Roadkill: Nesting Laser-Cut Objects for Fast Assembly

MUHAMMAD ABDULLAH, ROMEO SOMMERFELD, LAURENZ SEIDEL, JONAS NOACK, RAN ZHANG, THIJS ROUMEN, PATRICK BAUDISCH

Hasso Plattner Institute, University of Potsdam, Germany, firstname.lastname@hpi.de

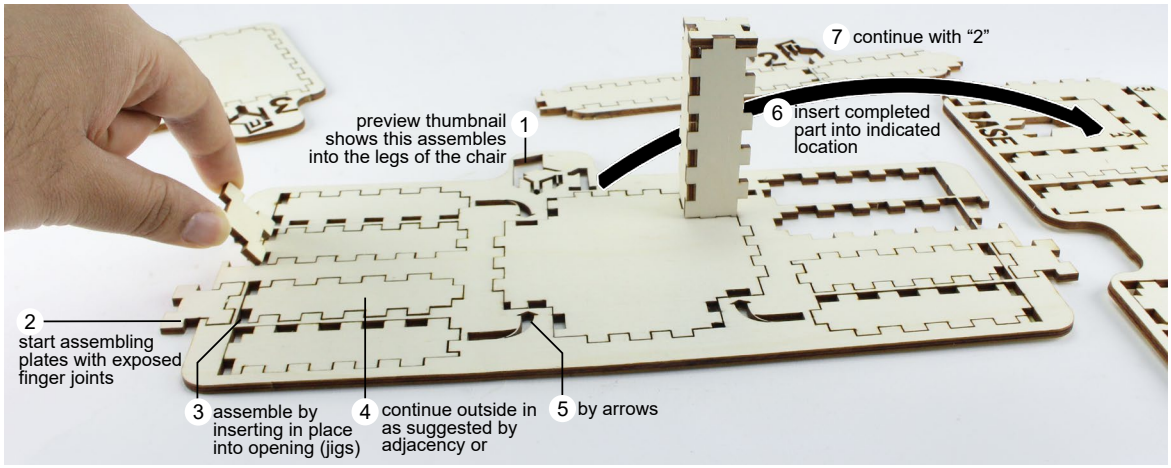

Figure 1: Our software tool *Roadkill* converts 3D models to 2D cutting plans for laser cutting—such that the resulting layout allows for *fast assembly*. Roadkill achieves this with the help of a visual language that conveys assembly instructions directly in the generated layout and by collocating plates to be joined, thereby minimizing visual search.

We present Roadkill, a software tool that converts 3D models to 2D cutting plans for laser cutting—such that the resulting layouts allow for *fast assembly*. Roadkill achieves this by putting all relevant information into the cutting plan: (1) *Thumbnails* indicate which area of the model a set of parts belongs to. (2) *Parts with exposed finger joints* are easy to access, thereby suggesting to start assembly here. (3) Openings in the sheet act as *jigs*, affording assembly *within* the sheet. (4) Users continue assembly by inserting what has already been assembled into parts that are immediately *adjacent* or are pointed to by *arrows*. Roadkill maximizes the number of joints rendered in immediate *adjacency* by breaking down models into "subassemblies." Within a subassembly, Roadkill holds the parts together using *break-away tabs*. (5) Users complete subassemblies according to their labels 1, 2, 3…, following 1 -> 1 links to insert subassemblies into other subassemblies, until all parts come together. In our user study, Roadkill allowed participants to assemble layouts *2.4 times faster* than layouts generated by a traditional pair-wise labeling of plates.

CCS CONCEPTS • •Human-centered computing~Human computer interaction (HCI)~Interactive systems and tools

Additional Keywords and Phrases: Personal fabrication, laser cutting, rapid prototyping, manual assembly

ACM Reference Format:

Anonymous for review

1 INTRODUCTION

While 3D printing continues to be one of the key technologies in rapid prototyping, laser cutting is catching up. With the ability to produce even large parts within seconds, laser cutters often allow producing models orders of magnitude faster than 3D printers [4]. A recent milestone in this evolution towards fast fabrication with laser cutting is a series of software systems that allow users to design laser cut 3D models directly in 3D and then automatically convert these 3D models to the 2D cutting plans demanded by the laser cutter (*FlatFitFab* [25], *Platener* [6], *Kyub* [5]).

As a result, the first two phases of the laser cutting workflow *design and cutting*, have become very fast and, as a result, it is now often the third phase, *assembly*, that becomes the bottleneck. For example the laser cut chair model shown in Figure 1 took 3 minutes to design in Kyub [5] and 3 minutes to laser-cut on a *Trotec Speedy 360* laser cutter [44], but it took our study participants an average of 22 minutes to assemble, resulting in the assembly time accounting for 80% of the overall fabrication time. Since our objective is to accelerate laser cutting further, assembly is the phase to tackle.

Closer examination reveals why assembly is slow. As shown in Figure 2, the cutting plans produced by *FlatFitFab* [25] and *Kyub* [5] use pairs of IDs (here numbers) to communicate which parts the users are supposed to join during assembly. When assembling a joint, users pick a joint, read its ID, and then visually search the cutting plan for a matching ID. This visual search consumes time proportional to the number of IDs on the cutting plan and given that users do it at least once per plate, the resulting overall user effort caused by visual search is *quadratic* in the number of plates. With all the other steps in the workflow being of *linear* time complexity, this *quadratic* effort not only forms the bottleneck of the assembly, but the bottleneck of the entire laser cutting workflow.

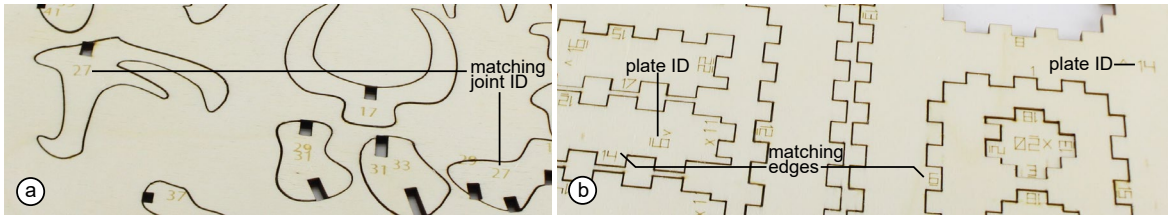

Figure 2: (a) Cutting plans produced by *FlatFitFab* [25] use pairs of numbers to communicate which parts to join during assembly. (b) *Kyub* [5] follows the same approach. Locating a matching number takes users time proportional to the number of plates.

In this paper, we present *Roadkill*, a software tool that produces 2D cutting plans that are *not* subject to this bottleneck. As illustrated by Figure 1, Roadkill features a visual language encompassing 10 design elements, the key element of which is that cutting plans produced by Roadkill *guide* the users from joints to their matching counterpart: in most cases users find the counterpart in immediate adjacency; in some cases, users follow arrows which Roadkill adds to the cutting plan. In our user study, participants assembled the cutting plans generated by Roadkill 2.4 times faster than traditional cutting plans which are based on a pair-wise labeling of plates.

The obvious benefit of Roadkill is that it offers a substantial quantitative benefit: a speed-up of 2.4x. This quantitative benefit translates into a qualitative benefit, in that it enables a novel application scenario: Similar to how 3D printing enabled "overnight prototyping", and how software-supported laser cutting enabled "between meeting prototyping", we see Roadkill enabling "within-meeting prototyping." We see Roadkill allowing prototyping to take place within a brainstorming session, i.e., it allows participants exchanging ideas to support these ideas with prototypes designed and fabricated during the meeting.

2 ROADKILL WALKTHROUGH

Expanding on the example of Figure 1, the following walkthrough shows how a user assembles a chair model.

As illustrated by Figure 3a, Roadkill fabricates the chair model in four separate parts, which we will refer to as "subassemblies." Subassemblies are a necessary side effect that allows Roadkill to create easy-to-assemble layouts, as we explain in detail below. The four subassemblies are labeled "1", "2", "3", and "base". Users start with subassembly "1". The number of subassemblies is small (here 4), and therefore, requires a minimal amount of visual search. (d) Subassemblies are surrounded by a frame that is cut through but (c) moving them around is easy, as plates within a subassembly are held in place by tiny "break-away tabs".

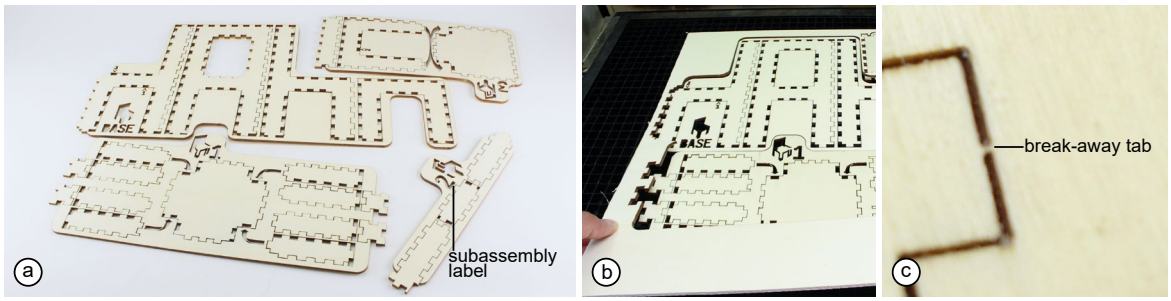

Figure 3: Roadkill has exported the chair in the form of (a) four "subassemblies" surrounded by frames that are (b) cut through while plates within are connected to the frame by (c) breakaway tabs.

Figure 4 shows a user assembling subassembly "1". (a) A thumbnail shows that this subassembly builds the bottom of the chair and its legs. Like all visual elements created by Roadkill, the thumbnail is *cut*, so it can be seen from either side. Allowing models to have engraved decorative elements without the need to flip the plate in the laser cutter. (b) The subassembly is enclosed in a thin rounded frame, but the finger joints of one or more plates (here 4) are sticking out; these are easy to grab, which is how Roadkill indicates where to start. (c) The user grabs one of the exposed "starting plates" and bends it *upwards*—breaking the tab that was holding it in place. *Breakaway tabs* are placed along the inner edge allowing the plate to act as a lever and affording this specific movement.

This plate has a single immediate neighbor, which is Roadkill's default way of communicating which plate to join. Rectangular cavities in this neighboring plate form a "jig", affording the small plate to be inserted. (d) The user inserts the small starting plate. Note how the entire subassembly acts as jig, holding the emerging 3D model in place. Roadkill enables this by ensuring that plates are always inserted *from above*.

As before, this plate has only a single immediate neighbor. The user thus extracts the two-plate-structure from the subassembly, and (e) inserts them into the jig of the adjacent neighbor.

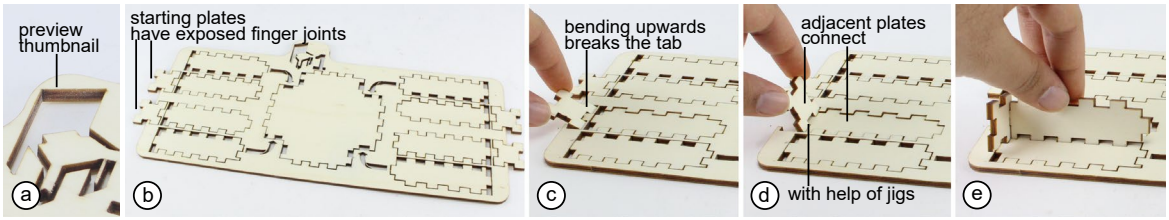

Figure 4: (a) A "preview thumbnail" indicates the subassemblies location in the model. Roadkill affords a simple "outside-in" assembly order. (b) Starting plates are indicated by "exposing" them outside the frame which allows the user to (c) easily break them out and (d) connect with the adjacent plates using "jigs".

As illustrated by Figure 5a, this plate has no immediate neighbor (the reason is that the local geometry of the chair forms a cavity here, preventing plates from being in immediate adjacency (see *algorithm*). Instead, this time an arrow points to the jig to insert into. (b) This arrow has a twist which suggests *rotating* the plate before insertion, while the asymmetric jig prevent erroneous insertion. (c) The user performs this rotation and inserts the plates. The user has now completed (the inner part of) a first chair leg.

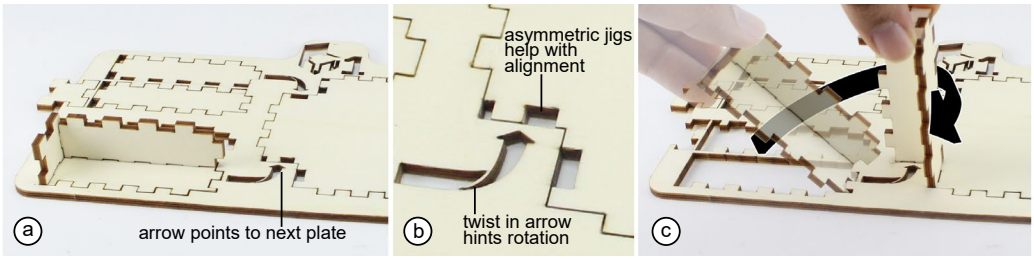

Figure 5: (a) In case the next plate in the assembly order cannot be placed adjacently, Roadkill uses an arrow to indicate it. (b) A twist in the arrow hints that the plate has to be (c) rotated before insertion, while asymmetric jigs help with alignment.

At this point the user has reached the "central plate" of this subassembly. As shown by Figure 6, the central plate cannot be connected elsewhere in the subassembly as it features neither adjacent jigs nor outward pointing arrows. The user thus continues assembling with the remaining three starting plates, assembling all four legs. Within the subassembly, Roadkill ensures that all sequences initiated by the starting plates can be assembled in any order—or in parallel, in case multiple users would like to collaborate.

(b) Upon completion of subassembly "1", the user extracts the emerging model and inserts it into the subassembly that contains the target label "1" + "arrow hat", here the "base". The placement of the "1" on the first subassembly and the placement of "1" + "arrow hat" on the second subassembly disambiguates the necessary rotation. As before, the asymmetric jig prevents erroneous insertion.

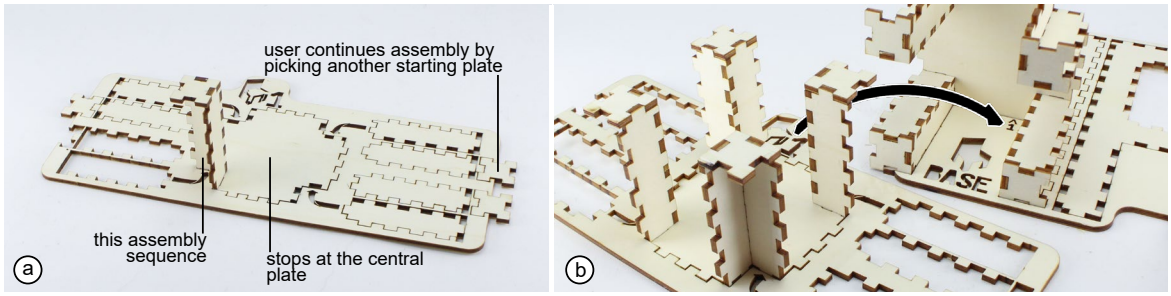

Figure 6: (a) All assembly sequences are terminated at a central plate and are assembled independently of each other. (b) After completing subassembly "1", the user takes it out and inserts it into the "base" subassembly as indicated by the matching numeric label.

As shown in Figure 7,(a) the user now assembles subassembly "2", (b) inserts it into subassembly "3". Note how the arrow flips the "backrest" before (c) inserting it into the "seat". To ensure the aforementioned assembly from above the "seat" is mirrored (see *algorithm*). (d) The user then inserts subassembly "3" into the only remaining subassembly (called "base"), which (e) completes the chair model.

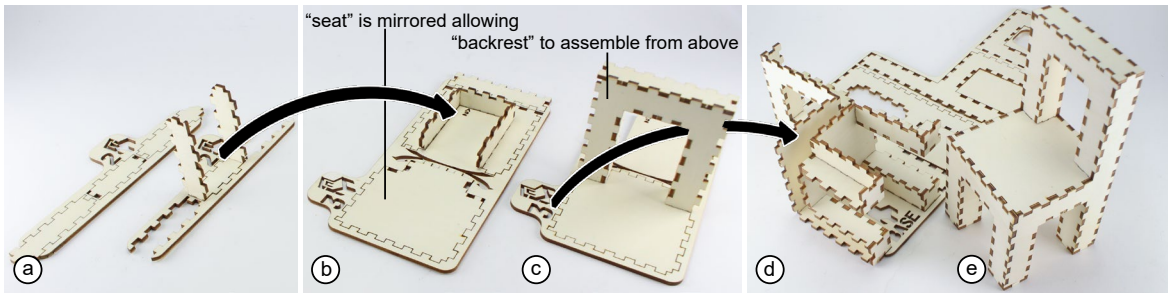

Figure 7: (a) The user assembles "2" and (b) inserts it into "3". Here the "seat" plate is mirrored to allow the "backrest" to be (c) inserted from above (d) The subassembly "base" receives "3" and is assembled last to (e) complete the chair model.

The key to the Roadkill assembly process is the use of adjacency and/or arrows as a means of eliminating visual search, thereby achieving fast assembly. For trivial models, such as fully convex, models Roadkill can place plates in adjacency by simply "unrolling" (aka "flattening" [36]) the model. For more complex models, including the chair model shown above, the naïve attempt to flatten the model results in multiple plates being placed at the same position, resulting in an erroneous layout. Roadkill resolves this by breaking down the model into multiple parts—the aforementioned subassemblies. The algorithm that breaks down models into just the right subassemblies is the core algorithmic contribution of this paper.

3 CONTRIBUTIONS & LIMITATIONS

In this paper, we make four contributions.

First, we identify manual assembly as the current bottleneck of rapid prototyping on laser-cutters based on its quadratic complexity. Second, we propose a novel cutting plan layout as well as a visual language communicating this layout. The main benefit of Roadkill is fast assembly. In our user study, participants assembled a test object 2.4x faster when that model was laid out by Roadkill than when laid out using the traditional approach which uses pairs of numbers.

Third, we present an algorithm and software tool that converts 3D models into this layout and visual language. Fourth, we integrate our tool into an existing 3D editor (*Kyub* [5]).

Roadkill's visual language encompasses the following 10 design elements: Roadkill communicates matching joints by (1) means of *adjacency* and (2) by means of *arrows*. Both are enabled by (3) *subassemblies*, held together by (4) *break-away tabs*. (5) *Thumbnails* preview the contents of each subassembly. Users start assembly with plates marked by (6) *exposed finger joints* and continue (7) *outside-in*. (8) *Subassembly labels* indicate assembly order. Plates/subassemblies are always (9) *inserted from above* into t-joint shaped (10) *jigs*.

Roadkill is subject to three main limitations. First, Roadkill does not handle cross joints. Second, models that are heavily interlocked, i.e., models where the removal of one plate allows access to only one additional plate, can result in a high number of subassemblies. Third, Roadkill consumes additional material (37% in the case of Figure 1). It uses the extra space to get the annotations that where historically engraved into parts, off the parts.

4 RELATED WORK

This work builds on research on rapid prototyping, assembly/disassembly in fabrication, techniques supporting manual assembly, and nesting.

4.1 Rapid prototyping

Speeding up prototyping is a key research topic in personal fabrication. Different approaches have been applied to speed up 3D printing, e.g. Wireprint [27], which prints fast wireframe previews and faBrickation [28], which substitutes parts of models with building blocks reducing the amount of 3D printing required. More recently, in Toward Support-Free 3D Printing [49], Wei et al. reduce fabrication time by partitioning the model in a way that it no longer requires support material. In Pop-up Print [30], Noma et al. propose a technique to 3D print objects in a folded state which reduces the required volume and support material, thus speeding up the fabrication process. Similarly, researchers have used other techniques besides 3D printing to facilitate rapid prototyping, such as WireFab [23] that bends metal wires to create structures and ProtoMold [54] that provides interactive vacuum forming to facilitate rapid prototyping.

Laser cutting on the other hand is by default a fast fabrication technology, due to the fact that it can quickly produce multiple plates at once. In StackMold [46], Valkeneers et al. propose using it to quickly create molds, while LamiFold [20] and LaserFactory [29] propose integrating mechanisms and electronics during the fabrication process to quickly produce a working prototype. Typically, 2D SVGs are used for developing and sharing laser cutting plans and tools e.g. Constructable [26] and Joinery [56] have helped users design their models. Similarly, CutCAD [17] helps users model 3D objects using a 2D format.

However, 2D modelling slows down the design process creating a bottleneck for laser cutting based fabrication. Recent advances in this domain such as FlatFitFab [25], Platener [6] and Kyub [5] allow users to model in 3D effectively speeding up the modelling process. However, this has highlighted another part of the fabrication process, *assembly*, as the new bottleneck. With Roadkill, we aim to remove this bottleneck and bring the next speed up to personal fabrication.

4.2 Assembly and disassembly in fabrication

Finding a valid sequence of steps to assemble an object has been a widely researched topic. Wilson et al. [50] proposed directional blocking graphs as a way to identify and store moveable parts in an assembly at every stage, allowing for efficient computation of an assembly plan. This approach has been widely used to analyze the complexity of assembly sequences [33], to find valid assembly sequences for furniture design [55] and more recently to create a framework for

designing interlocking assemblies [47]. Jimenez et al. [18] and Ghandi et al. [15] provide a comprehensive surveys of assembly planning techniques.

Approaches based on using disassembly sequences to find an assembly sequence have also been explored. In Fabrication-aware Design with Intersecting Planar Pieces [37], Schwartzburg et al., leverage this approach to generate models that have a valid assembly tree. Cignoni et al. use a similar approach to create an assembly order with the additional constraint that the structure remains stable at each step [10]. Similar approaches have been used to create interlocking puzzles [43], interlocking assemblies [48], and LEGO Technic based models [53]. In Assembling Self-Supporting Structures [12], Deuss et al. propose a method to choose an optimal assembly sequence that ensures sturdy gradual construction using the least number of materials. Fu et al. disassemble pieces out of a parts graph to create a valid assembly tree for interlocking furniture [14], while Song et al. extend this technique to create reconfigurable Interlocking Furniture [41]. Converse to this approach, Skouras et al. [40] used the modelling history of the object to generate an assembly order.

We build our work on the disassembly-based approach by finding branches in the assembly tree that can be assembled independently of each other, while creating subassembly groups for branches that cannot be assembled independently. This simplifies the assembly process, reducing the burden on the user building the model and in turn speeding up the process.

4.3 Supporting the manual assembly process

Assembly instructions are typically an integral part of a product. Many commercial products such as flat packed furniture (e.g., IKEA [13]) or constructions kits (e.g. LEGO [21]) come with an assembly instruction manual. Agrawala et al. generalized this approach and provided a framework for creating an instruction manual for different kinds of assemblies [2]. To achieve this, they use directional blocking graphs proposed by [50] to find valid assembly sequences and then create a manual based on them. Li et al. [22] and Guo et al. [16] have developed similar approaches to depict an exploded view (disassembly) of 3D models, while Antifakos et al. [3] propose a proactive appraoch that provides appropriate instructions based on the current state of the assembly. Shao et al. propose the reverse process by recovering the 3D model of the object through analysis of the assembly instruction manual [39]. Assembler^3 performs a similar process by reconstructing a 3D model of an object by parsing the 2D laser cutting plan [34].

However, an effective method to communicate assembly instructions has been missing in the context of laser cutting. A probable reason is that typically laser-cut models were designed in 2D and only had a small number of plates, thus lacking a complex assembly sequence. More complex models on the other hand, designed by experts and sold as construction kits (e.g. uGears [45]), provide assembly instructions using a combination of plate grouping, break-away tabs, engraved numbers and paper manuals.

However, recent advances in 3D modelling software for laser cutting such as FlatFitFab [25] and Kyub [5] allow novice users to quickly create highly complex models consisting of hundreds of plates. While these software provide assembly instructions in the form of engraved pair-wise numbers, we deem this approach time consuming and propose Roadkill as a solution to effectively communicate assembly instructions and speed up the assembly process.

4.4 Generating 2D cutting layouts

Creating a 2D non-overlapping arrangement of parts is typically known as the 2D packing problem or nesting [1]. The goal is usually to produce a tight packing and the most common logic used is the no fit polygon (NFP) which typically provides good polygon placement based on detected overlaps. Burke et al., used an approximate NFP approach along

with metaheuristic algorithms to provide a practical packing solution [7][8]. Several open-source projects such as SVGnest [42] and Deepnest [11] are based on the NFP approach.

Researchers have also explored interactive nesting solutions. Camera based manual nesting solutions are provided by VisiCut [31] and PacCam [35], while a projector based approach is explored by VAL [51] and ProjecTables [32]. JigFab [19] analyzes the users design and alerts them if their design cannot fit on the materials available to them. Similarly, Fabricaide [38] proposes a real-time nesting view that updates during the design process, keeping the user aware of how much material is required, allowing them to adjust their design accordingly. Other than reducing packing volume to save material, researchers have also looked at how to increase efficiency. In Carpentry Compiler [52], Wu et al. layout parts in a way that they share the maximum amount of edges effectively reducing the number of cuts required.

In our approach, we focus on creating a layout that can be assembled efficiently. Plates within a subassembly are laid out by the Roadkill algorithm either adjacently or linked through arrows (based on assembly order), while the subassemblies are nested using the approach discussed in [9].

5 THE ROADKILL ALGORITHM

We now present the Roadkill algorithm that converts a 3D model into the type of 2D cutting plan presented earlier. We again use the example of the chair model from Figure 1.

The algorithm proceeds in four steps. (1) the algorithm determines an assembly order, (2) subdivides it into subassemblies with independently assemblable branches, (3) creates the 2D layout and finally (4) creates the design elements (arrows, labels, jigs, etc.) to generate the 2D cutting plan (which it then saves in SVG format).

5.1 Step 1: Roadkill determines assembly order by exploring how to *disassemble* the model

Roadkill determines a possible assembly order "in reverse", i.e., it starts with the fully assembled model and explores ways of *removing* plates from it, until fully disassembled. It operationalizes this process in the form of a "disassembly tree". Roadkill then obtains the desired assembly tree by inverting the disassembly tree.

In order to create the disassembly tree, Roadkill starts by creating a list of plates that can be removed. It then chooses the plate that features the largest number of joints (the largest area, in case of a draw) to act as the "key" plate. At each hierarchy level of the disassembly tree, the key plate acts as the central node. Multiple key plates can be present at the same hierarchy level if the model splits into disconnected components. Roadkill removes the key plate from the model and adds it to the disassembly tree. In case of the chair model (Figure 8a), it chooses the chair's "back" plate. (b) In the next step, Roadkill inspects all plates that were previously connected to the key plate for removal. It removes all plates that fulfill this criterion from the model and adds them to the disassembly tree, where they form "child" plates of the key plate. While evaluating each plate for removal, Roadkill assumes that all other possible child plates that can be removed in the same step are still part of the model. This assumption is crucial as it ensures that all "sibling" plates (child nodes of the key plate) can be (dis)assembled independently of their siblings.

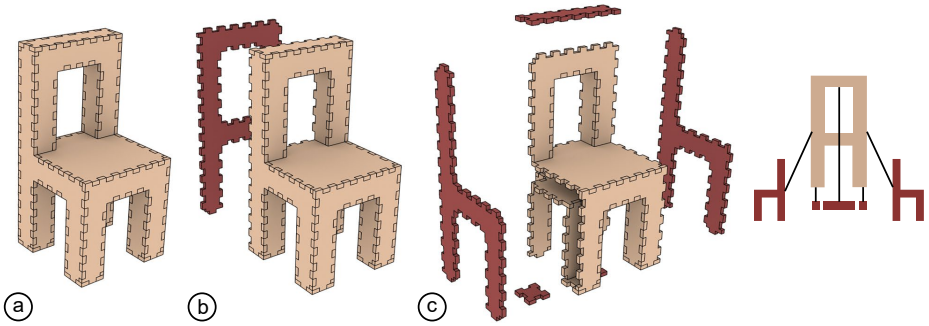

Figure 8: When Roadkill looks for plates that can be removed from (a) the chair model, (b) it chooses the "back" plate due to a high number of connecting joints and large area. (c) Roadkill then explores plates that *were* connected to the back plate for removal, removes them, and adds them to the disassembly tree.

Roadkill then evaluates all sibling plates to find another key plate using the same heuristics as before. For the chair it chooses one of the "side" plates (Figure 9a). In this case, two more plates are added as child plates, as they were both connected to the key plate and they can be removed. (b) The process is repeated iteratively until (c) all the plates are part of the disassembly tree. Algorithm 1 summarizes the process.

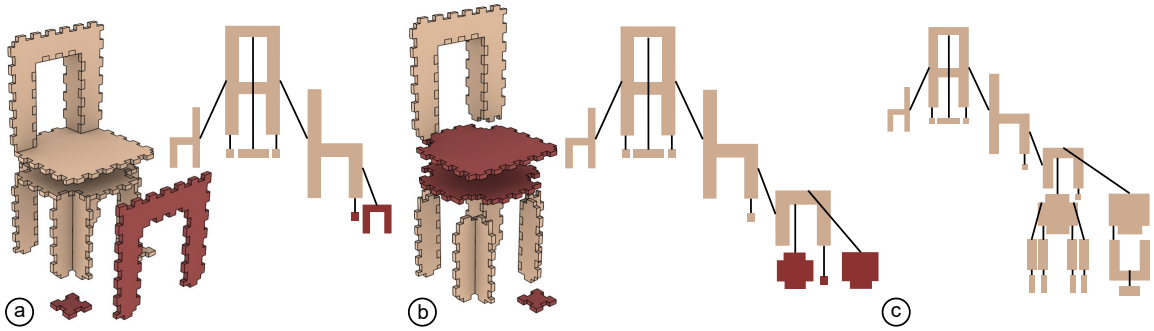

Figure 9: (a) Roadkill chooses the side plate (right) as the next key plate and then removes the plates that were connected to it. (b) Roadkill repeats the process until all plates are part of (c) the disassembly tree.

Once the disassembly tree is generated, Roadkill determines the assembly sequence by traversing the tree bottom-up, where each branch can be assembled outside-in, i.e., by starting from the leaves.

However, not all branches are independent of each other and some must be assembled in a certain order. Roadkill simplifies this with the following step.

Figure 10 illustrates how Roadkill determines if a plate is removable. (a) Two plates are connected by means of finger joints. Assuming the front plate is rigid, these finger joints prevent movement (translation) of the top plate in the four directions indicated by red arrows. (b) Now the top plate is connected to four other plates by means of multiple finger joints. As a result, constraints add up, so that this plate can now only be removed by (c) translating it along a direction of movement that is constrained by *none* of the joints. (d) In specific cases if a plate is blocked from translating along a movement direction by only one of its joints (hidden at the bottom), it is (e) removed by "tilting" it around that joint. This case relies on material compliance as fingers closer to the joint will have to compress in order to accommodate this

movement. Certain models from our test set (extracted from the Kyub repository) require such tilting in order to be (dis)assembled. However, where possible, our algorithm gives preference to translation over tilting.

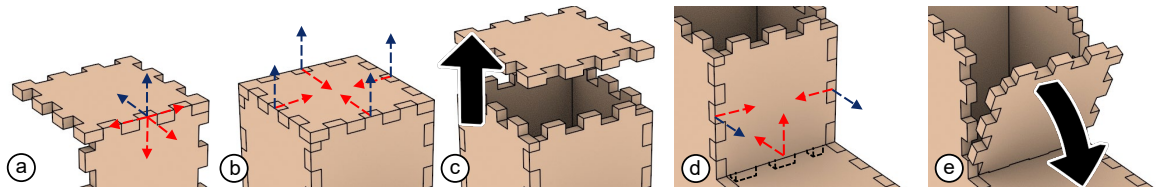

Figure 10: (a) Assuming that the front plate is rigid, the finger joints constrain the top plate from translating in four directions (red). (b) Adding additional joints constrains it further, so that (c) the only way to remove it is translating along the remaining unconstrained direction of movement. (d) If only one joint restricts translation along a movement direction, (e) the plate may be tilted out.

As discussed earlier, the general approach of using disassembly to determine an assembly order is well-established in the related work, e.g., [37][47][14]. However, Roadkill extends the status quo by creating subassemblies with branches that can be assembled *in any order*. This simplifies the assembly process as the user only has to pay attention to the inter-subassembly order while the intra-subassembly order can now be communicated implicitly through adjacency and arrows.

ALGORITHM 1: Creating the disassembly tree

generate_disassembly_tree:
Input: *model*
Output: disassembly order tree
get all plates *p* of model
extract key plate *k* of model
return generate_disassembly_tree_rec(*p, k*)

generate_disassembly_tree_rec:
Input: list of plates *p*, key plate *k*
Output: disassembly order tree
create empty tree data structure *root*
find connected components of plates
for each connected component:
    get neighboring plates *n* of key plate *k* in component
    find removable plates *r* of neighbors *n*
    choose a key plate *k** from removable plates *r*
    subtree = generate_disassembly_tree_rec(component without *k**, *k**)
    add it as a subtree of *root*
end
return root

5.2 Step 2: Roadkill creates subassemblies with independently assemblable branches

As discussed in the *walkthrough* section, Roadkill commonly enables multiple assembly "sequences" within each subassembly, each of which can be assembled independently. Roadkill achieves this by identifying these sequences or "branches" in the disassembly tree that cannot be assembled independently from each other. Once identified, Roadkill tries to merge the dependent branches together. If that is not possible, Roadkill splits the branch off as a subassembly, where assembly order labels ensure that users will assemble in the correct order.

Two branches can be assembled independently if their originating key plates have no siblings or if the siblings belong to a disconnected group. Roadkill starts by analyzing the key plates at the lowest hierarchy of the disassembly tree. As shown in Figure 11a, "green" key plates either do not have siblings or their siblings are part of a disconnected group and hence their branches are considered independent of each other. However, further up in the hierarchy, the "blue" key plate has a sibling (dark brown) belonging to the same group that could potentially block a plate in its branch.

(b) Roadkill evaluates this condition by creating a state where plates from the key plate branch and the blocking sibling plate are present and then tries to remove the key plate branch along its removal direction. If none of the plates in the branch are constrained and can be removed, then both siblings are independent of each other. However, in this case the sibling plate (dark brown) is constraining one of the child plates from the key plate branch. (c) Roadkill resolves this by merging the sibling plate into the branch, disconnecting it from its current parent and connecting it to one of the constrained plates. This combination makes the branch independent and avoids creating a subassembly.

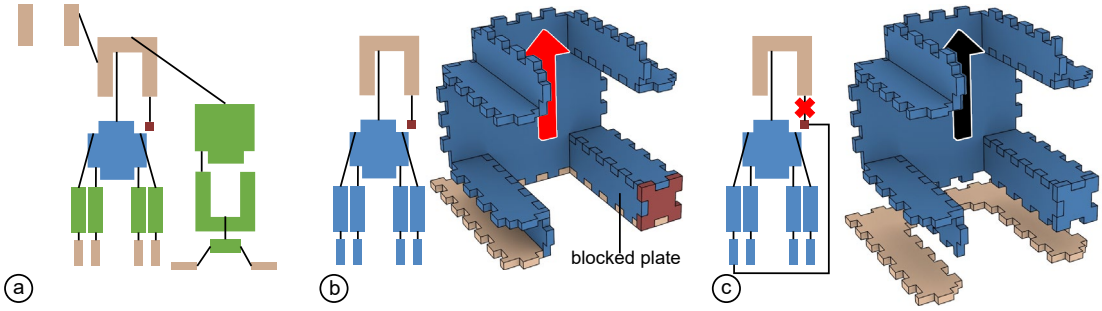

Figure 11: (a) While all "green" plates do not need to be tested against each other, the "blue" key plate and "dark brown" plate are siblings in the same group. (b) Roadkill finds that the "dark brown" plate blocks one of the child plates. (c) Roadkill resolves this by disconnecting the blocking plate from its parent and connecting it to the blocked plate.

However, not all blocking plates can be merged. As shown in Figure 12a, one of the side plates blocks two plates across different branches, thus combining it with one of the constrained plates cannot solve the problem. In this case, the branches starting with blocked key plates have to be assembled before the blocking "side plate" is inserted. (b) Roadkill resolves this by designating the branches as subassemblies and directly linking them to the blocking "side" plate. This ensures that they are assembled separately and inserted into the "side" plate before the rest of the model is assembled.

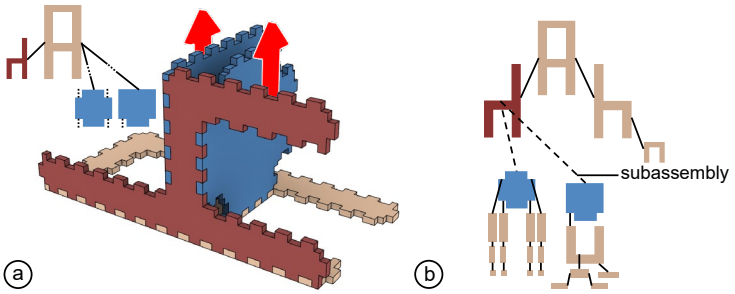

Figure 12: (a) In this case a side plate "dark brown" blocks two of the key plates "blue" and some of their child plates (not shown for simplicity) (b) Roadkill resolves this by declaring both key plates (and their children) as subassemblies and connects them to the blocking plate, this ensures that they will be assembled first and inserted into the blocking plate before it is assembled.

5.3 Step 3: Roadkill creates a 2D layout by placing plates adjacently, linking them through arrows or subdividing into subassemblies

Roadkill now starts to generate a separate provisional 2D cutting plan for each subassembly. As illustrated by Figure 13a, Roadkill starts by placing the root plate ("seat") of the subassembly and then traverses the tree placing the child plates ("backrest" etc.) adjacent to their connecting edges while checking for overlaps. Here the "seat" and "backrest" plates overlap, thus they cannot be placed adjacently. Similarly, some child plates of the backrest also overlap and cannot be placed adjacent to the connecting edge. To resolve this issue, Roadkill connects these plates using arrows. Roadkill collects the child plates that overlapped when placed on the adjacent edge (from the disassembly tree) and uses a simple physics engine (Matter.js [24]) to find a valid layout based on the following criteria: (1) maximize the number of parts that can be connected via arrows, (2) minimize the distance between the connecting edges to have shorter arrows, while (3) packing the plates together to conserve space. While finding the globally optimal solution to the packing problem is NP-hard, Roadkill's use of the rigid-body physics engine allows it to find a feasible locally optimal solution. It achieves this by representing the compactness of the connected parts as elastic energy of the connecting edges while prohibiting overlaps by collision detections of rigid bodies. Plate rotation and arrows with large number of bends are penalized. In this case both sets of plates cannot be connected using arrows, (c) Roadkill chooses to link the larger plates using arrows (shorter arrows) while designating the others as a subassembly.

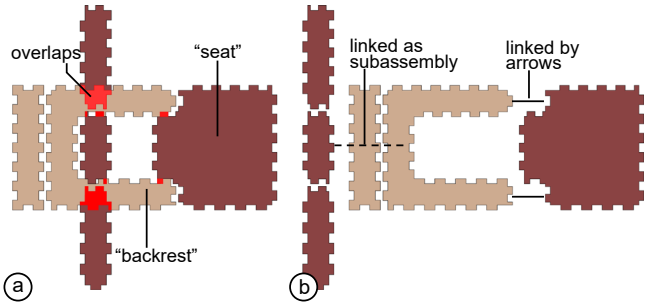

Figure 13: (a) Roadkill cannot place all plates adjacently and links overlapping (red) plates through arrows or designates them as subassemblies. (b) Based on the result of a physics simulation Roadkill chooses to link the "seat" and "backrest" through arrows while designating the other set of plates as a subassembly.

As mentioned previously (*see walkthrough*), Roadkill affords an assembly process where the user can insert all plates from above. Roadkill finds conflicts by comparing the assembly direction for each edge in the disassembly tree. As Figure 14a shows, in this case the "backrest" can receive the subassembly and adjacent plate from above, but (b) itself is inserted into the seat plate from below. (c) To resolve this conflict Roadkill mirrors the seat plate. This way the backrest will have to be rotated but it can be inserted from above. To enable decorative engraving on the mirrored plate Roadkill can break it off as a separate subassembly and the user can flip it manually before connecting to it.

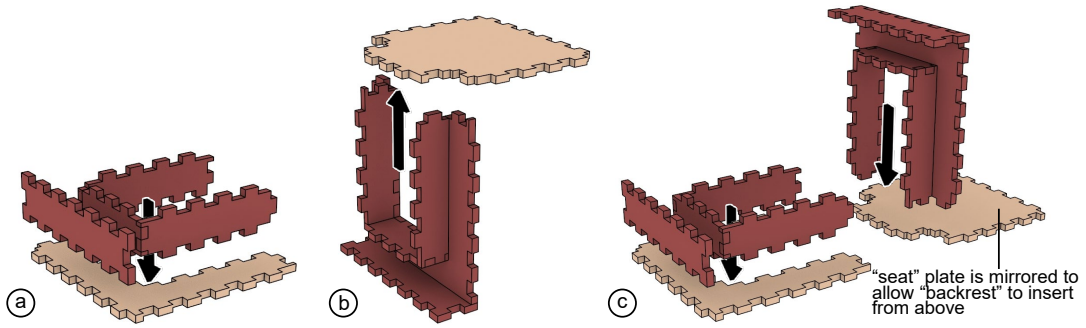

Figure 14: (a) The "backrest" plate can receive the connecting plates from above but (b) needs to be inserted into "seat" plate from below. (c) To ensure all plates can be inserted from above, Roadkill mirrors the "seat" plate. This allows the "backrest" to be rotated and inserted from above.

5.4 Step 4: Roadkill creates design elements: arrows, labels, jigs, etc. and generates the 2D cutting plan

As illustrated by Figure 15, in the final step Roadkill creates all design elements and generates the SVG.

Arrows are placed by Roadkill between edges that were linked in the previous step using a Dijkstra-based path planning algorithm (based on output from Matter.js). When a plate is required to be rotated before insertion, Roadkill ensures that an appropriate twist is added to the arrow and that joints connecting the two plates are asymmetric to help the user with alignment and ensure correct insertion.

Assembly order labels are placed on the edge that connects the subassembly to its parent. If the subassembly needs to be rotated before insertion the label is placed on a joint that is offset from the center. A matching smaller label is placed along the receiving edge along with a "arrow hat" (^) symbol pointing to the joint where the subassembly will be inserted. The last subassembly gets the "base" label on its starting plate.

Preview thumbnails are generated by combining a simplified angled silhouette of the outermost plates of the model (posed in a three.js canvas) and an eroded silhouette of the assembled subassembly plates form the same angle. The preview thumbnail is placed next to the order label.

A **frame outline** for the subassembly is generated by merging the simplified polygon outline of each element. The outline is then dilated and rounded. The merge includes all elements i.e. plates, arrows, order labels and preview thumbnails. Gaps and concavities are simplified to allow more stability for the frame.

Starting plates for each subassembly (leaves of its disassembly tree) are given a partial polygon outline, allowing them to protrude outside the frame outline.

Break-away tabs are generated for all plates to temporarily hold them in the frame after cutting. The number of tabs received by each plate depends on its size, while the thickness of the tabs is determined by the kerf of the laser cutter. The majority of the tabs for each plate (except the last plate) are placed along the connecting edge. This allows the user to guess where the tabs are allowing them to easily and intuitively disconnect the plate.

T-joint shaped jigs are created by placing a cutline next to the joint(s) along an edge. Multiple edges on a plate may have jigs based on how many plates are connecting to it in the current assembly step.

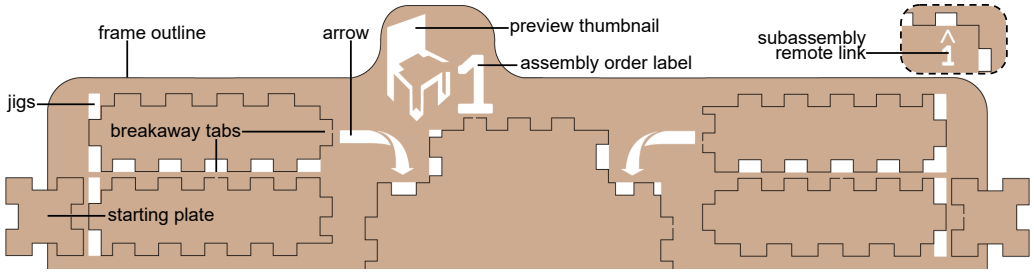

Figure 15: Roadkill creates the design elements and generates the SVG.

6 TECHNICAL EVALUATION

We integrated Roadkill into an existing 3D editor, Kyub [5]. In order to evaluate the performance of the Roadkill algorithm we used it to export the top 60 models (based on popularity) from the Kyub repository. Roadkill was able to generate layouts for 55 models (92%), where on average 81% of connecting plates were either placed adjacent or connected through arrows. 5 Models (8%) were heavily interlocked and failed to export. Figure 16 shows the layouts generated by Roadkill for 8 models from the Kyub repository test set. Design features are not included in the image for clarity.

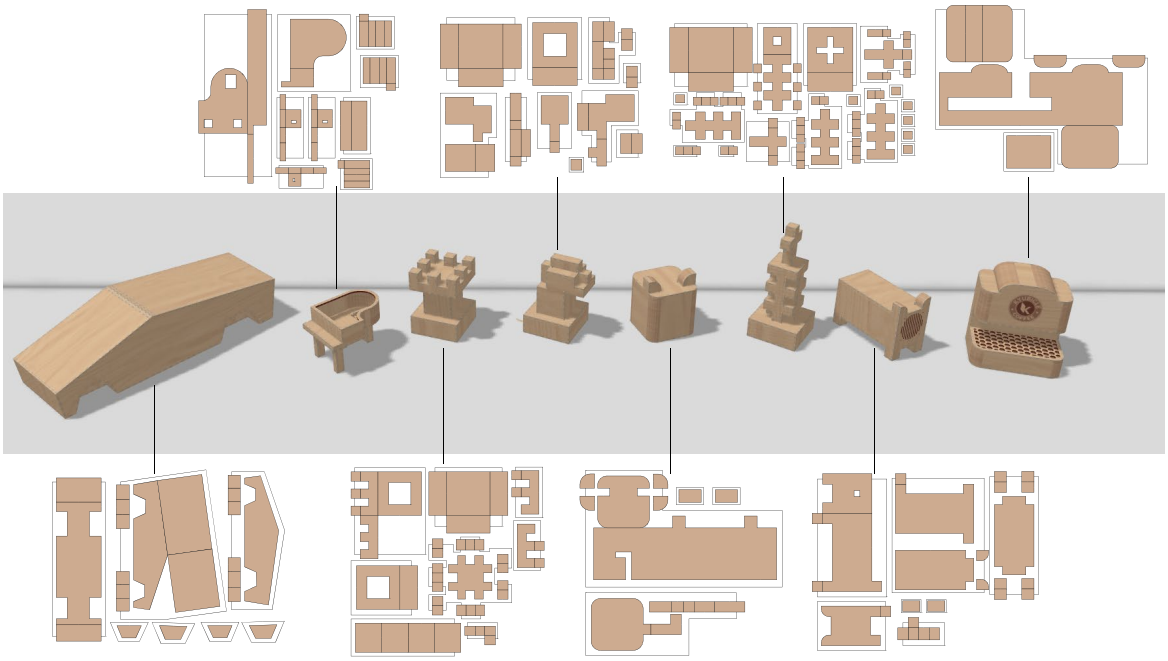

Figure 16: Roadkill layouts of some models from the Kyub repository.

The Roadkill algorithm adds 1.11 seconds to the export of the chair model, 2.81 seconds to the most complex model in the test set (99 plates), and on average of 0.89 seconds to all models in the test set. Given the assembly time saved (12 minutes in case of the chair model) the computation time can be considered negligible.

7 USER STUDY: LAYOUTS CREATED BY ROADKILL SPEED UP ASSEMBLY

In order to validate our claims regarding the assembly speed enabled by Roadkill, we conducted a user study. The participants' task was to assemble the chair model from Figure 1 from a 2D cutting plan produced by Roadkill, as well as from a 2D cutting plan produced by the traditional approach based on pairs of numbers (*Kyub* [5]). We hypothesized that participants would complete the task faster in the Roadkill condition.

7.1 Task

Participants assembled the chair model from Figure 1 (23 plates), from 4mm poplar plywood. Kerf was calibrated so as to ensure comfortable assembly by hand, i.e., without the use of tools.

7.2 Interface conditions

Participants assembled one chair for each of the following two interface conditions:

In the *Roadkill condition* 2D cutting plans were generated using the Roadkill algorithm presented throughout this paper (see Figure 1).

In the *pairsOfNumbers* control condition 2D cutting plans were generated using a 3D modeling tool for laser cutting (*Kyub* [1]), which follows the traditional approach of linking joints by engraved pairs of numbers, as shown in Figure 2b. In addition to pairing up joints, numbers also indicated assembly order, so that participants would start by joining the two edges labeled 1, then 2, and so on.

Participants received training in the form of two 2:40 min videos that illustrated how to assemble a simple object using the *pairsOfNumbers* interface (Figure 17b) and the *Roadkill* interface (Figure 17c) respectively. After watching the video, the participants physically assembled this training model. During the study the participants were allowed to look at a picture of the 3D model they were assembling.

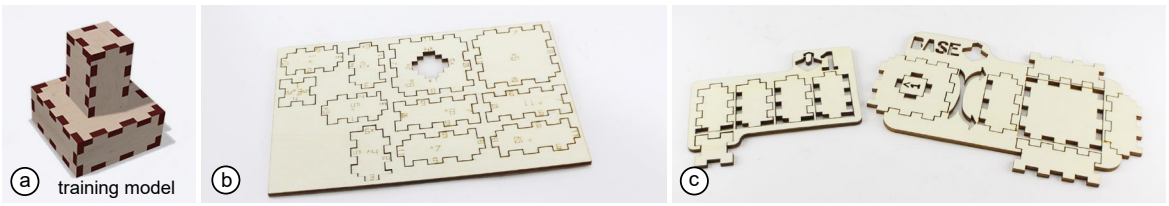

Figure 17: (a) A simple object was used to create instructional videos to show participants how to assemble both (b) traditional and (c) Roadkill layouts.

Each participant assembled the chair under both interface conditions (within-subject design) in counter-balanced order and their completion times were measured. After completing each condition, they filled in a questionnaire. All participants completed the study in under 1 hour.

7.3 Participants

We recruited 12 participants (9 male, 3 female, average age = 23.4 years, SD = 3.4) from our institution (plus one participant who had to abort the study, when the provided materials turned out to be warped). 8 participants had no prior experience assembling laser-cut objects, while the remaining 4 had designed, laser cut, and assembled models in the past.

7.4 Results

Completion time: Figure 18 summarizes the assembly times for all participants. As expected, the participants spent more time assembling the traditional layout, 22 minutes on average, while an average of 10 minutes were spent on assembling the Roadkill layout. Differences were significant ($t(11) = 7.8$, $p < 0.001$, $d = 2.25$) and the effect was substantial: participants were on average 2.4 times faster in the Roadkill condition.

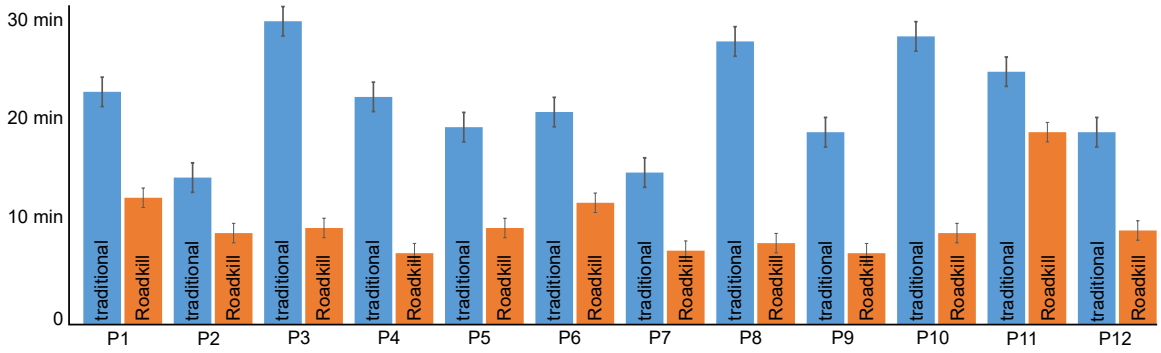

Figure 18: Results: Participants assembled the chair model on average 2.4x faster with the Roadkill interface. Error bars indicate standard error.

Subjective feedback: Figure 19 shows the result of the questionnaire. Participants rated "finding matching plates" as "easy" for the Roadkill layout, had a sense of how to assemble before starting and felt that the Roadkill communicated assembly order clearly. They enjoyed assembling the Roadkill layout.

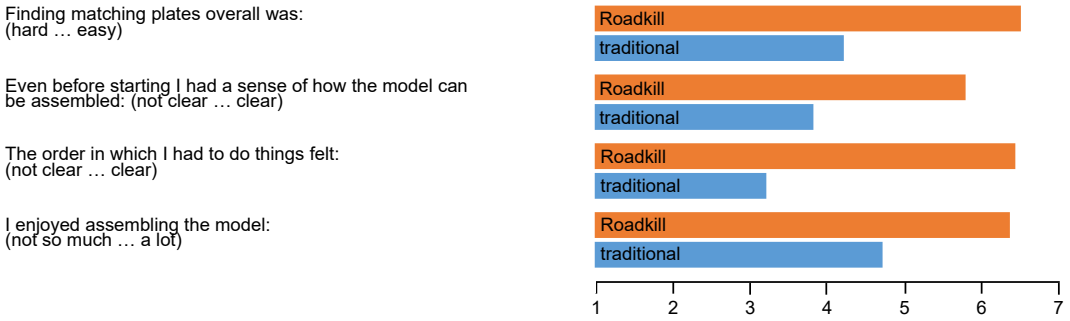

Figure 19: The results of the questionnaire that was conducted after the participants had completed the respective study.

Specifically for the Roadkill layout (Figure 20), all participants rated finding starting plates for each subassembly as "easy", and joining multiple plates/subassemblies using the jigs was also rated "moderately easy". Finding matching

plates and joining them based on adjacency and numeric symbols was also rated as "easy", while a few participants had trouble joining plates based on arrows. Some participants also found it slightly difficult to insert a completed subassembly into the receiving part.

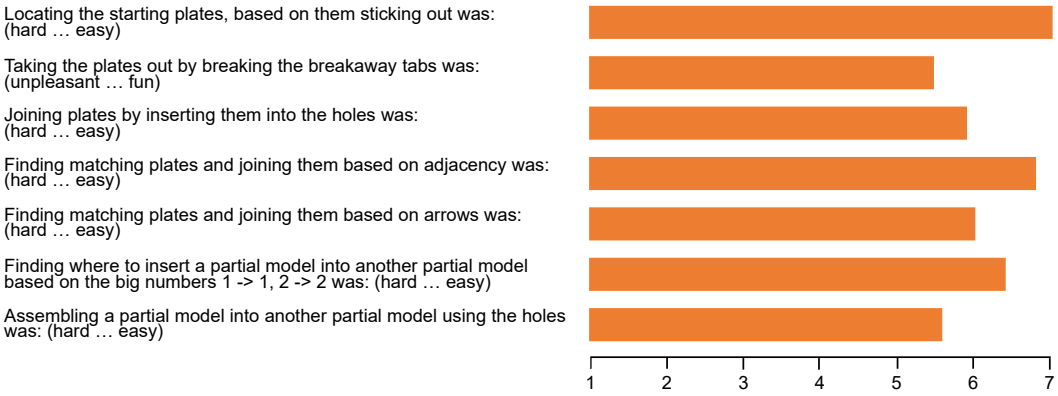

Figure 20: The results of the questionnaire conducted specifically for the Roadkill interface.

Participants mentioned that the thumbnails helped them get a sense of how the model would be assembled, with P6 saying "the picture of the result and the part visualizations [it] made clear what goes where". All participants agreed that locating starting positions was easy with P9 mentioning "if felt very natural to start with these rough parts". A few participants also noted that the layout felt intuitive, with P3 saying "no need to think about the order". Two participants (P4, P6) mentioned that while it was easy to insert a few plates, inserting a subassembly consisting of multiple plates was more difficult, as multiple joints have to be aligned at the same time; P6 added that this part "took a while".

Since Roadkill places instructions *next* to plates, while the traditional approach engraves plate IDs *into* plates, we used the questionnaire to find out whether that made a difference to participants. Specifically, we asked the participants whether they found engraved IDs acceptable in the context of models created for personal use. 7/12 participants strongly disagreed, while 3 participants said they would accept them if they were only building a prototype model but would like to remove them when building a presentation model.

7.5 Discussion

The results confirmed our hypothesis that layouts generated by Roadkill result in faster assembly. The speed up was substantial ~2.4x. Participants who had no previous experience in assembling laser-cut objects reported the highest increase in assembly speed, with 4 participants (P3, P4, P8, P10) exhibiting more than a 3x speed up. Experienced participants (e.g., P2, P12 > 50 laser-cut models built) also spent less time assembling the Roadkill layout. To further explore these observations we subdivided our 12 participants into "experienced participants" (P2, P5, P6 and P12) and "novice participants" and found that "experienced participants" were on average 1.86 times faster ($t(3) = 8.212, p < 0.002, d = 4.1$) under the Roadkill condition, while "novice participants" were on average 2.66 times faster ($t(7) = 7.421, p < 0.001, d = 2.62$). P11 spent the longest time assembling the Roadkill layout, the reason being that they did not fully connect the plates before moving to the next step, resulting in backtracking.

While visual search appeared indeed as the cause of slow assembly in the traditional layout condition, we also observed that participants had a hard time following the assembly order. The reason was that the assembly order for the chair (as for any slightly more complex laser-cut model) is a tree structure and a linear set of numbers cannot effectively

communicate a tree-based order. Thus, when the process required participants to assemble multiple branches separately, participants had a hard time figuring out when to combine them. In some cases, this required them to backtrack. Conversely, participants showed no difficulties following the grouping provided by Roadkill.

8 CONCLUSION AND FUTURE WORK

We presented Roadkill, a software tool that converts 3D models to 2D cutting plans for laser cutting, such that the resulting layouts allow for *fast assembly*. Roadkill achieves this by putting all relevant information into the cutting plan.

Zooming out, Roadkill contributes to pushing the speed boundaries of fast laser cutting. It achieves this by shifting the focus from the design of laser models to the entire design, cut and assembly process. Given that design and cutting tend to be fast already, Roadkill's 2.4x speed-up of the assembly process also speeds up the end-to-end process; in the example of the chair model, Roadkill speeds up the entire design, cut, assemble by a factor of 2: from the idea to a physical chair model in 16 minutes.

As future work, we plan on achieving further speed-ups by exploring models that combine laser cut joints with concepts known from foldable models.

ACKNOWLEDGEMENTS

Left blank for anonymous submission.

REFERENCES

[1] Michael Adamowicz, and Antonio Albano. "Nesting two-dimensional shapes in rectangular modules." Computer-Aided Design 8, no. 1 (1976): 27-33.

[2] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. "Designing effective step-by-step assembly instructions." ACM Transactions on Graphics (TOG) 22, no. 3 (2003): 828-837.

[3] Stavros Antifakos, Florian Michahelles, and Bernt Schiele. "Proactive instructions for furniture assembly." In International Conference on Ubiquitous Computing, pp. 351-360. Springer, Berlin, Heidelberg, 2002.

[4] Patrick Baudisch, and Stefanie Mueller. "Personal fabrication." Foundations and Trends in Human-Computer Interaction 10, no. 3-4 (2016): 165-293.

[5] 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 Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). ACM, New York, NY, USA, Paper 566, 12 pages. DOI: https://doi.org/10.1145/3290605.3300796

[6] 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 (CHI '15). Association for Computing Machinery, New York, NY,USA, 1799–1806. DOI:https://doi.org/10.1145/2702123.2702225

[7] Edmund K Burke and Graham Kendall. 2002. A New Approach to Packing Non-Convex Polygons Using the No Fit Polygon and Meta-Heuristic and Evolutionary Algorithms. In Adaptive Computing in Design and Manufacture.

[8] Edmund K Burke, Robert SR Hellier, Graham Kendall, and Glenn Whitwell. 2007. Complete and robust no-fit polygon generation for the irregular stock cutting problem. European Journal of Operational Research 179, 1 (2007), 27–49.

[9] Eunice López-Camacho, Gabriela Ochoa, Hugo Terashima-Marín, and Edmund K. Burke. "An effective heuristic for the two-dimensional irregular bin packing problem." Annals of Operations Research 206, no. 1 (2013): 241-264.

[10] Paolo Cignoni, Nico Pietroni, Luigi Malomo, and Roberto Scopigno. "Field-aligned mesh joinery." ACM Transactions on Graphics (TOG) 33, no. 1 (2014): 1-12.

[11] Deepnest, https://deepnest.io/, last accessed April 2021.

[12] Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sorkine-Hornung, and Mark Pauly. "Assembling self-supporting structures." ACM Trans. Graph. 33, no. 6 (2014): 214-1.

[13] IKEA, https://www.ikea.com/, last accessed April 2021.

[14] 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 (July 2015), 11 pages. DOI: https://doi.org/10.1145/2766892

[15] Somaye Ghandi, and Ellips Masehian. "Review and taxonomies of assembly and disassembly path planning problems and approaches." Computer-Aided Design 67 (2015): 58-86.

[16] Jianwei Guo, Dong-Ming Yan, Er Li, Weiming Dong, Peter Wonka, and Xiaopeng Zhang. "Illustrating the disassembly of 3D models." Computers &

graphics 37, no. 6 (2013): 574-581.

[17] 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). Association for Computing Machinery, New York, NY, USA, 1135–1139. DOI:https://doi.org/10.1145/3196709.3196800

[18] Pablo Jiménez. "Survey on assembly sequencing: a combinatorial and geometrical perspective." Journal of Intelligent Manufacturing 24, no. 2 (2013): 235-250.

[19] Danny Leen, Tom Veuskens, Kris Luyten, and Raf Ramakers. "JigFab: Computational fabrication of constraints to facilitate woodworking with power tools." In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1-12. 2019.

[20] 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.

[21] LEGO, https://www.lego.com/, last accessed April 2021.

[22] Wilmot Li, Maneesh Agrawala, Brian Curless, and David Salesin. "Automated generation of interactive 3D exploded view diagrams." ACM Transactions on Graphics (TOG) 27, no. 3 (2008): 1-7.

[23] Min Liu, Yunbo Zhang, Jing Bai, Yuanzhi Cao, Jeffrey M. Alperovich, and Karthik Ramani. "WireFab: Mix-Dimensional Modeling and Fabrication for 3D Mesh Models." In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 965-976. 2017.

[24] Matter.js, 2D physics engine for the web, https://brm.io/matter-js/, last accessed April 2021.

[25] 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). Association for Computing Machinery, New York, NY, USA, 13–22. DOI:https://doi.org/10.1145/2642918.2647388

[26] 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

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

[28] Stefanie Mueller, Tobias Mohr, Kerstin Guenther, Johannes Frohnhofen, and Patrick Baudisch. "faBrickation: fast 3D printing of functional objects by integrating construction kit building blocks." In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 3827-3834. 2014.

[29] Martin Nisser, Christina Liao, YuChen Chai, Aradhana Adhikari, Steve Hodges and Stefanie Mueller. " LaserFactory: A Laser Cutter-based Electromechanical Assembly and Fabrication Platform to Make Functional Devices & Robots" In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2021.

[30] Yuta Noma, Koya Narumi, Fuminori Okuya, and Yoshihiro Kawahara. "Pop-up Print: Rapidly 3D Printing Mechanically Reversible Objects in the Folded State." In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology, pp. 58-70. 2020.

[31] Thomas Oster. "Visicut: An application genre for laser cutting in personal fabrication." RWTH Aachen Univ (2011).

[32] Troels A. Rasmussen, and Timothy Merritt. "ProjecTables: Augmented CNC tools for sustainable creative practices." International Journal of Architectural Computing 16, no. 3 (2018): 227-242.

[33] Bruce Romney, Cyprien Godard, Michael Goldwasser, and G. Ramkumar. "An efficient system for geometric assembly sequence generation and evaluation." Computers in Engineering (1995): 699-712.

[34] Thijs Roumen, Yannis Kommana, Ingo Apel, Conrad Lempert, Markus Brand, Erik Brendel, Laurenz Seidel, Lukas Rambold, Carl Goedecken, Pascal Crenzin, Ben Hurdelhey, Muhammad Abdullah, Patrick Baudisch "Assembler 3: 3D Reconstruction of Laser-Cut Models." In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '21).

[35] Daniel Saakes, Thomas Cambazard, Jun Mitani, and Takeo Igarashi. "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, pp. 441-446. 2013.

[36] Rohan Sawhney, and Keenan Crane. "Boundary first flattening." ACM Transactions on Graphics (ToG) 37, no. 1 (2017): 1-14.

[37] 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.

[38] Ticha Sethapakdi, Daniel Anderson, Adrian Reginald Chua Sy and Stefanie Mueller. "Fabricaide: Fabrication-Aware Design for 2D Cutting Machines" In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '21).

[39] Tianjia Shao, Dongping Li, Yuliang Rong, Changxi Zheng, and Kun Zhou. "Dynamic furniture modeling through assembly instructions." ACM Transactions on Graphics 35, no. 6 (2016).

[40] Mélina Skouras, Stelian Coros, Eitan Grinspun, and Bernhard Thomaszewski. "Interactive surface design with interlocking elements." ACM Transactions on Graphics (TOG) 34, no. 6 (2015): 1-7.

[41] Peng Song, Chi-Wing Fu, Yueming Jin, Hongfei Xu, Ligang Liu, Pheng-Ann Heng, and Daniel Cohen-Or. "Reconfigurable interlocking furniture." ACM Transactions on Graphics (TOG) 36, no. 6 (2017): 1-14.

[42] SVGnest, https://svgnest.com/, last accessed April 2021.

[43] Keke Tang, Peng Song, Xiaofei Wang, Bailin Deng, Chi-Wing Fu, and Ligang Liu. "Computational design of steady 3d dissection puzzles." In Computer Graphics Forum, vol. 38, no. 2, pp. 291-303. 2019.

[44] Trotec speedy 360 laser cutter, https://www.troteclaser.com/, last accessed April 2021.

[45] Ugears, https://ugearsmodels.com/, last accessed April 2021.

[46] 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.

[47] Ziqi Wang, Peng Song, and Mark Pauly. 2018. DESIA: a general framework for designing interlocking assemblies. ACM Trans. Graph. 37, 6, Article 191 (December 2018), 14 pages. DOI:https://doi.org/10.1145/3272127.3275034

[48] Ziqi Wang, Peng Song, Florin Isvoranu, and Mark Pauly. "Design and structural optimization of topological interlocking assemblies." ACM Transactions on Graphics (TOG) 38, no. 6 (2019): 1-13.

[49] Xiangzhi Wei, Siqi Qiu, Lin Zhu, Ruiliang Feng, Yaobin Tian, Juntong Xi, and Youyi Zheng. "Toward support-free 3D printing: A skeletal approach for partitioning models." Ieee Transactions on visualization and computer graphics 24, no. 10 (2017): 2799-2812.

[50] Randall H. Wilson, On Geometric Assembly Planning. STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1992.

[51] Kristoffer Winge, Rune Haugaard, and Tim Merritt. "VAL: Visually Augmented Laser cutting to enhance and support creativity." In 2014 IEEE International Symposium on Mixed and Augmented Reality-Media, Art, Social Science, Humanities and Design (ISMAR-MASH'D), pp. 31-34. IEEE, 2014.

[52] Chenming Wu, Haisen Zhao, Chandrakana Nandi, Jeffrey I. Lipton, Zachary Tatlock, and Adriana Schulz. "Carpentry compiler." ACM Transactions on Graphics (TOG) 38, no. 6 (2019): 1-14.

[53] Hao Xu, Ka-Hei Hui, Chi-Wing Fu, and Hao Zhang. 2019. Computational LEGO technic design. ACM Trans. Graph. 38, 6, Article 196 (November 2019), 14 pages. DOI:https://doi.org/10.1145/3355089.3371979

[54] Junichi Yamaoka, and Yasuaki Kakehi. "ProtoMold: An interactive vacuum forming system for rapid prototyping." In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 2106-2115. 2017.

[55] Jiaxian Yao, Danny M. Kaufman, Yotam Gingold, and Maneesh Agrawala. 2017. Interactive Design and Stability Analysis of Decorative Joinery for Furniture. ACM Trans. Graph. 36, 2, pages. DOI: https://doi.org/10.1145/3054740

[56] Clement Zheng, Ellen Yi-Luen Do, and Jim Budd. 2017. Joinery: Parametric Joint Generation for Laser Cut Assemblies. In Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition (C&C '17). Association for Computing Machinery, New York, NY, USA, 63–74. DOI:https://doi.org/10.1145/3059454.3059459