Structure-Preserving Editing of Plates and Volumes for Laser Cutting

AUTHOR

City, Country, e-mail address

AUTHOR

City, Country, e-mail address

AUTHOR

City, Country, e-mail address

AUTHOR

City, Country, e-mail address

AUTHOR

City, Country, e-mail address

AUTHOR

City, Country, e-mail address

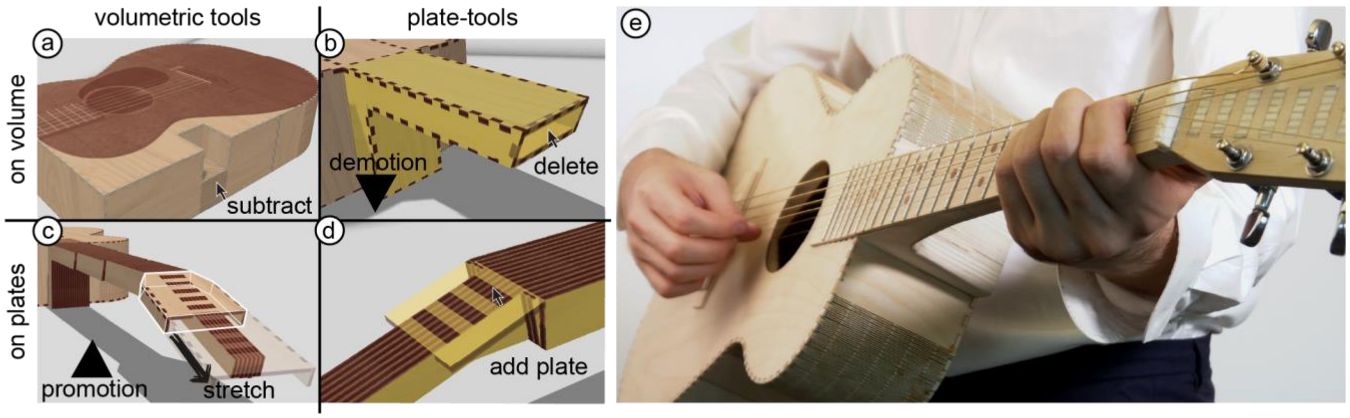

Figure 1: Structure-preserving editing for laser cutting (a) represents laser-cut 3D models as volumes, whenever possible. This allows users to manipulate models *efficiently* using volume-based tools. (d) It represents laser-cut 3D models as a 3D arrangement of plates, when users want to manipulate models *in detail* using plate-based tools. (b) The key to making volumetric and plate-based representations work within the same model is that our architecture *demotes* models represented as volume to plates, when users apply plate-based tools, and it (c) *promotes* models represented as plates to volumes, when users apply volume tools anywhere. (e) This approach allows users to manipulate 3D models that are complete plate-like elements with volumetric elements, resulting in a level of complexity not possible with previous tools.

We present a 3D editor for laser cutting that extends the range of models that users can manipulate. Our system gives users control over the detailed elements of laser cutting, i.e., individual plates and the associated joints, yet at the same time also allows for efficient editing by means of volumetric tools while preserving the structure of plates in the model. Our system consists of four functional groups: (1) We started with a fabrication-aware 3D editor capable of handling volumetric models (kyub [Baudisch 2019]). This subsystem represents 3D models as a single volume. (2) We added a second subsystem that represents laser-cut models as an arrangement of plates in 3D. This allowed us to add tools that allow manipulating individual plates. (3) We unified these two subsystems by adding a demotion mechanism that breaks volumes down into multiple plates, to allow users to apply plate tools to volumes, as well as (4) a promotion mechanism, which infers volumetric substructures from sets of plates, to allow users to apply volume-based tools to plate structures. We validated the resulting system by recreating the 100-model benchmark of assembler^3 [Roumen 2021]. Our combined system successfully recreated 87 of the models, compared to 9 with a volume-only baseline system (kyub [Baudisch 2019]) and 15 with a plate-only baseline system (flatFitFab [McCrae 2014]).

CSS Concepts **Human-centered computing~Human computer interaction (HCI)~Interactive systems and tools**

Additional keywords and phrases: Laser Cutting, Digital Fabrication, Personal Fabrication, 3D modeling, CAD

ACM Reference Format:

First Author's Name, Initials, and Last Name, Second Author's Name, Initials, and Last Name, and Third Author's Name, Initials, and Last Name. 2018. The Title of the Paper: ACM Conference Proceedings Manuscript Submission Template: This is the subtitle of the paper, this document both explains and embodies the submission format for authors using Word. In Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY. ACM, New York, NY, USA, 10 pages. NOTE: This block will be automatically generated when manuscripts are processed after acceptance.

1 INTRODUCTION

Historically, there have been two distinct approaches to 3D editing for laser cutting: (1) systems aiming at maximizing expressiveness in terms of shape represent 3D models as an arrangement of intersecting plates, such as *FlatFitFab* [McCrae 2014], *Fabrication-aware design* [Schwartzburg 2013], or *SketchChair* [Saul 2011]. In contrast, (2) systems aiming at maximizing sturdiness and efficiency represent 3D models as a volume, such as the conversion tools *Platener* [Beyer 2015], *Slicer for Fusion360* or *Fresh Press Modeler* [Chen 2016], various box makers (such as *make-a-box.io*), or the volumetric editor *kyub* [Baudisch 2019].

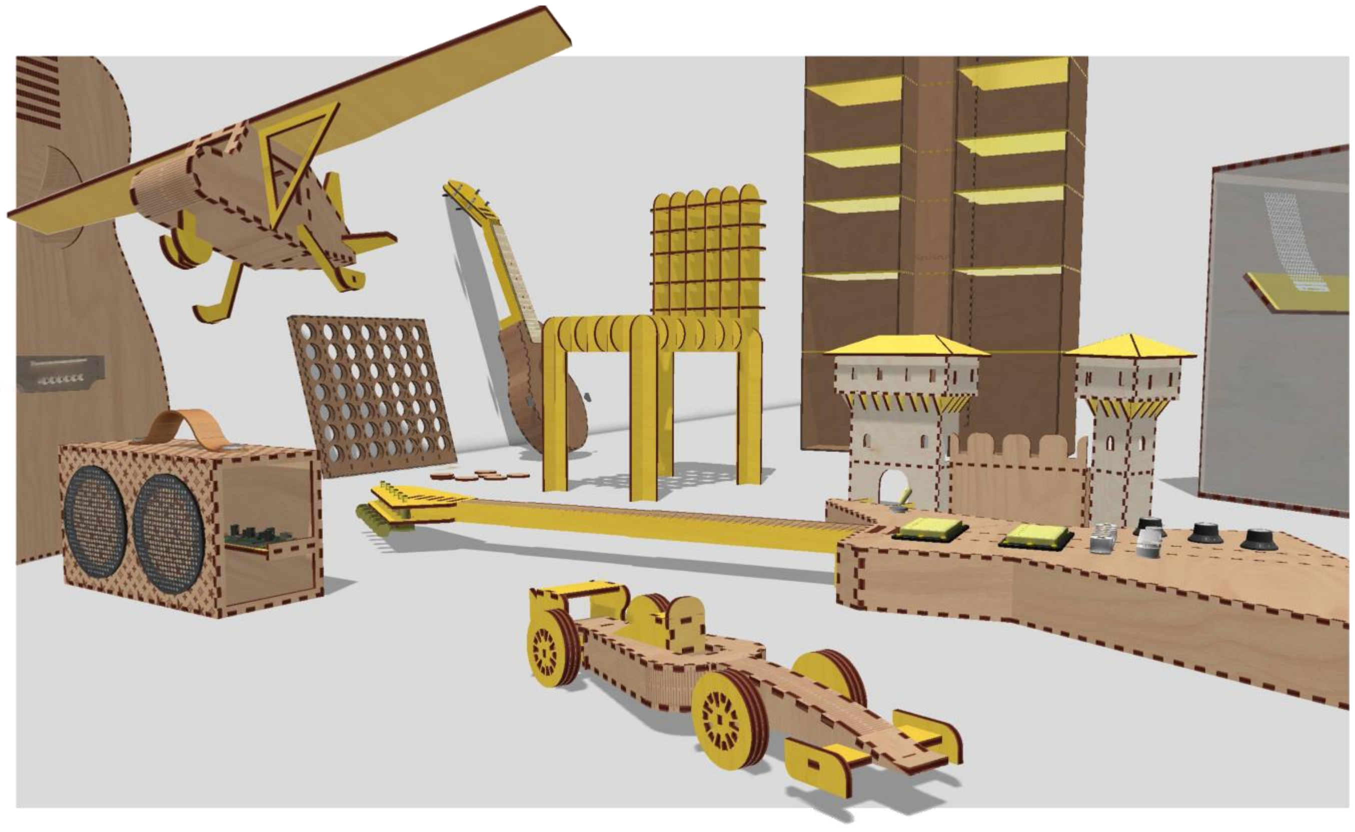

Figure 2: Structure-Preserving Editing allows users to create models that traditionally could only be created and manipulated by hand using "fabrication unaware" modeling. These hybrid models contain plates (highlighted in yellow) and volumetric elements

Unfortunately, many laser-cut models are *neither* all-volume *nor* all-plate, such as musical instruments, furniture, and scale models. As of today, these can be neither created nor manipulated with any of the tools and systems listed above, forcing designers to revert to general-purpose 3D editors, such as *Fusion360* or *OnShape.com* or even to 2D drawing programs, such as *CorelDraw* or *Adobe Illustrator*. Neither of these offers any specific support for laser-cutting, thus making users an order of magnitude less efficient.

In this paper, we tackle the editing of volume + plate models from a system-building angle. As illustrated by Figure 1, we proceed in four steps. Our *structure-preserving editing* for laser cutting (a) represents laser-cut 3D models as volumes, whenever possible. This allows users to manipulate models *efficiently* using volume-based tools. (d) It represents laser-cut 3D models as a 3D arrangement of plates, when users want to manipulate models *in detail* using plate-based tools. (b) The key to making volumetric and plate-based representations work within the same model is that our architecture demotes models represented as volume to plates, when users apply plate-based tools, and it (c) promotes models represented as plates to volumes, when using volumetric tools.

As illustrated by Figure 2, our approach allows users to create and manipulate 3D models that are neither all-plate nor all-volume, resulting in a level of complexity not possible with previous tools. This paper presents a data-structure contribution, we implement it at the example of integration with kyub tools, but it is not limited to that specific implementation.

We have validated the resulting system by recreating the 100-model *assembler³ benchmark* ([Roumen 2021], which in turn is based on models from *Thingiverse*.com).Our combined system successfully recreated 87 of the models, compared to 9 with a volume-only baseline system (*kyub* [Baudisch 2019]) and 15 with a plate-only baseline system (*flatFitFab* [McCrae 2014]).

2 CONTRIBUTION, LIMITATIONS & BENEFITS

Our main contribution is the integration of volumetric and plate-based modeling paradigms in a single system that allows users to edit laser-cut models in structure-preserving fashion. Building on a volume-based editor [Baudisch 2019], we add three key elements, i.e., (a) a subsystem for plate-based editing structurally similar to volume-based editing so as to allow for a tight integration, (b) a demotion mechanism from volumes to plates, and (c) a promotion mechanism from plates to volumes. It is the combination of these four elements that addresses the challenge.

We are thus making a *systems contribution*, i.e., our main contribution is not a single, iconic invention, but our contribution lies in how we put multiple (some novel, some previously explored) elements together, forming a new whole. Our main *algorithmic contribution* is the presented promotion mechanism.

While we demonstrate our approach by building on an existing 3D editor for laser cutting (kyub) the concept of a two-tiered system that represents some parts of a model as plates while representing others as volumes and allows switching between them, either by means of promotion and demotion is *independent* of the specific implementation, making our insights equally relevant to researchers working with other platforms (such as, *FlatFitFab* [McCrae 2014]).

The presented system allows users to create models previously only possible with the help of general-purpose 3D or 2D editors, but with the efficiency of a fabrication-aware tool, as we demonstrate by recreating 87 of the 100 models from the *assembler³* benchmark [Roumen 2021], as well as new complex models, such as the playable acoustic guitar shown in Figure 1e and models shown in Figure 2.

Limitations of our system include that our current set of plate tools does not offer tools for free-form editing (as offered, for example, by *FlatFitFab* [McCrae 2014]) and offers only limited control over alignment, precision, and symmetry. The system is built on the assumption of rigid materials.

3 THE PLATE-BASED SUBSYSTEM

We start by presenting our plate-based subsystem. As illustrated by Figure 3, we designed these tools to be consistent with the volume-based tools provided by the platform we built on [Baudisch 2019].

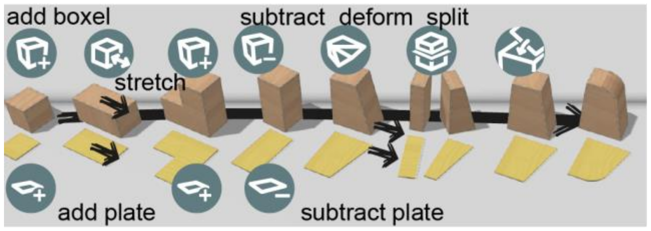

Figure 3: We designed the tools of the plate-based subsystem to be consistent with the volume-based tools provided by the platform we built on [Baudisch 2019].

This consistency across subsystems allows for a reduced user interface: as illustrated by Figure 3, it allows us to overload the *edit* functions for plates onto the same functions that manipulate volumes.

In addition to the volume-inspired tools shown above, we added tools that help to arrange plates in 3D. The workflow shown in Figure 4 adds plates at right angles or stacks them onto existing plates. The *move* tool and *rotate* tool allow users to fine-tune the arrangement.

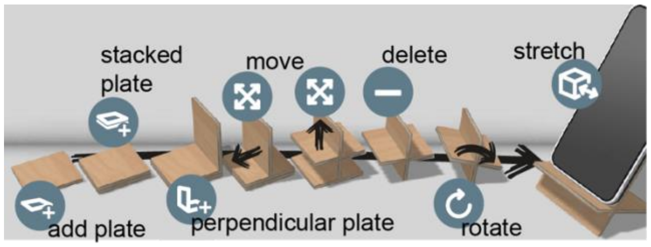

Figure 4: Various *add plate* tools allow arranging plates in 3D. The move tool allows users to fine-tune their positioning.

The *attach* tool shown in Figure 5 also extends to plates but presents additional options to users on how to arrange plates in 3D after attaching.

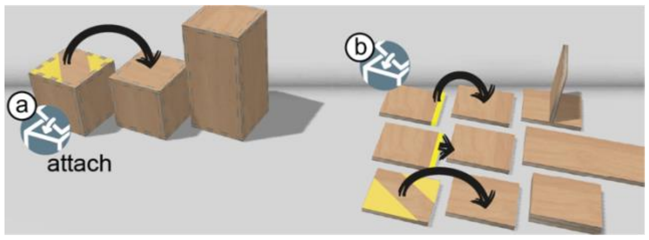

Figure 5: (a) In contrast to the attach tool of the volumetric subsystem, (b) the plate-attach tool provides additional 3D arrangement options.

The plate tools shown above allow constructing a range of basic models, such as the ones shown in Figure 6.

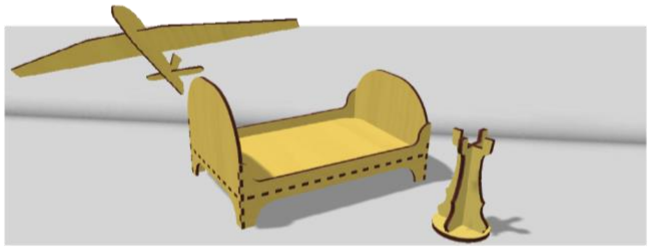

Figure 6: Simple models made using plate tools alone.

The same tools also allow somewhat more complex models, such as the VR headset shown in Figure 7. However, this workflow already hints at the limited efficiency of a purely plate-based workflow.

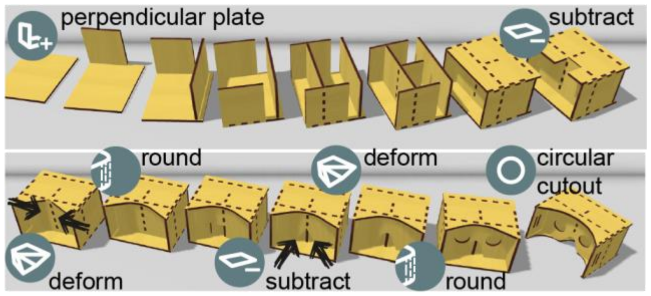

Figure 7: Plate and edit tools allow creating a wide range of models, albeit with limited efficiency (VR headset, id:638605).

4 PROMOTION

The inefficiency of a purely plate-based workflow becomes obvious when we try to modify the model from Figure 7. As illustrated by Figure 8, making the headset taller now requires users to stretch five plates, move the top plate, doing so in the right order, and getting the resulting alignment right. This is obviously not desirable.

What we want instead is to pull up the top plate and have the rest of the model follow its lead as shown in Figure 8b, similar to *pushpull++* [Lipp 2014]. We get this type of *volume-based* operation naturally from models that live in the volume-based subsystem. Naturally, we want this type of functionality also for models that originated in the plate-based subsystem.

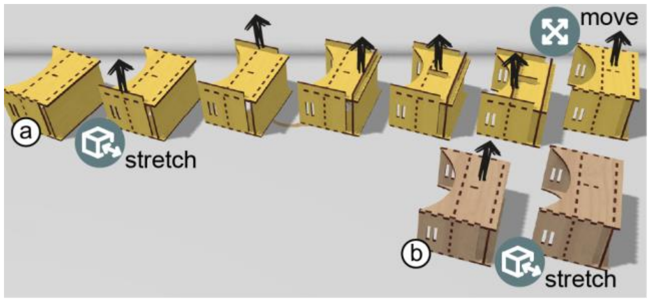

Figure 8: (a) Once demoted to plates, making a VR headset 1cm taller requires six user interactions. (b) Making the same *volumetric* modification is a single interaction.

We address this by adding what we call the *promoter.* The promoter is invoked whenever users apply a volume-based tool. The promoter now checks the clicked model: if it is already in volumetric representation, it is done and simply invokes the tool. If the model is in plate-based format, however, the promoter searches the model for *volume-like substructures,* translates them into a volumetric representation (the *promotion*), and then applies the tool.

As illustrated by Figure 9, this allows volumetric structures created from plates to be manipulated using volumetric tools, here "*stretch*".

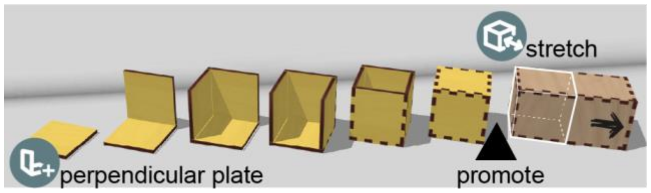

Figure 9: Consecutive add plate tools allow constructing a volume. When applying a volumetric stretch tool, the promoter detects the volume and stretches the plates accordingly.

But the promoter does more. As illustrated by Figure 10a, it identifies volumes also when these are incomplete, and when they are part of slanted models (Figure 10b).

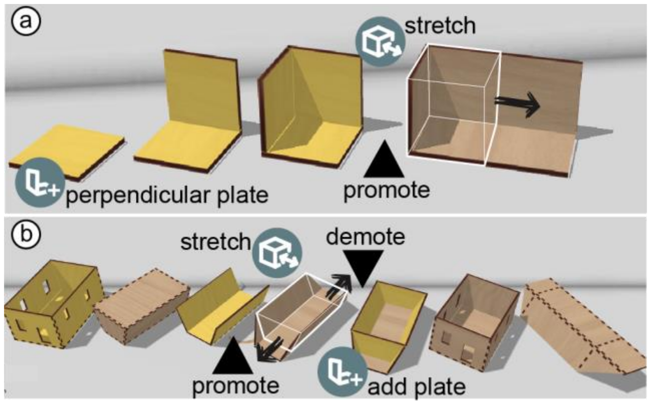

Figure 10: (a) The promoter also identifies incomplete volumes. (b) And works for slanted volumes, here to make a separate rooftop for a dollhouse. To apply the plate tool after, it gets demoted (see next section on demotion).

The key benefit of the promotion mechanism is that it relieves users from the burden to know about how a structure originated, as two structures that look the same can now be treated the same way. Figure 11 shows a three-plate corner created by removing plates from a box, as well as a three-plate corner created by assembling plates. With the help of the promoter, running in the background hidden from the user, either one can be stretched using the stretch tool, producing the same result.

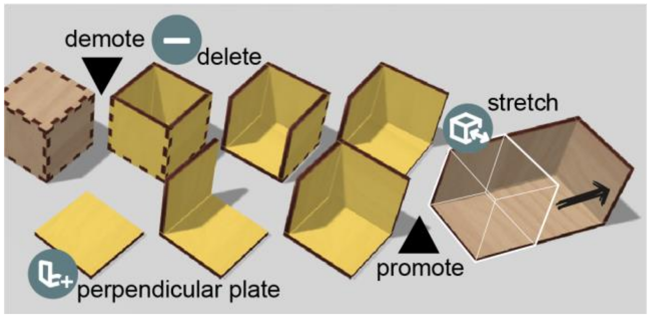

Figure 11: The promoter treats the shown 3-plate assembly the same, irrespective of whether it was created by combining three plates or by removing three plates from a box.

5 DEMOTION

Going back to the headset, the workflow shown in Figure 7 clearly is not the most efficient way of creating this 3D model. As illustrated by Figure 12, the tools from the volume-based subsystem get users started much faster. However, eventually users need to use plate tools to get the details right, such as the divider between the eyes and the overextended plates.

We enable this scenario with the counterpart to the promoter, the *demoter*. As shown in Figure 12, when users try to apply a plate tool to a model that lives in the volume-based subsystem, the demoter breaks the plates that are touched by the plate into plates, allowing individual plates to be moved or stretched.

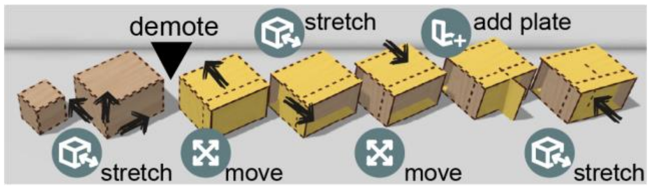

Figure 12: Starting with a volume allows re-creating the VR headset from Figure 7 more efficiently. The part of the model shown in yellow is demoted to plates to allow for the plate tools to apply.

We found this demoter-based workflow, i.e., volume-based tools first, then refinement using plate-based tools to be efficient and the basis for many common models (Figure 13). The promoter, however, is equally crucial for this approach to modeling, as it allows making late modifications, rather than enforcing a strict "waterfall" process.

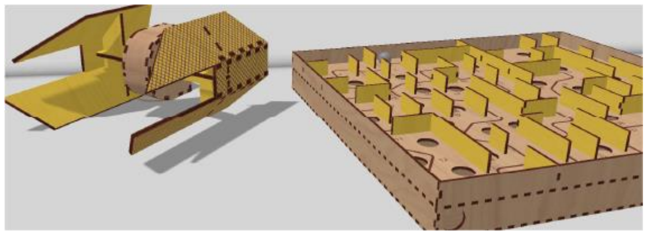

Figure 13: Volume-tools first, then refinement using plate tools is an efficient and thus common workflow.

Figure 2 shows a range of models that were created using this general "top-down" approach from volume to plates. Most of these models were created by starting with a volumetric element, then adding details using the plate tools, e.g., for structural reasons (e.g., guitar, chair), to mount components inside volumes (e.g., cajon, speaker), or to create small scale structures on a larger model (e.g., race car, airplane). The workflows of more complex models, such as the one shown in Figure 14, may contain multiple invocations of promoter and demoter.

Figure 14 shows the workflow of modeling the guitar of Figure 1 using multiple promotion and demotion invocations. (a) Users start to shape the model with volumetric tools (b) the demoter turns the neck into plates as the user deletes plates and inserts a stack (c) the neck is promoted to a volume when stretching it longer, to then be demoted again as the user modifies detailed plates (d) to make the head, the stretch tool uses the promoter, and to add individual plates the demoter turns it back into plates (e) finally the promoter allows the head plate to be stretched into a volume and (f) the user finishes the model by adding a sound hole, bridge, fretboard and tuners.

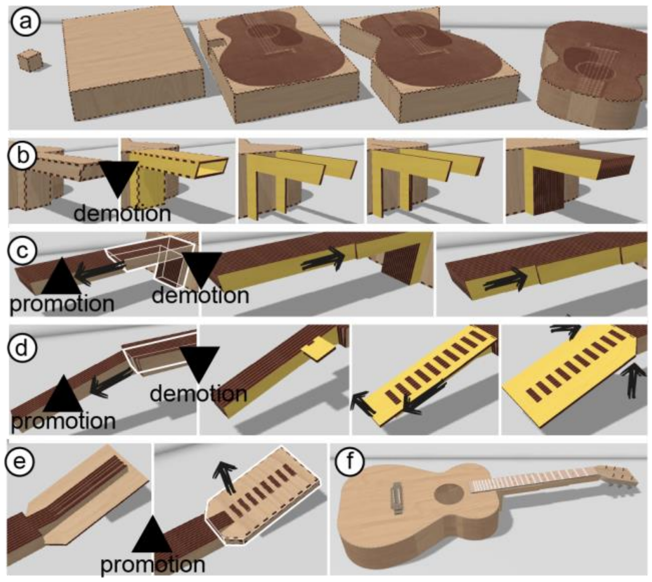

Figure 14: The workflows of more complex models may contain multiple invocations of promoter and demoter.

6 ALGORITHM AND DATA STRUCTURES

In this section we present the mechanisms of promotion and demotion. To understand demotion, we take a closer look at the data structure of plates and volumes. As volumes inherently consist of plates, we can break them down relatively easily. To reconstruct a volume, especially when the volume is incomplete, we present the promoter algorithm.

6.1 Volume-based vs. plate-based data structures

The promoter and demoter transition the representation of models between volume-based and plate-based data structures.

As shown in Figure 15a, data structures in the volumetric subsystem consist of a single `Mesh` per model, which has its own coordinate system (`Transform`) and operates on a series of linked surfaces (and related edges). Individual plates on the other hand have their own coordinate systems, allowing them to be manipulated without interfering with other plates.

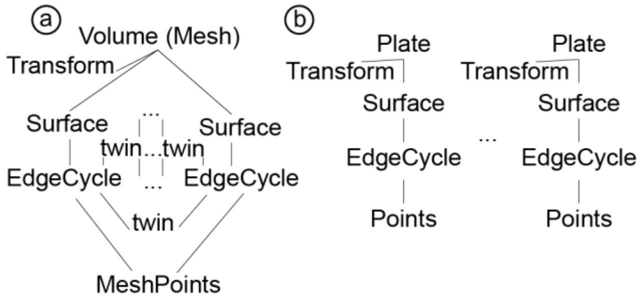

Figure 15: (a) The data structure of a volume vs. (b) data structure if the same model is represented by individual plates.

As shown in Figure 15b, the moment a `Mesh` is "damaged", e.g., by removing a plate, it cannot easily be represented as a `Mesh`. The linked `EdgeCycles` no longer form a fully linked chain, which breaks some of the assumptions the volumetric tools use when operating on Meshes. Our system demotes it to a set of plates, as illustrated by going through the EdgeCycles and assigning them their own Transforms. The cycles remain connected but no longer share MeshPoints or a common Transform. This gives the plate tools the ability to move them away from one another.

This may seem benign at first, but the demotion means that the volumetric tools no longer apply, as they operate on that single coordinate system and assume full connectedness of the EdgeCycles, turning what could have been a single volume interaction into a long sequence of primitive plate interactions.

This discussion of data structures extends beyond kyub in that fabrication-aware modeling environments for laser cutting would have *some representation* of plates and how they come together in terms of volumes. While it is possible to maintain both formats in parallel, the volumetric representation remains incomplete upon removal of plates so either the data structure or the resulting volumetric tools are required to handle this.

To achieve consistent integration in the kyub system, we opted for a modeless transition between the plate-based and volume-based representations. Alternative implementation strategies such as a plate and volume mode for an editor, more in line with traditional CAD tools, could benefit from the same underlying mechanisms of promotion and demotion.

6.2 Promoter Algorithm

At the heart of the presented system lies the promoter. Its purpose is to generate a volumetric description of the model, that tools utilize for volumetric editing operations, such as stretching.

In the example shown in Figure 16, three "plates" are missing to turn the model into a volume. The promoter constructs proxy planes by finding connected edges across two coplanar plates. The L shape on the top of the model, for example, consists of two edges connected at one corner. These edges are coplanar and stretch across two plates. The promoter constructs a proxy plane through these edges and repeats these steps for all connected coplanar edges.

When multiple such connected coplanar edges share a corner, the promoter inserts a proxy edge into the model at the intersection between the proxy planes. When both corners of the edge are shared with other connected edges, the promoter constructs all three planes and inserts a proxy corner at the point where these planes intersect. Finally, it inserts edges between the proxy corner and the edges of the model, resulting in a closed volume.

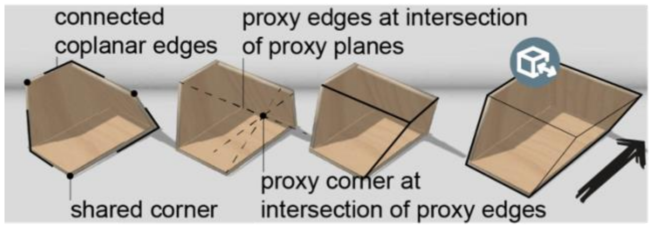
connected coplanar edges
proxy edges at intersection of proxy planes
shared corner
proxy corner at intersection of proxy edges

Figure 16: When coplanar edges touch in a corner, they form larger volumes with the adjacent coplanar edges.

When there are no shared corners between sets of connected coplanar edges, there is too little information for the algorithm to locate a proxy corner in 3D. Instead, as shown in Figure 17, the promoter runs the *2D QuickHull algorithm* [Barber 1996] (which runs in O(n log(n))) on the constructed plane and inserts result as edges into the model. In this case forming a basic prism, which can then be used by the volumetric tools. (b) The desk organizer model shows this using a real-world example: After the promoter found the rectilinear volumes, there is a single plate sticking out. Because the convex hull algorithm includes this as a volume as well, it stretches along when users make the model wider.

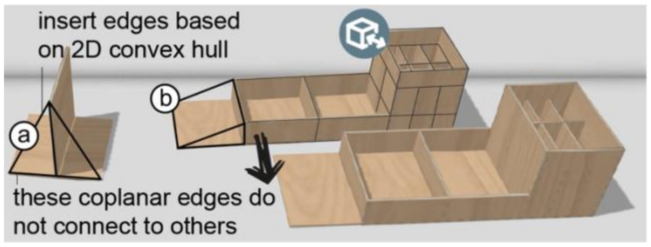

Figure 17: (a) The convex hull of objects where the connected coplanar edges do not share a corner. (b) a practical implication of this case at the example of a desk organizer: because of the proxy prism on the left the base plate stretches with the side plates.

Before the algorithm handles the cases presented thus far, it looks for closed volumes in the overall model. The previous cases therefore typically constitute of the last few plates that were not part of a volume yet. As shown in Figure 18, to detect volumes, the promoter iterates over the edges in the model and groups plates together when an edge connects exactly two adjacent plates. This effectively results in a flood fill for simple, closed volumes, such as the guitar stand of Figure 18.

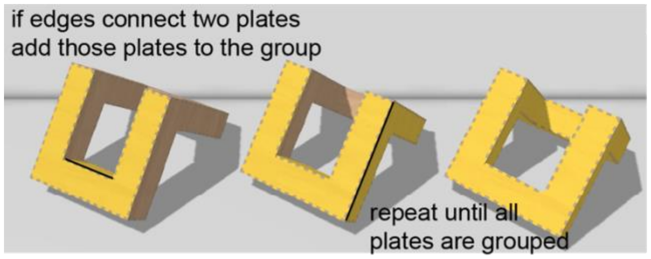

Figure 18: Inferring closed volumes on this guitar stand, the yellow plates are added to the closed group.

With full control over plates and volumes, it is possible to construct models which have plates *within* a volume. To respect these, the promoter runs 2D face detection (based on [Muller 1978]) on the planes before detecting closed volumes. As demonstrated in Figure 19, internal plates within the volume are identified as additional faces (in this case there are four such internal plates connecting in an H shape to the top plate), which results in an edge within the top plate that connects to three plates instead of two. This ensures that the internal plate is not simply discarded, but rather causes the volume to be split into two cells when executing the flood fill algorithm, such that volumetric tools behave accordingly. For example, in a stretching operation, the union of the volumes is used, but after stretching, the individual cells restore the internal plate.

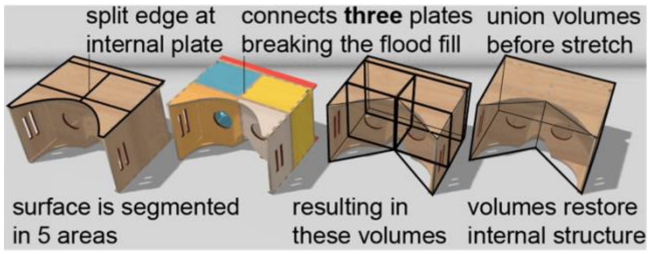

Figure 19: The promoter detects internal structures using face detection.

A special case of volumes are stacks of plates. Unlike any of the other plates in models, they are not connected using joints but instead glued on top of each other by users. Because there is no internal structure within a stack (it is all inherently filled with plate), the promoter simply creates a volume composed of the edges of the stack.

Figure 20 shows how the algorithm detected volumetric cells in three example models, and how volumetric stretch operations modify the cells yet keep the overall structure intact.

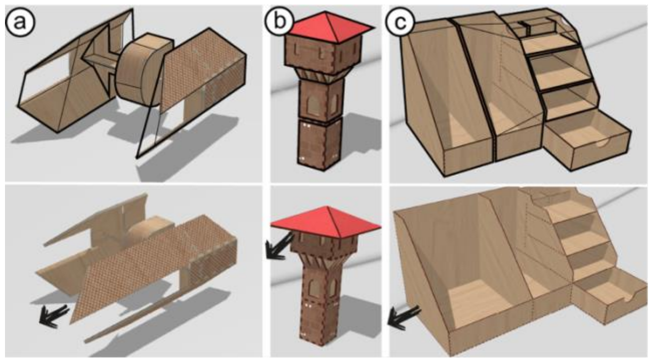

Figure 20: Three example models with their associated volumes as individual cells, the images below show how stretch operations applied to these models stretch these cells while keeping the structure of the model intact.

The explanation of the algorithm so far followed a bottom-up explanation; however, the actual algorithm proceeds in the opposite order, as shown in Algorithm 1. The algorithm recursively inserts proxy planes until all edges of the model are included in a volume. These planes in the next iteration are included as if they were actual plates often resulting in additional or bigger volumes to be found. This approach makes it easy to cache volumes as each tool interaction on the model only requires computing volumes on the newly added plates, extending the previously inferred volume.

ALGORITHM 1: promoter

Input: List of Edges in the model

Output: Constructed Volume, Cells

Internal data structures: Edges contain a Pointer to their Plate and what Edges on other Plates they connect to, Plates contain a Transform which orients them in 3D space and an EdgeCycle which is a linked list of the related Plates. CoplanarEdges contain pointer to the Edges they belong to.

```
// find all coplanar edges in the graph and store as coplanarEdges
coplanarEdges <- getCoplanarEdges(Edges)
cycles <- []
// 2D face detection splits up edges at internal plates, propagate the reference to edges
for plane in coplanarEdges:
  cycles.add (faceDetection(plane), plane.Edges)
clusters <- []
// flood fill closed cycles that share 2 plates along an edge
for cycle in cycles:
  for edge in cycle:
    if connecting two plates, add as cluster, remove from cycles
// handle non closed volumes and internal plates
while there are still cycles: // internal plates, add to both adjacent clusters
  for edge in cycle:
    if edge connects > two plates, add duplicate of cycle to clusters, remove from cycles
    // check if the cycle connects other cycles and insert proxy edges
    if edge connects to other cycle
      construct two planes through the points in both cycles and add proxy edge at the intersection,
      add to clusters,
      remove cycle from cycles
  // non-closing edges, use convex hull to construct proxy edges into the cycle add to clusters
  clusters.Add(2DQuickHull(cycle)), remove cycle from cycles
cells <- []
```

```
// construct cells
for cluster in clusters:
    if cluster contains proxy edges, insert corners at intersection between edges or proxy edges, generate proxy planes
    cell <- new Volume from linked list of plates in cluster, unify transforms of plates
    cells.add(cell)
// create the encompassing volume
Volume <-- union all cells
return Volume,cells
```

A limitation of this algorithm are models where it fails to construct corners because edges are all curved. Typical examples are skeleton structures with curved "ribs", the algorithm instead considers every point a corner and creates a lot of proxy faces. These produce the right volume, but no currently implemented volumetric tool makes productive use of that. More expressive fabrication-aware versions of volumetric operations like *Interactive Images* [Zheng 2012] and symmetry preserving editing [Lin 2011] support this, but that falls beyond the scope of this paper. As shown in Figure 21b, curved edges perform fine when stretching along the normal of the plane.

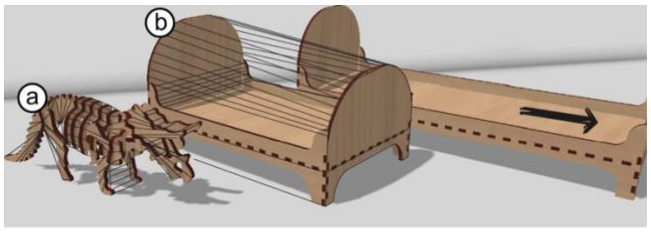

Figure 21: (a) Detected, but less useful volumes. (b) in this case the volume is still useful when stretched along the normal of the plane.

7 RELATED WORK

Our work builds on research in structure-preserving shape processing, inferring user intent, interoperability of CAD representations, and 3D editing for laser cutting.

7.1 Structure-Preserving Shape Processing

Preserving structure during editing tasks is crucial in various modeling workflows, and thus, has been a topic of interest in different fields. [Lin 2011] demonstrate its use in the context of preserving patterns in architectural design. Users select the structure to preserve and repeat. When they apply high-level volumetric changes, such as stretching, the algorithm updates the model to reflect the target aesthetic. [Fu 2016] instead machine-learn assembly structures in objects, so that when users edit them, they look for semantically similar objects that make up the same assembly structure. They demonstrate this by stretching an office chair, upon which the system returns a bench in similar style. *Im2Struct* [Niu 2018] uses a Convolutional Neural Network to detect similar 3D assembly structures in 2D images. Their approach is powerful in that it affords volumetric modifications to the virtual 3D model that are directly reflected in the 2D photograph using *interactive images* [Zheng 2012]. [Tian 2019] infer *3D shape programs*, which they use as a higher level of abstraction to improve 3D reconstruction. Their parametric shape programs contain semantic information about assembly (e.g., four "leg" objects connected to a "seat").

Surface2Volume [Araújo 2019] infers the internal structure of interlocking 3D printed shapes based on the color of their surface. [Kratt 2018] allow users to make sketch-based modifications to 3D models. They infer structures in models similar to the one users sketch on. Subsequently, their algorithm applies the suggested modification to similar sub-structures.

Structure-Aware Mesh Decimation [Salinas 2015] use the opposite approach to what we do; they start out with a messy volume and decimate it by finding planar surfaces. Their goal is to simplify a mesh. They maintain the structural properties of models during decimation following three topological rules based on a graph of proxy surfaces: (1) if two nodes of the graph are not connected, they cannot be connected through decimation (2) the proxies cannot be degenerated into a single vertex or edge, and (3) they construct and preserve corners. In structuring our algorithm, we were inspired by the reasoning behind these rules and applied some "in reverse"; using inferred corners to construct surface proxies, splitting volumes along intersecting plates to avoid swallowing them up into a larger volume (avoiding degeneration).

7.2 Inferring user intent

The promoter discussed in this work heavily builds on the idea of inferring the intention of users to predict the best way to support their workflow. [Gross 1996] approached this problem for 2D drawing, by inferring intent while the user is drawing. The more context users add to the drawing, the better the system detects their intention, avoiding the need for mode switches (e.g., between software to draw electronic circuits, charts, or images). Similarly, *Teddy* [Igarashi 1999] lets users draw 2D shapes and infers the volume intended by users.

Chateau [Igarashi 2007] shows a different approach to deal with ambiguous user intention by suggesting different outcomes and letting users disambiguate. The extreme of this is using statistical modeling to *predict* user behavior [Zukerman 2001].

Beyond inferring and predicting users' interactions, the notion of design intent in the context of 3D modeling is intended high-level parametric modifications to models [Kimura 1989]. [Otey 2018], show that this is more than just user-specified constraints and ranges anywhere from a single modeling step to the entire rationale underlying a model. Or as [Ganeshan 1994] describe in their framework for representing design intent: "it is not just about how but also *why* the design evolves as it does". Researchers therefore have looked for ways to capture such design intent in abstract representations, the *PARTs framework* [Hofmann 2018] allows users to specify *functional geometry objects* to represent object functionality and constraints. *Grafter* [Roumen 2018] similarly represents parts of 3D printed machines by representing the function of individual mechanisms. [Barbero 2018], show that capturing design intent not only helps in modeling, but it also serves to extend the level of expertise implicitly embedded in models.

7.3 Interoperability of CAD representations

Heterogeneous representations are a common issue in the domain of 3D modeling, [Attene 2018] provide an overview of representations and the trade-offs between them, their high-level categories are representations based on volumes, surfaces, primitives, and procedural generation. This variety of representations forces users to balance upfront how to represent objects for their given use-case [Gerbino 2003]. There are interchange formats like STEP [Pratt 2001], which in principle support the full range of representations, but in turn require a definition and maintenance of the content in all different representations and then allow switching between representations.

There are tools to convert between representations, especially converting from shape models as composed in general purpose 3D modeling environments to fabrication-aware representations. These are typically one-way conversions making it hard or impossible to reverse the process. In the context of laser-cutting, the most common conversion tool is *Slicer for Fusion360* (discontinued since 2020). This tool allowed users to convert 3D models to a range of different typical structures of plates approximating the initial shape. *Slices* [McCrae 2011] is a specialized version of this achieving an even stronger relation between the initial volume and resulting plate structure. *Fresh Press Modeler* [Chen 2016] and the follow-up publication on bevel joints [Su 2018] achieves such conversion specifically for volumetric and watertight models. Furthermore, *Platener* [Beyer 2015] and *CoFiFab* [Song 2016] convert generic shape models to partially laser-cut and partially 3D printed structures for fast fabrication and iteration.

Outside of the laser-cutting domain, there are various conversion tools [Gao 2004] however this tends to be lossy, making features that exist in another mode undiscoverable, and typically the conversion comes at a cost of expressivity, or even require fixing of models that break in the process [Pauwels 2011]. For example, [Wu 2005] recover structure of meshes that may result from poor 3D scans. *InverseCSG* [Du 2018] converts primitive models based on triangular boundary representations to a CSG tree, enabling powerful volumetric editing. Other approaches aim to identify higher level structures in the models such as [Fish 2014] who represent shape families, [Tulsiani 2017] who machine-learn using primitive volumes in models to identify abstract shapes, and *Grass* [Li 2017] detects shapes patterns allowing for high-level parametric operations. Finally, *Shape-up* [Bouaziz 2012] presents a geometry processing framework using projection operators that works reliably across polygonal meshes, volumetric meshes, point clouds and other discrete geometry representations.

7.4 3D editing for laser cutting

While laser cutters have been used for 2D fabrication traditionally, they gain momentum in producing advanced 3D models. Recent research investigated numerous ways to empower users in the process of creating 3D laser-cut models as part of personal fabrication [Baudisch 2017]. *LaserStacker* [Umapathi 2015] allows users to make 2.5D models by fusing stacks of plates together, *StackMold* [Valkeneers 2019] uses the same idea of plate stacks, but to create molds, which in turn create 3D objects. Furthermore, *LamiFold* [Leen 2020] enables users to create functional mechanisms using essentially 2.5D constructions of laminated plates. *Constructable*

[Mueller 2015] lets users construct 3D models for laser-cutting interactively in the machine. It forgoes the notion of an explicit 3D model, but encodes the artifact in combinations of elements users create. *LaserFactory* [Nisser 2021] extends laser cutters with other fabrication capacities to create an integrated assembly workflow from material to product.

Originally, making 3D models by joining laser-cut plates was achieved using 2D vector graphics tools like *CorelDraw* or *Adobe Illustrator*. Tools like *Joinery* [Zheng 2017] enable generation of reliable joints, where springFit [Roumen 2019] and kerfCanceler [Roumen 2020] maintain such joints when fabricated on a different machine. With the help of *CutCAD* [Heller 2018], users get an early preview of the resulting 3D design. Albeit domain specific, *SketchChair* [Saul 2011] presented an early 3D modeling environment for laser cutting based on sketched lines. *CODA* [Veuskens 2021] is a plugin for general-purpose 3D modeling environments (Fusion 360) that supports users by providing fabrication related constraints. *FlatFitFab* [McCrae 2014] and later *kyub* [Baudisch 2019] facilitated advances in 3D modelling for laser cutting by working directly on the assembled model instead of the 2D cutting plans. Models shared in a 3D format allow users to make high-level volumetric modifications, making these models much more valuable. *Assembler³* [Roumen 2021a] therefore allows users to convert 2D cutting plans to such 3D models, which was later automated by *autoAssembler* [Roumen 2021b].

In the closely related field of interactive carpentry, similar work in *fabrication-aware design* [Schwartzburg 2013] lead to computational support for the construction of interlocking furniture [Fu 2015]. [Rogeau 2021] generate joints for large-scale timber structures. Additionally, recent work [Noeckel 2021] reconstructs carpentry models based on images to allow for parametric modifications in 3D.

8 TECHNICAL EVALUATION: RE-CREATING 100 MODELS

To evaluate *structure-preserving editing*, we used our system to try and recreate the 100 models from the (assembler³ benchmark [Roumen 2021b], originally from thingiverse.com). the models from this benchmark were originally created using generic modeling software, thus exhibit a wide variety of construction methods.

We attempted to recreate these models using three systems, i.e., (1) volume-based (original, non-modified *kyub* [Baudisch 2019]), (2) plate-based (*FlatFitFab* [McCrae 2014]), and (3) volume + plate (the *structure-preserving* system presented above).

8.1 Results

Figure 22 shows the number of models we managed to recreate with each of the three approaches.

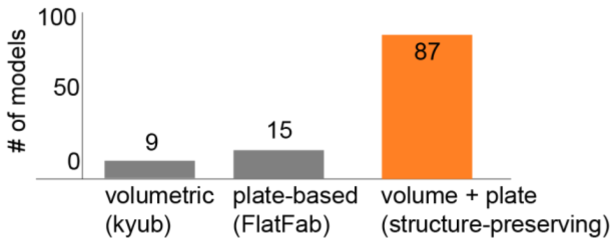

Figure 22: Models recreated using volumetric modeling (kyub), plate-based modeling (FlatFitFab) and our system.

As shown in Figure 23, the models exhibited different modeling workflows: (c) we recreated 53 models with multiple usages of the demoter/promoter, alternating between plate and volumetric workflows, (b) for 12 models we could use a waterfall process where the process is entirely volumetric (if done efficiently) with at the end plate tools demoting the model *exactly once*, (a) and we made the remaining 35 models using plate tools only like the ones shown in Figure 6.

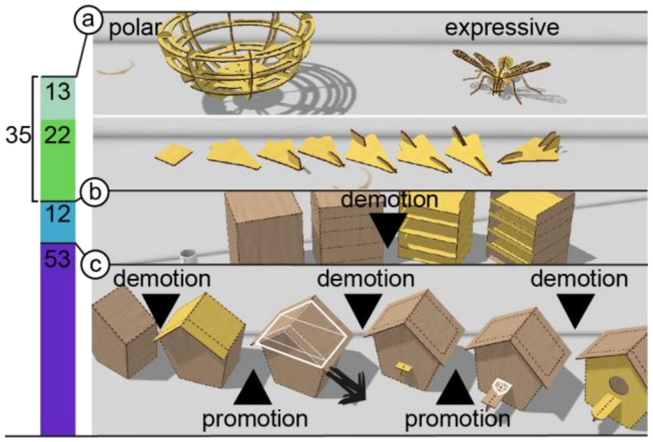

Figure 23: The 100 models of assembler[3] benchmark fall in three categories: (a) 35 models made using individual plate tools (b) 12 models made using a waterfall workflow and (c) 53 models that largely benefit from promotion/demotion in the modeling process.

All 13 models we could not recreate using our tools all fall in that last category; they do not benefit from promotion/demotion but would require a different set of plate tools. Six of them contain plates that are mapped to a polar coordinate system, our tools operate on a cartesian coordinate system, making it hard/impossible to recreate those. The other seven contain highly expressive plates, our system allows for curvature, but such detail is better done achieved using tools optimized for expressiveness (e.g., *FlatFitFab* [McCrae 2014]).

9 CONCLUSION

In this paper, we presented structure-preserving editing, an approach to bringing together the efficiency of volume-based editing with the control over details and structure offered by plate-based editing. We implemented this as two subsystems—one for volumes and one for plates. The key elements that hold the subsystems together are the promoter and demoter, which allow all tools to be applied to all structures.

Our approach of starting with a volumetric tool allowed us to create a particularly uniform tool set, as the limited expressiveness of the boxel-based building style made it easy to create matching plate tools. Our approach of plate-based tools + volume-based tools + promoter + demoter is generic in nature and inherently applicable to range of tools.

As shown in our technical evaluation, our approach increases the space of laser-cut 3D models substantially past those than could be created using either system, allowing those models to be created efficiently in fabrication-aware 3D environments, rather than generic 3D or even 2D editors.

Given that this paper is making a systems contribution, we did *not* present a user study [Greenberg 2008]. Nonetheless, we have put the system to actual use, such as by running a workshop were 8 teams of students from our department performed workflows similar to the one shown Figure 14 to design and fabricate playable custom string instruments (Figure 24).

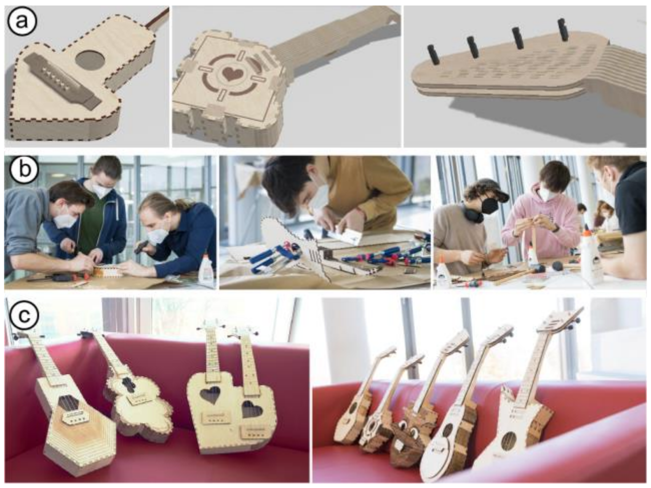

Figure 24: (a) 8 teams of students designed and (b) assembled (c) their instruments.

We expect structure-preserving editing to be impactful in that it will not only allow users to efficiently create models more advanced than seen previously, but also to increase the range of laser-cut 3D models that can be shared and remixed efficiently. The guitar shown in Figure 1, in that sense, is not about one guitar, but in the context of structure-preserving editing a starting point that allows users to create a wide-range of *custom* guitars efficiently. In that sense, the work presented in this paper is another important step in the transition from historical, machine-specific, and thus hard-to-modify laser cutting formats, (such as SVG), to easy-to-customize, sharing-friendly formats, 3D formats [Roumen 2021a] [Roumen 2021b].

REFERENCES

[1] Marco Attene, Marco Livesu, Sylvain Lefebvre, Thomas Funkhouser, Szymon Rusinkiewicz, Stefano Ellero, Jonàs Martínez, and Amit Haim Bermano. "Design, representations, and processing for additive manufacturing." Synthesis Lectures on Visual Computing: Computer Graphics, Animation, Computational Photography, and Imaging 10, no. 2 (2018): 1-146.

[2] Adobe Illustrator, https://www.adobe.com/products/illustrator.html, last accessed April 2022.

[3] Chrystiano Araújo, Daniela Cabiddu, Marco Attene, Marco Livesu, Nicholas Vining, and Alla Sheffer. 2019. Surface2Volume: surface segmentation conforming assemblable volumetric partition. ACM Trans. Graph. 38, 4, Article 80 (August 2019), 16 pages. DOI:https://doi.org/10.1145/3306346.3323004

[4] AutoDesk Fusion360. https://www.autodesk.com/products/fusion-360 last accessed April 2022.

[5] Bradford C. Barber, David P. Dobkin, and Hannu Huhdanpaa. "The quickhull algorithm for convex hulls." *ACM Transactions on Mathematical Software (TOMS)* 22, no. 4 (1996): 469-483.

[6] Basilio Ramos Barbero, Carlos Melgosa Pedrosa, and Gabriel Castrillo Peña. "The importance of adaptive expertise in CAD learning: maintaining design intent." *Journal of Engineering Design* 29, no. 10 (2018): 569-595.

[7] Patrick Baudisch and Stefanie Mueller. 2017. Personal Fabrication, Foundations and Trends in Human–Computer Interaction: Vol. 10: No. 3–4, pp 165-293. DOI: http://dx.doi.org/10.1561/1100000055

[8] Patrick Baudisch, Arthur Silber, Yannis Kommana, Milan Gruner, Ludwig Wall, Kevin Reuss, Lukas Heilman, Robert Kovacs, Daniel Rechlitz, and Thijs Roumen. 2019. Kyub: a 3D Editor for Modeling Sturdy Laser-Cut Objects. In *2019 CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019)*, May 4–9, 2019, Glasgow, Scotland, UK. ACM, New York, NY, USA. DOI: https://doi.org/10.1145/3290605.3300796

[9] Dustin Beyer, Serafima Gurevich, Stefanie Mueller, Hsiang-Ting Chen, and Patrick Baudisch. "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*, pp. 1799-1806. 2015.

[10] Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. "Shape-up: Shaping discrete geometry with projections." In Computer Graphics Forum, vol. 31, no. 5, pp. 1657-1667. Oxford, UK: Blackwell Publishing Ltd, 2012.

[11] Lujie Chen, and Lawrence Sass. "Fresh press modeler: A generative system for physically based low fidelity prototyping." Computers & Graphics 54 (2016): 157-165.

[12] James McCrae, Karan Singh, and Niloy J. Mitra. 2011. Slices: a shape-proxy based on planar sections. ACM Trans. Graph. 30, 6 (December 2011), 1–12. DOI:https://doi.org/10.1145/2070781.2024202

[13] James McCrae, Nobuyuki Umetani, and Karan Singh. 2014. FlatFitFab: interactive modeling with planar sections. In *Proceedings of the 27th annual ACM symposium on User interface software and technology (UIST '14).* ACM, New York, NY, USA, 13-22. DOI: https://doi.org/10.1145/2642918.2647388

[14] CorelDRAW.com, last accessed April 2022.

[15] Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. "Inversecsg: Automatic conversion of 3d models to csg trees." *ACM Transactions on Graphics (TOG)*37, no. 6 (2018): 1-16.

[16] Noa Fish, Melinos Averkiou, Oliver van Kaick, Olga Sorkine-Hornung, Daniel Cohen-Or, and Niloy J. Mitra. 2014. Meta-representation of shape families. *ACM Trans. Graph.* 33, 4, Article 34 (July 2014), 11 pages. DOI:https://doi.org/10.1145/2601097.2601185

[17] Chi-Wing Fu, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. 2015. Computational interlocking furniture assembly. *ACM Trans. Graph.* 34, 4, Article 91 (August 2015), 11 pages. DOI:https://doi.org/10.1145/2766892

[18] Qiang Fu, Xiaowu Chen, Xiaoyu Su, Jia Li, and Hongbo Fu. "Structure-adaptive shape editing for man-made objects." In Computer Graphics Forum, vol. 35, no. 2, pp. 27-36. 2016.

[19] Rajaram Ganeshan, James Garrett, and Susan Finger. "A framework for representing design intent." *Design Studies* 15, no. 1 (1994): 59-84.

[20] Jian Gao, Detao Zheng, and Nabil Gindy. "Mathematical representation of feature conversion for CAD/CAM system integration." *Robotics and Computer-Integrated Manufacturing*20, no. 5 (2004): 457-467.

[21] Salvatore Gerbino, "Tools for the interoperability among CAD systems." In Proc. XIII ADM-XV INGEGRAF Int. Conf. Tools and Methods Evolution in Engineering Design. 2003.

[22] Saul Greenberg and Bill Buxton. 2008. Usability evaluation considered harmful (some of the time). In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08). Association for Computing Machinery, New York, NY, USA, 111–120. DOI:https://doi.org/10.1145/1357054.1357074

[23] Mark D. Gross and Ellen Yi-Luen Do. 1996. Ambiguous intentions: a paper-like interface for creative design*. In Proceedings of the 9th annual ACM symposium on User interface software and technology* (UIST '96). Association for Computing Machinery, New York, NY, USA, 183–192. DOI:https://doi.org/10.1145/237091.237119

[24] Florian Heller, Jan Thar, Dennis Lewandowski, Mirko Hartmann, Pierre Schoonbrood, Sophy Stönner, Simon Voelker, and Jan Borchers. 2018. CutCAD - An Open-source Tool to Design 3D Objects in 2D. In *Proceedings of the 2018 Designing Interactive Systems Conference (DIS '18).* ACM, New York, NY, USA, 1135-1139. DOI: https://doi.org/10.1145/3196709.3196800

[25] Megan Hofmann, Gabriella Hann, Scott E. Hudson, and Jennifer Mankoff. 2018. Greater than the Sum of its PARTs: Expressing and Reusing Design Intent in 3D Models. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18).* ACM, New York, NY, USA, Paper 301, 12 pages. DOI: https://doi.org/10.1145/3173574.3173875

[26] Takeo Igarashi and John F. Hughes. 2007. A suggestive interface for 3D drawing*. In ACM SIGGRAPH 2007 courses(*SIGGRAPH '07). Association for Computing Machinery, New York, NY, USA, 20–es. DOI:https://doi.org/10.1145/1281500.1281531

[27] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99*). ACM Press/Addison-Wesley Publishing Co., USA, 409–416. https://doi.org/10.1145/311535.311602

[28] Fumihiko Kimura, and Hiromasa Suzuki. "A CAD system for efficient product design based on design intent." *CIRP annals*38, no. 1 (1989): 149-152.

[29] Julian Kratt , Till Niese, Ruizhen Hu, Hui Huang, Sören Pirk, Andrei Sharf, Daniel Cohen-Or, and Oliver Deussen. "Sketching in gestalt space: Interactive shape abstraction through perceptual reasoning." In Computer Graphics Forum, vol. 37, no. 6, pp. 188-204. 2018.

[30] Danny Leen, Nadya Peek, and Raf Ramakers. "LamiFold: Fabricating Objects with Integrated Mechanisms Using a Laser Cutter Lamination Workflow." In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pp. 304-316. 2020.

[31] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. "Grass: Generative recursive autoencoders for shape structures." *ACM Transactions on Graphics (TOG)* 36, no. 4 (2017): 1-14.

[32] Markus Lipp, Peter Wonka, and Pascal Müller. 2014. PushPull++. ACM Trans. Graph. 33, 4, Article 130 (July 2014), 9 pages. DOI:https://doi.org/10.1145/2601097.2601197

[33] Jinjie Lin, Daniel Cohen-Or, Hao Zhang, Cheng Liang, Andrei Sharf, Oliver Deussen, and Baoquan Chen. 2011. Structure-preserving retargeting of irregular 3D architecture. ACM Trans. Graph. 30, 6 (December 2011), 1–10. DOI:https://doi.org/10.1145/2070781.2024217

[34] Makeabox.io, last accessed March 2022

[35] Stefanie Mueller, Pedro Lopes, and Patrick Baudisch. 2012. Interactive construction: interactive fabrication of functional mechanical devices*. In Proceedings of the 25th annual ACM symposium on User interface software and technology(UIST '12).* Association for Computing Machinery, New York, NY, USA, 599–606. DOI:https://doi.org/10.1145/2380116.2380191

[36] David E. Muller, and Franco P. Preparata. "Finding the intersection of two convex polyhedra." *Theoretical Computer Science* 7, no. 2 (1978): 217-236

[37] Martin Nisser, Christina Chen Liao, Yuchen Chai, Aradhana Adhikari, Steve Hodges, and Stefanie Mueller. 2021. LaserFactory: A Laser Cutter-based Electromechanical Assembly and Fabrication Platform to Make Functional Devices & Robots*. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21).* Association for Computing Machinery, New York, NY, USA, Article 663, 1–15. DOI:https://doi.org/10.1145/3411764.3445692

[38] Chengjie Niu, Jun Li, and Kai Xu. "Im2struct: Recovering 3d shape structure from a single rgb image." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4521-4529. 2018.

[39] James Noeckel, Haisen Zhao, Brian Curless, and Adriana Schulz. "Fabrication-Aware Reverse Engineering for Carpentry." In *Computer Graphics Forum*, vol. 40, no. 5, pp. 301-314. 2021.

[40] Jeffrey Otey, Pedro Company, Manuel Contero, and Jorge D. Camba. "Revisiting the design intent concept in the context of mechanical CAD education." *Computer-Aided Design and Applications* 15, no. 1 (2018): 47-60.

[41] OnShape.com, last accessed April 2022

[42] Pieter Pauwels, Davy Van Deursen, Jos De Roo, Tim Van Ackere, Ronald De Meyer, Rik Van de Walle, and Jan Van Campenhout. "Three-dimensional information exchange over the semantic web for the domain of architecture, engineering, and construction." *Ai Edam* 25, no. 4 (2011): 317-332.

[43] Michael J. Pratt, "Introduction to ISO 10303—the STEP standard for product data exchange." *Journal of Computing and Information Science in Engineering* 1, no. 1 (2001): 102-103.

[44] Nicolas Rogeau, Pierre Latteur, and Yves Weinand. "An integrated design tool for timber plate structures to generate joints geometry, fabrication toolpath, and robot trajectories." *Automation in Construction* 130 (2021): 103875.

[45] Thijs Roumen, Ingo Apel, Jotaro Shigeyama, Abdullah Muhammad, and Patrick Baudisch. 2020. Kerf-Canceling Mechanisms: Making Laser-Cut Mechanisms Operate across Different Laser Cutters. *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology.* Association for Computing Machinery, New York, NY, USA, 293–303. https://doi.org/10.1145/3379337.3415895

[46] Thijs Roumen, Yannis Kommana, Ingo Apel, Conrad Lempert, Markus Brand, Erik Brendel, Laurenz Seidel, Lukas Rambold, Carl Goedecken, Pascal Crenzin, Ben Hurdelhey, Muhammad Abdullah and Patrick Baudisch. 2021. Assembler^3: 3D reconstruction of laser-cut models. In *CHI Conference on Human Factors in Computing Systems* (CHI '21), May 8–13, 2021, Yokohama, Japan. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3411764.3445453

[47] Thijs 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 (CHI '18).* ACM, New York, NY, USA, Paper 63, 12 pages. DOI: https://doi.org/10.1145/3173574.3173637

[48] Thijs Roumen, Conrad Lempert, Ingo Apel, Erik Brendel, Markus Brand, Laurenz Seidel, Lukas Rambold, and Patrick Baudisch. 2021. autoAssembler: Automatic Reconstruction of Laser-Cut 3D Models. In The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA. ACM, New York, NY, USA, 11 pages. https: //doi.org/10.1145/3472749.3474776

[49] Thijs Roumen, Jotaro Shigeyama, Julius Cosmo Romeo Rudolph, Felix Grzelka, and Patrick Baudisch. 2019. SpringFit: Joints and Mounts that Fabricate on Any Laser Cutter*. In Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19).* Association for Computing Machinery, New York, NY, USA, 727–738. https://doi.org/10.1145/3332165.3347930

[50] David Salinas, Florent Lafarge, and Pierre Alliez. "Structure-aware mesh decimation." In Computer Graphics Forum, vol. 34, no. 6, pp. 211-227. 2015.

[51] Greg Saul, Manfred Lau, Jun Mitani, and Takeo Igarashi. 2011. SketchChair: an all-in-one chair design system for end users*. In Proceedings of the fifth international*

conference on Tangible, embedded, and embodied interaction (TEI '11). Association for Computing Machinery, New York, NY, USA, 73–80. DOI:https://doi.org/10.1145/1935701.1935717

[52] Yuliy Schwartzburg, and Mark Pauly. "Fabrication-aware design with intersecting planar pieces." In *Computer Graphics Forum*, vol. 32, no. 2pt3, pp. 317-326. Oxford, UK: Blackwell Publishing Ltd, 2013.

[53] Slicer for Fusion 360, last accessed March 2022, https://knowledge.autodesk.com/support/fusion-360/downloads/caas/downloads/content/slicer-for-fusion-360.html

[54] Peng Song, Bailin Deng, Ziqi Wang, Zhichao Dong, Wei Li, Chi-Wing Fu, and Ligang Liu. 2016. CofiFab: coarse-to-fine fabrication of large 3D objects. ACM Trans. Graph. 35, 4, Article 45 (July 2016), 11 pages. DOI:https://doi.org/10.1145/2897824.2925876

[55] Zhilong Su, Lujie Chen, Xiaoyuan He, Fujun Yang, and Lawrence Sass. "Planar structures with automatically generated bevel joints." Computers & Graphics 72 (2018): 98-105.

[56] Thingiverse, last accessed April 2022, http://www.thingiverse.com

[57] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. "Learning to infer and execute 3d shape programs." arXiv preprint arXiv:1901.02875 (2019).

[58] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. "Learning shape abstractions by assembling volumetric primitives." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2635-2643. 2017.

[59] Udayan Umapathi, Hsiang-Ting Chen, Stefanie Mueller, Ludwig Wall, Anna Seufert, and Patrick Baudisch. "LaserStacker: Fabricating 3D objects by laser cutting and welding." In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pp. 575-582. 2015.

[60] Tom Valkeneers, Danny Leen, Daniel Ashbrook, and Raf Ramakers. "StackMold: Rapid Prototyping of Functional Multi-Material Objects with Selective Levels of Surface Details." In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pp. 687-699. 2019.

[61] Tom Veuskens Florian Heller, and Raf Ramakers. "CODA: A Design Assistant to Facilitate Specifying Constraints and Parametric Behavior in CAD Models." *In Proceedings of the 47th Graphics Interface Conference on Proceedings of Graphics Interface 2021 (GI'21)*

[62] Jianhua Wu, and Leif Kobbelt. "Structure Recovery via Hybrid Variational Surface Approximation." In *Comput. Graph. Forum*, vol. 24, no. 3, pp. 277-284. 2005.

[63] Clement Zheng, Ellen Yi-Luen Do, and Jim Budd. "Joinery: Parametric joint generation for laser cut assemblies." In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, pp. 63-74. 2017.

[64] Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and Niloy J. Mitra. 2012. Interactive images: cuboid proxies for smart image manipulation. ACM Trans. Graph. 31, 4, Article 99 (July 2012), 11 pages. DOI:https://doi.org/10.1145/2185520.2185595

[65] Ingrid Zukerman, and David W. Albrecht. "Predictive statistical models for user modeling." *User Modeling and User-Adapted Interaction* 11, no. 1 (2001): 5-18.