Substiports: User-Inserted Ad Hoc Objects as Reusable Structural Support for Unmodified FDM 3D Printers

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

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

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

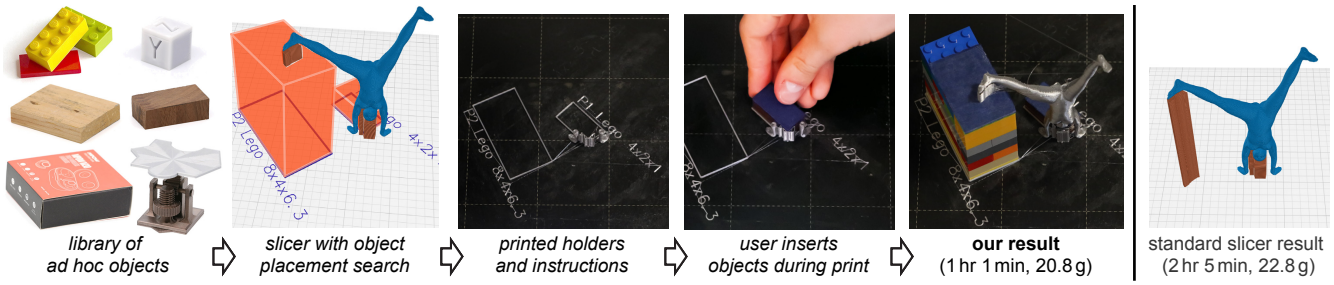


Figure 1: Our approach uses a modified slicer to select reusable objects to use as ad hoc structural support. Object "holders" and instructions are printed, then the user inserts objects during one or more pauses during the print. In this example, our approach reduced print time by 1 hour and saved 87% of support (2.0g out of 2.3g): even a small savings in support material can result in pronounced time savings due to reduced printer movements.

ABSTRACT

We contribute a technical solution to reduce print time and material with unmodified fused deposition modelling printers. The approach uses ad hoc objects inserted by a user during printing as a replacement for printed support of overhanging structures. Examples of objects include household items like books, toy bricks, and custom mechanisms like a screw jack. A software-only system is integrated into existing slicing software to analyze generated support print paths, search a library of objects to find suitable replacements, optimize combinations of replacement objects, and make necessary adjustments to impacted printing layers and paths. During printing, the user is prompted to insert objects with the help of lightweight printed holders to guide placement and prevent movement. Instructions printed on the build-plate help identify and position objects. A technical evaluation measures performance and benefits with different sets of ad hoc objects and different levels of user involvement.

CCS CONCEPTS

• **Human-centered computing** → **Interaction tech**.

UIST '23, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0132-0/23/10…$15.00
https://doi.org/10.1145/3586183.3606718

KEYWORDS

fabrication, interactive 3D printing, sustainability

ACM Reference Format:
Ludwig Wilhelm Wall, Oliver Schneider, and Daniel Vogel. 2023. Substiports: User-Inserted Ad Hoc Objects as Reusable Structural Support for Unmodified FDM 3D Printers. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3586183.3606718

1 INTRODUCTION

3D printers that employ fused deposition modelling (FDM) use printable support structures to reliably manufacture models with overhangs, bridges, and other geometry extending more than 45° from the Z-axis (we collectively refer to all of these as "overhangs"). Disposable support structures are pure waste, costing additional material and time, so it is desirable to minimize them.

One strategy is to optimize the support structure itself, for example, using lightweight structures that branch out from lower parts of the model to support overhangs [10]. However, this introduces additional connections between support material and the model, marring the surface finish. In general, optimizing support geometry increases the chance of failure: thin tall structures are more severely affected by thermal deformation and their low stiffness can result in accidental displacement. For these and similar reasons, support structures typically generated by slicers are sturdy with a substantial connection to the build plate and filling much of the volume beneath overhangs [1] .

Another approach is to replace support structures with something else. For example, Hongyao et al. [13], Xu et al. [39] replaced a printer build plate with a mechanically actuated grid of supporting surfaces and Yiğit et al. [40] propose using a robot arm to place a set

of uniform blocks. These both replace support material with other kinds of sturdy structures, but they would add cost and complexity due to additional hardware and major printer alterations. Our approach achieves the same goal by enabling users to manually insert ad hoc objects during the printing process, this requires no special hardware or printer modifications, and there is no need to adhere to a grid or certain object type for temporary support structures.

Our system is implemented primarily as modifications to an open-source slicer. After default print paths are generated for support, our system performs a geometric analysis and optimization to select and position objects from a user-maintained library (Figure 1a). Users can tune solutions to consider the number and complexity of object insertions, and a scheduler can further constrain insertions to convenient times. Once a solution is selected, customized G-code incorporates the replacement objects as support during the print, and adds minimal 3D printed guides to position and stabilize each object with object identification codes printed on the build plate (Figure 1b). By leveraging software-based optimization and human dexterity instead of a complex mechanical system, our system is compatible with unmodified FDM 3D printers and it is able to use a broad definition of ad hoc objects as support structures. An object is usable as long as it reliably provides an elevated horizontal surface to print on. Different sizes and shapes of objects may be placed at any position in the print volume, increasing compatibility with model geometry that conform to a fixed-size actuated grid or axis-aligned blocks.

A technical evaluation with a 15-model dataset achieved an average savings of 52.0% of support material and 50.4% of support print time (34.8% of overall printing time). The best performing model replaced 97.0% of support material and reduced total print time by 77.8%. Using Xu et al.'s "gymnast" model [39], our system saves as much as 93.5% compared to 78.1% for their actuated print bed approach. We make two contributions: (1) a software-only system for reducing support for a popular form of 3D printing; and (2) a technical evaluation showing the utility for different levels of user interaction. Code and data is available[1].

2 RELATED WORK

Trying to limit the amount of support required to achieve a successful print is a key topic in FDM 3D printing. Methods that mainly adjust the geometry of the support to reduce material usage, such as merging supports into smaller columns, forming truss, scaffolding, or tree structures [7, 10, 14, 16, 17, 22, 27, 31, 35], or filling the support volume with material efficient micro structures [32] can be used in principle with our method.

2.1 Printing Adjustments to Reduce Support

In some cases printing support material underneath overhangs can be avoided entirely. Print on Air [4, 9] creates horizontal overhangs by carefully attaching material sideways onto existing geometry by making use of reduced printing speeds and strong cooling. This method was shown to be successful for overhangs of 90° relative to the printing direction. A similar effect of depositing material sideways onto existing geometry can be achieved by adjusting the slicing process to divide the printed model into non-planar slices [25]. When following these non-planar tool paths, the print head moves slightly up and down relative to the build plate. This up and down motion aids in the sideways deposition of material and can enable printing overhangs slightly above 90°, limited by print head geometry. Our system can be used to support arbitrary overhangs like the printed supports that it replaces. It furthermore does not necessitate slowing down printing speeds.

Reorienting models can be an effective tool to minimize required support material [2, 6, 8, 12], for example by bringing parts of the model that require support closer to the build plate, or by allowing parts of the model to be better supported by the model itself. Some support material may still be required after adjusting the model orientation. Users may also prefer using specific orientations to optimize part strength or the visibility of layer lines instead. Our method applies in these cases.

2.2 Model Adjustments to Reduce Support

By accepting design limitations during modelling, such as keeping overhangs below 45° and bridging less than 10mm, support can be avoided or reduced. Reiner and Lefebvre [28] developed an interactive system to apply these types of limitations to generate support-free models. However, in practice there is a large class of models that cannot adhere to these limitations. For these kinds of models, PackMerger [36] is an approach to divide a model into segments arranged on the print bed to minimize support. Since the segments must be assembled to form the original model, this method is not applicable when there are mechanical requirements and additional user effort is needed. Our method also requires additional user effort but it does not alter the input model. When printing multiple models at once, Jiang et. al [18] propose placing smaller objects underneath overhangs of larger models. This way, a smaller model acts as an elevated platform to print on, replacing some of the support material required by a larger model. Our approach is more general since it works with a single model and uses a large class of re-usable objects instead of assuming other models will be printed.

2.3 Mechanical Methods to Replace Support

Shen et al. [13] subdivide the build plate into 6 x 6 square segments, each 50 by 50 mm, which can be raised using motors to create elevated print surfaces. They report saving 35.5% of material for two models, but the proportion of support material is not specified. Xu et al. [39] simplify the complex hardware requirements for this approach by allowing all segments to be raised by a single motor after a user manually inserts lifting rods of varying heights. This also enabled a larger 11 x 9 grid with smaller 12.7 by 12.7 mm segments. Average support material savings of 64.7% are reported for five test models of simple to moderate complexity, including the "gymnast" model we include in our test set. Our approach pursues the same goal by involving users to manually provide elevated printing surfaces. Unlike our approach, Xu et al. require a single up-front user interaction, but one that appears time intensive and complex (though user effort or time is not discussed). In addition to requiring highly specialized printer hardware, a grid-based subdivision is discrete, limiting how support can be replaced, and gaps between

[1] https://github.com/exii-uw/substiports

segments can cause surface artifacts and potential problems for first layer adhesion.

Yigit et al. [40] propose using reusable support replacement blocks placed by a robot arm. The system uses a discretized 1 cm grid and a sliding window algorithm to place axis-aligned 4x4 cm blocks. An average of 75.5% support material savings are reported for two test models, including the Stanford bunny that is also in our test set. However, this was only a software simulation where virtual blocks were placed and stacked using a virtual robot arm during a simulated print. Practical issues like adhesion, stability, and mechanical precision are not considered. Our approach pursues the same goal, but with a fully implemented system that handles practical details.

In comparison to these works, our approach requires no additional hardware by leveraging simple human abilities. A fully automated approach introduces points of failure. For example, small debris could alter a placed block position by a few degrees, enough to cause collisions or topple towers of blocks later. Detecting and removing such debris is trivial for a person when placing a block. In addition, we use a more advanced and general optimization method to find optimal ways to place and combine heterogeneous replacement objects at arbitrary positions and orientations.

2.4 Printing onto Objects

Integrating existing objects into the design and fabrication of 3D models has been explored previously in various ways, including printing on top of other objects. Encore [5] is a method to print on top of, or through holes of existing objects to enhance them. The user places the object onto a partially printed structure midway through the print. RoMA [26] is a prototyping tool to design and print directly on existing objects. It requires extensive hardware, such as a 3D scanner and robot arm. Patching Physical Objects [33] also enables printing onto existing objects with the purpose of making physical adjustments to the model or performing repairs. Under the right circumstances this can avoid printing support material entirely by avoiding printing a new model, but also requires additional hardware in the form of a depth camera, a mill, and a 5-axis rotating platform. RevoMaker [11] prints onto a rotating laser cut base cube to avoid some types of support at the cost of design limitations and additional hardware requirements. Rivera et al. [30] suggest securing flexible objects like fabric pieces in place with double sided tape, which also works for our replacement objects.

Scrappy [37] reduces infill material by creating a hollow pocket into which a user inserts a scrap object during a print. Only a single object insertion is supported, objects are primarily simple cuboids and cylinders, and the method requires extensive 3D geometry processing on a server to select and orient the object. Our system supports multiple objects of varying complexity, and it is able to run in a web browser. We expand their object library implementation to additionally differentiate between distinct types of objects, such as replacement objects that have a variable height. Note that infill and structure support are independent parts of a printed model, so our system is compatible.

3 WALKTHROUGH

This section introduces the three main parts of the system from the user's perspective and highlights a scheduling feature. Figure 1 illustrates the main user steps and Figure 2 provides additional slicing and insertion examples.

Object Library. A library defines what objects are available to use as support replacements. A base library of standard objects includes items that people may have already, like office supplies and standard packaging, and items often generated when tuning FDM 3D printers, like calibration cubes and certain test models. The user can delete, modify, or add objects to match what they have at hand. This is done primarily using web-based generator forms, which use caliper measurements or uploaded 2D line drawings to define the key dimensions. This largely follows the process used in Scrappy [37], they provide implementation details for generators and an explanation of the library interface and management. Unlike the Scrappy library, generating a 3D model mesh file is optional for most support replacement objects. For example, cuboids are the most basic type, and only their length, width, and height is needed by our system to use them to replace support. Similarly, for irregular prism shapes, it is sufficient to know the shape of the base, shape of the top, and the total height. Our version of the library also supports variable objects, like a miniature screw-jack with adjustable height and stack-able objects like toy building blocks.

Slicing. To prepare a print with support replacement objects, the user imports a 3D model into our customized slicer. They select usual slicing settings, with the addition of a new setting for their desired level of involvement during printing. During slicing, the system searches the library for one or more objects to use as support

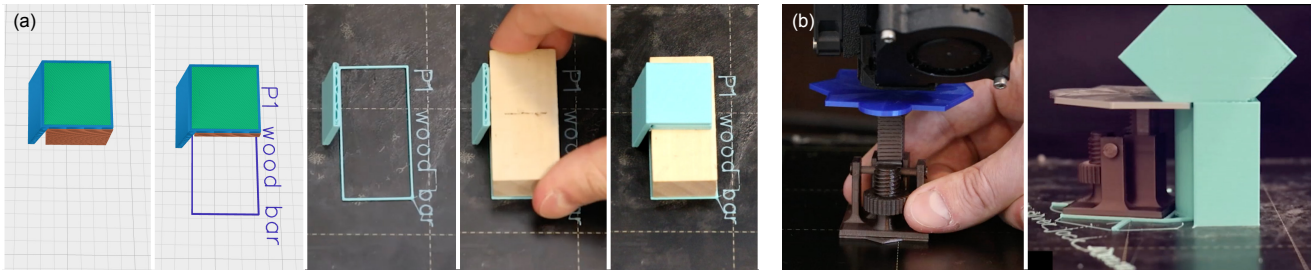


Figure 2: Slicing and insertion examples: (a) Angle Bracket model sliced to use a wood off-cut as a replacement object, the wood is inserted into a "holder" marked with instructions during the print; (b) the screw-jack replacement object is adjusted to match the nozzle height of the parked print head, a screw-jack used as support for the Angled Cylinder model.

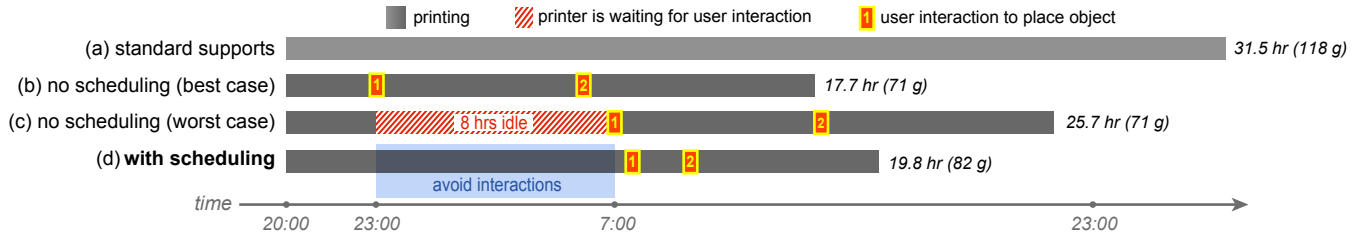


Figure 3: Example print times for the Arch Bridge model with print parameters held constant, with and without scheduling: (a) without our system all supports are printed; (b) an optimal solution using our system without scheduling constraints, and the user places objects during the night; (c) the same optimal solution without scheduling, but the printer idles until the user places the first object at 7am; (d) an optimal solution using the scheduler to avoid object insertions during the night.

replacements. The slicing result is directly modified to accommodate the replacement objects and rendered showing object positions for the user to inspect.

Inserting Objects. When printing, exported G-code includes commands to generate instructions and "object holders". When the print reaches the predetermined height where an object must be inserted, the printer pauses, parks the print-head away from the model, and beeps to prompt the user with instructions. Each object is identified by a name and ID printed on the build plate. The user locates the object, then inserts it into the custom printed holder which ensures it is aligned and secured in place. Depending on object type, the insertion process might differ. For example, a stack of toy bricks may need to be assembled to match a height printed on the build plate (i.e. number of bricks). Similarly, an adjustable model, like a screw jack, may need to be adjusted until it lightly touches the print nozzle (Figure 2b), which our custom G-code positions to the required support height. After inserting one or more objects, the user resumes the print, and this process of automatic pausing, object insertion, and resuming repeats as needed.

3.1 Scheduling User Interactions

The system includes a scheduler to search for solutions that avoid user interactions during certain periods of the day, for example during the night. Figure 3 provides illustrative examples for the Arch Bridge model shown in Appendix A. Example c is a worst case without scheduling where the printer idles for a full eight hours before printing can continue. Example d shows a solution using the scheduler to avoid interactions between 11 pm and 7 am. This results in a much shorter print time than the worst case without scheduling, but does increase time and material from example b, which is the optimal solution without any schedule constraints where the user is available for interactions regardless of time. Note that all three examples with our system, even the worst case, are still faster than example a, which is a default print with standard supports.

Scheduling restrictions are built into the system by marking solutions with interactions during unavailable hours as invalid, and by severely reducing their fitness value. To operationalize the scheduling constraint during search, the user can enter a print start time, with the current time used as default.

4 AD HOC REPLACEMENT OBJECTS

Our system supports replacement objects with two parallel flat surfaces, these can be gathered from a variety of sources. Found objects include books, household waste like cardboard boxes and other packaging material, or workshop scraps such as wooden off-cuts. 3D printing scrap like calibration cubes or support towers from previous prints can be reused as support replacements as well. Users may opt to create custom reusable objects that aid in replacing support material further. For example, a custom object perfectly suited to replace the support of a recurring print job that acts as a jig, or an adjustable screw jack that can vary in height.

We group replacement objects into three types based on their height variability. Objects with a fixed height, such as books or wooden off-cuts. Standardized objects that can be easily stacked to form variable discrete heights, such as toy building bricks, stacks of DVD cases, or poker chips. And objects where the height is continuously variable, such as a 3D printed screw jack that adjusts in height by turning a knob.

4.1 Technical Tests of Object Properties

We conducted technical tests of various object and filament properties. The general results are discussed below, please see Appendix B for test descriptions and more specific results.

Test B.1 Object surface adhesion. During empirical tests of 17 materials (Table B.1), only low density foam blocks proved problematic. Materials should have a hardness similar to medium density foam or harder. Materials that are softer might have insufficient physical resistance to reliably deposit printing material since FDM printing uses some downward pressure for layer adhesion. Materials should also remain stable at the temperature of the heated build plate and the filament when extruded. We found even materials with a melting point below 200°C can be used, albeit with some marring and deformation to the object surface. Such materials can be reused multiple times, but not indefinitely. Adding a layer of masking tape to the top surface of an object will prevent most surface marring. In practice, the adhesive properties of the replacement object material in relation to the printing material is unimportant as long as the top object surface is covered with masking tape, a thin layer of paper glue, or a similar coating.

Test B.2 Adhesion with different filaments. Different types of support printing material like PLA, PETG, ABS, and TPU can be used.

Depending on the printing material, tape or glue may have to be applied to a different set of replacement object materials.

Test B.3 Object measurement precision. Objects should be measured with an accuracy of at least ±0.25 mm to ensure a secure fit inside printed holders and to ensure ideal print adhesion. Measurements can be error-prone [21], but cuboids should be simple to measure and using standardized objects like toy bricks requires no measuring. The top surface of objects should be mostly planar. However, adhesion remains reliable for height irregularities or measurement errors of up to 0.3 mm. Additionally, irregular objects beyond that range, such as a slightly arched softcover book, may still be used if the print head can compress them to be within tolerance.

Test B.4 Stability with build plate movement. The size, shape, and density of replacement objects should be such that the object is stable and unlikely to tip. In general, heavy objects with a low centre of gravity are better suited than lightweight objects. The object holders that our system prints onto the build plate provide significant additional lateral stability, which prevents movement and tipping even for light and top-heavy objects. Objects remain secure even if the build plate moves at maximum speed. Holders can be secured to the build plate further by printing a single layer high brim. Brim width is adjustable as a slicing parameter.

Test B.5 Finish quality after long pauses. When printing with PLA, neither pausing nor extended idle periods cause noticeable deformations or surface finish degradation. Changes in geometry may change the printing direction of perimeters, which might be visible at certain reflection angles.

Test B.6 Compatibility with support structure types. The geometry or layout of support structures seems to have no effect on adhesion, making our approach likely compatible with any type.

4.2 Interaction Density and Complexity

There are a variety of ways that one or more replacement objects can be inserted into a print. We consider user effort to insert objects as the *density* and *complexity* of interactions. Our technical implementation attempts to balance this expected user effort with material and time savings.

Interaction Density. There can be one or more object insertions each time the printer pauses. Density refers to how frequently the user is interrupted during a print, and how many insertion steps are required during each interruption. The rows in Figure 4 illustrate this dimension.

If the print requires only a single replacement object, then there is only one interruption with a single insertion, making the interaction density low (Figure 4 bottom row). If multiple objects are used as support replacements, and all objects have the same height, there is still a single interruption but with multiple interactions to insert each object (Figure 4 middle row). This increases the density of a single interaction. Common types of uniform ad hoc objects exist, such as wooden off-cuts sharing the same standard thickness. When there are multiple replacement objects with different

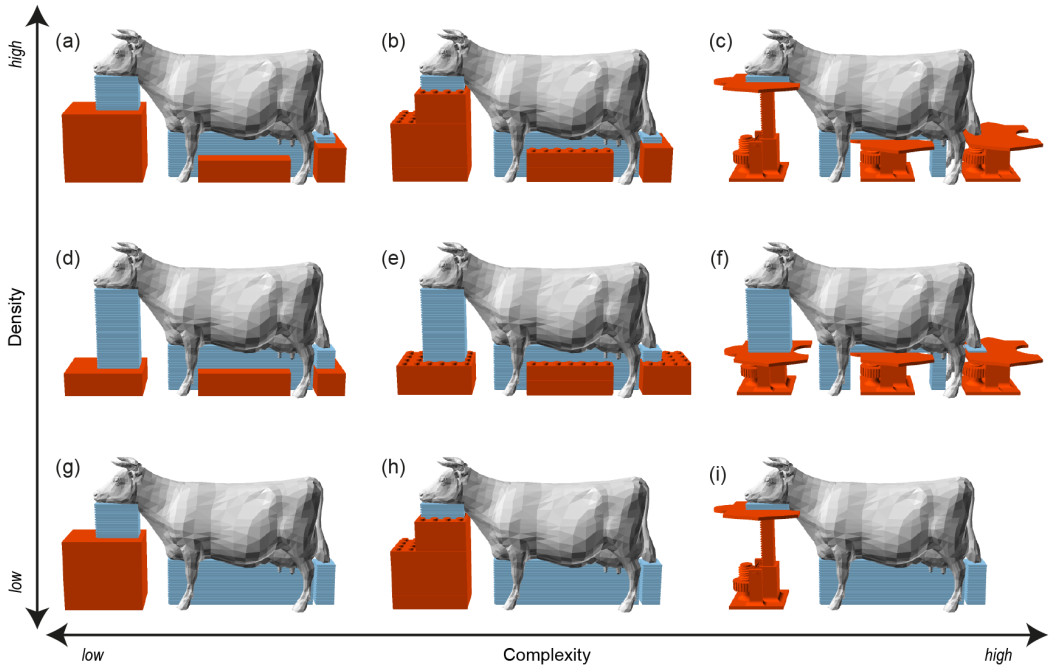


Figure 4: Interaction density and complexity: (g) a single cuboid replacement object requires only a single simple user interaction for insertion; (e) multiple more complex replacement objects with the same height also creates only a single interruption, but it requires more time to locate, assemble and insert multiple stacks of objects; (c) high density and complexity result in multiple user interactions throughout the print process and require additional precise adjustment steps at insertion time.

heights, there will be multiple interruptions creating a high density of interactions during the print (Figure 4 top row). There are other timing characteristics that affect interaction density. If multiple interruptions occur near each other, perhaps separated by a few minutes, the perceived interaction density is likely lower than if the same number of interruptions were spaced out during the print. The actual time of the interruption also likely affects perceptions of interaction density: consider object insertions during working hours versus in the middle of the night.

Interaction Complexity. Complexity is an orthogonal dimension to interaction density that captures how difficult, time consuming, or error-prone a single interaction during a single object insertion step is. The columns in Figure 4 illustrate the range of this dimension with three examples.

Examples of low complexity interactions are inserting fixed height replacement objects (Figure 4 left column). Objects only need to be located and inserted into the printed holder. A medium complexity interaction example is stacking multiple objects to achieve a desired support height (Figure 4 middle column). This requires more effort to gather the stacked objects, stack them, and check the height. It is more error-prone since it is possible to misunderstand stacking instructions, or introduce unexpected dimensional height variations due to how well the objects stack. An example of a high complexity interaction is using an adjustable replacement object like a screw jack (Figure 4 right column). There is increased complexity to fabricate the object itself, though this is a one-time cost. Setting a precise height requires some additional time and could be error-prone, so our system parks the print head to form a gauge for the user to "dial in" an adjustable height. The top of the screw jack shields the user's hand from a potentially hot nozzle.

In practice, complexity and density could achieve similar results in terms of support savings. For example, Figure 4a has three pauses each with a single simple insertion, while Figure 4i has a single pause but requires a more complex interaction to set a screw jack height; both achieve similar results. Likewise, Figure 4b and f use varying mixtures of complexity and density to achieve comparable results. This means the user has multiple options to choose based on their goals, constraints, and preferences.

4.3 Insertion Feasibility Test

The fundamental action of inserting an object into a holder is simple, but unfamiliar in the context of 3D printing. To evaluate this crucial step, we recruited 6 people with 3D printer experience ranging from none to expert. We explained how to assemble toy bricks securely and how to adjust a screw jack to match the print nozzle height. Using a separate build plate, we demonstrated how to align an object with the holder, insert it, then wiggle it to test stability.

After this introduction, the participant inserted three objects of varying complexity during a single printing pause: a simple cuboid, a 4x4 stack of toy bricks that required assembly to match the correct height, and a screw jack that had to be height adjusted. They were asked to test stability after insertion, and comment about their confidence in its placement. We recorded the time from when they began the task until they told us they were done inserting all three objects. Then, the 3D printer was restarted, and 10 layers were printed on all inserted objects to objectively test the placement.

For these first time users, the average time to assemble/adjust, insert, and test stability of all objects was 54 s (median 46, min 34, max 98, sd 22). All participants expressed confidence in their final insertion, and all objects were printed on successfully. A participant with little 3D printing experience described the task as "easy." We believe these times would be much faster with experience. For example, two participants did not initially understand how much pressure was needed to snap objects into holders, and one participant chose to re-insert objects multiple times, "just to be sure."

5 IMPLEMENTATION DETAILS

The system is implemented into the open source Kiri:Moto slicer [3]. In the modified user interface, the desired level of interaction density and complexity are adjustable, there are optional scheduling constraints expressed as time periods to avoid inserting objects (see above), and a search quality setting adjusts the optimization stopping condition and how similar object placements are pruned to speed up the process. When the user initiates slicing, our system searches for solutions given these user parameters and the available object library.

Our method uses the sliced 2D polygons representing the 3D model and support areas to select and position support replacement objects. To optimize computation, a copy of these 2D polygons are simplified using a heuristic method to conservatively remove excess precision. Then, our system proceeds in three steps: (1) separating the support into area clusters; (2) searching for candidate replacement objects and positioning them within each cluster; and (3) combining compatible solutions from each cluster into a list of global solutions. After determining the best global solution, the slicing process continues as normal, generating printing paths for the modified set of 2D polygons to accommodate the replacement objects. Our system further modifies G-code to add object holders, insert pauses with brief messages to alert the user to insert objects, and to make layer adjustments to accommodate arbitrary object heights. Working directly with slices and G-code means our technique is compatible with support generation strategies other than standard support fill, such as cones or trees.

5.1 Separate Support into Clusters

To help the subsequent search step find solutions focused on areas with large volumes of contiguous support, K-Means [24] is used to separate the total support material volume into sets of clusters. For efficient parallel computation, the maximum k is chosen so each cluster can be explored in parallel. As an example, 5 cluster sets for $k = 1$ up to $k = 5$ generate 15 unique clusters (i.e. 1+2+3+4+5 = 15) which can all fit within a 16 core processor.

Clustering is performed on a height map of support. Support polygons from all slices are discretized onto a $1x1$ *mm* grid, with each cell set to the number of slices that contributed. Considering height is important since adjacent support areas in X/Y should not be in the same cluster if they extend to vastly different heights in Z. After clustering, a concave hull (concavity factor 100) is generated for each cluster. Figure 5 shows an example of $k = 5$ clusters.

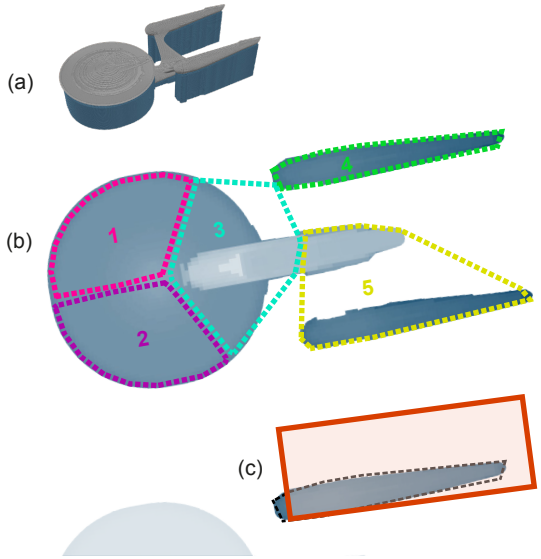


Figure 5: Example support material clustering and search: (a) sliced model with support shown in blue; (b) support material height map (darker means more support) with $k = 5$ clustering and convex hulls; (c) each cluster is searched independently, here cluster 4 is shown with a random initial object position (orange rectangle) aligned to part of the hull outline. In practice, many different replacement objects and positions are found for each cluster.

5.2 Find Solutions for Each Support Cluster

We use Particle Swarm Optimization (PSO) [20] to explore the solution space since it offers robust control parameters, is sufficiently efficient, and used in related geometry applications [15, 23, 37]. In parallel, PSO searches are performed within each support cluster hull generated by the sets of clusters. Each particle describes a target cuboid volume with: bottom X/Y corner position; length, width and height; and 2D rotations around the X/Y corner and object centre. Including two rotation parameters enables the particle volume to swivel around cylindrical model features while maintaining alignment with other particles. At each optimization step, a candidate replacement object is selected from the library that best approximates the particle's target length, width, and height. If available, this can be an assembly constructed from a set of toy bricks, or a variable height object like a screw-jack.

When the PSO search starts, each particle target cuboid volume is initialized randomly to span the smallest and largest replacement object in the library. The particle positions and orientations are initialized so 90% of the particles have one edge aligned with part of the cluster hull outline (Figure 5). An object placed beyond the hull is unlikely to overlap additional support from that cluster, and the hull typically defines a border between support and the model, suggesting a non-colliding position with large support area coverage. The remaining 10% of particles have entirely random positions and orientations to provide a way for the PSO to escape early local maxima. Then, the PSO search iterates in a loop: adjusting particle positions, orientations, and sizes; selecting matching objects for each particle; then calculating their fitness (see "Fitness Function" below). As is typical for PSO, the particles initially move at high speeds to explore a wide range of possibilities, then gradually reduce speeds so each settles into local fitness maxima. The search stops when it no longer yields a significant improvement in fitness. This is defined as multiple subsequent steps where an accumulated and decayed fitness improvement is below a relative threshold. The search quality setting in the user interface indirectly adjusts the decay and relative threshold to make the stopping condition more or less lenient. On completion, each support cluster PSO returns a list of candidate objects that do not overlap the model.

5.2.1 Fitness Function. Each particle fitness is a product:

$$f_i = S \times o \times c \times b \tag{1}$$

where:

S is the *volume of support replaced* by the candidate object (mm^3).

o is an *overlap penalty* defined using $\omega \in [0, 1]$, the proportion of the object that *does not* intersect with the model:

$$o = \begin{cases} (1-\omega)/10, & if\, \omega < 1 \text{ (there is overlap)} \\ 1, & \text{otherwise (no overlap)} \end{cases}$$

Note that any overlap invalidates the candidate from being used in a final solution, but scaling o by a factor of $1/10$ when there is any overlap creates a soft constraint, resulting in a smoother fitness space for the PSO to explore.

c is an *object complexity penalty* defined using an object type complexity factor α and the level of interaction complexity selected by the user $I_c \in [0, 1]$:

$$c = 1/\alpha^{(1-I_c)}$$

Fixed size, simple objects have $\alpha = 1$; more complex objects that must be assembled prior to insertion, such as stacking floppy discs or assembling toy bricks, have $\alpha = 2$; the screw jack, which requires manual height adjustment has $\alpha = 3$. The interaction complexity factor is used to condition the object complexity (e.g. $I_c = 0$ means keep complexity low, $I_c = 1$ means high complexity is acceptable).

$b \in (1.0, 1.05]$ is a *small footprint bonus*. Objects with smaller contact area on the baseplate are less likely to interfere with other replacement objects when combining local solutions into global solutions. b is mapped to the size of the smallest and largest replacement objects, such that the smallest object receives full 5% bonus and the largest receives almost none.

5.2.2 Constructing Ad Hoc Towers. If ad hoc towers are enabled in the library, the system also attempts to recursively place objects on top of other candidate objects, using those returned by the main PSO as a base. This tower creation uses a PSO with fewer particles, earlier stopping criteria, and a search area restricted to the base particle area. The fitness function is the same, except that b is reversed to assign a bonus to larger objects: this increases the potential for additional objects in the tower by providing more surface area for the next object.

5.2.3 Run-time Optimizations. Strategically chosen subsets of layers are used to approximate support material savings S and for early detection of model collisions. Initially, the calculation uses the lowest and highest layer according to the particle height, along

with 5% of the layers between. If the approximation of S is above a threshold (based on maximum material savings of previous candidates) and no collision is detected, the remaining layers are added to the calculation in random order, each time checking for the same stopping conditions. This method significantly decreases objective function computation since particles with collisions or poor support savings are not fully calculated.

Replacement objects with variable height, such as toy construction bricks and screw-jacks, are explored in an optimal way. They are first evaluated at their minimum height (e.g. a fully retracted screw jack) to confirm base placement validity. Then, the adjustable height is gradually increased within the possible range until the most support material is replaced without causing a collision. This way, all possible height variations for a candidate object are evaluated at once to find an optimal configuration.

5.2.4 Final List of Candidate Objects. When the main PSO search and ad hoc tower searches have completed, all candidate object placements that do not collide with the model and replace more than a minimum amount of support material are saved. The minimum threshold ϵ (in mm^3) is based on the search quality setting $q_\epsilon \in [0.2, 0.3]$ and the replaced volume of the current best candidate S_b, where $\epsilon = q_\epsilon \times S_b$. This list can be very large, so it is further pruned heuristically to select a diverse set of options with a focus on the most beneficial ones. Duplicates of equivalent results are removed. Results are sorted by support material saved, then every ith result is selected where $i = \lfloor 1.5(i-1) \rceil$. This indexing method selects the very best results, some good results, and a few poor ones. This selection process is performed separately for lists of individual objects and lists for each tower size (i.e. 2 object towers, 3 object towers, etc.). This selection process ensures a relatively stable number of outputs, even though the number of valid results from the search can vary wildly. A diverse set of candidates is useful when creating global solutions across objects selected for each support cluster.

5.3 Combine into Global Solutions

The system now combines different objects across support clusters into global solutions of replacement objects. Each global solution can use a specific physical object at most once, and all objects in a solution must not collide. Global solutions are constructed using each single cluster set (e.g. considering only the solutions from 4 clusters in the $k = 4$ set) and across all cluster sets. Sets of global solutions are generated in parallel on separate threads.

Finding the best combination of candidate objects to form a global solution is equivalent to determining the solution to a *maximum weighted clique problem* on a graph describing object compatibility. A graph is constructed where each vertex represents a candidate object (or tower) at the solution position, with the vertex weight set to the objective function value for the associated particle (f_i in Equation 1). Vertices are connected by edges if the objects they represent are not the same replacement object and they do not collide. A custom solver is used to find the maximum weighted clique for clique sizes ranging from 1 to 8 (we consider a single vertex a clique of size 1). Larger cliques translate to solutions with more objects which require more user interactions, so finding global solutions with smaller cliques (representing fewer objects) is important to provide options to the user.

The conventional weight of a clique is the sum of included vertices, but we set the clique weight using a global objective function that also considers the user's desired level of interaction density:

$$f_g = \frac{\sum_{x=i}^{N} f_i}{0.75P^{(1-I_d)} + 0.25N^{(1-I_d)}}$$

where:

f_i is the *fitness of a candidate replacement object* (Equation 1);
N is the *number of objects to be placed*;
P is the *number of printing pauses* (i.e., number of "interruptions");
$I_d \in [0, 1]$ is the user's *interaction density setting*.

Note the number of printing pauses is weighted 3 times greater than the number of objects to be placed (i.e. $0.75P$ vs. $0.25N$). This codifies an assumption that more pauses are more disruptive than placing more objects in a single pause. However, these weights can easily be adjusted, in the future they could be included in the user interface.

5.3.1 Run-time Optimizations. We prune some vertices using heuristics to decrease computation for the maximum weight clique solver. Since maximum cliques are evaluated in order of increasing clique size, vertices with fewer edges than the target clique size minus one are pruned to speed up following iterations. For example, when searching for a clique of size 4, a vertex with only 2 neighbours can be pruned. If multiple vertices are not connected, but each has identical neighbours, then they likely replace a similar area of support and are interchangeable within a solution. Among sets of interchangeable vertices, only the highest weight vertices representing a single object and a tower are kept in the graph. If more than 150,000 combinations of vertices remain after the previous two steps, additional vertices are pruned randomly to avoid excessive computation times.

5.4 G-code Modifications

Once the most desirable global solution is found, all support material print paths that are replaced by objects are removed from the slicing result. Our system then modifies the height of layer boundaries near the top surfaces of objects, adds printed object holders, printed text instructions on the baseplate, and inserts printing pauses.

The layer boundary is the height at which filament from a new layer touches the previous layer, and it rarely matches the height of a replacement object precisely. Without adjustment, these differences can cause visual imperfections and structural weaknesses. Our system adjusts adjacent layers to match the height of inserted objects by considering the four cases illustrated in Figure 6. When the top surfaces of two or more inserted objects fall within the same layer boundary, each is handled separately.

Custom holders and instructions are created by inserting parameterized G-code generated by our system. Each object holder is composed of 6 layers of filament extruded in a single outline path offset from the object by half the nozzle thickness (e.g. 0.2 mm for a 0.4mm nozzle). A brim for increased adhesion can be added using a user setting. Brief instructions are printed on the baseplate near the object as a single-layer vector line font with a user-defined letter

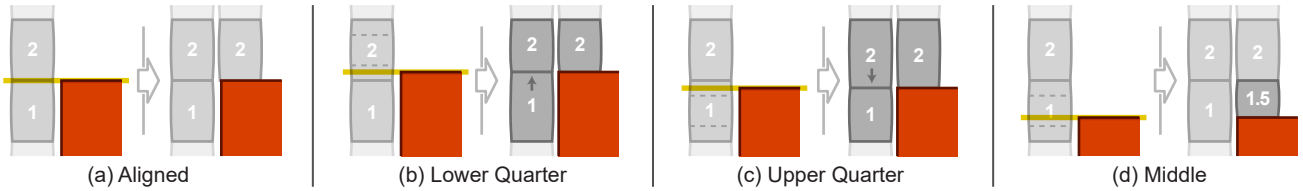


Figure 6: Layer height adjustments cases showing two adjacent boundary layers (grey shapes) before and after adjustment to object (orange shape): (a) object height and layer boundary are aligned, no adjustment necessary; (b) object top is within lower quarter of layer 2, layer boundary adjusted upwards; (c) object height is within upper quarter of layer 1, layer boundary is adjusted downwards; (d) object top is within middle 50% of layer, new intermediate layer inserted between boundary layers.

height in mm. Instructions provide a human-readable replacement object id (also searchable in the library) with dimensions for simple objects or manual adjustment for variable objects, such as screw jack height or toy brick assembly size. The G-code that pauses the print for the user to insert an object can be customized, such as allowing time for a heated enclosure to reach a desired temperature. The printing material and time required to print the holders and instructions are included in the measurements throughout the paper.

6 EVALUATION

To evaluate our approach, we conducted a technical evaluation using the system with a dataset of diverse models. The primary measures are proportions of printing time and support material saved. We also explore variations relating to the interaction density and complexity "usage space" factors discussed earlier. These are operationalized as slicer parameters and what types of ad hoc objects are available in the library.

6.1 Test Dataset of 3D Models

We assembled 15 3D models to test various conditions. Nine are models identified in previous research related to 3D printed support structures and related topics. Some were reconstructed from descriptions within the related work. The remaining six are canonical 3D models used in computer graphics research. Note that not all previous works identify the models used, and few provide mesh files or references. We also limited our selection to models that can be accessed freely for replication. Some models were scaled to fit into the build volume of the printer preset used in the evaluation. Models were printed in their original orientation, which in some cases is non-optimal to showcase extreme support cases. Our goal was to gather an assortment of objects with reasonable diversity, replicability, and comparability, and using objects from previous work mitigated selection biases. The complete dataset is shown in Figure A.1 of Appendix A and the mesh source files are provided in supplementary materials.

6.2 Ad Hoc Objects

The ad hoc objects used for the evaluation were primarily collected from the authors' households. This included 22 objects with a fixed height, such as a book, woodworking off-cuts, 3D printing calibration cubes, and small cardboard packaging boxes of various sizes. Object dimensions ranged from 73 × 10 × 9.5 mm to to 172 × 144 × 37 mm. Nine floppy discs were used as one type of stack-able object. A toy brick construction kit was included as an option in the library, which essentially enables an arbitrary variety of ad hoc objects by assembling blocks into a required size as needed. Finally, there were 3 adjustable objects with continuously variable height in the form of 3D-printed screw-jacks with different height ranges (29-46 mm, 49-66 mm, and 78-95 mm). All three screw jacks used the same prism shape cap. In our experiments, we use these objects with different levels of complexity to form different libraries as a dependent variable to evaluate the impact of interaction complexity.

6.3 Procedure

We first generated machine printable G-code for all dataset models with the unmodified Kiri:Moto slicer (version 3.3.D12) [3]. We used default slicing settings, with the exception of enabling automatic support and using a support pillar size of 8 instead of 6. Increasing the support pillar size ensures that continuous areas of support are merged into a single structure, rather than forming numerous small disjoint towers. This reduces execution time by reducing the number of support polygons. These default slicing results are baselines for material weight and printing time. All models were also sliced without any support to obtain the model material weight to compute proportional savings in material and printing time.

We then generated machine printable G-code for all models in the dataset using our system. In these tests we use the default print volume (300 × 175 × 150 mm) with all models centred on the buildplate. Since replacement objects must be placed fully within the buildplate area, the position of model affects the possible solution space. All tests were run on a high end consumer Windows PC with 24 available processing threads using the Chrome browser.

6.4 Independent Variables

There are three independent variables: two settings offered by our customized slicer (search quality and interaction level) and which library of replacement objects is available.

- *Search Quality – Fast or Thorough.* This setting adjusts details of the PSO search, such as the number of search particles; how persistent the search should be, as well as thresholds for early exit conditions during the search.
- *Interaction Level – Low, Medium, or High.* This evaluates increasing levels of user interaction as a combination of interaction density and complexity, where I_d and I_c are 0 for *Low*, 0.5 for *Medium*, and 1.0 for *High*.
- *Library – Simple, Towers, or Variable.* The tests are run using three libraries containing progressively more versatile and adjustable

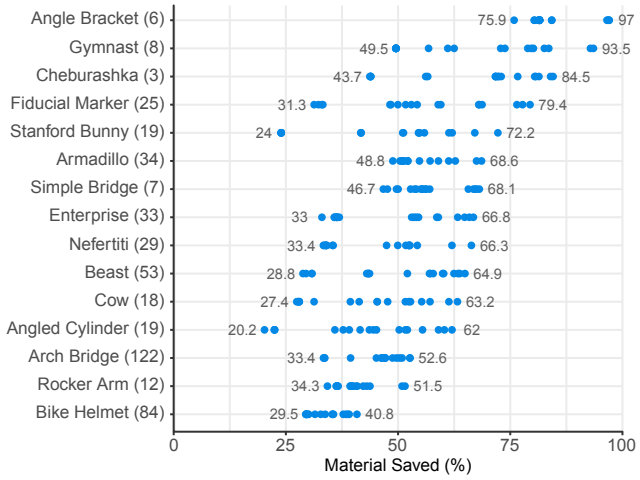


Figure 7: *Material Saved* by model for 12 test runs spanning all conditions. The default support material mass for each model is shown in parenthesis (in grams).

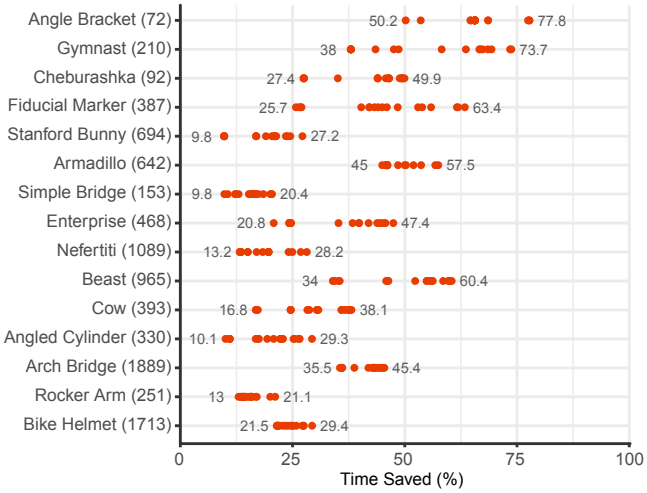


Figure 8: *Time Saved* by model for 12 test runs spanning all conditions. The proportion of print time saved by ad hoc objects varied depending on model and how much default printing time is spent creating support. The default print time for each model is shown in parenthesis (in minutes).

ad hoc objects and arrangements. The *Simple* condition is a library with 22 fixed height objects without stacking; the *Towers* condition adds the easily stack-able floppy discs and permits object stacking to form ad hoc towers; and the *Variable* condition adds 3 height-variable screw-jacks with different height ranges, and enables assembling objects of varying sizes using a toy brick construction kit.

6.5 Dependent Variables

For each combination of independent variables, we conducted a search with each test model and calculated three metrics:

- *Material Saved.* This is the proportion of printed support material weight generated by our system to the support material weight generated with the unmodified slicer. We calculate support material weight based on the printing weight estimates provided by the slicer itself, which is defined as the total length of extruded filament times the area of a cross section of the filament, times the average density of PLA.
- *Time Saved.* We use printing time estimates provided by the slicer itself for the total time to print the model including all support material. We define time saved as a proportional savings of print time using the ratio of time to print G-code generated with replacements to the time to print G-code generated with the unmodified slicer.
- *Search Time.* The run time of our system was measured during the ad hoc object search process and during the entire slicing process to generate all G-code.

6.6 Results

For each of the 15 models, we ran the system for 12 test runs spanning all conditions: 2 *Search Qualities* (fast, thorough) × 3 *Interaction Levels* (low, medium, high) × 3 *Libraries* (simple, towers, variable). This produced 270 data points for each dependent variable.

Each dependent variable is analyzed using a 2 × 3 × 3 repeated-measures ANOVA with test model as a random blocking variable. Visual inspection of Q-Q plots revealed no severe violations of normality; Greenhouse-Geisser corrections are applied to p-values when Mauchly's test for sphericity was significant; significant differences are reported at a 5% level. All post hoc tests use Bonferroni-corrected pairwise t-tests.

6.6.1 Support Material Saved. Overall, replacement objects reduced support material by 52.0% (95CI [50.0, 54.1]). *Material Saved* ranged from 20.2% (worst result for the Angled Cylinder model) to 97.0% (best result for the Angle Bracket) (see Figure 7).

Increasing levels of user interaction saved more material, libraries providing more versatile and adjustable ad hoc objects saved more material, and using a more thorough PSO search saved more material (see Figure 9). There was a main effect for *Interaction Level* ($F_{2,28} = 29.1, p < .001$), *Library* ($F_{2,28} = 42.2, p < .001$), and *Search Quality* ($F_{1,14} = 65.2, p < .001$). Post hoc tests found significant increases in material savings for *Low* (44.0%), *Medium* (53.8%), and *High* (58.5%) *Interaction Level* (all $p < .05$) and significant increases for the *Simple* (45.1%), *Towers* (50.6%), and *Variable* (60.6%) *Libraries* (all $p < .05$). *Search Quality* had less of an effect, with 50.7% support saved for *Fast* versus 53.5% saved for *Thorough*.

With the *Towers* library, moving from a *Low* to *Medium Interaction Level* made the most pronounced increase (14.6%), unsurprising as only *Medium* and *High Interaction* permit the additional pauses required to construct towers, with little or no additional savings moving to the highest level of interaction. For the other libraries, there was a steady increase in average material savings ranging from 4.7% to 7.9% when moving to the next higher *Interaction Level*. There was a *Interaction Level* × *Library* interaction ($F_{4,56} = 3.17, p = 0.048$). Post hoc tests examining *Interaction Levels* for each *Library* found significant differences in all cases (all $p < .05$) except between *Medium* and *High Interaction Level* with the *Towers Library*.

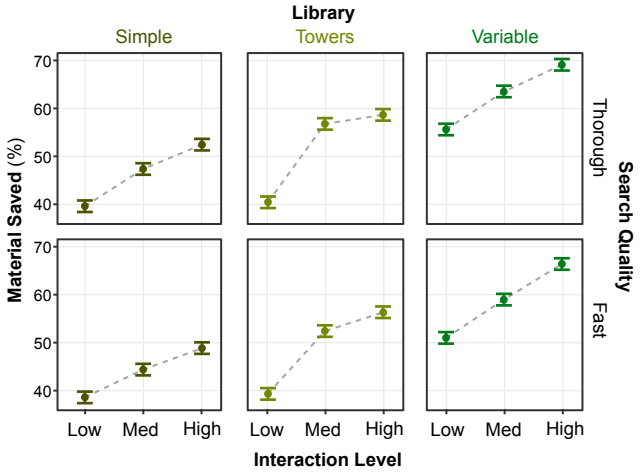


Figure 9: Percentage of *Material Saved* for each combination of *Interaction Level*, *Library*, and *Search Speed*. Error bars are 95% confidence intervals.

6.6.2 Time Saved. Overall, replacing support with objects reduced print time by 34.8% (95CI [32.8, 36.9]). *Time Saved* ranged from 9.8% (worst performance for the Stanford Bunny) to 77.8% (best result for the Angle Bracket) (see Figure 8).

Additional print time was saved with increasing levels of user interaction, with libraries providing more versatile and adjustable ad hoc objects, and with a thorough PSO search. There were main effects for *Search Quality* ($F_{1,14} = 70.0, p < .001$), *Interaction Level* ($F_{2,28} = 17.3, p < .001$), and *Library* ($F_{2,28} = 22.5, p < .001$). Post-hoc tests found *Time Saved* increased with each *Interaction Level*: *Low* (29.7%), *Medium* (36.1%), and *High* (38.6%). The same trend was found with *Library*: *Simple* (30.8%), *Towers* (34.4%), and *Variable* (39.2%). Though significant, *Search Quality* had less of an effect, with 34.4% time saved for *Fast* versus 35.2% time saved for *Thorough*.

6.6.3 Search time. On average, *Search Time* took 20.9s (sd 23.5 s, (95CI [18.1, 23.7])). *Search Time* ranged from 0.9 s (Angle Bracket, fast, simple, low interaction) to 124 s (Bike Helmet, thorough, towers, low interaction), 18 searches took longer than a minute. For comparison, the average slicing time without using our system is 16.7 s. Table 1 summarizes *Search Time* results.

Searching the *Simple* library was consistently fastest and, as expected, a *Thorough* search takes more time. There were main effects for *Library* ($F_{2,28} = 6.32, p = .016$) and *Search Quality* ($F_{1,14} = 19.5, p < .001$). Post hoc tests found *Simple* (14.9s) faster than *Towers* (27.2s) and *Variable* (20.8 s) (all $p < .05$). *Search Quality* was unsurprisingly faster on average with *Fast* (12.1s) than *Thorough* (29.7s).

Interestingly, the most versatile *Variable* library actually reduced search time compared to *Towers* with ad hoc object stacking. This makes sense, as objects with variable height may already reach up to the desired height, avoiding the need to construct towers recursively in many cases. Within each library, *Fast* quality search times are approximately half *Thorough* quality search times. There was a two-way interaction effect for Search Quality and Library ($F_{2,28} = 10.5, p < .001$). Post-hoc tests found that for *Fast* search, *Tower* (14.5s) and *Variable* (12.7s) were significantly longer than

Table 1: *Search Time* by *Library* and *Search Quality* (mean time in seconds, stdev in parenthesis).

	Simple	*Towers*	*Variable*
Fast	9.2 (8.6)	14.5 (15.0)	12.7 (10.9)
Thorough	20.6 (20.4)	39.9 (36.8)	28.8 (25.0)

Simple (9.2s) ($p < .05$) but not different from each other. However, for *Thorough* search, all three values were different: *Towers* (39.9s) was the longest, followed by *Variable* (28.8s), then *Simple* (20.6s) (all $p < .05$). For all libraries, *Thorough* search took longer than the equivalent *Fast* search.

7 DISCUSSION

Replacing support material by involving the user in the fabrication process rather than using automated motion controlled mechanisms provides additional flexibility. Unlike previous solutions, this approach is not limited to a rectangular grid. This enables custom object shapes and arbitrary positions for support replacement, which can reach and replace additional support material. The flexibility comes at a cost of increased computational complexity. By applying domain knowledge and placement heuristics to streamline the optimization, and careful pruning to shrink the solution space to a manageable size, our system overcomes the complexity to provide the sustainability benefits of reusing support to anyone who owns a FDM 3D printer.

Our system must consider the importance of a user's time. With decreasing size of print models, the benefits of using the system can become small enough that they might not justify any additional user interaction. The system avoids interactions that only save a token amount of support and exports the unaltered slicing result if no worthwhile user interactions can be made. Though not being able to save support for any model is an undesirable outcome, the affected small prints produce less waste from support material in the first place. In contrast, the system is most effective for large models that would generate excessive amounts of waste. Overall, the system provides the most benefit per user interaction for models that require large amounts of continuous support.

7.1 Ad Hoc Objects for Sustainability

From our 15 diverse models, using ad hoc objects reduced support material by 52.0% and print time by 34.8% on average. The best cases for all models allowed replacing more than two thirds of support material on average. Print time has a lower effect generally, since the print time depends on both support and model material. Other techniques, such as in-fill optimization [38] or objects being inserted inside the model [37] could further reduce print time.

The efficacy of using ad hoc objects as support replacements depends on the specific arrangement of support material. In some cases, the effects can be dramatic. For models with large, planar overhangs like the Angle Bracket, using replacement objects can remove almost all support material (97.0% savings), and reduce printing time by more than half (77.8%). However, in other cases, results are limited. In some models much of the support material is build on top of model geometry and unreachable from the build plate. Given the proposed "usage space", extensions can make it

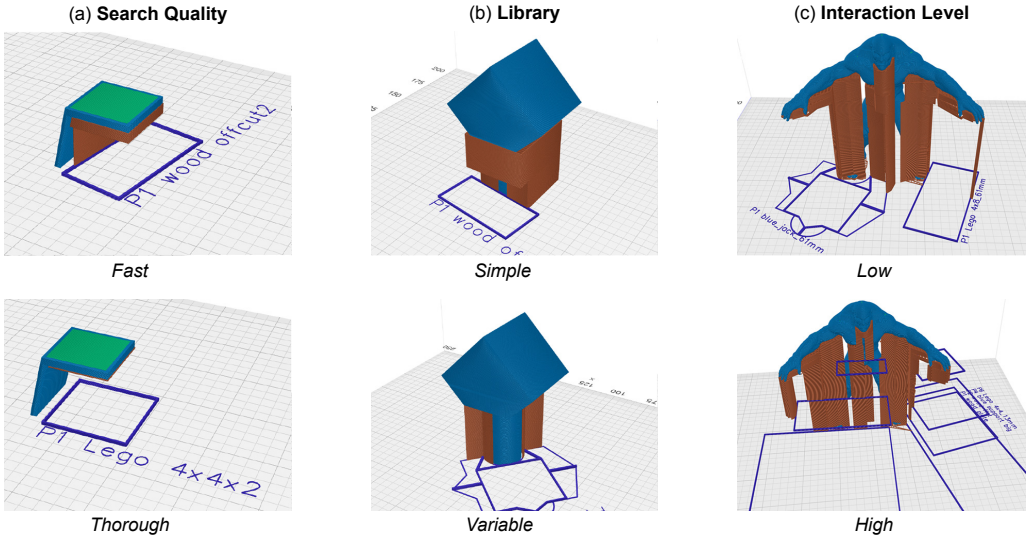


Figure 10: Examples of solution variations: (a) a fast search solution for Angle Bracket places a single wood replacement with sub-optimal height; a thorough search solution finds a more optimized object constructed with toy bricks; (b) using a library of simple objects, a single block of wood is used to save some support; using the variable object library, non-rectangular objects are available to better fit the curved surface of the Angled Cylinder; (c) a solution restricted to low interaction can still replace a large volume of support with multiple objects placed during a single pause; a solution that permits a high level of interactions can replace even more support, but requiring 7 print pauses for this example.

possible for ad hoc objects to also replace support in these situations, but likely with some added interaction complexity. Figure 10 illustrates more examples of solution variations.

7.2 Comparison with Previous Approaches

Comparing our average savings to previous work is problematic, as each work uses a very different set of test models. Xu et al. [39] test 5 models and Yigit et al. [40] test only 2. We use a larger dataset of 15 models and many are more complex in terms of support requirements. This is an argument to standardize test model datasets.

Perhaps most instructive is to compare using a common model. From the models tested by Xu et al., we were able to source the "gymnast". With this, our system achieves maximum support material savings of 93.5% compared to their 78.1%. From models tested by Yigit et al., we were able to source the "bunny". Here, our system achieves 72.2% savings compared to their 85%. However, even these more direct model comparisons are limited by system evaluation conditions, with different slicers and different slicing settings. For example, more than 19% of support is located above the body and below the ears of the bunny in the slicing result used in our evaluation. Those upper support regions are unreachable for both our system and the Yigit et al. system, so it is unclear how more than 81% of material could be saved.

In terms of run time, Xu et al. report an average optimization time of 27.4 s, which is longer than our average run time of 20.9 s, despite our more complex set of test objects. For simple test models, our system can deliver results orders of magnitude faster: our lowest search time is less than 1 second, compared to their 25.6 s.

7.3 Alternative types of support structures

We used the default kiri:moto supports in our evaluation, which generates support towers that are shaped to match the outline of model parts that require support. This resembles "Snug" supports in the Prusa slicer [29]. Other types of support structures, such as Prusa Organic or Cura [34] Tree supports can further reduce the amount of support material. The slicer estimates for printing time and material used by kiri:moto, Prusa, and Cura and each of their respective support types can be found in Appendix C. The data suggests that there is no single optimal type of supports. One model failed to slice with Cura Tree supports. Printing Prusa Organic supports requires additional printing time, and for some of the models, they also required more material than Prusa Snug supports.

Comparing "between-slicers" is less controlled than the "within-slicer" comparisons above and in our submission, as slicers may use different approaches to calculate the material and time cost estimates. The slicing results alone also provide little insight whether or how much the supports impact printing reliability and visual print quality. By correcting for relative differences in support weight using a per-model ratio of material used by Prusa Snug to kiri:moto supports, we can estimate material savings of our system over Prusa Organic supports. The estimated maximum material savings of our system are 63% on average, ranging from 37% to 95%. Using the worst case material savings of our system, the average improvement drops to 26%. Interestingly, Prusa Organic support is estimated to outperform the worst case of our system for the Nefertiti and Angled Cylinder models. Our method is in principle compatible with any software support generation method. Based on the Prusa and Cura slicing data, we estimate an additional 3-8% of all support

beyond our presented results can be saved by combining advanced support generation methods with support replacements.

7.4 Cost-Benefit of Mid-Print Interactions

Correlated with support material saved, using replacement objects can have dramatic results on printing time. For example, reducing print time from 31.5 h to 14.3 h with the Arch Bridge model (45.4% savings) or from 16.1 h to 9.7 h with the Beast model (60.4% savings). The lowest time saved in the test set was 15 minutes. User insertion time is not included in these values, but since insertions take less than a minute to complete, time savings remain relevant even for small models. Our system prints the names of all replacement objects on the first layer, so the user can search for all objects during printing and place them nearby ready to be inserted. The dimensions of stacks of objects are printed on the build plate as well, so they can also be assembled ahead of time. A specific assembly order is not required. For example, a stack of toy bricks can have multiple holes and cavities without issue, as long as the stack meets the overall dimensions and has a solid top surface. In addition, the effort to manage a library of ad hoc objects is more of an up-front setup cost. We use a library management interface like Scrappy [37] which has a relatively easy way to enter and locate objects, so in practice this is not a time-consuming aspect.

The main cost is not the time to complete the interaction, but when the interaction must be performed. With our approach, someone has to manually insert objects at one or more intervals during a long print job. We note that similar kinds of interruptions exist for manual tasks with appliances: for example when loading and unloading clothes into household washing and drying machines. The scheduler in our system can ensure that interruptions occur at more convenient times, such as during daylight hours. One can imagine design solutions for further reducing the disruptive nature of the notification: setting up timers or calendar alerts to facilitate scheduling, or even recruiting passersby to insert the objects in a public area or shared workplace. Finally, the adaptability of our fitness function allows minimizing this cost, even restricting solutions to a single mid-print interaction if desired, while still providing worthwhile material and time savings.

Future work could more closely examine the associated cost-benefit trade off. We imagine two complementary approaches: releasing the software to log actual users as they choose different schedules to trade-off interaction complexity; and a longitudinal study using a generic interaction task that can be controlled for density and complexity.

7.5 More Complex Replacement Objects

Further increasing the complexity of interactions could replace support material at places that are otherwise unreachable. For example, an ad hoc bridge could be constructed from three objects, such as a thin piece of wood taped onto two construction brick stacks (Figure 11a). This could enable supporting a model with large openings. To save even more support, a thin plate could be custom printed to perfectly fit into irregular object profiles (Figure 11b). Depending on the size and height of the supported area, the overhead to print this kind of custom object before the main print could still save overall time and material, especially when manufacturing multiple copies of the same model.

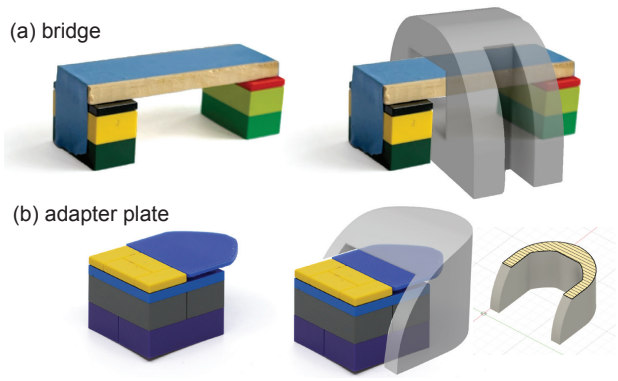


Figure 11: Mock-ups of more complex replacement objects: (a) combinations of objects to reach internal support structures like through-holes; (b) a custom adapter printed to perfectly fit into irregular object profiles.

7.6 Improving Mechanized Approaches

Our more advanced optimization method system is versatile enough to find object placement solutions to work in tandem with automated approaches from prior work. For example, Yigit et al. [40] propose using standardized cubes placed over a 1 cm grid resolution by a robot arm. This major simplification would reduce the search time for solutions, but our results show that ad hoc objects that are not restricted to a grid or a standardized shape are more flexible in placement and can save more material. Using a robot arm (or even the printer itself [19]) to place heterogeneous objects would be challenging, and robot actuators likely lack the dexterity to use our printed holders for object stability. However, our approach would enable more variation in block size and height, and eliminate the limitation of a 1 cm grid resolution. Our optimization algorithm could also be modified to find the optimal positioning of a model on an actuated build plate [13, 39] by treating each grid cell as an adjustable height object at a constrained location. Our approach of relying on users to insert replacement objects could enable a lower-cost alternative to an actuated build platform. Instead, users could place a collection of standardized raised platforms on top of the build plate. Systems that rotate the build platform [11] to avoid printing support may still require some support structures to print complex models. As our printed holders can hold objects even when the build plate is flipped upside down, our approach could work at steep angles in conjunction with such systems.

7.7 Practical Considerations

Although pausing a print is common in certain applications, such as printing ranges of layers in different colours, or may be required to continue a print when the filament spool runs out, some users may be unfamiliar with the process. To make best use of printing pauses, they should be dialed in like any other slicing parameter. For example, our system includes machine commands to prime the nozzle after a pause, but users may wish to adjust the details of

this process based on the type of extruder used. The primed material may be loosely attached to the print. Although easily removed and typically attached to a support tower, custom adjustments to the pausing instructions could prevent such minor blemishes. Our system inserts pauses prior to printing support material. Should a pause cause blemishes, they will typically be constrained to the support material itself. If the build plate is movable, inserting objects may push the build plate out of position. After a pause, the build plate axis is homed to reset its location and avoid issues from accidental movements. Using printing materials that require a heated build chamber might necessitate custom pause handling for prolonged pauses. Maintaining a minimum temperature during the pause or adding delays to allow the temperature within the enclosure to equalize slowly before resuming the print could prevent deformations. The system includes a basic scheduler to help avoid long idle periods.

Using support replacement objects offers additional modes of printing failure recovery. If a user notices a lack of adhesion on the object after a pause, they may still be able to adjust the height of a screw-jack or add layers of blue tape to match the current printing height to restore adhesion after skipping some layers.

Printed supports sometimes stick thoroughly to the supported part of the model. Replacement objects can in some cases essentially allow supporting the area with blue tape rather than filament. The tape is easier to remove sequentially which may prevent damaging fragile models. The smooth finish of the supported area may also be desirable. Support material can typically be removed cleanly from the replacement objects by hand or with a knife.

7.8 Limitations

Depending on the build plate and filament colour, the printed instructions can be difficult to read under certain lighting conditions. Users may require hand tools such as calipers to measure replacement objects with sufficient accuracy.

The system slightly adjusts the thickness of some layers, which are usually imperceptible, but may still be undesirable when printing mechanical objects. Instead of adjusting layer thicknesses, it would be possible to print a layer of material underneath inserted objects to raise them to the desired height. This solution however would come at the cost of using some additional support material.

Our findings, especially trends within our dataset, may not generalize to all models. Prior work often used very small sets of test models and models chosen or built to demonstrate benefits. We intentionally chose a diverse, but representative set of 15 models, and we were able to observe a wide range of outcomes. The current system would be unable to save any support material for pathological examples like a hollow egg model, which only includes internal support. In the worst case scenario where the system fails to find an opportunity to replace support material, the unaltered slicing result is exported instead.

8 CONCLUSION

We introduced a practical software-only solution where people manually insert ad hoc objects to replace support material during a FDM 3D print. A technical evaluation shows support material is reduced by an average of 52.0% across 15 models (ranging from 20% to 97%) and revealed how model characteristics influence support savings. Possible future improvements include using a print head depth probe to work with non-planar replacement objects and to measure replacement objects with the printer itself, perhaps circumventing the need for an object library. Regarding user interaction, a single basic mid-print interaction saved 39.1% of support material on average, but with increased density and complexity of interactions, average savings increased to 60.1%. By involving users even a little bit during the print job, 3D printing can be faster and more sustainable.

ACKNOWLEDGMENTS

This work made possible by NSERC Discovery Grant 2018-05187 and the Canada Foundation for Innovation Infrastructure Fund 33151 "Facility for Fully Interactive Physio-digital Spaces".

REFERENCES

[1] Marc Alexa, Kristian Hildebrand, and Sylvain Lefebvre. 2017. Optimal Discrete Slicing. *ACM Trans. Graph.* 36, 1, Article 12 (jan 2017), 16 pages. https://doi.org/10.1145/2999536

[2] Paul Alexander, Seth Allen, and Debasish Dutta. 1998. Part orientation and build cost determination in layered manufacturing. *Computer-Aided Design* 30, 5 (1998), 343–356. https://doi.org/10.1016/S0010-4485(97)00083-3

[3] Stewart Allen. 2020. Kiri:Moto: An open-source slicer for Laser cutting, 3D printing, CNC milling . https://grid.space/kiri/. [Online; accessed 4-April-2023].

[4] Gianluca Barile, Alfiero Leoni, Mirco Muttillo, Romina Paolucci, Gianfranco Fazzini, and Leonardo Pantoli. 2020. Fused-Deposition-Material 3D-Printing Procedure and Algorithm Avoiding Use of Any Supports. *Sensors* 20, 2 (Jan. 2020), 470. https://doi.org/10.3390/s20020470 Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[5] 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

[6] Paramita Das, Ramya Chandran, Rutuja Samant, and Sam Anand. 2015. Optimum Part Build Orientation in Additive Manufacturing for Minimizing Part Errors and Support Structures. *Procedia Manufacturing* 1 (2015), 343–354. https://doi.org/10.1016/j.promfg.2015.09.041 43rd North American Manufacturing Research Conference, NAMRC 43, 8-12 June 2015, UNC Charlotte, North Carolina, United States.

[7] Jérémie Dumas, Jean Hergel, and Sylvain Lefebvre. 2014. Bridging the gap: automated steady scaffoldings for 3D printing. *ACM Transactions on Graphics* 33, 4 (July 2014), 98:1–98:10. https://doi.org/10.1145/2601097.2601153

[8] Ben Ezair, Fady Massarwi, and Gershon Elber. 2015. Orientation analysis of 3D objects toward minimal support volume in 3D-printing. *Computers & Graphics* 51 (2015), 117–124. https://doi.org/10.1016/j.cag.2015.05.009 International Conference Shape Modeling International.

[9] Gianfranco Fazzini, Paola Paolini, Romina Paolucci, Daniela Chiulli, Gianluca Barile, Alfiero Leoni, Mirco Muttillo, Leonardo Pantoli, and Giuseppe Ferri. 2019. Print On Air: FDM 3D Printing Without Supports. In *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)*. 350–354. https://doi.org/10.1109/METROI4.2019.8792846

[10] Ruiliang Feng, Xianda Li, Lin Zhu, Atul Thakur, and Xiangzhi Wei. 2021. An Improved Two-Level Support Structure for Extrusion-Based Additive Manufacturing. *Robotics and Computer-Integrated Manufacturing* 67 (Feb. 2021), 101972. https://doi.org/10.1016/j.rcim.2020.101972

[11] Wei Gao, Yunbo Zhang, Diogo C. Nazzetta, Karthik Ramani, and Raymond J. Cipra. 2015. RevoMaker: Enabling Multi-Directional and Functionally-Embedded 3D Printing Using a Rotational Cuboidal Platform. 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, 437–446. https://doi.org/10.1145/2807442.2807476

[12] Jiman Hong, D.L. Wu, D.C. Li, and B.H. Lu. 2001. Multi-objective optimization of the part building orientation in stereolithography. 35 (05 2001), 506–509.

[13] Shen Hongyao, Ye Xiaoxiang, and Fu Jianzhong. 2018. Research on the flexible support platform for fused deposition modeling. *The International Journal of Advanced Manufacturing Technology* 97, 9 (Aug. 2018), 3205–3221. https://doi.org/10.1007/s00170-018-2046-2

[14] Xiaomao Huang, Chunsheng Ye, Siyu Wu, Kaibo Guo, and Jian-hua Mo. 2009. Sloping wall structure support generation for fused deposition modeling. *International Journal of Advanced Manufacturing Technology* 42 (June 2009), 1074–1081. https://doi.org/10.1007/s00170-008-1675-2

[15] Alec Jacobson. 2017. Generalized Matryoshka: Computational Design of Nesting Objects. *Computer Graphics Forum* 36 (08 2017), 27–35. https://doi.org/10.1111/cgf.13242

[16] Seongje Jang, Byungjin Moon, and Kunwoo Lee. 2020. Free-Floating Support Structure Generation. *Computer-Aided Design* 128 (Nov. 2020), 102908. https://doi.org/10.1016/j.cad.2020.102908

[17] Jingchao Jiang, Jonathan Stringer, and Xun Xu. 2018. Support Optimization for Flat Features via Path Planning in Additive Manufacturing. *3D Printing and Additive Manufacturing* 6, 3 (Nov. 2018), 171–179. https://doi.org/10.1089/3dp.2017.0124 Publisher: Mary Ann Liebert, Inc., publishers.

[18] Jingchao Jiang, Xun Xu, and Jonathan Stringer. 2019. Optimisation of multi-part production in additive manufacturing for reducing support waste. *Virtual and Physical Prototyping* 14, 3 (July 2019), 219–228. https://doi.org/10.1080/17452759.2019.1585555 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/17452759.2019.1585555.

[19] Shohei Katakura, Yuto Kuroki, and Keita Watanabe. 2019. A 3D Printer Head as a Robotic Manipulator. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 535–548. https://doi.org/10.1145/3332165.3347885

[20] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol. 4. 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

[21] Jeeeun Kim, Anhong Guo, Tom Yeh, Scott E. Hudson, and Jennifer Mankoff. 2017. Understanding Uncertainty in Measurement and Accommodating Its Impact in 3D Modeling and Printing. In *Proceedings of the 2017 Conference on Designing Interactive Systems* (Edinburgh, United Kingdom) *(DIS '17)*. Association for Computing Machinery, New York, NY, USA, 1067–1078. https://doi.org/10.1145/3064663.3064690

[22] Jusung Lee and Kunwoo Lee. 2017. Block-based inner support structure generation algorithm for 3D printing using fused deposition modeling. *The International Journal of Advanced Manufacturing Technology* 89 (03 2017), 2151–2163. https://doi.org/10.1007/s00170-016-9239-3

[23] Kwang Y. Lee and Jong-bae Park. 2006. Application of Particle Swarm Optimization to Economic Dispatch Problem: Advantages and Disadvantages. In *2006 IEEE PES Power Systems Conference and Exposition*. 188–192. https://doi.org/10.1109/PSCE.2006.296295

[24] S. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489

[25] René K. Müller. 2021. 3D Printing: 90° Overhangs without Support Structure with Non-Planar Slicing on 3-axis Printer . https://xyzdims.com/2021/03/03/3d-printing-90-overhangs-without-support-structure-with-non-planar-slicing-on-3-axis-printer/. [Online; accessed 4-April-2023].

[26] Huaishu Peng, Jimmy Briggs, Cheng-Yao Wang, Kevin Guo, Joseph Kider, Stefanie Mueller, Patrick Baudisch, and François Guimbretière. 2018. RoMA: Interactive Fabrication with Augmented Reality and a Robotic 3D Printer. 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/10.1145/3173574.3174153

[27] Jiankang Qiu, Lifang Wu, and Yuxin Mao. 2015. A Novel Supporting Structure Generation Scheme to 3D Printing. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service* (Zhangjiajie, Hunan, China) *(ICIMCS '15)*. Association for Computing Machinery, New York, NY, USA, Article 69, 4 pages. https://doi.org/10.1145/2808492.2808561

[28] Tim Reiner and Sylvain Lefebvre. 2016. Interactive Modeling of Support-free Shapes for Fabrication. In *EG 2016 - Short Papers*, T. Bashford-Rogers and L. P. Santos (Eds.). The Eurographics Association. https://doi.org/10.2312/egsh.20161006

[29] Prusa Research. 2016. PrusaSlicer: An open-source, feature-rich, frequently updated tool that contains everything you need to export the perfect print files for your 3D printer. https://github.com/prusa3d/PrusaSlicer/. [Online; accessed 27-July-2023].

[30] Michael L. Rivera, Melissa Moukperian, Daniel Ashbrook, Jennifer Mankoff, and Scott E. Hudson. 2017. Stretching the Bounds of 3D Printing with Embedded Textiles. 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, 497–508. https://doi.org/10.1145/3025453.3025460

[31] Ryan Schmidt and Nobuyuki Umetani. 2014. Branching Support Structures for 3D Printing. In *ACM SIGGRAPH 2014 Studio* (Vancouver, Canada) *(SIGGRAPH '14)*. Association for Computing Machinery, New York, NY, USA, Article 9, 1 pages. https://doi.org/10.1145/2619195.2656293

[32] G. Strano, L. Hao, R. M. Everson, and K. E. Evans. 2013. A new approach to the design and optimisation of support structures in additive manufacturing. *The International Journal of Advanced Manufacturing Technology* 66, 9 (June 2013), 1247–1254. https://doi.org/10.1007/s00170-012-4403-x

[33] Alexander Teibrich, Stefanie Mueller, François Guimbretière, Robert Kovacs, Stefan Neubert, and Patrick Baudisch. 2015. Patching Physical Objects. 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, 83–91. https://doi.org/10.1145/2807442.2807467

[34] UltiMaker. 2014. UltiMaker Cura: free, easy-to-use 3D printing software trusted by millions of users. Fine-tune your 3D model with 400+ settings for the best slicing and printing results. https://ultimaker.com/software/ultimaker-cura/. [Online; accessed 27-July-2023].

[35] J. Vanek, J. A. G. Galicia, and B. Benes. 2014. Clever Support: Efficient Support Structure Generation for Digital Fabrication. *Computer Graphics Forum* 33, 5 (2014), 117–125. https://doi.org/10.1111/cgf.12437 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12437

[36] J. Vanek, J. A. Garcia Galicia, B. Benes, R. Měch, N. Carr, O. Stava, and G. S. Miller. 2014. PackMerger: A 3D Print Volume Optimizer. *Computer Graphics Forum* 33, 6 (2014), 322–332. https://doi.org/10.1111/cgf.12353 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12353.

[37] Ludwig Wilhelm Wall, Alec Jacobson, Daniel Vogel, and Oliver Schneider. 2021. Scrappy: Using Scrap Material as Infill to Make Fabrication More Sustainable. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 665, 12 pages. https://doi.org/10.1145/3411764.3445187

[38] 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.

[39] Yang Xu, Ziqi Wang, Siyu Gong, and Yong Chen. 2021. Reusable support for additive manufacturing. *Additive Manufacturing* 39 (March 2021), 101840. https://doi.org/10.1016/j.addma.2021.101840

[40] I. E. Yiğit, M. Isa, and I. Lazoglu. 2018. ADDITIVE MANUFACTURING WITH MODULAR SUPPORT STRUCTURES.

A MODEL TEST SET

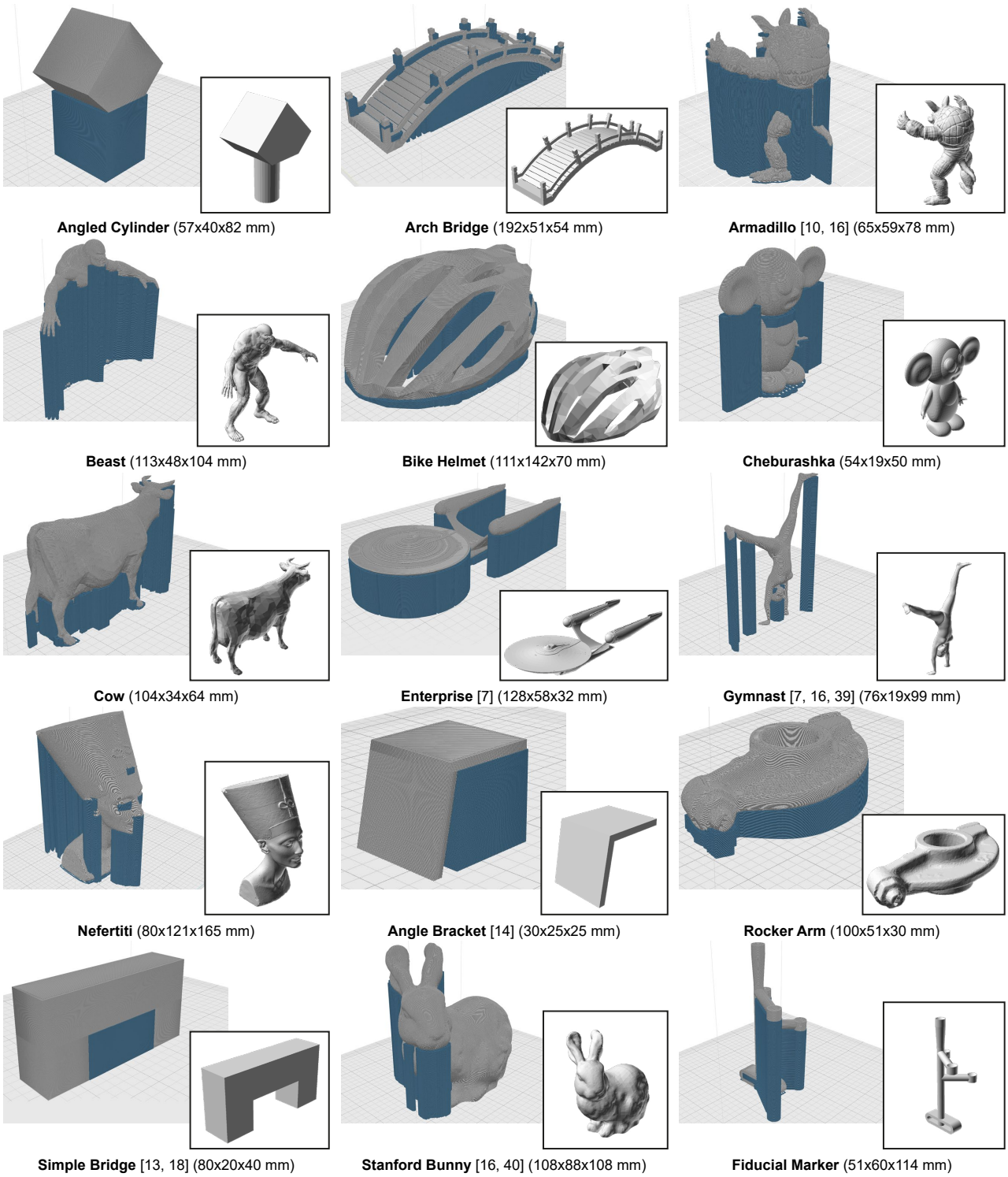


Figure A.1: The 15 models that were used for testing the system.

B TESTING PRINTING ON OBJECTS

This section reports on six tests related to printing on top of replacement objects: adhesion to different materials and their re-usability; filament type; tolerances; effects of extended idle periods; support structure type; and tipping resistance of replacement objects. All tests use a standard 0.4 mm nozzle and recommended print settings for the filament type used.

B.1 Object Surface Adhesion

The approach relies on the ability to print on top of foreign support replacement objects. We conducted tests to examine filament adhesion with common replacement object materials. A 1 mm thick (5 layer) 2*cm* square of standard PLA was printed on top of 14 different replacement objects with different material properties. Additionally, 1 to 3 mm of support was printed in between the square and the replacement objects. The level of adhesion was judged by visual inspection and light forces applied by hand. In case of poor adhesion, the print was repeated when object surface was covered in blue tape and glue stick applied. The printed square was removed and the material surface inspected for degradation to test for re-usability.

B.1.1 Results. Table B.1 provides details for the 14 tests. The results show good adhesion when printing on objects made of PLA, PVC, Medium Density Fibreboard (MDF), high density foam, medium density foam, and cardboard. Tape or glue is required when printing on ABS, wood, metal, and paper book covers. Low density foam is not recommended for replacement objects due to heat warping.

Most object surfaces exhibited little or no degradation caused by printing suggesting they can be re-used multiple times. Cardboard, high density foam, and medium density foam had noticeable marring, but maintained structural and dimensional integrity suggesting they could be re-used a few times. A single layer of blue masking tape would reduce or eliminate marring for these objects, and we found the same tape surface could be reused multiple times.

B.2 Adhesion with Different Filaments

We test printing different filament materials on top of PLA (a common material used for replacement objects) and blue masking tape (which can be applied to any object). A 1 mm thick (5 layer) 1.8*cm* square and 3 mm of support (15 layers) were printed on a PLA calibration cube using PLA, PETG, ABS, and TPU.

B.2.1 Results. Figure B.1 shows results. TPU, PETG, and PLA have strong adhesion. For ABS, adding tape or glue is recommended. In principle, covering the top of an object with tape or glue should enable printing with most types of filament (e.g. soluble PVA, Nylon).

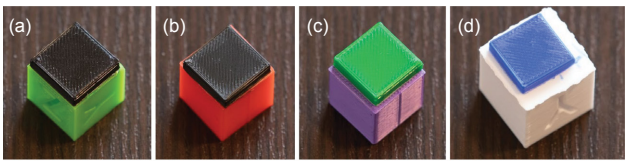


Figure B.1: Adhesion test: (a) TPU; (b) ABS; (c) PETG; (d) PLA.

B.3 Object Measurement Precision

Printed holders ensure that replacement objects are securely held in position. Fit depends on the size of the objects and the holders, so objects need to be measured with a reasonable level of precision. Similarly, objects should have the correct height to be able to print on top of them reliably. We adjusted the "measured size" of a 4x4 LEGO block to cause deliberate "measurement errors" and attempted inserting the block into the resulting printed holder and then attempted to print on top of the block. The error was increased in steps of 0.05 mm at a time.

B.3.1 Results. We found that a ±0.25 mm error range around the true width and length allowed for a secure fit. If smaller, the object can no longer fit inside the printed holder. If larger, insertion will no longer have a noticeable snap. Eventually, lightweight/tall objects may move or start to tip. Only one dimension has to be in the 0.5 wide mm "snapping range" for the tactile insertion feedback. If the holder is overall too large, or was accidentally knocked off the build plate, it may still be used as a positioning guideline, while double sided tape ensures that the replacement object stays in the correct position. For the height of the object, printing was reliable between -0.2 mm to +0.3 mm error compared to the actual height of the object. Below this range, a collision may be detected by the printer, or the nozzle may dig into material that can melt. At +0.35 mm, printing succeeded but adhesion started to become spotty, and for larger errors adhesion will eventually fail fully. This range may depend slightly on printing layer height, which was set to 0.2 mm for the tests. Based on this range of acceptable heights, we were able to print on solid objects with noticeably angled top surfaces, such as a rough-cut piece of wood.

Flexible materials (such as cardboard boxes or foam plates) may allow successful printing even for very imprecise measurements (e.g. measurement errors greater than 1mm), such as when squeezing the material with the calipers during measurement. As the print head is lowered onto the support replacement object from above, the print head can compress such objects without triggering collision detection. Many slicers enable this movement pattern, as they lift the print head during retractions (i.e., when switching between printing locations). We found that top surfaces with extreme deviations that exceed the lifting height (up to several millimetres) still do not cause issues if the deformations peak near the centre and the surface is sufficiently pliable. This is because the print head gradually pushes the object surface down as it prints, essentially undoing deformations temporarily.

B.4 Stability with Build Plate Movement

As replacement objects are located on the build plate, moving the build plate could potentially tip replacement objects over. We placed an otherwise unsupported 4x4 stack of LEGO into a printed holder. The stack was more than twice as high as it was wide. The build plate was then moved back and forth at the maximum speed allowed by the printer firmware. The length of each movement was reduced consecutively each time after switching the movement direction.

B.4.1 Results. The stack remained in its holder without tipping. Figure B.2 shows the test setup.

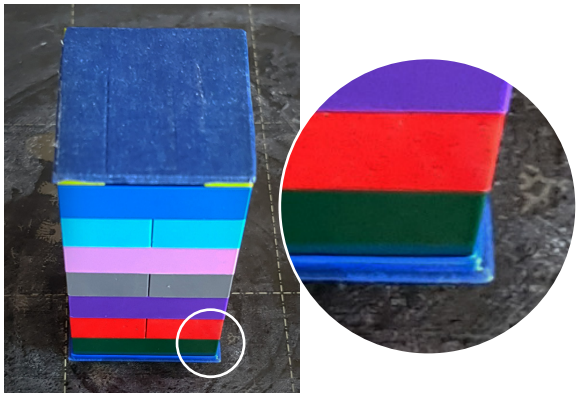


Figure B.2: Stability test setup: a tall 4x4 stack of bricks held in place by a printed holder.

B.5 Finish Quality after Long Pauses

If no user is available to perform an interaction, the printer may idle for extended periods of time. We compare the same print job when a pause is handled immediately and when a pause is handled after an 8-hour idle period.

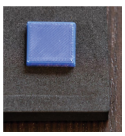
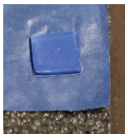
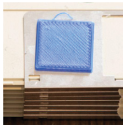
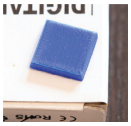
B.5.1 Results. Figure B.3 shows results. For standard PLA printing processes, rapid and extended pauses (8 hours) seem to have no noticeable effect on print quality. For processes that require for example a heated printing chamber, pausing parameters may need to be customized, for example to ensure the temperature equalizes within the chamber after reheating when continuing after an extended pause. Heating could be maintained for a while during the pause to accommodate shorter pauses more readily. Any change in geometry may alter the printing direction of outer perimeters, which may be visible at certain reflection angles.

B.6 Compatibility with Support Structure Types

The approach should work with any type of support structure, as long as printing on top of the chosen support replacement objects is possible. The modified slicer currently only supports generating one type of support structure. We tested extreme slicing settings of this type of support generation to gauge whether the method is

Table B.1: Object surface adhesion tests.

Material	Adhesion	Degradation	Notes	
PLA	Strong	Minor surface marring (melting)	Using support objects that match the printing material results in strong adhesion, but can usually still be separated easily by hand, much like regular printed supports.	
ABS	Glue/tape required	Minor surface marring (melting)	Large assemblies of toy bricks should be tightened before use. In case of the stacks used, simply pressing down on the stack with some force is sufficient.	
PVC (Floppy Disc)	Strong	Little to none	Surface ridges and paper labels have no noticeable effect on adhesion, but the metal shutter may require glue/tape.	
Medium Density Fiberboard (MDF)	Strong	Little to none		
Hardwood (Walnut)	Recommend Glue/tape	Little to none	The level of adhesion depends on the type of wood and its surface finish, ranging from strong to poor.	
Softwood (Pine)	Recommend Glue/tape	Little to none	The level of adhesion depends on the type of wood and its surface finish, ranging from strong to poor.	
High Density Foam (isolation board)	Strong	Surface marring (melting)	Some compression during printing, some minor melting due to print.	

Material	Adhesion	Degradation	Notes	
Medium Density Foam	Strong	Surface marring (melting)	Some compression during printing, some minor melting due to print.	
Low Density Foam	Warping	Major melting	Even with tape applied, the foam below may melt, which can cause warping to the material above. For foam with large pores, the tape itself may not adhere well to the foam.	
Metal	Glue/tape required	Little to none		
Cardboard (box)	Strong	Surface marring	Some compression during printing. Some delamination after removing printed material.	
Hardcover Book	Recommend Glue/tape	Surface marring (delamination)	To protect the cover from surface degradation, blue tape should be applied. The level of adhesion otherwise may also depend on the cover material.	
Softcover Book	Recommend Glue/tape	Surface marring (delamination)	To protect the cover from surface degradation, blue tape should be applied. The level of adhesion otherwise may also depend on the cover material.	

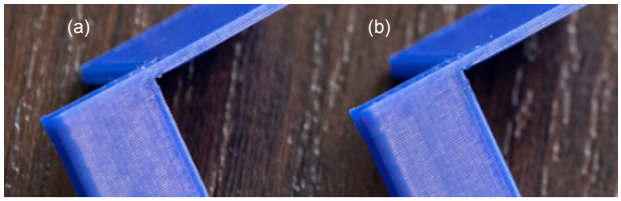


Figure B.3: Finish quality after long pauses: (a) resuming printing immediately; (b) resuming printing after 8 hours.

affected by different types of support. We printed on top of support replacement objects with varied support slicing settings: no support infill (i.e., printing support outlines only); 100% support infill (i.e., printing solid supports); and thin support towers of 1 to 3 mm width to emulate tree-style supports.

B.6.1 Results. Different support fill methods showed no impact on adhesion. When testing the 1 to 3 mm support towers, all 37 adhered strongly to the support replacement object.

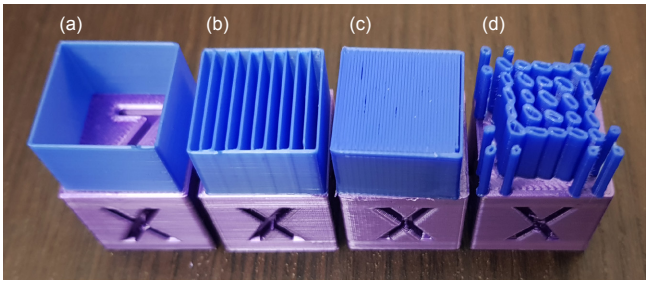


Figure B.4: Adhesion using different support settings: (a) no support infill; (b) default support infill; (c) 100% support infill; (d) thin tree-style supports.

C COMPARISON WITH OTHER SUPPORT GENERATION METHODS

Table C.1: *Support Material* in grams used by different slicer support generation methods and by our system.

Model	Prusa Grid	Prusa Snug	Prusa Organic	Cura Normal	Cura Tree	Kiri Snug	Min. Replaced	Max. Replaced
Angle Bracket	3.4	3.1	1.9	3.7	1.8	4.3	1.0	0.1
Gymnast	9.6	6.3	7.0	8.4	5.2	5.9	3.0	0.4
Cheburashka	3.5	1.9	3.7	2.4	2.3	2.0	1.1	0.3
Fiducial Marker	15.2	9.8	7.0	13.7	4.9	18.1	12.4	3.7
Stanford Bunny	21.1	14.4	13.3	14.5	8.3	13.8	10.5	3.8
Armadillo	20.5	15.6	15.1	20.3	11.9	24.2	12.4	7.6
Simple Bridge	3.9	3.7	2.4	4.1	2.6	5.1	2.7	1.6
Enterprise	21.7	17.7	17.5	18.1	14.4	23.7	15.8	7.9
Nefertiti	96.8	70.8	40.7	33.8	23.3	20.9	13.9	7.0
Beast	30.2	22.2	16.4	27.4	11.6	38.0	27.1	13.3
Cow	13.4	10.1	9.2	11.2	6.3	12.7	9.2	4.7
Angled Cylinder	19.0	17.8	11.1	9.5	4.4	13.3	10.6	5.1
Arch Bridge	55.8	44.6	47.3	51.7	-	87.0	57.9	41.2
Rocker Arm	7.7	6.1	7.2	6.0	4.3	8.3	5.5	4.0
Bike Helmet	66.2	50.9	47.7	54.3	32.5	60.0	42.3	35.5

Table C.2: *Support Printing Time* in minutes used by different slicer support generation methods and by our system.

Model	Prusa Grid	Prusa Snug	Prusa Organic	Cura Normal	Cura Tree	Kiri Snug	Min. Replaced	Max. Replaced
Angle Bracket	9.0	8.0	21.0	16.0	15.0	57.3	21.4	1.7
Gymnast	73.0	45.0	60.0	66.0	37.0	162.5	82.6	7.4
Cheburashka	30.0	17.0	39.0	21.0	17.0	53.4	28.2	7.5
Fiducial Marker	104.0	58.0	94.0	65.0	27.0	324.4	224.7	79.0
Stanford Bunny	214.0	152.0	166.0	104.0	65.0	264.3	196.6	75.6
Armadillo	199.0	161.0	170.0	124.0	86.0	543.4	254.8	174.6
Simple Bridge	17.0	16.0	37.0	18.0	20.0	65.0	50.1	33.9
Enterprise	183.0	143.0	267.0	106.0	119.0	352.5	255.2	130.6
Nefertiti	840.0	626.0	437.0	293.0	175.0	467.2	323.4	160.4
Beast	259.0	188.0	186.0	163.0	82.0	845.4	517.5	261.9
Cow	123.0	93.0	129.0	71.0	57.0	255.1	189.2	105.4
Angled Cylinder	86.0	89.0	134.0	51.0	32.0	205.2	172.0	108.4
Arch Bridge	424.0	343.0	704.0	260.0	-	1610.1	938.7	751.9
Rocker Arm	71.0	55.0	113.0	144.0	153.0	124.9	92.4	71.9
Bike Helmet	628.0	492.0	621.0	346.0	258.0	1199.3	831.3	696.5

Some slicers offer a variety of support generation methods. We found the kiri:moto method we used in our submission is most similar to Prusa Snug. Prusa Organic saved an average of 30.9% material compared to Prusa Grid, but only saved 7.5% compared to Snug. In 4 models, Organic used 6% to 96% more material than Snug. Prusa Organic required an average of 18% more time to print compared to Snug. Cura Tree saved an average of 39.5% compared to Cura Normal, but the Arch Bridge model failed to slice with Tree supports.