

SplitCAD: A Split-2D/3D CAD Program for Rapid Design of 3D-Printed Mechanical Parts by Novice Users

James M. Bern
Williams College
Williamstown, Massachusetts, USA
jamesmbern@gmail.com

Michael Faulkner*
MathWorks
Cambridge, Massachusetts, USA
mfaulkner1155@gmail.com

Nathan Vosburg
Williams College
Williamstown, Massachusetts, USA
nathanjvosburg@gmail.com

Abstract

We present SplitCAD, an experimental computer-aided design (CAD) program for novice designers of 3D-printed mechanical parts. SplitCAD features an unconventional split-2D/3D workflow supported by a computational geometric registration algorithm. We deployed SplitCAD as the primary mechanical CAD program in a semester-long elective on Robotics and Digital Fabrication for computer science undergraduates at a small liberal arts college. We also performed a within-subject user study and found participants completed a mechanical modeling task more quickly and with fewer failures in SplitCAD than in OnShape, which is a browser-based CAD program commonly used in education. Finally, we present several fabrication-aware sketching tools for designing 3D-printed mechanical parts as well as user-interface animations.

CCS Concepts

• Human-centered computing → Interactive systems and tools; • Applied computing → Computer-aided design.

Keywords

computer-aided design, 3D printing, robotics, education

ACM Reference Format:

James M. Bern, Michael Faulkner, and Nathan Vosburg. 2026. SplitCAD: A Split-2D/3D CAD Program for Rapid Design of 3D-Printed Mechanical Parts by Novice Users. In *Proceedings of the 11th ACM Symposium on Computational Fabrication (SCF '26)*, December 06–08, 2026, Tokyo, Japan. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3828611.3845652>

1 Introduction

The advent of cheap and reliable FDM (fused deposition modeling) 3D printers promised to enable students and hobbyists to easily create their own custom mechanical parts. Unfortunately, early adopters of home 3D printers quickly learned a bitter lesson: *fabrication*—the actual building of things—is only half of the battle [Wade et al. 2017]. A 3D printer is limited by the accessibility of available *design* tools. Students and hobbyists might download and print a few 3D models from the Internet, but the 3D printer’s promise of extraordinary creative freedom remained unrealized.

*Work performed at Williams College.



This work is licensed under a Creative Commons Attribution 4.0 International License. *SCF '26, Tokyo, Japan*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2821-1/2026/12

<https://doi.org/10.1145/3828611.3845652>

Modern mechanical CAD programs like SolidWorks must meet the demanding needs of professional mechanical engineers designing extremely complex machines like aircraft and automobiles. For this reason, such programs include sophisticated algorithms like finite element analyses and fluid flow simulation.

In this project, we instead focus on hobbyists and students designing relatively simple 3D-printed mechanical parts, which can be modeled using roughly between one and five 3D features (extrudes and revolves). Our target audience likely does not require many of modern CAD’s more complex features. However, they still absolutely need the ability to create precise geometry in order to interface with off-the-shelf parts. For example, in a recent robotics and digital fabrication class taught by the primary author, our students created circular hole patterns for attaching to servos, hexagonal cavities for captive nuts, and profiles for seating bearings. A photo from the class is shown in Figure 1. Examples of parts of the type our students designed are in Figure 2.

The difference between the needs of professional engineers and those of novices makes us curious: **Are industry-grade CAD programs necessarily the best way for novice designers to learn how to design 3D-printed mechanical parts?** In this paper, we present and evaluate SplitCAD, an experimental CAD program for novice users to design 3D-printed mechanical parts.



Figure 1: Five-Bar Linkage Arm by Anonymous Students; course final project. A team of two students independently designed this yellow five-bar linkage robotic arm in SplitCAD. The arm is shown writing their names (“rein” and “nate”) in cursive. The stepper motor assembly was designed by the primary author, also in SplitCAD.

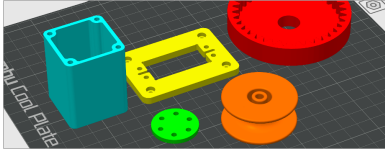


Figure 2: Examples of relatively simple mechanical parts designed to be 3D-printed.

2 Related Work

Sketchpad [Sutherland 1964], an early ancestor of modern computer-aided design (CAD) programs, ran on the MIT Lincoln Laboratory’s massive TX-2 computer and enabled engineers to create precise mechanical drawings more quickly than was possible with conventional pencil-and-paper drafting [Weisberg 2008]. Since then, personal computers have become fast enough to run CAD programs, and interest has grown in making them more accessible to novice designers.

The HCI and digital fabrication communities have explored new tools and interaction paradigms aimed at lowering the barriers to entry for novice designers [Arslan et al. 2025; Baudisch and Mueller 2017; Fossdal et al. 2021]. Of particular relevance to our paper is work on designing 3D models by drawing 2D silhouettes [Rivers et al. 2010]; like SplitCAD, this work can be seen as presenting a simplified variant of modern CAD’s ubiquitous “2D sketch → 3D geometry” workflow. Additionally, the computer graphics community has done work to automate geometry generation of entire 3D-printed mechanical toys [Zhu et al. 2012], automata [Coros et al. 2013], and robots [Yuan et al. 2018]. Sketching still plays a role in some of these works, albeit with the user sketching *target motion trajectories* rather than 2D part geometry.

In parallel, industry and open-source have developed full hobby-grade CAD programs. Often, these hobby-grade programs shun modern CAD’s “2D sketch → 3D geometry” workflow [Micallef 2015]. For example, TinkerCAD [AutoCAD Incorporation 2026] and Womp [Womp 3D Inc. 2026] both work by providing the user with a set of 3D primitives that can be transformed and combined together. The open-source programmatic OpenSCAD [Gohde 2024] and ManifoldCAD [Lalish 2026] also work this way, albeit with a radically different text-based interface for creating and modifying geometry; BlocksCAD [Berdik 2017] wraps OpenSCAD in a Scratch-like drag-and-drop programming environment. There have also been efforts to add CAD-like features to non-CAD modeling programs, for example CAD Sketcher [hlorus 2026]—which adds constraint-based sketching to Blender—and BlendSCAD [3DLIRIOUS 2017]—which adds OpenSCAD-like functionality to Blender.

Additionally, industry-grade CAD programs continue to evolve. The programs Shapr3D [Shapr3D Zrt. 2026] and Autodesk Fusion [Autodesk 2026] augment a typical parametric workflow with direct modeling features similar to those in programs like Ansys SpaceClaim [ANSYS Incorporation 2026]. Generative design capabilities are now available in mainstream CAD programs like Fusion, as well as more specialized programs like nTop [nTopology, Inc. 2025]. Finally, recent large-scale AI models can generate high-fidelity 3D geometry from text and images [Xiang et al. 2026], and are being applied to CAD [Backflip AI, Inc. 2026; Zhang et al. 2026].

3 Splitting 2D and 3D

3.1 Motivation

Our goal in creating SplitCAD was to simplify industry-grade CAD’s ubiquitous “2D sketch → 3D geometry” workflow while still retaining the ability to create precise 3D geometry. In SplitCAD, we aim to accomplish this by *splitting* (separating) 2D and 3D. Concretely, this means that SplitCAD does not store persistent spatial relationships between 2D sketches and 3D parts. Rather, a 2D sketch and a 3D part live in separate coordinate systems and are presented to the user in separate panes. In order to create new 3D geometry, a 2D sketch can be temporarily registered (aligned) to a 3D part, either manually or via an automatic registration algorithm. In contrast, mainstream parametric mechanical CAD programs like SolidWorks persistently locate 2D sketches relative to a parent 3D part, and present a part and its sketches in a single unified pane.

A potential advantage of our split 2D-3D workflow is that the design process is divided into two separate phases (first create a 2D sketch, then create a 3D part) instead of being interleaved. In the words of our user study participants:

[In SplitCAD,] I liked how I could do the whole 2d sketch before doing anything three dimensional whereas for onshape i had to constantly switch between [2d and 3d] which was confusing...
(User Study Participant 7)

[In SplitCAD,] it was a lot easier for me to actually select what faces/planes I was working off of. Having separate 2D and 3D windows made it much easier for me to visualize the shape I was working with, and I didn’t have to keep on fussing about the camera angle.
(User Study Participant 11)

Another potential advantage is that because the user interface is split into dedicated panes, each pane presents fewer options. In the words of our user study participants:

[SplitCAD features] simplicity, ‘fewer annoying buttons’.
(User Study Participant 14)

[SplitCAD required] less clicking around on different tools.
(User Study Participant 4)

A major limitation of SplitCAD’s workflow is that it does not currently include parametric history. Changes to the 2D sketch do not automatically propagate into a previously constructed 3D part; the part must be directly edited or recreated from the sketch. We aim to alleviate this burden in future work. We also note that parametric history—along with the enabling features of parametric sketching and persistent 3D localization of 2D sketches—is a double-edged sword. While parametric history is valuable for experts creating complicated machines like automobiles and airplanes, our course deployment demonstrates that novices can successfully design simple 3D-printed robotic devices without it. Additionally, in our formal user study, participants designed parts more quickly and with fewer failures in SplitCAD than in OnShape, a modern parametric CAD program. We observed that many participant failures and slowdowns with OnShape stemmed from difficulties with parametric sketching, such as adding incorrect constraints.

3.2 Split 2D/3D-User Interface

SplitCAD’s user interface is shown in Figure 3. It consists of two main panes. The *Sketch Pane* contains a large 2D sketch consisting of lines, arcs, and circles; in SplitCAD, it is convenient to include all of a mechanical assembly’s 2D geometry—including multiple views of its parts—in a single sketch. The *Mesh Pane* contains the 3D triangle mesh (part) currently being modeled. The sketch and mesh can be saved to and loaded from DXF and STL files respectively.

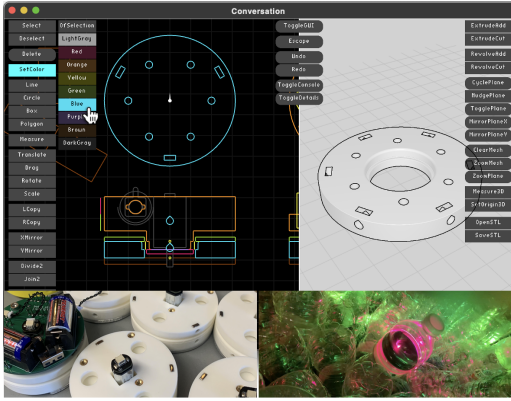


Figure 3: SplitCAD’s overall user interface, which has two separate panes, one for the 2D sketch (Sketch Pane) and one for a 3D mesh (Mesh Pane). Here the user is modeling a modular robotic water bottle for an interactive art piece.

Further details are shown in Figure 4. Clicking on the 3D part spawns a temporary *feature plane* tangent to the selected surface. Selected sketch 2D entities show up on this feature plane.

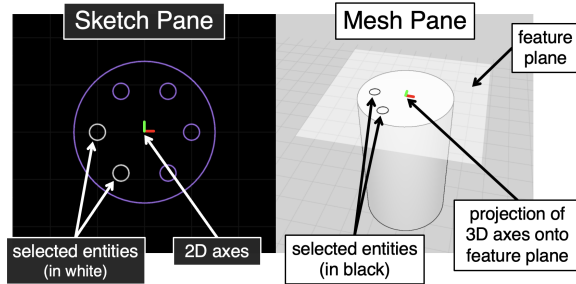


Figure 4: Annotated screenshot of detail of SplitCAD’s user interface. The user can select 2D entities in the Sketch Pane, and turn them into 3D geometry in the Mesh Pane.

3.3 Split 2D/3D-Modeling Workflow

SplitCAD’s core 2D $\rightarrow$ 3D modeling workflow is shown by example in Figure 5 and consists of (1) choosing a feature plane, (2) selecting sketch entities, and then (3) extruding (or revolving) those entities into 3D geometry. No persistent relationships are created between 2D sketches and 3D geometry. No parametric history (feature tree) is constructed. **The fundamental goal of this modeling workflow is to minimize the friction required to turn a 2D sketch into a 3D part.**



Figure 5: Annotated sequence of screenshots of a user modeling *Happy Box* in SplitCAD, starting from a single sketch of the box’s top and side views (shown in purple). Reading column by column, the user first chooses the world top plane to be the current feature plane, and selects the outer contour of the box’s top view (shown in white). The user then extrudes the box up (preview shown in green wireframe). The user then selects the top of the box as the current feature plane, selects the inner contour of the box’s top view, and cuts down into the box (preview shown in red wireframe). The user then snaps the 2D origin to the middle of the side view’s bottom line, selects the smiley face, and cuts through the box’s sides. Finally, the smiley face cut is repeated using another side of the box as the feature plane.

4 Automatic 2D/3D-Registration Algorithm

4.1 Motivation

SplitCAD’s core 2D $\rightarrow$ 3D modeling workflow (see Section 3.3) requires the 2D sketch to be temporarily located in 3D space in order to create a new 3D feature. In SplitCAD, this can be done semi-manually as follows. First, clicking on a face of the mesh spawns a feature plane tangent to the selected surface. Then, the feature plane can be transformed relative to the drawing using tools like `TranslatePlane` and `RotatePlane`. However, these manual adjustments can become cumbersome for more complex parts, especially when the user has to repeatedly switch between different portions of their 2D sketch.

To alleviate this editing burden, we develop a registration algorithm that attempts to automatically infer the user’s desired $2D \rightarrow 3D$ transform. While knowing the user’s true intent is impossible, our core insight is that an educated guess can be made by maximizing overlap between sketch entities and part face boundaries, as illustrated in Figure 6. For real-time footage of AutoRegister running, please see our video (screensaps in Figure 7).

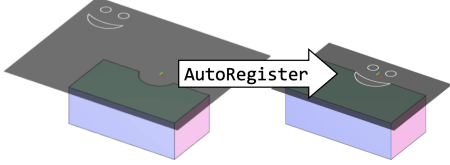


Figure 6: Here the user has chosen the top face of the 3D part to be the feature plane. An automatic registration algorithm should be able to detect that the top arc of the 2D sketch’s smiley face perfectly matches the curved face of the 3D part, and align the sketch accordingly.

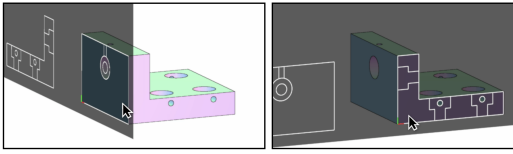


Figure 7: Screenshots of a user selecting different feature planes with AutoRegister turned on. SplitCAD automatically aligns the 2D sketch to match the 3D part as well as possible.

4.2 User Interaction Design

We provide two ways to invoke the AutoRegister tool: manually through a button or automatically through face selection. In manual mode, the user presses a button to compute and apply the highest-ranked candidate transform to the current feature plane. Repeated presses cycle through a truncated ranked list of transforms. In automatic mode, AutoRegister is invoked whenever the user clicks a face of the 3D model to select a feature plane, with subsequent clicks cycling through the same ranked transforms.

4.3 Optimization Formulation

We cast automatic registration as an optimization problem that formalizes the intuition from Section 4.1. Formally, an optimal registration transform

$$T^* = \arg \max_T R(T; S, P, M) \quad (1)$$

maximizes this scoring function, where S is the 2D sketch, P is the feature plane, and M is the 3D mesh. The reward function

$$R(T; S, P, M) = \text{overlap}(S_{\text{src}}, S_{\text{dst}}) \quad (2)$$

works by constructing two sketches and measuring their overlap. The source sketch $S_{\text{src}} = \text{transform}(T, S)$ is created by applying the candidate transform T to the 2D sketch S . The destination sketch $S_{\text{dst}} = \text{project}(P, M)$ is created by projecting the 3D mesh M onto

the feature plane P . The reward R accumulates the lengths of all entities that overlap exactly in the two sketches.

The registration transform is encoded as $T = (t, \theta, f)$, where $t = (t_x, t_y) \in \mathbb{R}^2$ is a 2D translation, $\theta \in \mathbb{R}$ is a 2D rotation, and $f \in \{-1, 1\}$ is a flag indicating whether to apply a reflection about the x -axis. These operations are applied as an (optional) reflection, followed by rotation, followed by translation. This can be written using affine transformation matrices as

$$\underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{translation}} \underbrace{\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{reflection}}. \quad (3)$$

4.4 Isometry Enumeration

We solve Equation (1) by enumerating and scoring candidate transforms that map entities in the source sketch onto entities in the destination sketch. As our reward function only counts perfect overlaps, we limit our search to cross-drawing entity pairs with identical intrinsic properties. For example, we consider pairs of lines with identical length. For each such pair, we enumerate the family of isometries that maps the entity in S_{src} onto the entity in S_{dst} . E.g., for a pair of equal length line segments with endpoints (a, b) and (c, d) , there are four such isometries given by

$$\begin{cases} T_1 = (c - R_{\theta_1}a, \theta_1, 1), \\ T_2 = (d - R_{\theta_1+\pi}a, \theta_1 + \pi, 1), \\ T_3 = (c - R_{\theta_3}Fa, \theta_3, -1), \\ T_4 = (d - R_{\theta_3+\pi}Fa, \theta_3 + \pi, -1), \end{cases} \quad (4)$$

where $F = \text{diag}(1, -1)$, $\theta_1 = \angle(b - a, d - c)$, $\theta_3 = \angle(F(b - a), d - c)$, and R_θ is a 2×2 rotation matrix.

4.5 Implementation Details

4.5.1 Mesh Projection and Cleaning. Note that naively projecting every edge of the triangle mesh onto the feature plane would create a sketch containing many extraneous edges from the interiors of part faces. Instead, we project only the *silhouette edges* of the mesh, shown in Figure 8. When viewed by an orthographic camera oriented along the feature plane’s normal n , a silhouette edge separates a visible triangle A from a nonvisible triangle B , satisfying

$$n \cdot n_A > 0 \wedge n \cdot n_B \leq 0. \quad (5)$$

Finally, the destination sketch must be *cleaned* by merging small connected line segments into larger entities, with connectivity queries accelerated by a spatial hash grid.

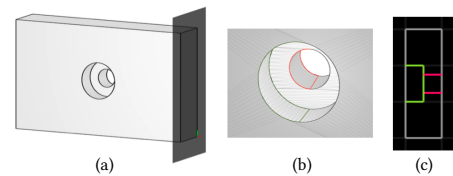


Figure 8: When projecting the mesh onto the feature plane in (a), we consider only the silhouette edges in (b), which, after cleaning, yield the destination sketch in (c).

4.5.2 Transform Hashing. Our enumeration procedure may produce the same transform multiple times. If these transforms are not merged, the AutoRegister tool may present the user with multiple “options” that correspond to the same transform. Since floating-point imprecision can cause equivalent transforms to have slightly different values, we round each component of a candidate transform’s 4-tuple encoding to the nearest thousandth before hashing.

5 Unconstrained 2D Sketching Engine

SplitCAD uses an *unconstrained (non-parametric, direct)* 2D sketching engine. This means SplitCAD’s sketches do not include the explicit *constraints (sketch relations)* common to programs like SolidWorks. A sketch in SplitCAD is a straightforward list of *entities* (lines, arcs, circles). Dimensions are first-class properties of these entities. A SplitCAD sketch never needs to be numerically “solved.”

Even though SplitCAD’s sketching system is unconstrained, it still supports exact numerical dimensions. The user can key in exact dimensions of entities at the time of creation. The user can also use *snaps* to, e.g., place the start of a line at the exact center of a circle (see Figure 9). Finally, the user can use tools like Translate, Rotate, and Scale to transform entities following their creation.

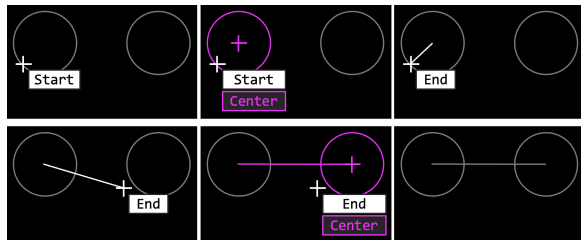


Figure 9: Sequence showing a user sketching with the Center snap, which snaps to the center of the nearest circle or arc. When the Center snap is active, SplitCAD highlights the circle or arc nearest to the user’s (white, crosshair style) cursor in magenta, and also draws a second “snapped” cursor, also in magenta. For mouse presses, this snapped cursor is treated as the primary one. Reading row by row, the user activates the Line tool (by pressing L on the keyboard), activates the Center snap (by pressing C), and then presses the left mouse button to place the line’s start point at the exact center of the left circle. The user then moves their mouse to be near the right circle, presses C, and clicks again. Note that snaps are *not* persistent. If the user were now to translate the circles away from their starting location, the line would not move.

While SplitCAD’s unconstrained sketching system is very different than the parametric sketching ubiquitous in modern 3D CAD software like SolidWorks, unconstrained sketching is actually quite common in 2D CAD software. For example, unconstrained sketching is used in the open-source 2D CAD software QCAD [RibbonSoft, GmbH 2026], the proprietary waterjet drawing software OMAX LAYOUT [OMAX Corporation 2026], and Autodesk’s AutoCAD [Autodesk Incorporation 2026]—which was first released in 1982 and did not add parametric sketching until 2010 [Ellis 2026].

We chose unconstrained sketching because our teaching experience suggested it to be less error-prone for our target audience and

use case. In our experience teaching as well as in our user study (see Section 8.2), we have observed that novice students using parametric sketching often make mistakes that are hard for them to correct. For example, they accidentally added dimensions between the wrong entities, omitted dimensions they actually needed, and automatically added constraints between the wrong entities. Unconstrained sketching in SplitCAD lacks these pitfalls.

Finally, SplitCAD’s sketching engine is *fast*. For reference, SplitCAD can render and edit a sketch with 100 000+ entities in real-time on an M1 MacBook Pro. The relative speed of SplitCAD’s sketching engine is largely attributable to its lack of need for constraint solving, but also owes to the design of its entity struct (see Section 10), and its use of spatial acceleration structures for proximity queries [Nystrom 2014].

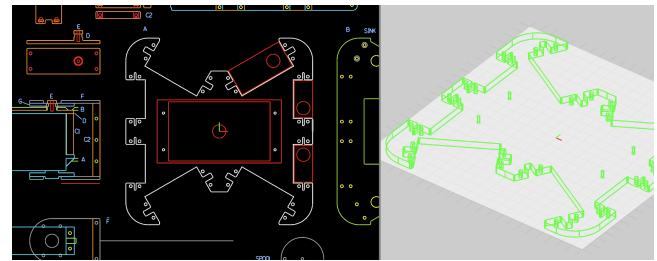


Figure 10: Screenshot of SplitCAD showing the user extruding 3D geometry from a large (6500+ entity) sketch. The sketch was created by the primary author, and is the complete design of an aquatic robot.

5.1 Fabrication-Aware Sketching Tools

No manufacturing process is perfect, and parts created via FDM 3D printing tend to have certain defects, also called *artifacts* [Lab 2026]. SplitCAD makes it easier to create designs that account for the inherent limitations of FDM 3D printing by providing multiple useful sketch tools.

5.1.1 ElfHat. The top surfaces of transverse holes in FDM 3D-printed parts tend to sag if support material is not used [Lab 2026]. Figure 11 shows how a triangular “hat” or “point” can be added to make the hole self-supporting [Slant 3D 2023].

5.1.2 DogBone. Sharp edges in FDM 3D-printed parts tend to get rounded out by the melting action of the printer. This means that, e.g., polygonal posts may not fit through matching holes and ball bearings may not seat properly. Related issues arise in processes like CNC routing, and Figure 11 shows how *dogbone fillets* can be added to ensure a proper fit [Imran Anees 2020].

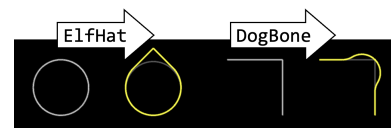


Figure 11: SplitCAD’s ElfHat and DogBone tools help users account for the inherent limitations of FDM 3D printers.

5.1.3 Drag. The relative imprecision of hobby-grade 3D printing means that students often need to adjust their designs and reprint them. To simplify the process of making routine edits, we provide a tool called Drag—similar to AutoCAD’s Stretch [Autodesk Inc. 2024]—shown in Figure 12. Internally, Drag uses a graph search to find entities incident to the current selection.

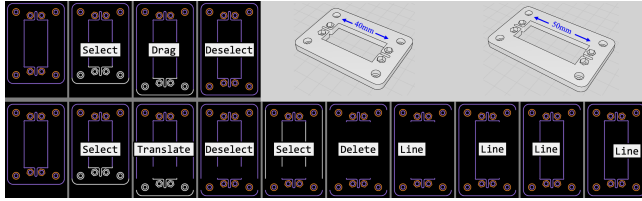


Figure 12: Annotated screenshots of a user editing the sketch of a servo bracket to accommodate a wider servo (top-right; shown in 3D). The top row shows how this can be quickly done with the Drag command; the bottom row shows how much more cumbersome it is to use lower-level commands.

6 Triangle Mesh-Based 3D Geometry Kernel

For its geometry kernel, SplitCAD uses the open-source Manifold library, which performs fast and accurate boolean operations on manifold triangle meshes [Lalish 2026]. We chose this kernel to simplify our implementation; we already need a triangle mesh to render geometry to the screen and Manifold lets us use this same mesh for modeling. We note that the hobby-grade programmatic mechanical CAD program OpenSCAD also uses a mesh-based kernel. While representing 3D geometry as triangle meshes is nonstandard in industry-grade mechanical CAD—with most mainstream programs using spline surfaces like NURBS—mesh representations are typical in digital sculpting (e.g., ZBrush [Maxon Computer GmbH 2026]) and box modeling (e.g., Blender [The Blender Foundation 2026]).

7 User Interface Animations

Animations (tweening, easing) can make user interfaces (UI) easier to understand and more engaging [Fleury 2022; Jonasson and Purho 2012]. SplitCAD makes extensive use of a frame-independent variant of the classic “lerp smoothing” update $a \leftarrow a + 0.1 * (b - a)$, which moves the quantity a a fixed fraction of the way to target value b [Holmér 2024]. For examples, see Figures 13 to 16.

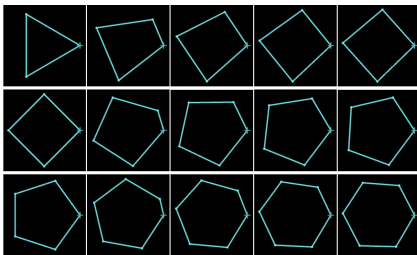


Figure 13: As the user changes the Polygon tool's number of sides, SplitCAD animates the preview of the resulting shape.

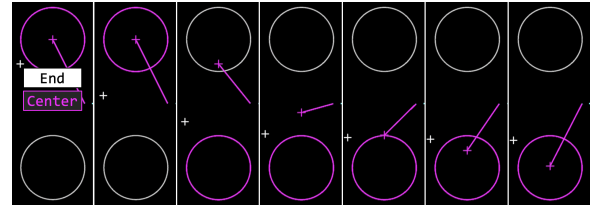


Figure 14: SplitCAD interpolates the preview of Line tool as the line's end snaps between the centers of the two circles.

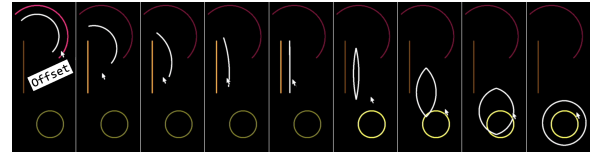


Figure 15: With the Offset tool active, the user moves the mouse from the red arc to the orange line, to the yellow circle. SplitCAD animates the preview of the offset white shape.

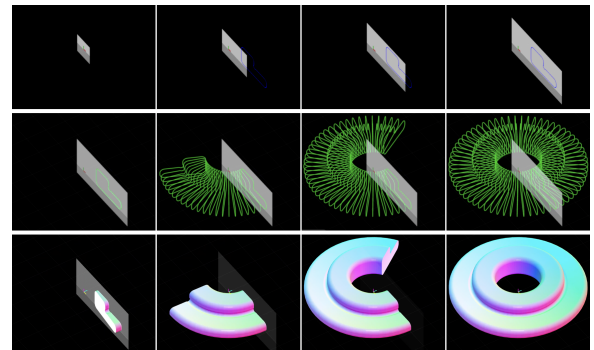


Figure 16: Here we show the different animations involved in a single application of the Revolve command. Reading row by row, SplitCAD animates the size of feature plane to accommodate selected sketch entities (top row), the preview of the revolve (middle row), and the application of the revolve—using clip planes [Andrei Gradinari 2005]—plus the plane’s opacity (bottom row).

8 Evaluation

8.1 College Course Deployment

We used SplitCAD as the primary mechanical CAD software for our semester-long, project-based elective *Robotics and Digital Fabrication*, taught in the computer science department of a small liberal arts college. The vast majority of our 23 students had no prior CAD experience, yet successfully used SplitCAD to design and fabricate custom 3D-printed robotic devices, as shown in Figures 1 and 17 and our video. In the school-administered anonymous student course survey, students rated how much the course contributed to their education an average of 6.0/7.0, compared with 5.5/7.0 for other STEM courses taught the same semester. The lab component, which used SplitCAD most weeks, received an average rating of 7.0/7.0.

After course grades were finalized, we solicited qualitative feedback from students regarding their experience using SplitCAD in the course. In addition to SplitCAD, our course included a week of SolidWorks, so all students had SolidWorks as a reference point. For clarity, we have substituted “SplitCAD” for the software’s working title in the quotes below; otherwise the quotes are unedited.

Before joining CS345 I already owned a 3D printer and was getting some amount of use out of it, but not nearly as much as I wanted. The reason for that was primarily driven by the complexity of the design tools that were available. I tried AutoDesk Fusion and SolidWorks, but neither provided an intuitive user experience and left me spending more time watching tutorials than making the item that I wanted. In contrast, within 30 minutes of downloading SplitCAD I was able to design and print multiple test pieces of different shapes and complexities. That speed to output compared to other programs gave me the confidence to try and build things that I wouldn’t have attempted otherwise because SplitCAD made the process of design and iteration so much more approachable than a similar process using another tool. (Student A)

As somebody who’s learned how to use many different types of software, using SplitCAD was very enjoyable because of the emphasis on using keybinds to perform actions and manipulations. The simplicity of the program’s workflow meant there was a relatively small pool of keybinds that I could learn quickly, and I felt like I could work efficiently after minimal practice. (Student B)

While SolidWorks may have greater capability for large-scale models and systems, I often found it tedious and somewhat unapproachable as a beginner in 3d modeling. In contrast, the workflow in SplitCAD is very intuitive and smooth, making it an amazing tool for robotics students. It’s not only fast and approachable, but very fun and satisfying to use. (Student C)

Overall, I felt SplitCAD was incredibly easy to pick up. A big hurdle when learning Solidworks was dealing with constraints, which can feel limiting and frustrating. SplitCAD was quite forgiving in that sense, and made simple modeling quick to learn. All features felt well-thought of, and when paired with key bindings, made the process quite intuitive. Solidworks, in comparison, sometimes had the problem of overwhelming the user with features they’d rarely need to use. (Student D)

The awesome thing about SplitCAD is that it has remarkably little learning curve. Within a few hours of using SplitCAD I was able to quickly design and iterate on structures for 3d printed robotics. The learning curve for a software like Solidworks is steep. Learning to constrain your sketches and work in both 2d and 3d space simultaneously takes time. I felt like Solidworks’ learning curve held back my ability as a maker to translate my ideas into 3d while I navigated its learning curve. SplitCAD’s fast learning curve and speed makes it uniquely suited for 3d printed robotics. (Student E)

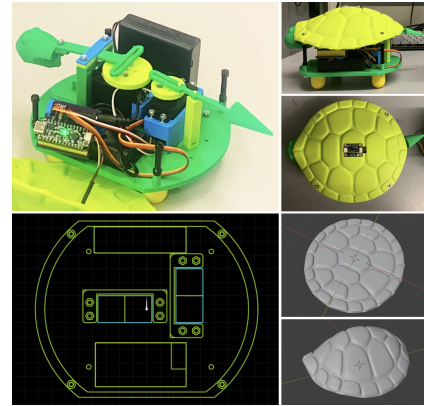


Figure 17: *Terry the Turtle* by Anonymous Students, course midterm project. A team of two students designed and fabricated this servo-driven turtle capable of hiding its head and wagging its tail in response to light. Scotch yoke mechanisms move the head and tail. All 3D-printed parts were modeled by the students in SplitCAD, except for the sculpt of the turtle’s head—which was found online—and the turtle’s shell—which the students first made in Blender and then modified in SplitCAD to include cutouts for a light sensor and screws.

8.2 Ease Of Acquisition User Study

To evaluate how readily SplitCAD is acquired by novices compared to a modern parametric CAD program, we conducted a formal, IRB-approved user study with 16 undergraduate participants who had little to no prior CAD experience recruited from our computer science department mailing list. Participants received a \$20 Amazon gift card. We compared specifically against OnShape because it is freely available in the browser and is popular in FIRST Robotics. We note that SplitCAD and OnShape have major differences beyond the fact that SplitCAD splits 2D and 3D, and OnShape does not. Importantly, SplitCAD uses unconstrained (direct) sketching, and OnShape uses constrained (parametric) sketching. Additional study would be required to isolate the effect of splitting 2D and 3D.

8.2.1 Study Design. The study involves creating parts from dimensioned mechanical drawings, which is a typical exercise used to learn CAD, and also a vital part of being able to design 3D-printed mechanical parts to mate with off-the-shelf parts like servos. The specific dimensioned drawings used in our study are shown in Figure 18. The study was meant to simulate students working with Part A in class before making the more complex Part B as homework.

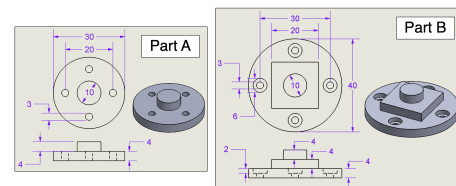


Figure 18: The dimensioned drawings used in our study. Students were told that both parts are rotationally symmetric.

Each participant completed two back-to-back sessions, one with SplitCAD and one with OnShape; session order was alternated across participants. Before starting, the dimensioned drawings were explained and the participant could ask any questions about them.

Each session began with a 15-minute *learning segment*, in which the participant worked with an instructor. During this segment, the instructor first demonstrated how to make Part A, while answering any questions. The participant then made Part A themselves, asking questions as they went. We note that for each CAD program, the instructor demonstrated a standard, simple workflow that was known to also work well for Part B. For SplitCAD, the instructor first copied over the entire sketch and then performed a sequence of simple extrude adds and cuts. For OnShape, the instructor repeatedly created a simple sketch on a plane or face and used it to perform an extrude add or cut. While other workflows are possible in OnShape—e.g., using a master sketch with projections or offset extrusions—these alternate workflows are, in our experience teaching CAD, much harder to learn for novices. The goal was not to compare expert workflows, but novice acquisition of workflows that we selected as representative. Next came a 15-minute *test segment*, during which participants attempted to independently recreate Part B from a dimensioned mechanical drawing. Questions were not allowed during the test segment. If the part was not finished once 15 minutes had elapsed, the segment was stopped. The second session followed this same overall procedure.

8.2.2 Study Results. Our main result is summarized Figure 19. Participants worked more quickly and with fewer failures in SplitCAD than OnShape. Participants failed the task in OnShape by, e.g., creating sketch entities of the wrong length, creating sketch entities at the wrong position, accidentally creating multiple solid bodies, and accidentally creating surface geometry instead of solid geometry.

The mean SplitCAD completion time was 3.5 minutes ($SD = 1.5$). The mean OnShape completion time was 9.0 minutes ($SD = 3.4$). A paired t-test showed a significant difference between the two conditions ($t(15) = -6.20, p < .0001$), with a large effect size ($d_z = 1.55$). The mean per-participant speedup from SplitCAD was $2.9\times$ ($SD = 1.4$). Every participant was faster with SplitCAD than OnShape. Every participant succeeded with SplitCAD but only 75% (12/16) succeeded with OnShape; this difference was not statistically significant (McNemar’s exact test, $p = .125$).

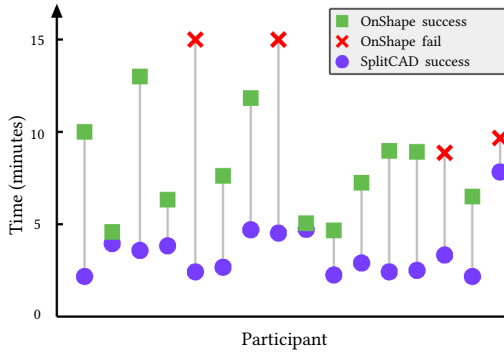


Figure 19: Per-participant completion times for the user study task of recreating Part B.

Likert-scale feedback is in Figure 20. Participants found SplitCAD to be easier, more efficient, and more enjoyable to use.

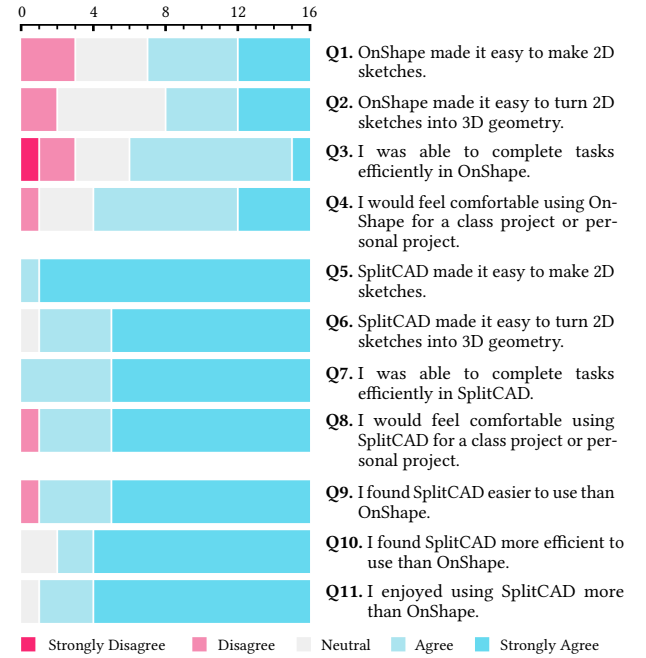


Figure 20: User study questionnaire comparing participants’ experiences of OnShape and SplitCAD. 93.75% (15/16) of participants found that SplitCAD made it easy to turn 2D sketches into 3D geometry, compared to only 50% (8/16) for OnShape. 93.75% (15/16) of participants found SplitCAD easier to use than OnShape.

8.3 Automatic Registration User Study

To evaluate the utility of our automatic registration algorithm—which was not yet available to students at the start of the course—we conducted a short, exploratory user study with 8 students from our course. The students were each tasked with using SplitCAD to turn a multi-view 2D sketch into a corresponding 3D mesh (see Figure 21). Each participant performed the task under two conditions: with the AutoRegister feature enabled, and with it disabled; condition order was alternated across participants.

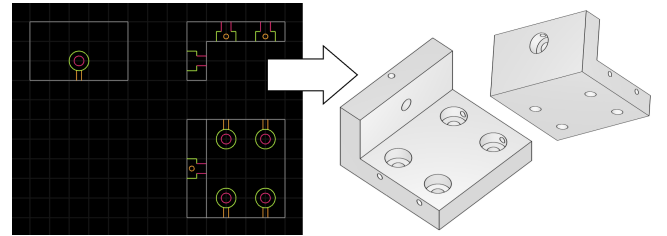


Figure 21: To evaluate the effectiveness of our AutoRegister feature, participants were tasked with turning a single multi-view 2D sketch (left) into a corresponding 3D part (right).

We find that AutoRegister reduced completion time and perceived difficulty. The mean completion time with the AutoRegister feature enabled was 5.0 minutes ($SD = 2.9$). The mean completion time with AutoRegister disabled was 8.2 minutes ($SD = 3.5$). The mean per-participant speedup from using AutoRegister was $2.1\times$ ($SD = 1.3$). These results agree with the Likert-scale participant impressions we summarize in Figure 22.

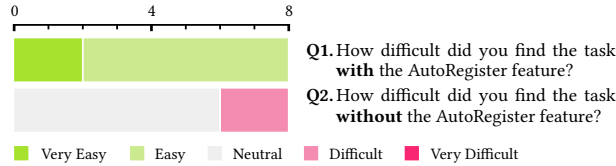


Figure 22: Participants evaluated how challenging they found the same 2D → 3D modeling task in both conditions. Participants consistently experienced the task as being easier when they had access to AutoRegister.

8.4 UI Animation User Study

To get an initial sense of student sentiment on UI animations, we performed a short study with the same participants from Section 8.2. We had participants watch two one-minute videos of the same part being modeled in SplitCAD: once with user animations enabled, and once with them disabled; video order was alternated across participants. Participants then answered a survey. As shown in Figure 23, most participants: 1) found that UI animations made it easier to follow what was happening, 2) found that UI animations helped them pay attention, and 3) preferred the video with UI animations.

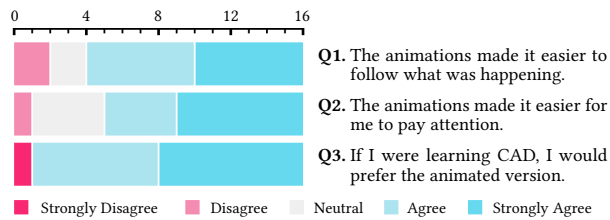


Figure 23: User study participant responses to questions about user interface animations. 93.75% (15/16) of participants would prefer learning CAD with UI animations.

9 Feature Set Comparison

In Table 1, we compare SplitCAD’s feature set to over ten other CAD programs. Among these programs, only SplitCAD features an automatic 2D/3D-registration algorithm. Additionally, only SplitCAD uses UI animations for 2D sketching and 3D features. Among the hobby-oriented 3D CAD programs we compare, only SplitCAD features precise 2D sketching. All programs are capable of creating precise 3D geometry, although they do it in different ways. Industry-grade programs tend to feature constraint-based sketching and a parametric feature tree, with the partial exception of AutoCAD.

10 Implementation Details

SplitCAD is developed in C. Its third-party dependencies are Manifold, GLFW, and OpenGL. Excluding these, the source is about 13 000 lines. SplitCAD builds without optimization in about 1 second. Its executable is about 1 MB. It opens in under 0.1 seconds on an M1 MacBook Pro.

Internally, SplitCAD uses a single, “fat” entity struct to minimize code paths [Fleury 2023]. An entity has three points: start, end, and center. No type field is needed. A circle has end set exactly to start, a line has center.x set to INFINITY, and an arc is neither a circle nor a line.

11 Discussion and Future Work

We presented SplitCAD, an experimental CAD program aimed at making it easier for novice designers to create 3D-printed mechanical parts. The most unconventional feature of our program is that it splits 2D and 3D both in its user interface and modeling workflow. The two sides of the program can be temporarily bridged using a computational geometric registration algorithm that automatically aligns the user’s 2D assembly sketch to the 3D part they are currently modeling. We found SplitCAD was well received in our demo deployment in a college computer science elective, and were encouraged to see students quickly gain confidence with 3D modeling. In our user study, students completed a modeling task more quickly and with fewer failures using SplitCAD than using OnShape. Additionally, we presented multiple fabrication-aware sketching features and explored adding user interface animations.

We are excited to use SplitCAD as a starting point for exploring additional unconventional CAD workflows in support of STEAM education. Two core ways we plan to improve and extend SplitCAD are included below.

11.1 History System

Currently, SplitCAD has only a basic undo/redo stack-based history system. This sufficed for our course deployment, but as we consider making parts with larger numbers of features, a more sophisticated history system becomes desirable. We are interested in developing a new history system that is compatible with split 2D/3D CAD, as a typical parametric history system tightly couples 2D sketches to a 3D part. Inspired by the observation that an OpenSCAD program encodes an editable modeling history, we plan to explore whether automatic OpenSCAD program generation could be integrated into our software. Such an approach could help reduce the effort required to edit long-history 3D parts, while still preserving the benefits our students and user study participants observed when using SplitCAD.

11.2 Organic Sculpting

In our course, we noticed that students were interested in integrating organic meshes into their devices, such as in Figure 17. Currently, this requires the use of a secondary program like Blender or ZBrush. One interesting avenue of future work would be to allow organic sculpting or box-modeling inside of SplitCAD. This would support additional STEAM educational use cases and enable a wider range of design workflows within a single environment.

Table 1: Feature comparison between SplitCAD and various other 3D CAD programs. By “Precise 2D Sketching,” we mean that the program can create arbitrarily precise 2D sketches. By “Precise 3D Geometry,” we mean that the program can create arbitrarily precise 3D meshes. Some programs only partially incorporate certain features, which we indicate with a “~”. E.g., Blender can be given some ability to do precise 2D sketching through the in-development open-source CAD Sketcher add-on, and TinkerCAD recently added basic 2D sketching functionality, but lacks circle and arc tools. TinkerCAD does not expose export resolution parameters, but it certainly could. In OpenSCAD, the user creates a model by writing a computer program; this program is—in an unconventional sense—a parametric feature tree. It can also encode a constrained sketch.

¹SolidWorks, Fusion, OnShape, Shapr3D, and FreeCAD have the same feature set in this table.

²OpenSCAD, ManifoldCAD and BricksCAD have the same feature set in this table.

	Automatic 2D/3D-Registration	User Interface Animations	Precise 2D Sketching	Precise 3D Geometry	Constraint-Based 2D Sketching	Parametric History
SplitCAD	✓	✓	✓	✓	✗	✗
SolidWorks-like ¹	✗	✗	✓	✓	✓	✓
AutoCAD	✗	✗	✓	✓	✓	✗
Blender	✗	✗	~	✓	~	✗
TinkerCAD	✗	✗	~	~	✗	✗
OpenSCAD-like ²	✗	✗	~	✓	~	~

References

- 3DLIRIOUS. 2017. BlendSCAD: OpenSCAD-ify Blender. <https://github.com/3DLIRIOUS/BlendSCAD>
- Andrei Gradinari. 2005. Advanced Clipping Techniques. <https://glbook.gamedev.net/GLBOOK/glbook.gamedev.net/mogllp/advclip.html>
- ANSYS Incorporation. 2026. SpaceClaim. <https://www.ansys.com/products/3d-design/ansys-spaceclaim>
- Oguz Arslan, Artun Akdoğan, and Mustafa Doga Dogan. 2025. TinkerXR: In-Situ, Reality-Aware CAD and 3D Printing Interface for Novices. In *Proceedings of the ACM Symposium on Computational Fabrication*. 1–19.
- AutoCAD Incorporation. 2026. TinkerCAD. <https://www.tinkercad.com>
- Autodesk. 2026. Autodesk Fusion. <https://www.autodesk.com/products/fusion-360/overview>
- Autodesk Inc. 2024. The Stretch and Lengthen Commands.
- Autodesk Incorporation. 2026. AutoCAD. <https://www.autodesk.com/products/autocad/>
- Backflip AI, Inc. 2026. Backflip. <https://www.backflip.ai>
- Patrick Baudisch and Stefanie Mueller. 2017. Personal fabrication. *Foundations and Trends® in Human-Computer Interaction* 10, 3–4 (2017), 165–293.
- Chris Berdik. 2017. Kids code their own 3D creations with new blocks-based design program. *Tech Directions* 76, 9 (2017), 23.
- Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.
- Rick Ellis. 2026. A Practical Guide to Parametric Drawing in AutoCAD. <https://www.autodesk.com/autodesk-university/class/Practical-Guide-Parametric-Drawing-AutoCAD-2019>
- Ryan Fleury. 2022. UI, Part 1: The Interaction Medium. <https://www.rfleury.com/p/ui-part-1-the-interaction-medium>
- Ryan Fleury. 2023. The Codepath Combinatoric Explosion. <https://www.rfleury.com/p/the-codepath-combinatoric-explosion>
- Frikk Fossdal, Rogardt Heldal, and Nadya Peek. 2021. Interactive digital fabrication machine control directly within a CAD environment. In *Proceedings of the 6th Annual ACM Symposium on Computational Fabrication*. 1–15.
- Justin Gohde. 2024. OpenSCAD. <https://openscad.org>
- hlorus. 2026. CAD Sketcher. https://github.com/hlorus/CAD_Sketcher
- Frey Holmér. 2024. Lerp smoothing is broken. <https://www.youtube.com/watch?v=LSNQuFED0yQ>
- Imran Anees. 2020. Everything You Need To Know About DogBone Fillets. <https://www.bricsys.com/en-eu/blog/everything-you-need-to-know-about-dogbone-fillets>
- Martin Jonasson and Petri Purho. 2012. Juice it or lose it. <https://www.youtube.com/watch?v=Fy0aCDmgnxg>
- Bambu Lab. 2026. How to Print Overhangs. <https://wiki.bambulab.com/en/filament-acc/filament/print-quality/overhang>
- Emmett Lalish. 2026. Manifold. <https://github.com/elalish/manifold>
- Maxon Computer GmbH. 2026. ZBrush. <https://www.maxon.net/en/zbrush>
- Joe Micallef. 2015. *Beginning design for 3D printing*. Apress.
- nTopology, Inc. 2025. nTop. <https://www.ntop.com>
- Robert Nystrom. 2014. *Game programming patterns*. Genever Benning.
- OMAX Corporation. 2026. LAYOUT. <https://www.omax.com/>
- RibbonSoft, GmbH. 2026. QCAD. <https://www.qcad.org/>
- Alec Rivers, Frédo Durand, and Takeo Igarashi. 2010. 3d modeling with silhouettes. In *ACM siggraph 2010 papers*. 1–8.
- Shapr3D Zrt. 2026. Shapr3D. <https://www.shapr3d.com>
- Slant 3D. 2023. Design Better Holes. <https://www.youtube.com/watch?v=Bd7Yyn61XWQ>
- Ivan E Sutherland. 1964. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*. 6–329.
- The Blender Foundation. 2026. Blender. <https://www.blender.org>
- Russell Wade, Nigel P Garland, and Gary Underwood. 2017. Challenges of 3D printing for home users. *International Conference on Engineering and Product Design Education* (2017).
- David E Weisberg. 2008. The engineering design revolution. *CADHistory.net* (2008).
- Womp 3D Inc. 2026. Womp. <https://www.womp.com>
- Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. 2026. Native and compact structured latents for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14419–14429.
- Ye Yuan, Changxi Zheng, and Stelian Coros. 2018. Computational design of transformables. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 103–113.
- Licheng Zhang, Bach Le, Naveed Akhtar, Siew-Kei Lam, and Duc Ngo. 2026. Large language models for computer-aided design: A survey. *Comput. Surveys* 58, 9 (2026), 1–39.
- Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. 2012. Motion-guided mechanical toy modeling. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–10.