

Our add-in exports the exterior and interior surfaces to our scrap-aware 3D printing slicing algorithm, and the printing begins. The generated G-Code commands include a `PAUSE` command at the moment when the user should insert the scrap, moving the print head out of the way for clearance. Precise insertion instructions are simultaneously visualized on the printer’s display screen. The user inserts the scrap and confirms that printing should resume.

The final result is a 3D printed model of a Tardis with a salt shaker embedded inside, saving time, material, and energy.

### 4.1 Populating the Scrap Library

The scrap library can be populated by uploading an existing 3D model, using the failed print generator, using one of the packaging generators, or the plate outline generator. In the previous example, the salt shaker mesh was created using the jar packaging generator.

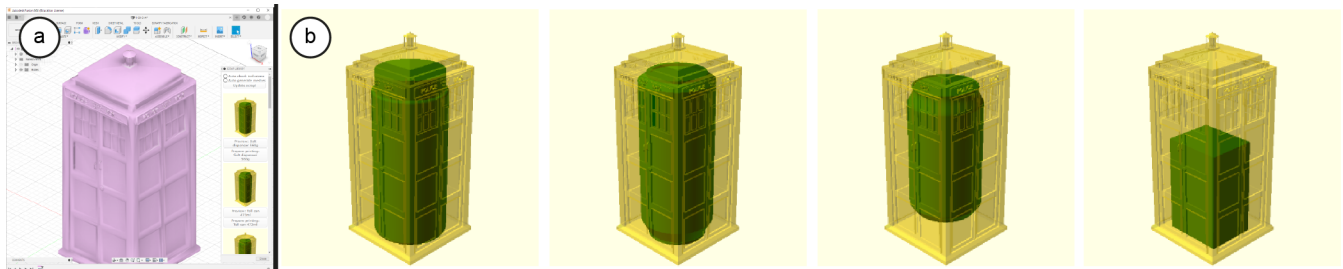
*Upload existing model.* When a 3D model of the scrap object readily exists, we can simply load it into the library. This covers a large number of scenarios: previous successful prints, objects with standardized or well-known geometries found online (e.g., CD cases, bottle caps, GameBoys), or 3D models acquired of unique physical objects through 3D scanning.

*Failed prints.* 3D printing does not always succeed. As a result, most makerspaces have an abundance of partial prints and failed prints. Our scrap library captures a common type of failed print, where printing proceeds successfully up until a point of failure. A variety of issues can lead to this kind of failure, such as a clogged printing nozzle, running out of filament, or the print getting knocked loose off the build plate. For other types of failures that continue printing, but with corrupted geometry, a user could cut through a failed print along the z-axis plane below the height at which the error occurred, only using the lower half with known geometry for scrap. In either case, the digital 3D model is faithful to the physical artifact until the height reached along the print axis of the failure or cut. This distance can be determined by the time of print failure or simply by measuring the height. Based on this height, we intersect the original 3D model with a half-space, recovering just the portion of the successful print. Debris (e.g., “FDM spaghetti”) should be pruned before including in the scrap library. Figure 3 shows the result of such a printing failure and the output of our generator.

*Packaging generators.* Many common objects have a simple shape parameterized by a few easy-to-determine measurements. For example, disposable coffee cups are generally conical and well approximated once the bottom and top radii are measured with a ruler or caliper. We provide web-based 3D-model generators for seven types of objects: boxes, jars, lids, two different types of cans, and two different types of cups. These generators use descriptively named measurements to approximate any of these common forms of one time packaging, including options for generating rounded corners and slopes.

Some types of packaging have large internal voids. For the purpose of inserting them as scrap it can be beneficial to treat these voids as filled with material as well. Simple physical scrap modifications can avoid potential bridging issues caused by the voids. For example, adding two strips of masking tape across the opening of a coffee cup provides enough support to print upon and bridge the void. Our packaging generators contain export flags to treat internal voids either as hollow or as filled to enable such scrap modifications.

*Plate outline generator.* Many 3D shapes (e.g., wood off-cuts and laser cutting scraps) are one-dimensional extrusions of a 2D shape. For such scraps, we can generate 3D geometry by allowing the user



**Figure 2: Fusion 360 integration: (a) our add-in panel displays a list of compatible scrap with previews; (b) detail of previews showing compatible scraps that would fit into a large model, such as a plastic salt shaker, a beer can, a soda can, or a cardboard packaging box.**



**Figure 3: Comparison between the result of a failed 3D print and the geometry created by our generator: (a) A 3D print that failed mid-fabrication. (b) The mesh generated of our failed print generator.**

to input a 2D outline (e.g., measured with a flat bed scanner or a camera-phone measurement app) and a height value. Precise digital outlines may already be available for off-cuts from laser cutting.

We provide an initial set of common scrap objects. This library of models can be accessed, altered, and extended via a website called scrap library. Our initial set contains 3D model files portraying common household garbage, such as disposable coffee cups, lids of jars, and over a dozen of different types of food containers. We furthermore initially include famous objects for 3D printing, such as a 20mm calibration cube, or the 'benchy' boat model. Also included are 3D models of common collectibles, such as an Ikea hex key, for

a total of 40 models. Renderings of all 40 objects can be seen in Figure 8 in Appendix A.

We provide software tools to maintain the library and simplify retrieval of scrap items in the makerspace or at home. Our library keeps track of customizable meta data for each scrap object. For example, users can specify how many of these objects they have on hand, or whether they generally have repeatable access to this kind of scrap, such as a daily disposable cup of coffee. The scrap library also holds an image of the object (either uploaded by the user, or automatically generated from the 3D model), and provides a unique ID that can be used to mark and later re-identify visually similar objects. It furthermore allows entering a name and a description per scrap item, such as a note where the user would usually keep this item stored, and finally also the option to upload a picture of the item in its storage location.

## 4.2 Examples

Our system can make use of many different kinds of scrap materials and geometries, the example objects that we fabricated highlight some of them. The example objects show a range of use cases, such as decorative models, tools, parts of mechanisms, and printer maintenance.

The example shown in Figure 4a inserts an outdated model into a new one. Geometries that have a flat surface on one side are well suited as scrap objects. Many 3D models have a flat bottom which securely affixes the model to the build plate during fabrication, which makes them well suited for inserting as scrap. Figure 4b demonstrates a jar lid inserted into a gear. The shape of the lid forms a bridge-like shape. During FDM 3D printing, support material is often used to create bridges, which supports perimeters that are printed at a steep angle, or that are disconnected from the rest of the model. This scrap performs a similar function. The broken handle inserted into the scraping tool shown in Figure 4c performed a similar function to the model that it is inserted into. Geometries of objects that perform similar functions are likely to resemble each other. As such, it is likely that such a scrap object is fitting well into such a model.

Figure 4d shows a model that is often printed repeatedly to calibrate 3D printers. With many flat sides and a large volume compared to its surface area, this geometry is likely to fit many different models. Since this type of model is being printed repeatedly, it can also be used to print a larger version of the same model using a comparable amount of material as the original print. Since this



**Figure 4: Examples showing scrap, computed mesh, and printed model: (a) an outdated print inside a new one; (b) household waste inside a mechanical part; (c) the broken handle of a hand tool inside another tool, (d) and a recurring print used to calibrate printers inside a bigger version of the same model.**

model is used for calibration purposes, it needs to maintain truthful printing dimensions and the look of the outer walls of the object. Our system does not alter outer walls, and we found that in practice, our algorithm and slicing manipulation creates a snug fit that does not flex the outer walls. We show the fabrication process for many of our example objects in our supplemental video.

## 5 SYSTEM IMPLEMENTATION

Each stage of our system would be difficult or time consuming to reproduce using existing tools or with manual effort. For example, populating a scrap library without generators would require resorting to expensive, time consuming, and error prone 3D scanning. Determining the best scrap object that fits into a given target shape is an intractable problem to solve with a standard 3D modelling tool. Small changes in the orientation and position of the inner object can cause the scrap to pierce the target shape. Finding a configuration where the scrap fits feasibly inside the target shape is not even a sufficient condition. It must be possible to *insert* the scrap object without being blocked by already printed parts of the target shape and it must be possible for the print head to move around to continue printing the remaining target shape after insertion. Given a feasible placement of the scrap, the 3D printer’s instructions must be adapted to leave a hollow space affording insertion: again, not

just the hollow space of the inserted scrap, but a hollow *path* for the scrap to move along. Naively sending an outer surface and inner surface to a standard 3D printing slicing algorithm could actually result in *longer* printing times and increased material usage. This is because standard slicers will reinforce *all* surfaces with shells to ensure a clean surface finish. Our modified slicing routine removes these material and time intensive print lines for the inner (non visible) surface. The synthesis of these user interface, algorithmic and design contributions combine into a time, energy and material saving system.

### 5.1 Fusion 360 add-in

Our system is integrated into the 3D modelling process in the form of a Python Fusion 360 add-in. Using an add-in allows users to benefit from the full modelling capabilities of popular CAD software, while also seamlessly integrating our technique. The add-in creates a new side panel within the Fusion 360 GUI. The panel shows previews of potential scrap able to fit inside the model edited. To gather possible scrap inserts in the first place, the add-in handles exporting the open 3D model, simplifying and potentially repairing it, and sending these files to the server side of our system. Upon receiving scrap insertion options from the server, the add-in creates preview images and displays those to the user.

The user may load any of the scrap meshes that were discovered to fit the current model directly into the active modelling scene to inspect it further, or to use it as reference while continuing the modelling process. After making a decision which scrap to include, the user can request the hollowed meshes from the server, if they were not already provided automatically in a background process. As a final step within Fusion 360, the user may then export the hollowed meshes, including the information when to pause during the printing process to allow inserting the scrap. While exporting, our add-in also provides guidance on how to locate stored scrap, as well as a link to the scrap library for the specific scrap item. The scrap database is then updated to record that the scrap items was used.

Most server communication and related computations are handled in asynchronous background threads to make sure that Fusion 360 remains responsive while scrap inserts are evaluated, or while the hollowed meshes are generated. Since most of the add-in functionality runs concurrently, all scrap operations are associated with time stamps and revision numbers to ensure that the add-in never presents outdated data to the user. Some of the functionality implemented by the add-in makes use of external open-source software. Preview images are rendered by invoking OpenSCAD [16] via the command line. Similarly, a Python script invokes Blender [10] to simplify and repair meshes. The add-in stores local copies of scrap models to minimize network traffic, only transmitting newly added files or files that have been modified. It also keeps a persistent storage of scrap metadata using JSON files to minimize start-up times of the add-in after the first use.

### 5.2 Insertion engine

Given a library of scrap models, our goal is to present the user with an ordered list of scraps that can be safely inserted into the target shape during printing. A valid scrap must not only fit within the target shape geometrically, there must also exist an insertion path

for the scrap without crossing the target shape’s surface. Furthermore, we must identify the time of insertion during the print so that the part of the scrap “sticking out” after insertion is so minimal that it doesn’t colliding with the print head when printing resumes.

Although quite distinct in motivation, this geometric problem bares many resemblances to the “Generalized Matryoshka” algorithm by Jacobson [13]. In that work, the goal is to create Russian nesting dolls for arbitrary input shapes: the outer object is split in half and the inner object is inserted within it. Like our scenario, their algorithm considers not just feasible containment, but also feasibility of the insertion path. Distinct from our problem, Jacobson’s method seeks to *maximize* the scale of the inner object to achieve tight nesting, while our inner object has a fixed scale as it already exists in the physical world. Jacobson’s method also has no requirement that the inserted object must not stick out too much.

We propose several modifications to Jacobson’s algorithm so that we can leverage their overall modelling of the problem and optimization technique. For readability, we give a brief overview of Jacobson’s method, but refer the reader to that paper [13] for details.

Jacobson’s algorithm first assumes that the inner object will be inserted along a straight trajectory (pure translation). Since the path of each point on the inner object follows a straight line, the problem can be recast as a visibility problem and Jacobson’s method implements the following algorithm on the GPU for maximal parallelization. The placement of the inner object is parameterized by the 3D position of its center of mass and its 3D rotational orientation. The insertion action is parameterized by the 3D cut plane (in Jacobson’s case, literally where the outer object is split in two; in our case, the height at which printing is paused) and 3D direction vectors describing insertion into the top and bottom halves. Given specific values for these altogether 16 parameters, Jacobson’s visibility based algorithm returns the maximal scale at which feasibility occurs (returning zero if no solution exists). In an outer loop, Jacobson’s method searches the 16-dimensional space using the particle swarm optimization method [15] to maximize the scale parameter. Particle swarm optimization is a global optimization method that does not require the objective function (maximum feasible scale) to be differentiable. While many other global optimization techniques exist, Jacobson reports this method works well for the low-dimensional matryoshka problem. The core subroutine that is needed must take as input a candidate configuration and output the maximal feasible scale. The optimization returns the largest encountered feasible scale after a fixed number of iterations.

Our problem can be understood as a modification of the Generalized Matryoshka problem; surprisingly, the extra constraints and assumptions made in our problem will in many ways simplify the problem and improve performance.

We assume that the user has determined the print direction for the outer object. This is reasonable and welcome as 3D printing surface quality, material strength and use of support material depend on the print direction. Since 3D printers proceed in layers perpendicular to the printing direction, this assumption implies that the “cut plane” must also be perpendicular to the known printing direction. Reducing the four parameters of the cut plane to a single degree of freedom. Similarly, we assume that the object

will be inserted along the printing direction (i.e., straight down), eliminating the insertion direction parameters.

We must prevent too much of the inserted scrap from sticking out above the cut plane, so we short circuit the optimization for any configurations where the maximum distance of the inserted object is too far (0.6mm for our setup) and return zero. This quick check not only avoids the more expensive visibility check, but will discourage the particle swarm optimization from exploring the vast regions of the configuration space where printer-head collisions would occur.

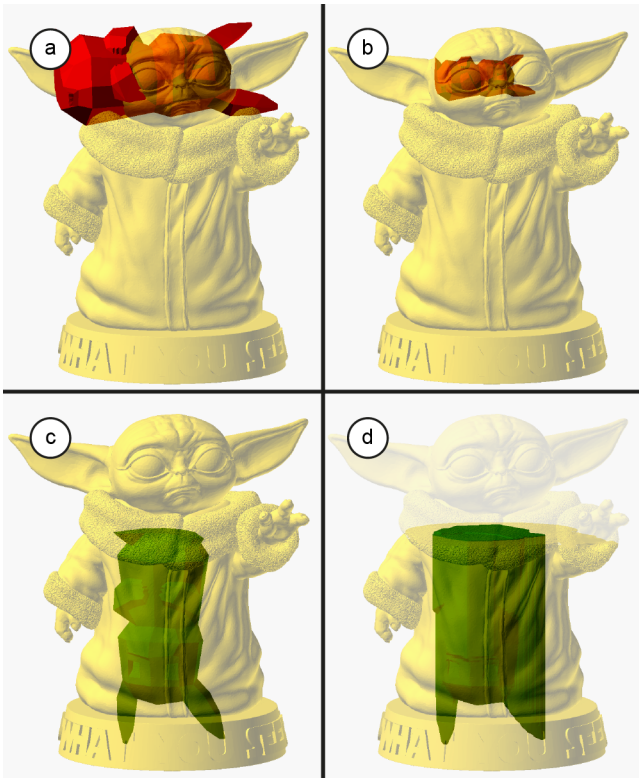
Unlike Jacobson’s problem, we ultimately do not want to *maximize* the scale of the inner object. Rather we want to find a feasible configuration where the scrap object can be inserted at 100% scale. Figure 5 demonstrates the feasibility of configurations. For a given configuration, Jacobson’s proposed method conducts a binary search to determine the largest scale possible that passes the visibility test. If we ever identify that a configuration is feasible at 100% scale, then we terminate the optimization early and return this solution. If the particle swarm optimization concludes without finding a configuration at 100% scale, then we declare that this scrap is not feasible and do not present it as an option to the user.

Since we must determine feasibility for each scrap in the library, we first conduct two inexpensive culling procedures before invoking the modified-Matryoshka optimization. If the scrap’s total volume is larger than the total volume of the target shape, we can reject immediately. For both shapes we construct a tight fitting oriented bounding box, and then exhaustively check whether any of 10000 random rotation of the scrap’s box would fit inside the target’s box with a 10% and 20% margin of error for the longer and shortest dimensions, respectively. While not strictly conservative, this simple test is extremely fast to compute and culls away most of the “obviously” infeasible scraps.

As proposed by Jacobson, we also pre-dilate the inner shape to account for 3D printing accuracy tolerances and pre-erode the target geometry to account for minimal wall thickness. When a scrap shape is selected by the user, we generate the interior surface geometry by conducting the swept volume of the inflated scrap geometry along the insertion direction. Figure 6 demonstrates this process. This swept volume’s surface is tagged as an “inner shell” so that our modified slicing algorithm can process it distinctly from the outer shell of the target shape.

### 5.3 Custom slicer

Our customized slicer is a modification of the Kiri:Moto slicer [3]. Typically, this and all other standard slicers will proceed layer-by-layer determining all intersections of the layer plane with the input surface geometry. This intersection curve is traced with multiple high-density “shell” paths and then the interior between shells is filled with a low-density, fast-to-print pattern (e.g., honeycomb). Shells ensure quality surface finish, but we do not care about this for the internal boundary between the fill pattern and the yet-to-be inserted scrap. Since we have tracked which surface parts correspond to the target shape and the inserted scrap’s swept volume, we post-process the layer paths from Kiri:Moto and remove the paths corresponding to shells of the inner surface (see Fig. 7).

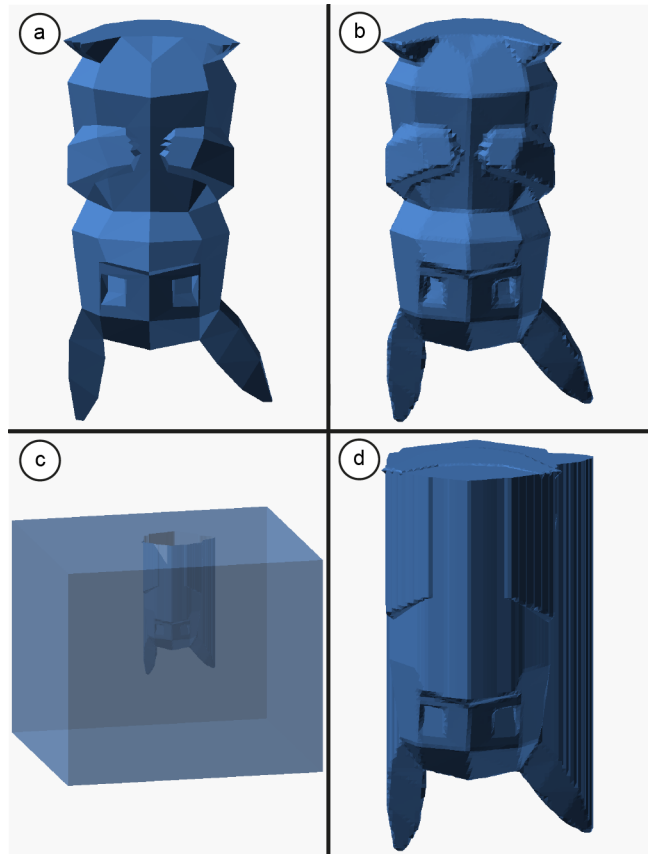


**Figure 5: The optimization process:** (a) This random initial alignment is not feasible for including the Pikachu scrap mesh in the Baby Yoda model, since the scrap mesh protrudes the model mesh. (b) At this alignment, it can only be scaled up to 45% of its initial size before colliding with the model mesh. (c) The optimization concludes after finding an alignment for which the scrap fits the model at its original size and can be inserted without collision. (d) Additional volume has to be removed from the model mesh to achieve an unobstructed insertion path through this insertion plane.

The slicer generates a G-Code file that serves as instructions to the 3D printer. The “cut plane” determined during the modified-Matryoshka optimization above corresponds to a specific height along the printing direction. We identify G-Code corresponding to the layer closest to this height and insert commands to stop printing, move the printer bed to an easily accessible position, move the print head out of the way, and play a beep melody to alert the user that it is time to insert the object. The LCD screen on the printer is triggered to display customized instructions and prompts the user to confirm when the scrap is inserted so that printing may resume.

#### 5.4 Scrap library

The usefulness of our system hinges on whether a user can find well-fitting scrap for their planned 3D prints. The chances of this to happen can be increased by having a large and varied collection of possible scrap objects. Users must, however, also be able to locate specific scrap objects. We provide a number of web-based tools that together form our scrap library to support maintaining and



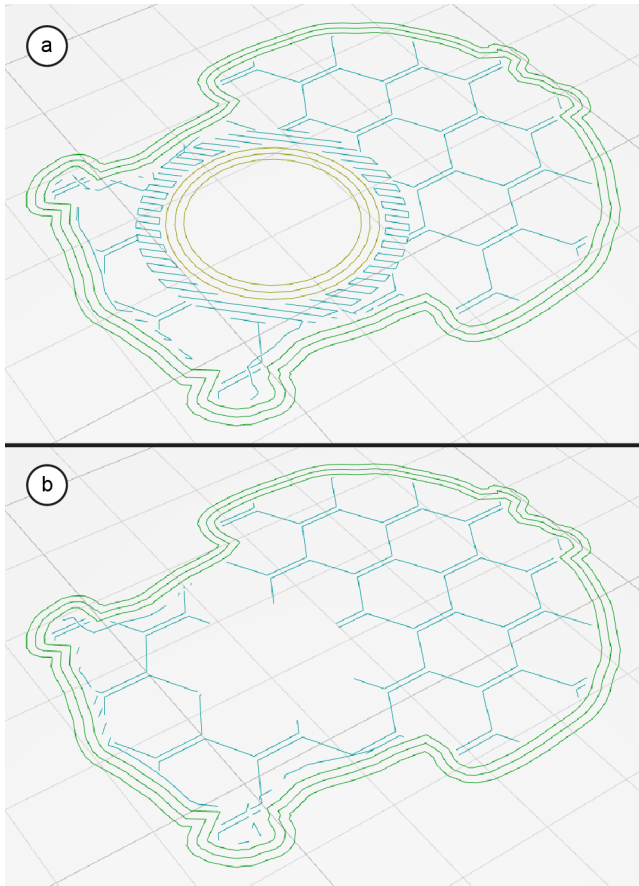
**Figure 6: Hollowed meshes generation:** (a) The aligned scrap model. (b) The scrap model is inflated to account for printer inaccuracies. (c) The scrap mesh is projected towards the insertion plane and the resulting volume is cut from the surrounding material halves. (d) A mesh of the cavity that has to be removed from the model mesh to allow the scrap insertion.

extending a collection and to identify and find specific scrap again later.

Not every user will have access to the same types of scrap objects, and even for a single user their access to certain types of scrap will vary over time. One of the maintenance use cases for the scrap library is therefore to keep it up to date and aligned with the user’s current scrap availability. To do so quickly, we include an overview page that shows images of the scrap objects and grants direct access to the availability status of any scrap object. This list can be filtered, for example by the method with which the scrap was entered into the data base. It can also be ordered, for example by scrap availability or volume. This way, users do not have to click through detail pages of every single scrap object, but can make adjustments in bulk.

We provide scrap generators that can be used to generate scrap meshes, instead of uploading a mesh directly. While entering parameters, the generator website already displays the scrap ID that will be assigned to this new library item. This can be useful to mark objects that might be hard to identify again later, such as wooden off-cuts. Users may also enter their own scrap name, a description





**Figure 7: Comparison of the slicing result of Kiri:Moto and our customized slicer: (a) The toolpaths generated by the original slicer contain internal walls and supports. The color of the internal walls signify that these lines will be printed slowly. (b) The toolpaths generated by our slicer do not contain internal walls or supports surrounding scrap.**

of where they store this kind of scrap, as well as a picture of the storage location, to help identify and locate the scrap again later. All of this information can be found on a scrap details page. If this scrap was recently exported to be included in a 3D print, this page also contains an animation of how to insert the scrap. This animation shows a partially transparent version of the surrounding model up to the insertion layer. It also shows an animation of the scrap object being inserted into the model along its insertion path.

## 6 EVALUATION

To test our system, we created a set of seven example objects. Each of them was printed with and without scrap inclusions to compare printing time and material usage. All but one of the models were downloaded from Thingiverse [18]. The phone stand was modelled by one of the authors. Some of the models were altered slightly before printing, such as scaling the model with the scrap preview in mind.

The scrap objects that were added to the scrap library are based on unused items that some of the authors collected randomly over

time, and common household waste items that were stored instead of being thrown out during the time the paper was written. The household waste items were added to the library using our generators. The collectibles were added by downloading appropriate 3D models from online databases like Thingiverse or by modelling them by hand. Some items in the scrap library are models previously printed by the authors. These items were added by uploading the mesh to fabricate them originally.

All models were printed in PLA at 185 °C using the default slicer settings for infill and perimeters. We chose not to alter these settings as novice users are likely to rely on predefined print settings. The benefits of our system would be more or less pronounced based on higher or lower numbers respectively for infill percentage and number of perimeters. Even at 0% infill, our system can still provide some benefit, since it can replace parts of inner perimeters. The default infill setting of the Kiri:Moto slicer is a 0.5 percentage hex pattern. This equals around 20% infill in the PrusaSlicer [23] software and between 15 to 20% in the Cura [5] slicer, both of which are among their default settings as well. It is not possible to provide a more detailed comparison to the Cura slicer, as it does not offer a hexagonal infill pattern. The default number of perimeters in Kiri:Moto is three, which is again among the default settings in PrusaSlicer and Cura.















Printing times in Table 1 include the time required to insert the scrap while printing. Printing these models with scrap objects inserted during fabrication saved on average 29.4% (SD = 13.9%) material. Printing times decreased on average by 26.4% (SD = 15.6%).

We evaluated the time that our algorithm requires to validate a scrap mesh. This evaluation was performed on a library of 40 scrap meshes. It was performed while running six parallel threads validating scrap meshes. The mean validation and mesh generation times for each of the seven example models are shown in Table 2. Validating the whole library of 40 scrap meshes took 32 s on average (SD = 4.1 s) among all examples while running on six parallel threads. Generating the hollowed meshes for our seven example objects took 58.0 s on average (SD = 48.7 s). Note that this process only has to be performed once when the user is ready to print. It replaces a traditional export step at the end of the design phase and is quite short compared to savings in printing times in the order of hours.

## 7 DISCUSSION

Though our system provides speed and cost benefits to users, it might also cause some detrimental effects in certain situations. For example, the mechanical properties of the resulting 3D prints can vary widely based on the type of scrap material that was integrated. Inclusions made from wood or metal are likely to increase compression and bending resistance of the model, but the tensile strength of the part may still be reduced due to the inclusion of scrap. This is mostly due to the way the scrap is held in position on the inside of the model. For scrap made from similar thermoplastics as the printing filament, chemical bonding can occur. This bond is limited to one side of the scrap and most other materials will not bond chemically to the printing filament. Since the scrap object is surrounded narrowly on all sides, it still fits tightly inside its cavity. Also, the common FDM quality issue of stringing actually benefits

**Table 1: Printing time and material weight of our example objects. Blue bars denote percentages of time or material saved when using our system.**

Model	Scrap	Print time without scrap	Scrappy print time savings	Material weight without scrap	Scrappy material weight savings
Tardis	Salt dispenser	1730 min		285 g	
Gear	Jar lid	150 min		25 g	
Fossil token	Rock	55 min		10 g	
Large XYZ-cube	XYZ-cube	33 min		7 g	
Baby Yoda	Pikachu	472 min		52 g	
Phone stand	I-Pod	272 min		58 g	
Scraping tool	Broken handle	160 min		33 g	

**Table 2: Average run times and standard deviations for validating insertion options for 40 scrap meshes and the time it took to generate the hollowed mesh for printing the chosen insertion option.**

Model	mean validation time	standard deviation	mesh generation time
Tardis	8.6 s	10.9 s	19 s
Gear	3.9 s	10.7 s	14 s
Fossil token	1.7 s	3.4 s	126 s
Large XYZ-cube	1.0 s	1.7 s	17 s
Baby Yoda	4.1 s	6.1 s	84 s
Phone stand	12.1 s	23.1 s	32 s
Scraping tool	2.2 s	4.8 s	114 s

our system, as the strings inside the cavity provide a tense cushion once scrap has been inserted, tightening the fit further. Adding a drop of glue during scrap insertion could further alleviate potential issues with stability of fit.

Our system prevents collisions of the print head with the inserted scrap object by restricting the insertion plane to the top of the scrap object. Knowledge about the geometry of the printer and the print head in particular could be used to generate the reachable volume after scrap insertion. Though having the scrap object reach high above the last printed layer would make many geometries impossible to complete, having more flexibility in choosing the insertion plane would enable the fabrication of others.

Since a scrap might spend a long time inside the model object, not all waste items may be suitable for insertion. Perishable items could decompose inside the model, putting the part under internal pressure, or slowly melting it from the inside. Adding breakable scrap, such as glass jars, could lead to a more dangerous failure mode of parts that come under mechanical stress.

Including multiple scrap objects in a single model would enable replacing even more infill, but would require additional user interactions or additional computation. Short of using a commercial cloud-based solution for the computation, this computational effort of trying to fit two or even more scrap objects into a model at the same time would likely increase the run time of the search too much to integrate this feature into an interactive modelling session.

The usefulness of some objects to be used as scrap may be dramatically improved by breaking them apart. For example, most of the volume of the Pikachu model is located in its body. It is much

easier to find suitable scrap placements for the two separate parts of a Pikachu whose tail had already snapped off. We would like our system to be able to recognize such circumstances and instruct the user to cut or break off specific parts of otherwise unwieldy scrap models on demand.

## 8 CONCLUSION

Based on a formative interview study with fabrication experts, we verified that material is often wasted on failed prints and early prototypes, print time is a limited resource, and makers care about sustainability. With this motivation, we built a system for FDM printing that makes use of these and other waste materials by using them to replace infill material. We extended an existing algorithm to determine possible scrap inclusions, and integrate this information into an existing CAD modelling workflow using our Fusion 360 add-in. Our method includes custom slicer adjustments to avoid printing unnecessary internal walls. We created an extensible web-based tool for makers to maintain a scrap library which provides instructions for locating scrap and integrating it during fabrication.

In future work, we plan to compare different methods of recruiting people nearby to insert scrap, such as printing a QR code that links to a website with instructions, or printing a virtual post-it note. We will also compare different fidelity levels of instructions, such as a written message, an instructive image, or an instruction animation, and we will contrast their performance when used by people nearby to the maker who started the print. Furthermore, we plan to extend our system to replace external support material using ad-hoc supporting structures. It would also be interesting to extend our system to evaluate the added benefit and feasibility of insertions that perform additional functions beyond saving infill material, such as inserting electronics or materials with mechanical benefits.

3D printing has the potential to decentralize the production of many tools and consumables, which are traditionally packaged and shipped around the globe. This decentralization has an important sustainability outcome, the reduction of packing materials and lowering emissions from transportation. However, 3D printing itself can generate significant waste. Our system contributes to make the 3D printing process itself more sustainable.

## ACKNOWLEDGMENTS

This work made possible by NSERC Discovery Grant 2018-05187, the Canada Foundation for Innovation Infrastructure Fund 33151

“Facility for Fully Interactive Physio-digital Spaces,” and Ontario Early Researcher Award ER16-12-184.

## REFERENCES

- [1] 3devo B.V. 2020. 3devo: Turning 3D printing into a closed-loop cycle. <https://3devo.com/our-process/>. [Online; accessed 17-September-2020].
- [2] Yoh Akiyama and Homei Miyashita. 2016. Fitter: A System for Easily Printing Objects That Fit Real Objects. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) (UIST '16 Adjunct). Association for Computing Machinery, New York, NY, USA, 129–131. <https://doi.org/10.1145/2984751.2985730>
- [3] Stewart Allen. 2020. Kiri:Moto: An open-source slicer for Laser cutting, 3D printing, CNC milling. <https://grid.space/kiri/>. [Online; accessed 17-September-2020].
- [4] Inc. Autodesk. 2020. Fusion360. <https://www.autodesk.ca/en/products/fusion-360/overview>. [Online; accessed 17-September-2020].
- [5] Ultimaker BV. 2020. Cura: an open source slicing application for 3D printers. <https://ultimaker.com/software/ultimaker-cura/>. [Online; accessed 17-September-2020].
- [6] Xiang 'Anthony' Chen, Stelian Coros, and Scott E. Hudson. 2018. Medley: A Library of Embeddables to Explore Rich Material Properties for 3D Printed Objects. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) (CHI '18). Association for Computing Machinery, New York, NY, USA, 1–12. <https://doi.org/10.1145/3173574.3173736>
- [7] Xiang 'Anthony' Chen, Stelian Coros, Jennifer Mankoff, and Scott E. Hudson. 2015. Encore: 3D Printed Augmentation of Everyday Objects with Printed-Over, Affixed and Interlocked Attachments. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (Charlotte, NC, USA) (UIST '15). Association for Computing Machinery, New York, NY, USA, 73–82. <https://doi.org/10.1145/2807442.2807498>
- [8] A. Dijkshoorn, P. Werkman, M. Welleweerd, G. Wolterink, B. Eijking, J. Delamare, R. Sanders, and G. J. M. Krijnen. 2018. Embedded sensing: integrating sensors in 3-D printed structures. *Journal of Sensors and Sensor Systems* 7, 1 (2018), 169–181. <https://doi.org/10.5194/jsss-7-169-2018>
- [9] Alfonso Fernández. 2019. *Development of Actuators Using Material Extrusion Additive Manufacturing with Embedded Shape Memory Alloy Wire*. Master's thesis. University of Texas at El Paso.
- [10] The Blender Foundation. 2020. Blender V2.8. <https://www.blender.org/>. [Online; accessed 17-September-2020].
- [11] James Gopsill, Jonathan Shindler, and Ben Hicks. 2017. Using finite element analysis to influence the infill design of fused deposition modelled parts. *Progress in Additive Manufacturing* 3 (11 2017). <https://doi.org/10.1007/s40964-017-0034-y>
- [12] Daniel Groeger, Elena Chong Loo, and Jürgen Steimle. 2016. HotFlex: Post-Print Customization of 3D Prints Using Embedded State Change. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) (CHI '16). Association for Computing Machinery, New York, NY, USA, 420–432. <https://doi.org/10.1145/2858036.2858191>
- [13] Alec Jacobson. 2017. Generalized Matryoshka: Computational Design of Nesting Objects. *Computer Graphics Forum* 36, 5 (2017), 27–35. <https://doi.org/10.1111/cgf.13242> arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13242>
- [14] Andrew Jones, Thomas Cameron, Benjamin Eichholz, Martin Eichers, Taylor Kray, and Jeremy Straub. 2019. Reducing space sensing and other mission cost with 3D printing infill optimization. In *Sensors and Systems for Space Applications XII*, Genshe Chen and Khanh D. Pham (Eds.), Vol. 11017. International Society for Optics and Photonics, SPIE, Bellingham, Washington USA, 69 – 74. <https://doi.org/10.1117/12.2520261>
- [15] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol. 4. IEEE, Perth, WA, Australia, Australia, 1942–1948 vol.4. <https://doi.org/10.1109/ICNN.1995.488968>
- [16] Marius Kintel and Clifford Wolf. 2020. OpenSCAD: The Programmers Solid 3D CAD Modeller. <https://www.openscad.org/>. [Online; accessed 17-September-2020].
- [17] Robert Kovacs, Anna Seufert, Ludwig Wall, Hsiang-Ting Chen, Florian Meinel, Willi Müller, Sijing You, Maximilian Brehm, Jonathan Striebel, Yannis Komman, Alexander Popiak, Thomas Bläsius, and Patrick Baudisch. 2017. TrussFab: Fabricating Sturdy Large-Scale Structures on Desktop 3D Printers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) (CHI '17). Association for Computing Machinery, New York, NY, USA, 2606–2616. <https://doi.org/10.1145/3025453.3026016>
- [18] LLC MakerBot Industries. 2020. MakerBot Thingiverse. <https://www.thingiverse.com/>. [Online; accessed 17-September-2020].
- [19] Stefanie Mueller, Sangha Im, Serafima Gurevich, Alexander Teibrich, Lisa Pfisterer, François Guimbretière, and Patrick Baudisch. 2014. WirePrint: 3D Printed Previews for Fast Prototyping. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) (UIST '14). Association for Computing Machinery, New York, NY, USA, 273–280. <https://doi.org/10.1145/2642918.2647359>
- [20] Stefanie Mueller, Tobias Mohr, Kerstin Guenther, Johannes Frohnhofen, and Patrick Baudisch. 2014. FaBrickation: Fast 3D Printing of Functional Objects by Integrating Construction Kit Building Blocks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) (CHI '14). Association for Computing Machinery, New York, NY, USA, 3827–3834. <https://doi.org/10.1145/2556288.2557005>
- [21] Jukka Pakkanen, Diego Manfredi, Paolo Minetola, and Luca Iuliano. 2017. About the Use of Recycled or Biodegradable Filaments for Sustainability of 3D Printing. In *Sustainable Design and Manufacturing 2017*, Giampaolo Campana, Robert J. Howlett, Rossi Setchi, and Barbara Cimatti (Eds.). Springer International Publishing, Cham, 776–785.
- [22] Huaishu Peng, Rundong Wu, Steve Marschner, and François Guimbretière. 2016. On-The-Fly Print: Incremental Printing While Modelling. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) (CHI '16). Association for Computing Machinery, New York, NY, USA, 887–896. <https://doi.org/10.1145/2858036.2858106>
- [23] Prusa Research. 2020. PrusaSlicer: G-code generator for 3D printers. <https://www.thingiverse.com/>. [Online; accessed 17-September-2020].
- [24] P. Frazer Seymour, Justin Matejka, Geoff Foulds, Ihor Petelycky, and Fraser Anderson. 2017. AMI: An Adaptable Music Interface to Support the Varying Needs of People with Dementia. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility* (Baltimore, Maryland, USA) (ASSETS '17). Association for Computing Machinery, New York, NY, USA, 150–154. <https://doi.org/10.1145/3132525.3132557>
- [25] C. Shemelya, F. Cedillos, E. Aguilera, D. Espalin, D. Muse, R. Wicker, and E. MacDonald. 2015. Encapsulated Copper Wire and Copper Mesh Capacitive Sensing for 3-D Printing Applications. *IEEE Sensors Journal* 15, 2 (2015), 1280–1286.
- [26] J. Wu, N. Aage, R. Westermann, and O. Sigmund. 2018. Infill Optimization for Additive Manufacturing—Approaching Bone-Like Porous Structures. *IEEE Transactions on Visualization and Computer Graphics* 24, 2 (2018), 1127–1140.
- [27] Suguru Yamada, Hironao Morishige, Hiroki Nozaki, Masaki Ogawa, Takuro Yonezawa, and Hideyuki Tokuda. 2016. ReFabricator: Integrating Everyday Objects for Digital Fabrication. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (San Jose, California, USA) (CHI EA '16). Association for Computing Machinery, New York, NY, USA, 3804–3807. <https://doi.org/10.1145/2851581.2890237>
- [28] Amanda K. Yung, Zhiyuan Li, and Daniel Ashbrook. 2018. Printy3D: In-Situ Tangible Three-Dimensional Design for Augmented Fabrication. In *Proceedings of the 17th ACM Conference on Interaction Design and Children* (Trondheim, Norway) (IDC '18). Association for Computing Machinery, New York, NY, USA, 181–194. <https://doi.org/10.1145/3202185.3202751>
- [29] Kening Zhu, Alexandru Dancu, and Shengdong Zhao. 2017. FusePrint: A DIY 2.5D Printing Technique for Good-Fit Fabrication with Daily Objects. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (Denver, Colorado, USA) (CHI EA '17). Association for Computing Machinery, New York, NY, USA, 413–416. <https://doi.org/10.1145/3027063.3050430>

## A EXAMPLE SCRAP OBJECTS

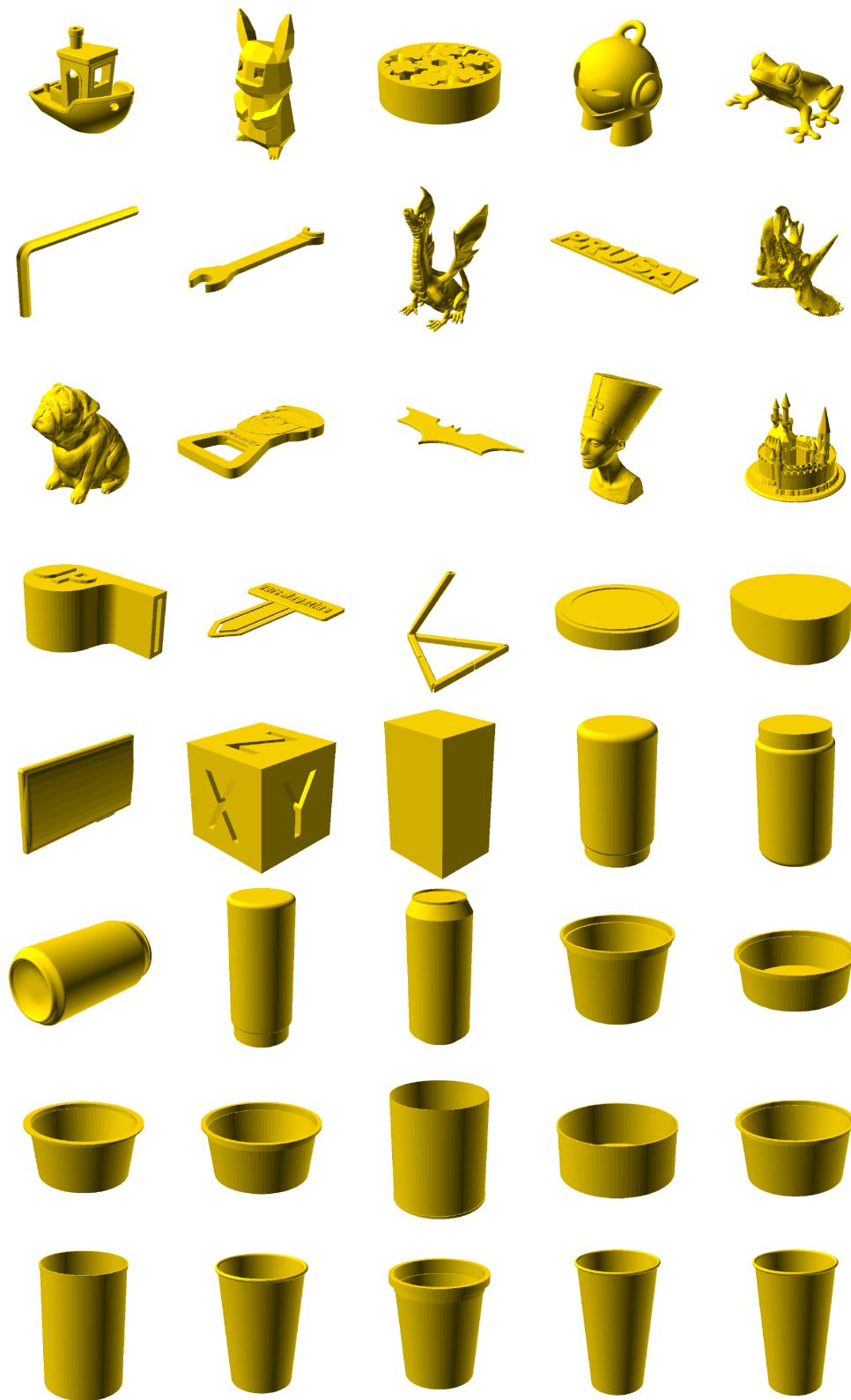


Figure 8: The 40 objects that form the initial scrap library used during evaluation.