SliceHub: Augmenting Shared 3D Model Repositories with Slicing Results for 3D Printing

Faraz Faruqi
MIT CSAIL
Cambridge, MA, USA
ffaruqi@mit.edu

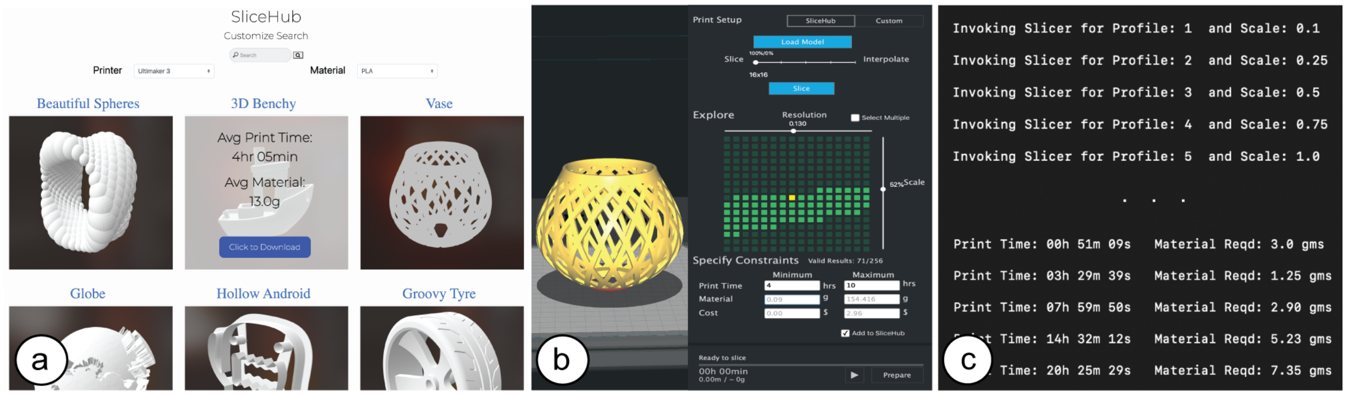
Kenneth Friedman
MIT CSAIL
Cambridge, MA, USA
ksf@mit.edu

Leon Cheng
MIT CSAIL
Cambridge, MA, USA
leonc@mit.edu

Michael Wessely
MIT CSAIL
Cambridge, MA, USA
wessely@mit.edu

Sriram Subramanian
University College London
London, United Kingdom
s.subramanian@ucl.ac.uk

Stefanie Mueller
MIT CSAIL
Cambridge, MA, USA
stefanie.mueller@mit.edu



```
Invoking Slicer for Profile: 1  and Scale: 0.1
Invoking Slicer for Profile: 2  and Scale: 0.25
Invoking Slicer for Profile: 3  and Scale: 0.5
Invoking Slicer for Profile: 4  and Scale: 0.75
Invoking Slicer for Profile: 5  and Scale: 1.0

                    . . .

Print Time: 00h 51m 09s   Material Reqd: 3.0 gms
Print Time: 03h 29m 39s   Material Reqd: 1.25 gms
Print Time: 07h 59m 50s   Material Reqd: 2.90 gms
   t Time: 14h 32m 12s   Material Reqd: 5.23 gms
     Time: 20h 25m 29s   Material Reqd: 7.35 gms
```

Figure 1: SliceHub's integrated system: (a) repository with slicing results, (b) user interface for exploring trade-offs between different print resolution profiles and model scales, (c) infrastructure for slicing and interpolation to generate new slicing results—the results can then be added to the repository further extending the available options.

ABSTRACT

In this paper, we explore how to augment shared 3D model repositories, such as *Thingiverse*, with slicing results that are readily available to all users. By having print time and material consumption for different print resolution profiles and model scales available in real-time, users are able to explore different slicing configurations efficiently to find the one that best fits their time and material constraints. To prototype this idea, we build a system called SliceHub, which consists of three components: (1) a repository with an evolving database of 3D models, for which we store the print time and material consumption for various print resolution profiles and model scales, (2) a user interface integrated into an existing slicer that allows users to explore the slicing information from the 3D models, and (3) a computational infrastructure to quickly generate new slicing results, either through parallel slicing of multiple print resolution profiles and model scales or through interpolation. We motivate our work with a formative study of the challenges faced by users of existing slicers and provide a technical evaluation of the SliceHub system.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**;

KEYWORDS

3D printing; slicing; online repositories.

1 INTRODUCTION

3D printing a digital model typically requires a user to make certain choices: users have to select the 3D printer and filament they would like to use, choose a print resolution profile, and scale the model to the desired size. This is done in a *slicer*, a program that uses the selected print settings to convert the 3D model into a set of machine instructions that the 3D printer can execute.

The required print time and material consumption are strongly influenced by these print settings. For instance, an object printed with a higher print resolution profile requires more layers and thus more print time than the same object printed with a lower print resolution profile that requires fewer layers. Similarly, the scale of the model is an important factor for print time and material consumption since larger models require more layers and more material per layer. Since 3D printing is slow and can take many hours even for hand-sized objects, users often find themselves in a situation where they have to trade-off between different print resolution profiles and model scales [Hudson et al. 2016; Ludwig et al. 2014].

Unfortunately, today, information on the required print time and material consumption for different print resolution profiles and model scales is not readily available to users. Since slicers first have to slice the model before they can display the expected print time and material consumption, users have to wait after each change of the print resolution profile and model scale. Since a single slicing process can take up to several minutes for complex 3D models, it makes the exploration of suitable print resolution profiles and model scales a time-consuming process.

In this paper, we present an integrated system that enables users to explore different print settings in real-time to find the trade-off between print resolution profile, model scale, print time and material consumption that best fits their needs. Our idea is to augment 3D model repositories, such as *Thingiverse*, where the same 3D model is downloaded and printed by hundreds of users [Makerbot 2015], with slicing results that have to be computed only once and subsequently are stored and made available to all users of the repository. This allows users to explore multiple print settings simultaneously in real-time without having to wait for slicing to finish.

While our long-term vision is to integrate the slicing results into existing 3D model platforms, such as *Thingiverse,* these commercial platforms are hard to extend with functionality due to limitations with their APIs. We therefore built the SliceHub repository to be able to prototype and study a system that stores and reuses slicing results. The repository contains for each 3D model, the print time and material consumption for different print resolution profiles and model scales, which users can download through each 3D model's page.

While current slicers restrict users to explore print resolution profile and model scales one at a time, having the print time and material consumption for different settings available in real-time allows users to explore multiple options simultaneously. To support this exploration, we developed a user interface embedded within an existing slicer that provides users with an overview of the print time and material consumption resulting from different print resolution profiles and model scales. The user interface also supports users in filtering results by their time and material constraints and then displays only those print resolution profiles and model scales that are within the time and material the user has available.

Finally, while SliceHub will contain increasing amounts of data over time, there will always be cases where either the desired print resolution profile and model scale have not yet been sliced or the 3D model is new to the shared repository and no data is available yet. For these cases, SliceHub provides infrastructure to either interpolate or slice the missing print resolution profiles and model scales in parallel. For parallel slicing, SliceHub interfaces with an existing slicing engine and instantiates multiple slicing processes with different print resolution profiles and model scales simultaneously. To make this scalable, SliceHub uses a cloud computing infrastructure that can return up to 1000 slicing results in parallel. SliceHub's parallel slicing infrastructure can thus return the slicing results for several hundred print resolution profiles and model scales combinations in the same time it would take to return results for only one.

In summary, we contribute:

- an augmented repository that stores for each 3D model the required print time and material consumption for different print resolution profiles and model scales, which allows future users to re-use the results in real-time;
- a user interface integrated into an existing slicer that supports exploration of multiple print resolution profiles and model scales at once, including functionality to narrow down available options by filtering based on time and material constraints;
- infrastructure to efficiently generate data for missing print resolution profiles and model scales on the shared repository either through interpolation or by slicing multiple print resolution profiles and model scales in parallel.

2 RELATED WORK

Our work is related to support tools for 3D printing, algorithms that improve the slicing process, research on finding speed-fidelity trade-offs, and data augmentations for 3D model repositories.

2.1 Supporting Users with 3D Printing

[Hudson et al. 2016] were among the first to provide an in-depth analysis of the issues novice users ('casual makers') encounter when using 3D printers including the creation of the 3D model itself and the subsequent slicing and fabrication process. Recent studies, including [Annett et al. 2019; Dew et al. 2019; Norouzi et al. 2021] analyze fabrication workflows for users in various settings, ranging from makerspaces and fabrication studios [Annett et al. 2019] to summer programs for young people [Norouzi et al. 2021]. In recent years, much of HCI research has focused on supporting users in the modeling process: Measurement Uncertainty [Kim et al. 2017], for instance, compensates for users' measurement errors in the initial modeling phase; RetroFab [Ramakers et al. 2016] supports users in creating 3D models that add functionality to existing devices; and Lamello [Savage et al. 2015] enables users to create 3D designs with interactive input components. While there is a large body of work in HCI on improving 3D modeling tools as part of maker software [Schmidt and Singh 2010], only few research projects have focused on the later steps when a design is prepared for fabrication. Most existing work on helping users prepare an object for fabrication has focused on laser cutting (VisiCut [Oster 2011], PacCam [Saakes et al. 2013], Fabricaide [Sethapakdi et al. 2021]). When explored in the context of 3D printing existing work created new workflows not based of conventional tools. For instance, Scrappy [Wall et al. 2021] suggests objects to be inserted into the 3D model to replace infill material. Our work, in contrast, builds onto slicing tools already in use today and can thus be directly integrated into the existing fabrication pipeline.

2.2 Improved Slicing Algorithms

Over the last years, several research projects in computer graphics have investigated how to improve slicing algorithms to improve print quality (Connected Fermat Spirals [Zhao et al. 2016], CurviSlicer [Etienne et al. 2019]), generate faster printing supports [Dumas et al. 2014; Schmidt and Umetani 2014; Vanek et al. 2014], create better packing layouts on the print platform (Chopper [Lu et al. 2014], Dapper [Chen et al. 2015]), improve visual quality of the print result (Perceptual Support [Zhang et al. 2015], Saliency Preserving Slicing [Wang et al. 2015]), and increase durability (Cross-Section Analysis [Umetani and Schmidt 2013], Orthogonal Slicing [Hildebrand et al. 2013], Build-to-last [Lu et al. 2014]). Finally, researchers have also investigated how to create slicing algorithms that create new object properties, such as infills distributed in a way that makes an otherwise unbalanced object stand [Prévost et al. 2013] or spin [Bächer et al. 2014] after fabrication. Rather than inventing new algorithms for slicers, our work investigates how to facilitate the exploration of slicing results for a 3D model.

2.3 Exploring Fabrication Trade-Offs

Supporting users in finding the best trade-off between speed and fidelity has a long history in HCI research and has recently also been explored in the context of 3D printing. Low-fidelity fabrication techniques [Mueller et al. 2015], such as WirePrint [Mueller et al. 2014] and Platener [Beyer et al. 2015], for instance, allow users to trade-off model fidelity with print speed during the fabrication stage. During the modeling stage, SPATA [Weichel et al. 2015] visualizes where support material will be generated, which allows designers to come to an informed decision about either modifying the design to avoid supports or spending the extra time on printing them. Our work also supports users in finding trade-offs between different options but applies it to the context of finding the most suitable print profile resolutions and model scales for 3D printing.

2.4 Augmentations to 3D Model Repositories

Several researchers explored how to augment existing 3D model repositories with additional data. Grafter [Roumen et al. 2018], for instance, annotates mechanisms embedded in shared 3D models resulting in a model-graph of mechanisms that can be used for remixing mechanical parts. ShapeNet [Chang et al. 2015] annotates 3D models with a rich set of geometric and language annotations that enable researchers to filter models via attributes like part decompositions and word taxonomies. Thingi10K [Zhou and Jacobson 2016], a dataset of 10,000 3D models, augments 3D models with several mesh complexity and quality metrics, such as the number of vertices and if the mesh is closed or self-intersects, which provides researchers with data that reflects real-world imperfect meshes instead of the clean data often used by researchers. Thingi-Pano [Berman and Quek 2020] is a large dataset, which includes 3-axis panoramic depth-map projections of 3D models together with images and meta-data, which can be used for machine learning. Finally, HowDIY [Berman et al. 2021] provides a dataset and platform that facilitates the exploration of various online resources for 3D printing. In contrast to these datasets and platforms, Slice-Hub augments 3D models with their slicing results (print time, material consumption) for a variety of print profile resolutions and model scales to facilitate the exploration of slicing settings.

3 FORMATIVE STUDY

To better understand the challenges regarding slicing and the strategies employed by users to achieve various slicing goals for 3D printing, we conducted interviews with 18 participants (6f/12m). Participants had used 3D printers for a diverse range of applications: such as 3D printing models for personal use, printing prototypes for research purposes, and for commercial and architectural use cases. We focused on topics such as the issues faced by participants when using slicers, types of print constraints faced while printing, experiences when exploring models on online 3D model repositories, and ideas for improvements. We identified several recurring break-downs.

Long Wait Times: Participants expressed frustration with the slow feedback of the slicer. Since the slicer by default re-slices the model after every print resolution profile and model scale change, participants stated that they often have to wait for the slicer even after making only minor changes. Participant P6, for instance, complained about the *"long calculation times"* and P8 stated *"every time I choose a profile, it needs a lot of time to calculate the print time."* Participants wondered if the wait time could be shortened. For instance, P6 stated *"It would be cool if instead of calculating the entire slice operation after every change, the program would only updated time/material use and then just do a full slice calculation once I choose to save."* P8 added that estimates for the print time and material consumption may be enough for initial exploration: *"I actually don't need the very precise time, I just need to know [...] it will take at least 10 hours."*

Lack of Information about how Print Resolution and Model Scale relate to Print Time and Material Consumption: Participants expressed frustration that there was no information on how different print resolution profile and model scale changes would impact the print time and material consumption. P1 recounted that, *"[there was] no intuition or indication of what would affect time and material amount and by how much. Had to do ridiculous binary searching to find optimal value."* Multiple participants (P2, P5, P8, P10, and P11) indicated a preference towards having a recommendation or optimization system that would "*compute the settings based on some desired quality heuristics*" (P10). P11 suggested that it would be useful "*if I can input a target print time, or filament usage, and it can recommend settings for me, then I have a baseline to start with.*"

Difficulty in Satisfying Print Constraints: Most 3D printers used by participants were shared, and when faced with a deadline or during rapid prototyping, participants (P8, P10, P12) worked under stressful time and material constraints. P8 recounted: *"I really, really needed to get that model done within 20 hours or else I [wasn't] going to make it to the deadline; I had to to find another way. [...] I spent around 20 minutes finding the correct solution so that I can get the thing done."* P10 added: *"[if] that means sacrificing a lot of resolution to get it, it's better than not having a product in the end."*

Participants (P7, P8, P10, P12) also expressed that exploring different print resolution profiles and model scales took them a large number of iterations before arriving at a suitable solution. They shared ideas for improving the process of finding suitable print settings under time or material constraints. P7, for instance, suggested that *"If the slicing program is really smart, then if I can type my purpose [goal], it may suggest some things"* and P8: *"[if] there's a bar, and it will tell me if you scale to this, how much material is used."*

Lack of Print Information on Online Repositories: Almost all participants had used 3D model repositories, such as *Thingiverse*, when looking for 3D models. But participants often had to go back-and-forth between the online repository and the slicer because the repository did not include any print information. P10 stated *"the website doesn't give you a lot of the things that the slicer software does."* Participants (P4, P5, P7, P8, P10, P11) mentioned that including print-specific information for models on these shared repositories would aid in determining if the model is suitable. P4, for instance, said *"having more information on the platform for a specific model before having to download it and testing it out would be great."* P5 added: *"you could have different size[s], so you could say with this particular size, this is the printing time [..]"*

To address the problem of long wait times, SliceHub's integrated system makes slicing results, i.e. the print time and material consumption for different print resolution profiles and model scales, available in real-time by storing slicing results and sharing them among all users. By making the print time and material amount of multiple print profile resolutions and model scales simultaneously visible in a user interface, it supports users in gaining an understanding of how they relate to each other. To support users in finding print profile resolutions and model scales that match a user's time and material constraints, SliceHub provides filter functionality that narrows down the options to only those that match the time and material the user has available. Finally, SliceHub embeds basic print information, such as the average print time and material consumption for a 3D model, on the 3D model repository page, which allows users to take this information into account already during the search for a suitable 3D model.

4 SLICEHUB

SliceHub is an integrated system that allows users to efficiently explore slicing results for a 3D model. It consists of three components: (1) a repository of 3D models that contains for each 3D model the slicing results, i.e. print time and material consumption, for various print resolution profiles and model scales; (2) a user interface integrated into an existing slicer, Ultimaker Cura [Ultimaker 2020], that supports simultaneous exploration of multiple print resolution profiles and model scales as well as filtering based on time- and material constraints; (3) a computational infrastructure to efficiently generate data for missing print resolution profiles and model scales either by estimating slicing results through interpolation or by slicing multiple print resolution profiles and model scales in parallel on an external cloud-based system. Together, these components allow users to explore trade-offs between different print resolution profiles and model scales with respect to the print time and material consumption for 3D printing a model.

Note that SliceHub's components work based on print resolution profiles, i.e. a set of slicer settings (layer resolution, print speed, infill density etc.) that taken together print the model at a certain resolution. This is similar to existing slicers which offer predefined print resolution profiles (e.g. 'fast(0.2mm)', 'fine(0.06mm)'). Varying individual slicer settings, when not done carefully by an expert user, can lead to invalid prints. We thus exclude this option from SliceHub and only use predefined print resolution profiles whose settings were calibrated by the manufacturer of the 3D printer.

In the next sections, we explain each of the three main components of SliceHub in more detail.

4.1 Repository of Slicing Data

SliceHub's shared repository stores for each 3D model the slicing results for different print resolution profiles and model scales when printed on different 3D printers and with different materials.

Content Stored in the Repository: For each 3D model, SliceHub stores the 3D model geometry (.stl) and a meta-data file (.json) that contains for each print resolution profile and model scale the print time and material consumption for fabricating the 3D model. This allows the user to explore various print settings in real-time to find the trade-off between print time, material consumption, print resolution and model scale that best fits their needs. Storing print time and material consumption only requires little storage space (12 KB in total for one model with 16 different print resolution profiles and 16 different model scales, i.e. 256 combinations), which is a requirement to make the repository scaleable. We achieve this small storage overhead by discarding the print path (.gcode) since it does not contain information required to make a decision on the trade-offs discussed above and, additionally, an average .gcode file has an approximate file size of 10 MB per print resolution profile and model scale. Storing 256 .gcode files, one for each combination of the 16 print profile resolutions and 16 model scales, would require 2.5 GB on average and, thus, does not scale for large repositories. Instead, we only store the meta-data in a .json file. At 12 KB per model, even large data bases with thousands of models could store and process this additional data.

Finding Content in the Repository: The repository can be accessed through the SliceHub website and displays the 3D models on its main page (Figure 2). The user starts by selecting the 3D printer and material they want to use. Next, the user can start a query for specific models (e.g., "Mobius"). After clicking on the "Search" Button, SliceHub lists the most relevant models to the query in a result matrix. To provide the user with initial information on the print time and material consumption for a 3D model, SliceHub shows this information at a medium print resolution profile (0.15mm) and at 100% model scale when the user hovers over a 3D model's thumbnail. In case no slicing results are available yet, the print time and material consumption are displayed as 'not available'. Clicking on the 3D model's thumbnail downloads a .zip archive containing the 3D model geometry (.stl) and a corresponding meta-data file (.json) that includes the slicing results for different print resolution profiles and model scales. As mentioned in the introduction, we envision that in the future the data from the SliceHub repository will be integrated with existing 3D model platforms, such as *Thingiverse*, with the print time and material consumption being made available during model search and on a 3D model's subpage, and the meta-data file with slicing results added for download with the 3D model's .stl file.

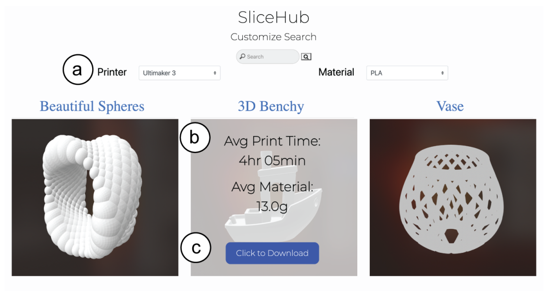


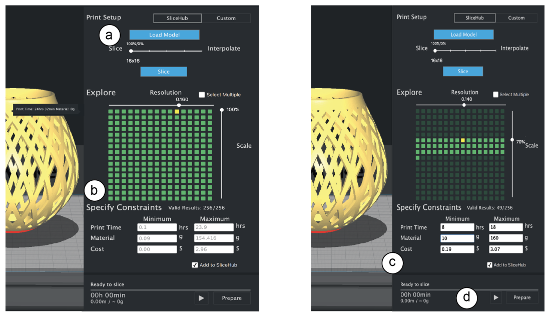
Figure 2: SliceHub repository: (a) select 3D printer and material, (b) for each model average print time and material consumption are shown, (c) choose model and download .zip file (contains model geometry (.stl) and slicing results in meta-data file (.json)).

Adding Content to the Repository: SliceHub's repository can be populated with data in two ways. First, users have the option to 'opt-in' for sharing their slicing results whenever they use SliceHub's parallel slicing infrastructure to generate new results (see Section 4.3). Second, SliceHub can generate slicing results itself using its cloud compute service and then subsequently add the results to the repository. Content can be added either to an existing model by providing slicing results for print resolution profiles and model scales that have no data yet or by contributing a new model that does not yet exist on the shared repository and adding an initial set of slicing results to it (see section '4.2 User Interface' for how users can add models). SliceHub identifies if a model already exists in the database or is new using the model's unique identifier (similar to *Thingiverse*), which is also encoded in the meta-data file. Thus, once the slicing results are available in the repository, they can be reused by future users without the need to re-compute them.

4.2 User Interface for Exploring Trade-Offs

After downloading a 3D model and its meta-data file from the repository, users can load the content into the SliceHub user interface, which is integrated into an existing slicer (i.e., Ultimaker Cura). The user interface allows users to compare multiple print resolution profiles and model scales simultaneously, which facilitates the exploration of trade-offs. In addition, the user interface enables users to filter print resolution profiles and model scales according to material and time constraints.

Loading Data into the User Interface The user interface takes as input the .zip file, which includes the 3D model geometry (.stl) and the meta-data file (.json) from the repository, which users can load using the 'Load Model' button (Figure 3a). Next, SliceHub extracts the slicing results for different print resolution profiles and model scales from the meta-data file and populates the user interface with the corresponding print times and material amounts (Figure 3b).



	Minimum		Maximum	
Print Time	0.1	hrs	23.9	hrs
Material	0.09	g	154.416	g
Cost	0.00	$	2.96	$

	Minimum		Maximum	
Print Time	8	hrs	18	hrs
Material	10	g	160	g
Cost	0.19	$	3.07	$

Figure 3: User Interface (Ultimaker Cura Plugin): (a) load .zip file (3D model (.stl) and meta-data (.json)), (b) UI updates to display print time and material consumption for each print profile resolution and model scale. (c) Adding lower and upper bounds for print time or material consumption shows only valid configurations. (d) Slicing the preferred configuration to generate the .gcode file for 3D printing.

Exploring Resolution and Scale Trade-Offs: The visualization of slicing results (Figure 3b) shows the print time and material amount for different print resolution profiles and model scales. Hovering over a slicing result displays the corresponding print time and material amount. The top left slicing result corresponds to the highest print resolution profile and largest model scale (0.06mm and 100%) whereas the bottom right slicing result corresponds to the lowest print resolution profile and smallest model scale (0.2mm and 10%). This allows users to compare different print profiles and model scales in real-time, which facilitates finding an appropriate trade-off. Since the focus of this paper is not on the particular visualization chosen to present the slicing results, we use a simple grid to represent the two-dimensional search space of print resolution profiles and model scales.

Print Time/Material Constraints: When 3D printing a model, users often work under time or material constraints as shown by our formative user study. For instance, users may not want to fabricate the model with a print resolution profile and model scale that requires printing overnight. SliceHub provides an option to narrow down the configurations to only those that print the model within the user's available time and material. Users can enter their constraints into the corresponding text boxes for lower and upper bounds (Figure 3c). To provide the user with information on the allowable range for these values, SliceHub initially populates the text boxes with the minimum and maximum values across all print resolution profiles and model scales for that particular 3D model (i.e., the print time and material consumption at the lowest resolution (0.2mm) and smallest scale (10%), and highest resolution (0.06mm) and original model scale (100%). As soon as the user enters the constraints, SliceHub updates the visualization, highlighting only those print resolution profiles and model scale combinations that are within the chosen bounds (Figure 3c). If users enter multiple constraints, such as both print time and material amount, both constraints are considered together.

Generating GCode: Once users settle on a print resolution profile and model scale combination by selecting it from the visualization, they can click the 'Slice' button (Figure 3d), which then slices the 3D model geometry (.stl) locally with the selected settings and afterwards loads the print path (.gcode) into the view. Users can then export the .gcode file using the existing slicer software functionality and then upload the .gcode file to their 3D printer to start the fabrication process.

4.3 Infrastructure for Parallel Slicing and Interpolation

SliceHub's infrastructure also addresses the need to generate new slicing results for print resolution profiles and model scales that have no data yet. To fill in the missing data, SliceHub can either interpolate the missing results, which can be done in real-time but is less accurate, or parallel slice the missing results, which is more accurate but requires additional time.

Interpolation of Missing Results: When slicing results are not yet available, SliceHub interpolates the missing results from the already sliced results. To indicate which print resolution profile and model scale combinations are sliced and which are interpolated, SliceHub marks the sliced and interpolated results with different symbols in the visualization (Figure 4). The interpolated values can be computed in real-time when the meta-data file (.json) is loaded. Thus, the interpolated results are immediately available for the user to explore. When the user hovers over one of these results, SliceHub shows the estimated print time and material consumption and also notes the prediction accuracy.

Parallel Slicing to Replace Interpolated Results: To compute the slicing results for one or more interpolated values, users select the interpolated slicing results and then hit the 'slice' button (Figure 4a). Alternatively, users can also use the 'interpolation percentage' slider (Figure 4b), which allows to decrease the number of interpolated results and thus increase the accuracy of the prediction. To speed up the process of generating new slicing results, SliceHub contains infrastructure to compute slicing results for multiple print resolution profiles and model scales combinations in parallel. Note that when all slicing processes run in parallel, the total time equals the time for a single slicing process, i.e., the time required for the setting that needs the most slicing time, which is the model at 100% scale printed with the highest print resolution profile. While the slicing results are being generated, the progress slider at the bottom of the interface updates to provide the user with an estimate of the remaining slicing time. Once the slicing results are available, the previously interpolated results update, i.e. convert their visual icon and on hover show the print time and material consumption for the sliced result rather than the interpolated estimate. When the 'Add to SliceHub' option in the user interface is checked (Figure 5c), all slicing results are added to the SliceHub repository for future

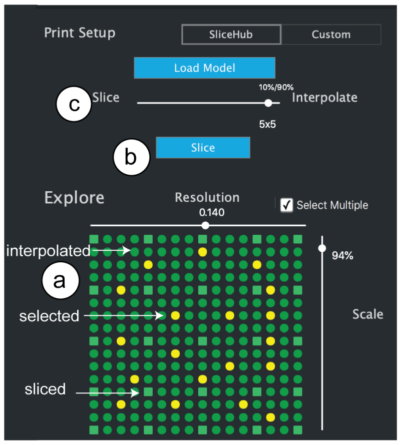


Figure 4: Adding slicing results: (a) by default, all missing results are interpolated, (b) selecting individual interpolated results and clicking 'slice' turns them into sliced result, (c) moving the 'slice/interpolation' slider increases the percentage of sliced results via parallel slicing.

re-use by other users (i.e., when SliceHub parallel slices in the cloud it stores the slicing results in the repository by updating the meta-data file of the model at the same time it returns the results to the user interface). However, users have the choice to override this setting by de-selecting the checkbox. The slicing results are then not stored on SliceHub and only saved as a new meta-data file on the user's local machine.

Periodically Adding Results to Popular Models: While we discussed in the previous sections how users can generate additional slicing results by selecting the desired print configurations in the user interface, SliceHub can also generate slicing results automatically through the repository. For instance, SliceHub has the capability to go over the models in its repository and identify those with missing slicing results. If SliceHub has processing capacity available, it generates the slicing results itself through its cloud compute service and adds them to the repository. If processing capacity is limited, SliceHub prioritizes popular models (by number of downloads) since the slicing results generated for those models will benefit the largest number of users on the shared repository. Once SliceHub added the missing results by slicing each print resolution profile and model scale combination once, they are available to all future users.

Adding a New Model: Slicing + Interpolation: To add a new 3D model, SliceHub uses a combination of slicing and interpolation. Users start by loading the 3D model geometry (.stl) into the SliceHub user interface using the 'Load Model' button (Figure 5a), which instantiates an empty visualization with no available slicing results yet. If users have the 'Add to SliceHub' option selected, SliceHub uploads the 3D model automatically to its cloud compute service and parallel slices it to generate 10% of the slicing results while interpolating the rest of the values (Figure 5b). SliceHub uses 10% since our technical evaluation shows that this fraction of sliced results leads to only a small interpolation error (Section 6.3). SliceHub then saves the 3D model together with a newly generated meta-data file that contains the slicing results (.json) on the server before returning the results to the user interface. After this step, the 3D model is available on the shared repository along with the generated slicing results. Users can decide to slice additional results, which are then added to the meta-data file on the server when SliceHub returns values from parallel slicing on its cloud compute service.

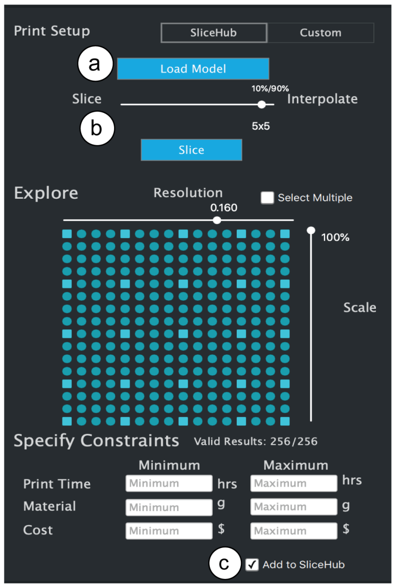


Figure 5: Adding a new model: (a) load model geometry (.stl), (b) SliceHub slices 10% of the print profile and model scale combinations and interpolates the rest, (c) SliceHub checks if users allowed sharing the model on the repository and then uploads the model and slicing results.

5 APPLICATION SCENARIOS

We next describe how SliceHub can be used for different applications.

Exploring Model Scale vs. Print Resolution Trade-Offs: In this scenario, an architect is short on time to print out a building model for an upcoming customer presentation. He loads the new 3D model into the SliceHub user interface and since the 3D model is not present in the repository, SliceHub proceeds to parallel slice 10% of the data and interpolate the rest (see Technical Evaluation 7.3). After looking at the SliceHub results for the highest print resolution profile and 100% model scale, the architect concludes that printing for 26 hours is beyond the time he has available since the customer presentation is in 16 hours. The architect wants to preserve all the fine features of the building design and thus would like to print the model at the highest print resolution (0.06mm) if possible. He thus decides to explore if scaling down the model would print within his time constraint. However, upon closer inspection he realizes that if he kept the highest print resolution, he would have to scale the model by more than 50%, which would make the model too small. He thus decides to slightly reduce print resolution to 0.1mm, which allows him to print the model at 80% scale within 15 hours and 27 minutes. The architect slices the building 3D model locally with the selected model scale, downloads the gcode file, and starts printing.

Exploring Models based on Print Times: In this application scenario, a user wants to 3D print a flower pot for a plant they want to give to their friend for their birthday, which is the next day. They have limited access to a shared 3D printer in the local library, and after looking at the open 3D printing times they see that for that particular day, there is only a 10 hour print window left. Since the user needs the flower pot within the day, she is searching for a model that prints within that time window. The user goes onto the SliceHub website and finds five different 3D models of flower pots, of which two have an average print time of under 10 hours. One of them looks more aesthetically pleasing, so the user decides to go with this model. She downloads the 3D model, opens it in the SliceHub user interface, generates the gcode for the default print settings that were shown on the repository, and sends the gcode file to the local library for printing.

Exploring how to Reduce Material Usage: In this application scenario, a maker is prototyping an interactive toy that has capacitive touch buttons across its surface. Since the conductive filament is expensive (Electrify Conductive Filament [Multi3D 2021], ca. $200 per filament roll) , the maker wants to make sure that the prints during prototype iteration are not using too much material. The maker loads the 3D model into SliceHub, which causes SliceHub to parallel slice 10% of the print profile resolutions and model scales. When exploring different model scales, the maker is surprised that scaling the model to 50% saves more than 80% of material since scaling down the model linearly leads to a cubic decrease in material volume. She thus decides that scaling the model to 75% is sufficient for saving a lot of material. After printing the first version and verifying that the toy works as expected, the maker prints the toy at full scale. Since the maker is happy to share the 3D model and slicing results with others, she selects the 'Add to SliceHub' checkbox, which makes the model together with the slicing results available on the repository for other users to reuse.

6 IMPLEMENTATION

SliceHub's components, i.e. the repository, the user interface, and the infrastructure for parallel slicing and interpolation can be implemented in conjunction with a variety of existing cloud services and slicers. For the purposes of building a prototype system, we are using Amazon Web Services for the cloud storage and cloud slicing, and the slicer Ultimaker Cura [Ultimaker 2020] for the user interface and slicer back-end. We provide details on the implementation of each of the components in the following sections.

6.1 Repository Backend and Frontend

There are three major components that make up the repository: (1) a website built with ThreeJS [Danchilla 2012], (2) a cloud storage that contains all the data for the 3D models and their slicing results

(AWS S3), and (3) cloud processing infrastructure for parallel slicing consisting of a Flask [Grinberg 2018] webserver connected to Amazon Web Services (AWS Lambda).

Data Structure: The cloud storage contains for every 3D model, the 3D model geometry (.stl file) and a meta-data file (.json) that contains for each print resolution profile and model scale the corresponding print time and material consumption. In addition, the cloud storage also saves a thumbnail for each model (generated from the .stl file using the numpy-stl library [Oliphant 2006]). The web server also contains a global meta-data file (.json) that contains links to each 3D model folder on the cloud storage.

Data Exchange: When the user hits the 'download' button, SliceHub packs both the 3D model geometry (.stl) and the meta-data file (.json) in a .zip container which can be opened in the SliceHub user interface embedded in the existing slicer Ultimaker Cura.

Adding Data: Every time SliceHub generates new slicing results, it updates the meta-data file of the 3D model on the cloud compute service. When a new 3D model is added, a new folder is created and the global meta-data file of the repository is updated with the new 3D model entry.

6.2 User Interface Plugin

The user interface is implemented as a plugin to Ultimaker Cura 3.6 using Cura's plugin support. The backend is written in Python and the front-end is written in QML and Javascript. When users load the .zip file from the repository, our slicer plugin unpacks the file and after loading the 3D model (.stl) into the view, populates the user interface, i.e. adds to each print resolution profile and model scale the print time and material consumption from the meta-data file (.json). Next, it fills the constraint boxes for print time and material consumption with the respective minimum and maximum values from the print resolution profiles with the highest/lowest resolutions and largest/smallest model scales. All elements in the user interface are linked with each other, for instance, as users select a slicing result with a smaller model scale, the model is resized in the view.

6.3 Parallel Slicing Infrastructure

To be able to slice multiple print resolution profiles and model scales at the same time, we automated the slicing process and then deployed the automated slicer on the SliceHub infrastructure that enables us to run multiple slicing instances in parallel.

Automated Slicing (CuraEngine): For automated slicing, we use CuraEngine, an open source C++ library that can be operated as a command line tool. CuraEngine's slice() function takes as input: (1) an .stl file for the 3D model, (2) a .json file containing the default print settings, and (3) a string containing the modified print parameters for each configuration (when the user selects a different print resolution profile the print settings, such as layer-height and print speed, change). After slicing, CuraEngine returns the print time and material amount to the terminal. SliceHub then reads the print time and material amount from the terminal and adds them to the model-specific meta-data file. Since SliceHub does not use the .gcode file that is also generated as a part of the slicing process, it discards it. Once the meta-data is updated, it can either be read back to the user interface embedded into the slicer or updated on the model repository website.

Parallel Slicing in the Cloud: To make slicing run in parallel, SliceHub uses the cloud-compute service Amazon Web Services and instantiates as many cloud computers in parallel as needed. Each cloud computer is instantiated with a .zip file that contains CuraEngine and a python script, which starts the slicing process as soon as the cloud computer is called. After slicing has finished, the print time and material are returned to the terminal by the cloud computer. The cloud computers then save the model-specific meta-data file on the cloud storage system. Depending on the print resolution profile and model scale, some slicing processes may finish faster than others. The user interface waits for all processes to return and then reads the model-specific meta-data file from the cloud storage to update the slicing results in the user interface.

6.4 Interpolation Algorithm

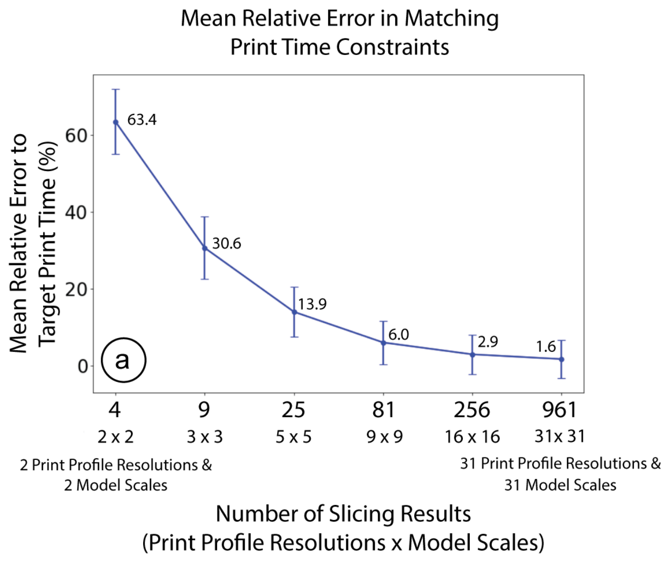
To be able to estimate slicing results for missing print resolution profiles and model scales, we fit a function over the existing slicing results of a 3D model. When users increase the percentage of sliced results using the 'interpolated/sliced' slider, we distribute sliced results uniformly across the space of print resolution profiles and model scales.

Fitting a Function over Slicing Results: To compute the print time and material consumption estimates for the interpolated slicing results, SliceHub fits a function to the existing slicing results of the currently loaded 3D model using the *scikit-learn* library. SliceHub fits a second degree polynomial to all slicing results that are available for that particular model. We use a polynomial function since changes in print time and material consumption do not change linearly with changes in scale. The reason for this is that the overall volume of an object shows an approximately quadratic increase with a linear increase in scale.

Choosing Which Print Profile Resolutions and Model Scales to Slice: When users increase the number of sliced results using the 'interpolated/sliced' slider, SliceHub computes which print resolution profiles and model scales should be sliced and which should remain interpolated. Given the selected percentage, SliceHub distributes the sliced results across the different print resolution profiles and model scales uniformly per dimension and uses the closest approximation when results cannot be uniformly distributed.

7 TECHNICAL EVALUATION

We based our design decisions for the individual SliceHub components, such as what information to store in the repository and how many print profiles and model scales to provide in the user interface, on a set of technical evaluations, which we describe below.



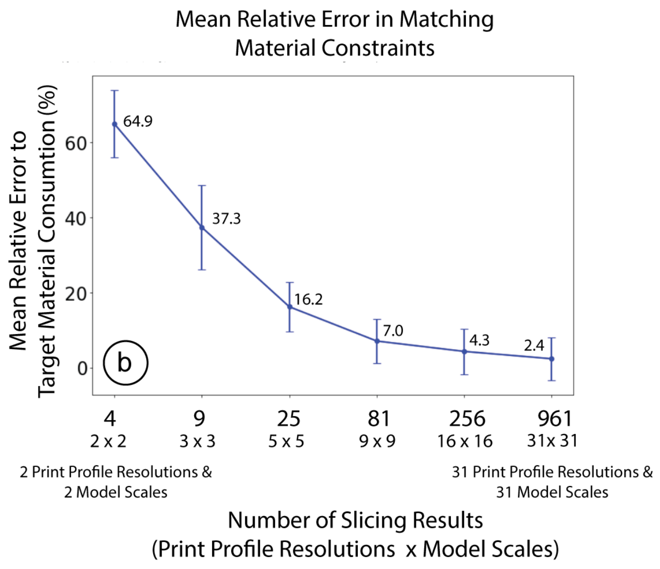


Figure 6: Mean relative error for different numbers of print resolution profiles and model scales offered in the user interface when either (a) a print time constraint or (b) a material amount constraint need to be matched.

7.1 Repository Scalability: Amount of Meta-Data per Model

We investigated how much storage space a model with all print resolution profiles and model scale combinations requires when all results are sliced and when a large fraction of results is interpolated.

Average File Size All Results Sliced: For the top 1200 Thingiverse models, we found that the meta-data file (.json) for one 3D printer and material combination with 16x16 print resolution profiles and model scales (256 combinations) and with all results sliced is approximately 12 KB. When the top 7 materials are supported, which cover 90% of the 3D printing demand in plastic [Hubs 2020], the file size increases to approximately 84 KB. Further supporting the top ten 3D printers, which cover 28.4% of all 3D printing users [Hubs 2020], increases the file size to approximately 840 KB.

Average File Size 90% Results Interpolated: When 90% of results are interpolated as recommended by our technical evaluation 7.3, the average file size is reduced by 90%, i.e. 1.2 KB rather than 12KB for one material/printer combination, 8.4 KB rather than 84KB for the top 7 materials, and 84 KB rather than 840KB for the top 10 printers.

Average File Size of Geometry (.stl) and .gcode: We found that for the top 1200 Thingiverse models, the average 3D model geometry file (.stl) was 6MB and the average size for the .gcode file was 10MB.

Based on this data, we conclude that adding the SliceHub meta-data to an existing repository, such as *Thingiverse,* for a range of popular print materials and 3D printers would only require an increase in the storage by 13.6% if all results are sliced (840 KB on top of the 6MB average model geometry file size), or 1.36% if 90% of values are interpolated (84 KB on top of the 6MB average model geometry file size). Storing the gcode, however, would add significant storage overhead to the repository. Since we generate 1750 slicing results per model (10% of 256 slicing results for 7 materials and 10 printers) and the size of an average gcode file is 10Mb, the overall storage increase would be approximately 17Gb per model (10Mb gcode * 1750 slicing results) and is thus not scale-able.

7.2 User Interface: Effect of Number of Print Profiles on Constraint Accuracy

When designing the user interface, we also had to decide on the number of print resolution profiles and model scales that the user can choose from. If there are only few options available, the user may not be able to find a good match for their time and material constraints. For instance, if only two print resolution profiles (e.g., 0.06mm and 0.2mm) are available, the high resolution profile may take much more time than the user has available while the fast low resolution may result in a worse print quality than the user would have had time for. Offering more options, however, increases computational cost to generate the slicing results. To find a good trade-off, we ran the following experiment.

Conditions: We created six conditions, i.e. six user interfaces with different numbers of print resolution profiles and model scales (ranging from 2 print resolution profiles and 2 model scales (4 slicing results) to 31 print profiles and 31 model scales (961 slicing results)). The user interface with the fewest options used the print resolution profiles with the highest resolution (0.06mm) and the lowest resolution (0.2mm), as well as the largest (100%) and smallest (10%) model scales. All other user interfaces iteratively added print resolution profiles and model scales at each midpoint of the existing values.

Procedure: We used our data-set of the top 1200 Thingiverse models and generated slicing results for each condition. We then randomly picked 20 print time and 20 material constraints from the available min/max bounds of each model, and for each user interface condition computed the mean relative error to the closest matching slicing result.

Experiment Results: As shown in Figure 6, the more print resolution profiles and model scales the user interface offers, the smaller the mean relative error is. For the smallest user interface (2x2) the mean relative error for the print time is 63.4% and material amount 64.9%, whereas for the largest user interface (31x31) the mean relative error for print time is only 1.6% and material amount 2.4%. However, while there is a strong reduction in error when a small user interface is expanded (e.g., for print time: 2x2: 63.4% vs 3x3: 30.6%), the effect is less emphasized for larger user interfaces (e.g., for print time there is only a 1.3% improvement between 16x16: 2.9% vs. 31x31: 1.6%). Since a user interface with 16 print resolution profiles and model scales requires only 26.6% of the data of a user interface with 31 print resolution profiles and model scales, and thus significantly lowers computational cost, we decided to use the user interface with 16 print resolution profiles and model scales as the default for SliceHub.

7.3 Effect of Number of Interpolated Slicing Results on Average Prediction Error

To decide how many slicing results SliceHub should by default slice and how many interpolate, we ran an experiment to evaluate the effect of an increasing number of interpolated results on the mean relative error for predicting print time and material consumption.

Experiment Procedure: We created five conditions, i.e. five user interfaces of 16x16 print resolution profiles and model scales that had different percentages of interpolated results. The user interfaces ranged from containing 0% interpolated results (all 16 print resolution profiles x 16 model scales combinations sliced) to 98% interpolated cells (only 2 print resolution profiles x 2 model scales sliced). We then calculated the relative mean error for the interpolated results from the computed slicing results (e.g., if the computed print time of an object is 2h but the interpolated print time is 1h, there is a relative error of 50%).

Experiment Results: As can be seen in Figure 7, the error in predicted print time and material amount decreases with number of sliced results since there are more data points available for accurately fitting the polynomial function. For instance, the interpolation error for print time when only 4 (2x2) out of 256 print resolution profiles and model scales are sliced is 36.31%, whereas the interpolation error is only 2.08% when 81 (9x9) out of 256 print resolution profiles and model scales are sliced. Comparing the interpolation error for different percentages of interpolated results shows a strong decrease in error when 98% vs 96% of results are interpolated (26.33% less interpolation error) and when 96% vs 90% are interpolated (6.31% less interpolation error). The reduction in interpolation error when 90% vs. 68% of results are interpolated is only marginal (1.79% less error). Since interpolation accuracy no longer strongly decreases after about 10% of sliced results (90% interpolation), SliceHub by default slices 10% of results when a new model is added.

8 DISCUSSION AND FUTURE WORK

We next discuss how SliceHub can be extended in the future.

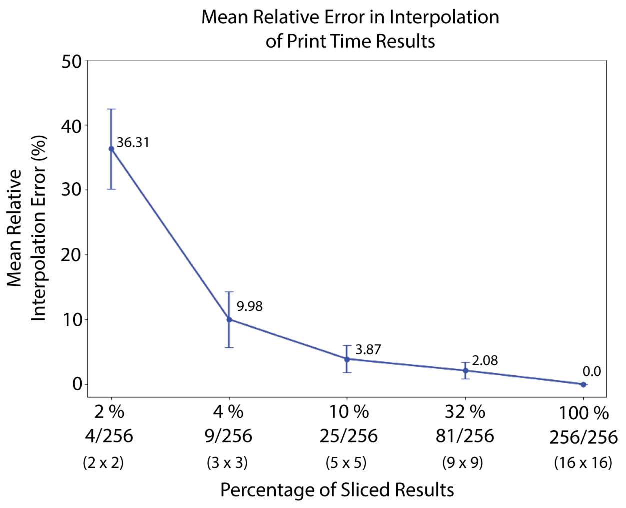
Indicating Failed Print Settings: While in our work, we have focused on making slicing results, such as print time and material consumption, readily available for exploration and comparison, we currently do not provide users with information if the chosen print profile resolutions and model scales lead to a successful print. For future work, we plan to allow the model designer or a user who printed the model to indicate if specific print profile resolutions and model scales worked or if the print did not complete. By integrating this information in the user interface, SliceHub can provide the information in a structured manner rather than through the informal comments section as is commonly done in 3D model repositories, such as Thingiverse.

Comparing Printers / Materials: While we focused on comparing different print resolution profiles and model scales of a 3D model, the data in SliceHub's repository can also be used for other use cases. For instance, SliceHub's data allows users to compare the print time of different 3D printers with each other enabling users to draw conclusions which 3D printer on the market is currently the fastest. Similarly, users can compare the print times of different filament types with each other, which may influence their choice of print material.

Extending the Slicing Infrastructure: Finally, we plan to extend SliceHub's slicing infrastructure. Our current implementation is based on the slicer Ultimaker Cura and thus only works for FDM 3D printers. By default, it is setup to generate slicing results for the Ultimaker 3D printer series but print resolution profiles for other FDM 3D printers (e.g., MakerBot, Prusa) can be imported and subsequently used for slicing. To support other 3D printing technologies, we are currently in the process of extending our slicing infrastructure with the slicer Slic3r [Slic3r 2011], which supports a wider variety of other 3D printers, such as those based on DLP printing (e.g., Prusa SL1 and Formlabs Form 3).

9 CONCLUSION

We presented SliceHub, an integrated system that enables users to explore different print settings in real-time to find the trade-off between print resolution profile, model scale, print time, and material consumption that best fits their needs. We discussed the design and technical implementation of the SliceHub repository for storing slicing results, the user interface for exploring trade-offs between multiple print resolution profiles and model scales simultaneously, and SliceHub's infrastructure for slicing and interpolation that further adds to the data available in the repository. We provided an evaluation of the data storage requirements for the repository and



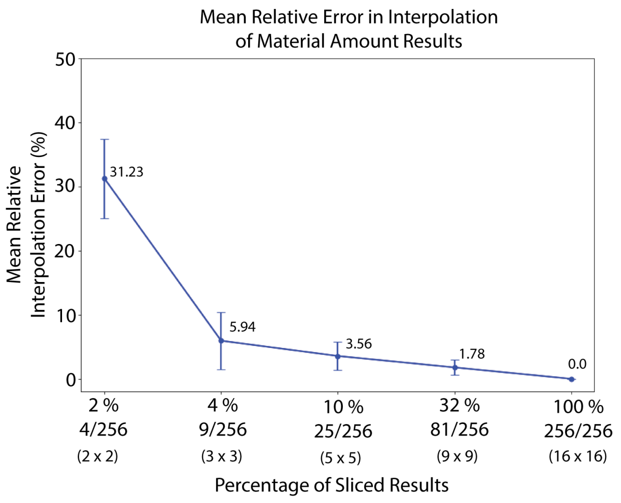


Figure 7: Higher percentages of interpolated results, lead to larger errors. The error stabilizes when 90% of results are interpolated (i.e., 10 % of results are sliced). We thus use this as the default setting when a new model is added to SliceHub.

reported evaluation results to support design decisions made in the user interface, such as the number of print resolution profiles and model scales the user can choose from. For future work, we plan to deploy the repository as a website and provide the plugin for the slicer through the Ultimaker Cura marketplace. In addition, we plan to track contributions to the repository and investigate users' activities on SliceHub as well as collect feedback from the maker community.

REFERENCES

Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2019. Exploring and understanding the role of workshop environments in personal fabrication processes. *ACM Transactions on Computer-Human Interaction (TOCHI)* 26, 2 (2019), 1–43.

Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it: optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Alexander Berman and Francis Quek. 2020. ThingiPano: A Large-Scale Dataset of 3D Printing Metadata, Images, and Panoramic Renderings for Exploring Design Reuse. In *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*. IEEE, 18–27.

Alexander Berman, Ketan Thakare, Joshua Howell, Francis Quek, and Jeeeun Kim. 2021. HowDIY: Towards Meta-Design Tools to Support Anyone to 3D Print Anywhere. In *26th International Conference on Intelligent User Interfaces*. 491–503.

Dustin Beyer, Serafima Gurevich, Stefanie Mueller, Hsiang-Ting Chen, and Patrick Baudisch. 2015. Platener: Low-fidelity fabrication of 3D objects by substituting 3D print with laser-cut plates. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 1799–1806.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository*. Technical Report arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Xuelin Chen, Hao Zhang, Jinjie Lin, Ruizhen Hu, Lin Lu, Qi-Xing Huang, Bedrich Benes, Daniel Cohen-Or, and Baoquan Chen. 2015. Dapper: decompose-and-pack for 3D printing. *ACM Trans. Graph.* 34, 6 (2015), 213–1.

Brian Danchilla. 2012. Three. js framework. In *Beginning WebGL for HTML5*. Springer, 173–203.

Kristin N Dew, Sophie Landwehr-Sydow, Daniela K Rosner, Alex Thayer, and Martin Jonsson. 2019. Producing printability: Articulation work and alignment in 3d printing. *Human–Computer Interaction* 34, 5-6 (2019), 433–469.

Jérémie Dumas, Jean Hergel, and Sylvain Lefebvre. 2014. Bridging the gap: automated steady scaffoldings for 3D printing. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Jimmy Etienne, Nicolas Ray, Daniele Panozzo, Samuel Hornus, Charlie CL Wang, Jonàs Martínez, Sara McMains, Marc Alexa, Brian Wyvill, and Sylvain Lefebvre. 2019. CurviSlicer: Slightly curved slicing for 3-axis printers. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–11.

Miguel Grinberg. 2018. *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.".

Kristian Hildebrand, Bernd Bickel, and Marc Alexa. 2013. Orthogonal slicing for additive manufacturing. *Computers & Graphics* 37, 6 (2013), 669–675.

3D Hubs. 2020. 3D printing Trends 2020 Report. https://www.3dhubs.com/get/trends/

Nathaniel Hudson, Celena Alcock, and Parmit K Chilana. 2016. Understanding newcomers to 3D printing: Motivations, workflows, and barriers of casual makers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 384–396.

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*. 1067–1078.

Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. 2014. Build-to-last: strength to weight 3D printed objects. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Thomas Ludwig, Oliver Stickel, Alexander Boden, and Volkmar Pipek. 2014. Towards sociable technologies: an empirical study on designing appropriation infrastructures for 3D printing. In *Proceedings of the 2014 conference on Designing interactive systems*. 835–844.

Makerbot. 2015. Celebrating a Maker Milestone: 1 Million Uploads on Makerbot's Thingiverse. https://www.makerbot.com/stories/2015/10/29/

Stefanie Mueller, Dustin Beyer, Tobias Mohr, Serafima Gurevich, Alexander Teibrich, Lisa Pfistere, Kerstin Guenther, Johannes Frohnhofen, Hsiang-Ting Chen, Patrick Baudisch, et al. 2015. Low-fidelity fabrication: Speeding up design iteration of 3D objects. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 327–330.

Stefanie Mueller, Sangha Im, Serafima Gurevich, Alexander Teibrich, Lisa Pfisterer, François Guimbretière, and Patrick Baudisch. 2014. WirePrint: 3D printed previews for fast prototyping. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 273–280.

Multi3D. 2021. Electrifi Conductive Filament. https://www.multi3dllc.com/product/electrifi/

Behnaz Norouzi, Marianne Kinnula, and Netta Iivari. 2021. Making Sense of 3D Modelling and 3D Printing Activities of Young People: A Nexus Analytic Inquiry. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–16.

Travis E Oliphant. 2006. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA.

Thomas Oster. 2011. Visicut: An application genre for lasercutting in personal fabrication. *RWTH Aachen Univ* (2011).

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make it stand: balancing shapes for 3D fabrication. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2016. Retrofab: A design tool for retrofitting physical interfaces using actuators, sensors and 3d printing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 409–419.

Thijs Jan Roumen, Willi Müller, and Patrick Baudisch. 2018. Grafter: Remixing 3D-printed machines. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

Daniel Saakes, Thomas Cambazard, Jun Mitani, and Takeo Igarashi. 2013. PacCAM: material capture and interactive 2D packing for efficient material usage on CNC cutting machines. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 441–446.

Valkyrie Savage, Andrew Head, Björn Hartmann, Dan B Goldman, Gautham Mysore, and Wilmot Li. 2015. Lamello: Passive acoustic sensing for tangible input components. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 1277–1280.

Ryan Schmidt and Karan Singh. 2010. Meshmixer: an interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks*. 1–1.

Ryan Schmidt and Nobuyuki Umetani. 2014. Branching support structures for 3D printing. In *ACM SIGGRAPH 2014 Studio*. 1–1.

Ticha Sethapakdi, Daniel Anderson, Adrian Reginald Chua Sy, and Stefanie Mueller. 2021. *Fabricaide: Fabrication-Aware Design for 2D Cutting Machines.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445345

Slic3r. 2011. Open source 3D printing toolbox. https://slic3r.org/

Ultimaker. 2020. Ultimaker Cura. https://ultimaker.com/software/ultimaker-cura

Nobuyuki Umetani and Ryan Schmidt. 2013. Cross-sectional structural analysis for 3D printing optimization.. In *SIGGRAPH Asia Technical Briefs*. Citeseer, 5–1.

Juraj Vanek, Jorge A Garcia Galicia, and Bedrich Benes. 2014. Clever support: Efficient support structure generation for digital fabrication. In *Computer graphics forum*, Vol. 33. Wiley Online Library, 117–125.

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*. 1–12.

Weiming Wang, Haiyuan Chao, Jing Tong, Zhouwang Yang, Xin Tong, Hang Li, Xiuping Liu, and Ligang Liu. 2015. Saliency-preserving slicing optimization for effective 3D printing. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 148–160.

Christian Weichel, Jason Alexander, Abhijit Karnik, and Hans Gellersen. 2015. SPATA: Spatio-tangible tools for fabrication-aware design. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*. 189–196.

Xiaoting Zhang, Xinyi Le, Athina Panotopoulou, Emily Whiting, and Charlie CL Wang. 2015. Perceptual models of preference in 3D printing direction. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–12.

Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2016. Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797* (2016).