The MechProcessor: Helping Novices Design Printable Mechanisms across Different Printers

Mark Fuge[a,*]**, Greg Carmean**[a]**, Jessica Cornelius**[a]**, Ryan Elder**[b]
Department of Mechanical Engineering[a]
Department of Computer Engineering[b]
University of Maryland
College Park, Maryland 20742
Email: {fuge;gcarmean;jcornell;relder}@umd.edu

Additive Manufacturing (AM), or 3D-printing, sits at the heart of the Maker Movement—the growing desire for wider-ranges of people to design physical objects. However, most users that wish to design functional moving devices face a prohibitive barrier-to-entry: they need fluency in a Computer-Aided Design (CAD) package. This limits most people to being merely consumers, rather than designers or makers. To solve this problem, we combine advances in mechanism synthesis, computer languages, and Design for Additive Manufacturing to create a computational framework, the MechProcessor, which allows novices to produce 3D-printable, moving mechanisms of varying complexity using simple and extendable interfaces. The paper describes how we use hierarchical cascading configuration languages, Breadth-First Search, and Mixed-Integer Linear Programming for mechanism synthesis, along with a nested, printable test case to detect and resolve the AM constraints needed to ensure the devices can be 3D printed.

We provide physical case studies and an open-source library of code and mechanisms that enable others to easily extend the MechProcessor framework. This encourages new research, commercial, and educational directions, including new types of customized printable robotics, business models for customer-driven design, and STEM education initiatives that involve non-technical audiences in mechanical design. By promoting novice interaction in complex design and fabrication of movable components, we can move society closer to the true promise of the Maker Movement: turning consumers into designers.

1 Introduction

Additive Manufacturing (AM)—or 3D printing—has blossomed substantially over the past decade, thanks in part to a growing philosophical culture called the *Maker Movement*: "people's need to engage passionately with objects in ways that make them more than just consumers" [1]. It posits that all people should be able to tinker with and invent new objects; a democratization of design that directly involves consumers in the design process.

This paper addresses a fundamental problem that consumers face when participating in the Maker Movement: "How can non-professionals generate or modify new designs that can be reliably printed, when they do not know how to use professional CAD tools or understand the limitations of AM equipment?" Currently, consumers have two options: 1) download pre-existing geometry files from a repository (*e.g.*, Thingiverse), limiting their ability to modify the design, or 2) use simplified CAD tools (*e.g.*, SketchUp, TinkerCAD, *etc.*), limiting a design's complexity.

We propose a third option, built off an analogy to how a word processor works. In a word processor, a user focuses on the content of the document (the words they type), while the processor uses a library of configurable settings (such as fonts and page styles) to typeset and layout the document. Users can extend a word processor by including third-party settings (*e.g.*, adding new fonts), or changing advanced layout settings themselves (*e.g.*, altering paragraph spacing or letter kerning). This abstraction provides different levels of control: complete novices can produce new printed documents easily by using default settings, while more advanced users can adjust settings manually or even provide new fonts or templates that can be shared with (or sold to) others.

In this paper, we apply this word processor analogy to mechanical devices, specifically mechanisms, in a framework we call the MechProcessor. The key difficulties lie in how to take ambiguous high-level mechanism graphs (*e.g.*, Fig. 4.2) and choose among (possibly equally good) design decisions, such as selecting components, connecting them in 3D space, and specifying dimensions such that it will print successfully. The paper's key contributions lie in a process for integrating advances across mechanism synthesis, computer languages, and Design for Additive Manufacturing to fully customize mechanical designs to printers of different qualities. Specifically, this paper contributes:

1. An extensible design representation that uses hierarchi-

*Address all correspondence to fuge@umd.edu

cal default configurations to resolve certain, pre-defined uncertainties not prescribed by the user, including specific component selections, part dimensions, and spatial orientations, subject to manufacturing constraints (Section 3.1).
2. Algorithms that use graph simplification techniques and Mixed-Integer Linear Programming to progressively reduce the mechanism design space complexity (Section 3.2).
3. A nested test geometry that, when printed, assesses a 3D printer's capabilities by only counting separated components (Section 3.3),allowing a novice user to customize a design to a particular machine.

While the above build upon existing computational synthesis methods, the key advance is the merger of three disparate areas (user-specifications of mechanisms, handling mechanical synthesis constraints, and capturing critical manufacturing information) to automatically adapt a design to various 3D printers. Section 4 demonstrates this through four-bar and six-bar linkages printed on both consumer- and professional-grade Fused-Deposition Modeling (FDM) machines, providing examples where successful parts must account for both user and manufacturing specifications. These contributions open up the possibility of permitting user-driven mechanical device design not only for specific engineering uses, but also for engineering education, new forms of online collaboration, and new business models.

2 Related Work

This paper builds upon advances in several different fields, including Mechanical Design, Computer Science, and Manufacturing. Specifically, we leverage techniques from Automated Mechanism Synthesis, Digital Fabrication, and Additive Manufacturing.

2.1 Design Representations for Mechanism Synthesis

Prior work in automated mechanism synthesis falls into roughly two areas: Graph Theoretic techniques for analyzing or computing mechanism properties, and Artificial Intelligence (AI) techniques for searching over mechanism structure or function. Our work combines these areas while integrating techniques specifically designed to address Design for Additive Manufacturing concerns (Section 3.3).

Kinematic analysis of mechanisms shares many important parallels with Graph Theory, and there are textbooks [2, 3] and articles [4] that provide a thorough review of both. Important extensions of this graph theoretic framework include Kota *et al.* 's work on decomposing mechanisms into fundamental building blocks that can be combined together to synthesize new designs through dual vector algebras [5] and matrix compositions [6, 7]. For particular classes of mechanisms, specific techniques can be used to synthesize new mechanism graphs, such as Sunkari's and Schmidt's [8] use of group-theory over graph isomorphisms.

Researchers have also approached mechanism synthesis through the usage of AI Techniques. Hoeltzel and Chieng [9] were one of the first in long research thread that combined Search-based techniques with Rule-checking and traditional kinematic analysis. In these approaches a program would exhaustively enumerate various possible mechanism configurations that might form a kinematic graph and then use rules or search strategies to prune the number of options, using a heuristic to arrive at a mechanism layout which satisfies all constraints or minimizes some objective function. To this framework, others have added functional reasoning [10, 11], search heuristics [12], additional types of rules [13], and new types of constraints [14]. Alternative approaches include case-based reasoning techniques, where new mechanisms are formed by modifying or extending those in an existing library [15], or through an evolutionary process, such as Genetic Algorithms [16, 17]. This also includes a body of work under the nomenclature "Computational Design Synthesis" [18], where researchers have used techniques from Graph-Grammars [19, 20, 14, 21] and Boolean Satisfiability [22] to optimize mechanical systems. These approaches tackle a larger design scope, such as mechanical functional synthesis, than this paper addresses, allowing us to leverage the structure of mechanism design to simplify the necessary constraint solution techniques.

In relation to Additive Manufacturing of mechanisms, Mavroidis *et al.* [23] provided some of the earliest work on printing joints for robotic manipulators using Selective Laser Sintering. Throughout the next decade researchers would begin to combine 3D printing with mechanism testing [24], however it was only within the past year that robotic toolkits have emerged which combine algorithms for synthesizing mechanisms with the intention of directly printing them [25, 26, 27, 28]. Our paper builds upon these efforts by connecting mechanism design and robotics with the AM test cases necessary to allow novices to use these algorithms on different 3D printers.

2.2 Computer Graphics and Digital Fabrication

The Computer Graphics community has also recently begun focusing on design tools for mechanisms and geometry, under the nomenclature of "Digital Fabrication" [29]. Prior approaches cluster into either translating user input into planar mechanism constraints, or generating new geometry using a model library. Researchers in Mechanical Engineering have also explored techniques for simplifying CAD interfaces, including Sketch-based CAD [30, 31] and Gesture-based CAD [32], and through design repositories [33] and model-based synthesis (*e.g.*, DARPA's Adaptive Vehicle Make program [34]).

For translating user input, researchers have used Motion Capture [35] or Sketch-based input to define desired motion paths [27, 28] or generate links [36]. Their contributions focus on optimization techniques for matching a small class of mechanism primitives (gears, revolute joints, *etc.*) to specific kinematic paths. In contrast, this paper connects these user interface techniques with machine-specific AM limitations.

When using component libraries to generate new designs, past techniques focused on hierarchical compositions

of parametric components [37, 38, 25, 26] or probabilistic models of large collections [39]. Their central challenge has been modeling the geometric relationships between parts of an object so that it can be modified while preserving the overall design intent and manufacturability. The closest work to ours in this regard is Mehta *et al.* [26] who reason about foldable robots. In contrast, this paper encompasses a wider array of mechanism components, tackles general mechanism graphs rather than hierarchies, and customizes the design to machine-specific manufacturability limitations.

2.3 Additive Manufacturing

Relevant past research around Design for Additive Manufacturing (AM) has focused in two areas: design tools for specific mixed material properties, and how to account for AM errors in early-stage design.

While certain types of AM allow users to print multiple types of material, designing for such processes is challenging. Ma *et al.* [40] use traditional Fused Deposition Modeling to produce mechanism housings that include break-away shells that can be used for casting flexible materials; this allowed them to produce multi-material functional hinges using traditional FDM hardware. With machines capable of printing multiple materials, Skouras *et al.* [41] develop a finite element formulation that allowed them to simulate and optimize the material elasticity and actuation points to automatically achieve certain user-specified deformations. For mechanisms where different surface friction properties are needed, Cali *et al.* [42] devised a method for altering internal friction in single material printed joints by creating offset ridges that provide appropriate friction when rotated out of position. Lastly, Stava *et al.* [43] account for the material limitations of a specific printer by using Finite Element Analysis as a pre-processing step to insert additional supporting structure into a part.

AM machines also have varying dimensional accuracy, and designing around this error has been approached in multiple ways. The approaches most similar to our work use physical test cases to calibrate specific machines. For example, Bochmann *et al.* [44] design a set of calibration geometry to measure the dimensional accuracy of FDM machines using professional coordinate measuring machines. Rather than directly measuring dimensional accuracy, Clemon *et al.* [45] and Cali *et al.* [42] use the physical characteristics of printed objects to determine minimum wall closure distances and frictional properties, respectively.

Dimensional errors are also introduced as part of slicing operations: Hildebrand *et al.* [46] improve accuracy across parts by optimizing the geometry's slicing direction and then re-assembling it post-printing. Nelaturi *et al.* [47] formalize the geometric deviation of layered printed parts by defining a "printability map" that can adjust geometry so that it can be printed with lower error. Lastly, Rajagopalan and Cutkosky developed analytical error analysis techniques for predicting the kinematic performance envelope of additively manufacturing linkages [48]; these kind of techniques are a useful design validation step which would complement the strategies we propose in Section 3.2.

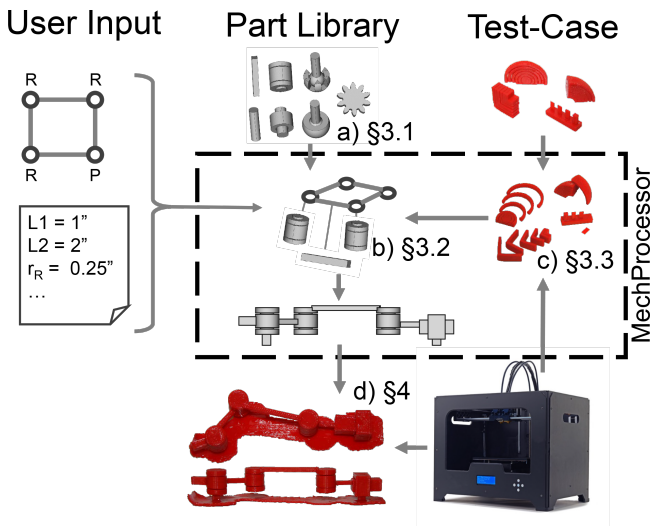

Fig. 1: An overview of MechProcessor framework. The user provides a kinematic graph, along with a set of desired default lengths, radii, *etc.*; a) a part library containing parametrized kinematic elements is b) matched to the input graph and adjusted to account for user-specified geometric constraints and c) machine capabilities, as captured by a calibration test-case. Our framework resolves these constraints and d) generates a customized STL file tuned to a particular AM machine's capabilities.

Our paper is most similar to the works of Cali *et al.* [42] and Bochmann *et al.* [44] in that we also create a printable test case to verifying certain AM manufacturing constraints for movable joints and mechanisms. Our work differs in that our test cases allow users with no technical background or measurement equipment to recover their AM machine's accuracy simply through counting (Section 3.3). This allows complete novices to easily calibrate a variety of printers (See Fig. 6).

3 The MechProcessor Framework

Our framework contains several key elements (shown in Fig. 1): a) a design representation that can be extended by users of different skill levels (Section 3.1); b) a constraint solver that handles geometric, manufacturing, and interface constraints (Section 3.2); and c) an additive manufacturing test case that identifies a machine's printing capabilities without high precision measurement tools (Section 3.3). Integrating these elements resolves three central ambiguities when transforming user input into a printable mechanism: ambiguities in user specifications, component selection, and machine capabilities.

3.1 Design Representation

Given a fully specified mechanism design, one can easily extract the underlying kinematic graph, however going the other way (from graph to design) is substantially more

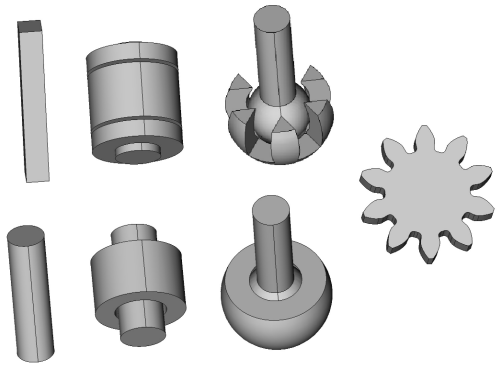

Fig. 2: Examples of some parametrized parts included in the library, include: Rod and Bar linkages; Cylindrical, Revolute, and Spherical joints; and involute gear pairs.

difficult [2]. This is due to an inherent loss-of-information: a graph only specifies the types of joints, which joints are connected to one another, and possibly specific lengths or angles that might constrain motion. That kinematic graph communicates what a mechanism should do (*e.g.*, its degrees of freedom, possible displacements, velocities, and accelerations, *etc.*), which is why novices and experts alike find it useful. However, that representation tells us nothing about many design details: How wide should a joint be? What does the physical joint look like, and how should links attach to that joint to maintain the desired motion? How large should gaps in joints be to create an acceptable running clearance on a particular printer? These types of detailed decisions tie the overall function (the mechanism graph) to specific geometric and manufacturing constraints.

The key challenge is specifying a design representation that is comprehensive enough to produce reasonable mechanisms given a general, high-level kinematic graph, but flexible enough to allow experts to specify design details. We accomplish this via two methods: 1) a parametric part library that provides specific instances of more general classes of mechanism components (*e.g.*, revolute joints), and 2) a configuration language that allows a graph to separately inherit information from users, default configurations, and manufacturing tests, and resolve inconsistencies or ambiguities.

For the part library, we use an object-oriented design representation where joints, linkages, or support structures inherit a common set of properties and actions, including: basic dimensions, the part's global placement in 3D space, and *interfaces* allowing other components to connect to the part (similar to port-based modeling methodologies [49], such as those used in SysML). Expert users can create new parts from scratch, or inherit and extend properties from existing parts. Figure 2 demonstrates typical joints and links, as well as sub-classing multiple instances of a joint (*e.g.*, the two spherical joint options).

For the configuration language, we use the YAML specification [50] to encode the kinematic graph and manufacturing constraints. Unlike XML and variants, YAML can store relational data structures and embed other YAML files as input, allowing one to nest mechanism descriptions or share complex designs. More importantly, one can define a separate set of default specifications, such as preferred joint components, dimensions, *etc.* that one can use to resolve ambiguities if the user does not specify particular details. This ability to fall back on prior configurations allows our algorithm to generate printable mechanisms when novice users only specify the high-level kinematic graph. It also permits a cascading specification pattern similar to how Cascading Style Sheets work in a web-browser: expert users can set particular details (*e.g.*, a particular desired thickness or joint type) and the algorithm can cascade backwards through a hierarchical set of default configurations for parameters that remain ambiguous after considering the user and manufacturing specifications.

For example, a user might specify that a joint should be a "Spherical Joint," however, given the two options in Fig. 2 it is unclear whether they mean a fully-enclosed joint or the cage-type socket. In this case, one of two things might occur: 1) If the options are equally preferable with respect to manufacturing limitations, the algorithm uses the one defined in the default specification (e.g., a fully-enclosed socket), much like a Word Processor might default to "Times New Roman"; 2) If the manufacturing specifications (Sec. 3.3) place limits on the part, such as minimum spherical clearances or single- versus dual- extrusion printing, then certain defaults might be eliminated and the algorithm would cascade back to other options. For example, the fully-enclosed socket in Fig. 2 can only be built using AM machines that print dissolve-able support, whereas the cage-type socket, while structurally weaker, can be printed in two parts on a single-extrusion machine and then snap-fit together after printing.

As another example, consider two different users building a crank-slider linkage (RRRP). For a user providing the minimum required kinematic graph as input and no additional details (Fig. 3), the algorithm would cascade back to using all default joints and linkage types. In contrast, a different user might specify particular joint and link details (such as increasing the link thicknesses, altering revolute joint sizes and types—Fig. 3-boxed), overriding some details and cascading back to defaults on others. By controlling this cascade process, the algorithm can ensure mechanism printability: in the boxes in Fig. 3 the user wants to decrease a joint size (d), however the algorithm might override those modifications if the test case (Section 3.3) indicates that the 3D printer could not produce free-running joints with that size or clearance.

3.2 Geometric Reasoning and Constraint Satisfaction

Next we resolve ambiguities in the specific configuration, placement, and dimensions of the mechanism. These include: 1) solving any position constraints regarding joint angles, *etc.*; 2) ensuring that parts correctly connect to adjoining parts to permit appropriate degrees-of-freedom; and 3) modifying any geometry so that it can be reliably printed on a specific AM machine. To solve these, we take advantage of the mechanism graph structure and use the following

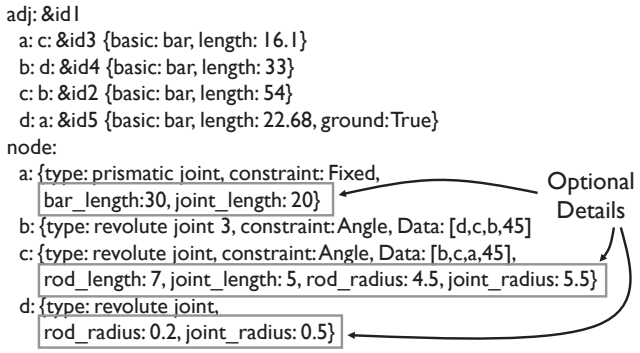

Fig. 3: An example of a user-input file for a crank-slider linkage (RRRP). The details inside the boxes indicate details that a user might provide to customize the generated mechanism. Removing these details would cause the algorithm to cascade back to default values. If user-provided values conflict with manufacturability limits from the test in Section 3.3, they would be overridden.

progression (Fig. 4).

First, we translate the user input (Fig. 4.1) into an idealized mechanism graph (Fig. 4.2) using the techniques in Section 3.1. We randomly project the node positions into 2D, and then use the geometric constraint solver proposed by van der Meiden and Bronsvoort [51] (Fig. 4.3) to resolve any geometric constraints.

If the graph is under-constrained (*e.g.*, missing certain lengths or angles), we substitute constraints as shown in Fig. 5. A common cause of under-constrained mechanisms lies in not constraining their driving Degree-of-Freedom (DoF): in a four- or six-bar linkage, there will typically be one angle or length that is needed to constrain the system to one particular point in its motion path. For CAD systems in general, automatically adding constraints is a indeterminate problem, however, one can leverage a mechanism's structure to solve special cases. For example, if a driving DoF is missing, we iterate over the mechanism graph as follows (Fig. 5):

1. Build a Breath-First Search graph over nodes in the graph, using one of the grounded nodes as the root,
2. Traverse the graph, adding candidate angle constraints between a node and two of its neighbors (for all permutations of neighbors),
3. Evaluate candidates using either a solver [51] or the Loop Mobility Criterion [Sec. 4.4] [2] to determine if the added constraint will determine the system,
4. If so, sweep angles in the constraint until the angle satisfies the rest of the mechanism constraints. If not, move to the next candidate.

Our present approach is limited to only slightly under-constrained systems (missing a single angle or length). For severely under-constrained systems missing several angles or lengths, one would need more advanced techniques, such as enumerating a mechanism's configuration space [53] and then selecting points among that space, or to elicit additional information from the user.

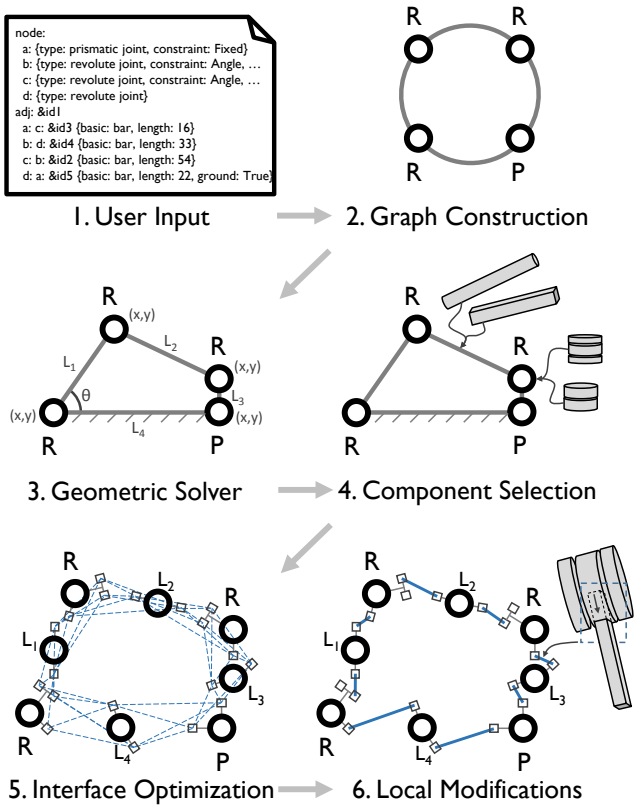

Fig. 4: How we progressively resolve ambiguities in the user input (1): (2) Create an un-directed mechanism graph, (3) solve for the ideal 2D positions using a geometric constraint solver [51], (4) substitute in user-selected components from the library, if specified, or appropriate defaults given the AM test (Section 3.3), (5) create an interface graph from the selected components and optimize build volume using Mixed-Integer Linear Programming [52], and (6) locally modify the geometry at the selected interfaces to ensure movement when printed.

If the input graph is over-constrained, or if the user specifies infeasible geometry, the algorithm raises an error to the user, who can modify the original input file. In cases where the model is correctly constrained, but the constraint solver [51] cannot find a solution, we restart with randomized positions to explore different local optima and then terminate if we are still unable to resolve the constraints.

Once the algorithm finds joint positions that satisfy geometric constraints for the graph (Fig. 4.3), it selects specific components (Fig. 4.4) from the part library (Section 3.1) to optimize further. It chooses a component from the library using either the user input file (if specified) or from a default configuration. At this point, the algorithm knows approximately where each part should go and which geometry it might use for each part, but not how to correctly position and interface those pieces.

To determine how components should connect, the algorithm requests the specific interfaces from each object (Section 3.1). Recall that each object (*e.g.*, revolute joint, bar linkage, prismatic joint, *etc.*) defines its own set of interface

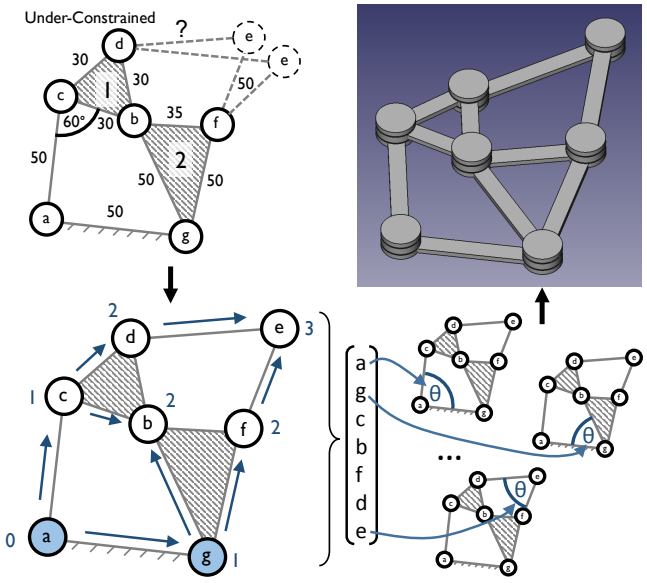

Fig. 5: For under-constrained systems, we attempt to reduce the mechanism degrees-of-freedom by adding angle constraints starting breadth-first from a randomly-selected ground node.

points on the geometry (*e.g.*, the end faces of a bar, or the outer collar of a revolute joint). The algorithm aggregates these interfaces and assumes that if two parts are linked in the graph, then any interfaces on one part could connect to any interfaces on another (Fig. 4.5).

The difficulty comes in selecting which particular interface pair to choose: 1) certain interfaces might be kinematically coupled (*e.g.*, the two ends of a revolute joints represent two different interfaces, despite sharing a common, rigidly-connected shaft), and 2) interfaces have physical consequences since a particular joint interface implies a specific height for a joint or linkage that must reconnect to the graph (See examples in Fig. 6).

To select appropriate interfaces, we formulate the task as a Mixed-Integer Linear Programming (MILP) problem: binary indicator variables activate or deactivate each interface and each part has a specific continuous z-height orthogonal to its movement direction. We then minimize the part's build volume (reducing printing time and support material), subject to the following constraints: 1) grounded links are at height = 0 w.r.t. the build plate; 2) two interfaces should connect at the same orthogonal height (*i.e.*, interfaces connected on the graph should be physically coincident in space); and 3) if a joint has interfaces for the same degree of freedom (*e.g.*, the two shaft ends in a revolute joint), then only one edge in the mechanism graph can connect to that degree of freedom (*i.e.*, edges passing through a node in the mechanism graph cannot share interfaces that are rigidly connected, since that would defeat the purpose of the joint). We then solve this MILP using Mitchell *et al.* [52]. For feasible solutions, the algorithm moves the parts so that the interface locations align correctly in 3D space. If the solver returns no feasible solutions, our algorithm terminates, since the joint parameters (such as heights, lengths, or depths) often over-constrain feasible solutions. One extension of this work would be to progressively relax non user-specified dimensions to expand the feasible region.

After the algorithm optimizes the positions and interfaces, it alters lengths, widths, radii, or other parameters of matching parts so that they fuse together correctly. For example, in Fig. 4.6, since the algorithm knows that the end face of a bar connects to the outside joint collar, it calls a function defined in the part library (Section 3.1) on the interface pair that modifies the bar by reducing its end length, such that it only connects to the outer collar without interfering with the joint's internal shaft.

3.3 Detecting Manufacturing Constraints

To print mechanisms on different printers, a user has to know their machine's capabilities. While one can approximate this using a manufacturer's specifications (*e.g.*, nozzle diameters, (x, y, z) tolerances, *etc.*), the printer's actual performance may vary substantially depending on how the machine was installed or its operating environment. In addition, the slicing software that turns a user's STL files into machine code often introduces additional variation. Given this variation, a user cannot predict how well his or her machine will produce mechanical parts. As mentioned in Section 2.3, past researchers have developed physical test cases for measuring machine accuracy, however these rely on measurement equipment and technical expertise.

To allow novice users to assess their machine's ability to produce movable joints, we designed a printable test-case that incorporates the principle used in a Matryoshka doll, also known as a Russian nesting doll: we nest similar parts on top of one another, monotically increasing their relative clearance distances (*e.g.*, 0.6mm, 0.5mm, 0.4mm, *etc.*), so that certain parts fuse together around the machine's threshold clearance.

In Fig. 6, the L-shaped, semi-circular, and hemispherical pieces test the clearances between planar, cylindrical, and spherical joints, respectively. The fourth piece varies the thickness from left to right (with dimensions 4mm, 2mm, 1mm, and 0.5mm, respectively), allowing a user to deflect each piece and qualitatively evaluate a material's strength.

Once a user prints the test-case, he or she simply counts the number of separated components (Fig. 6, upper row) and enters those counts into a configuration file. Based on these counts, our algorithm maps the number of separated components into approximate planar, circular, and spherical clearance tolerances. For example, given the number of separated parts in Fig. 6a, the minimum planar, revolute, and spherical clearances are 0.3mm, 0.4, and 0.6mm, respectively. One can also use this technique to compare printing performance along different orientations or at different locations in the build volume.

To verify the accuracy of our proposed test-case, we tested both its internal and external validity: if internally valid, our manufacturing test should correctly mirror the actual performance of printed joints, and if externally valid, it

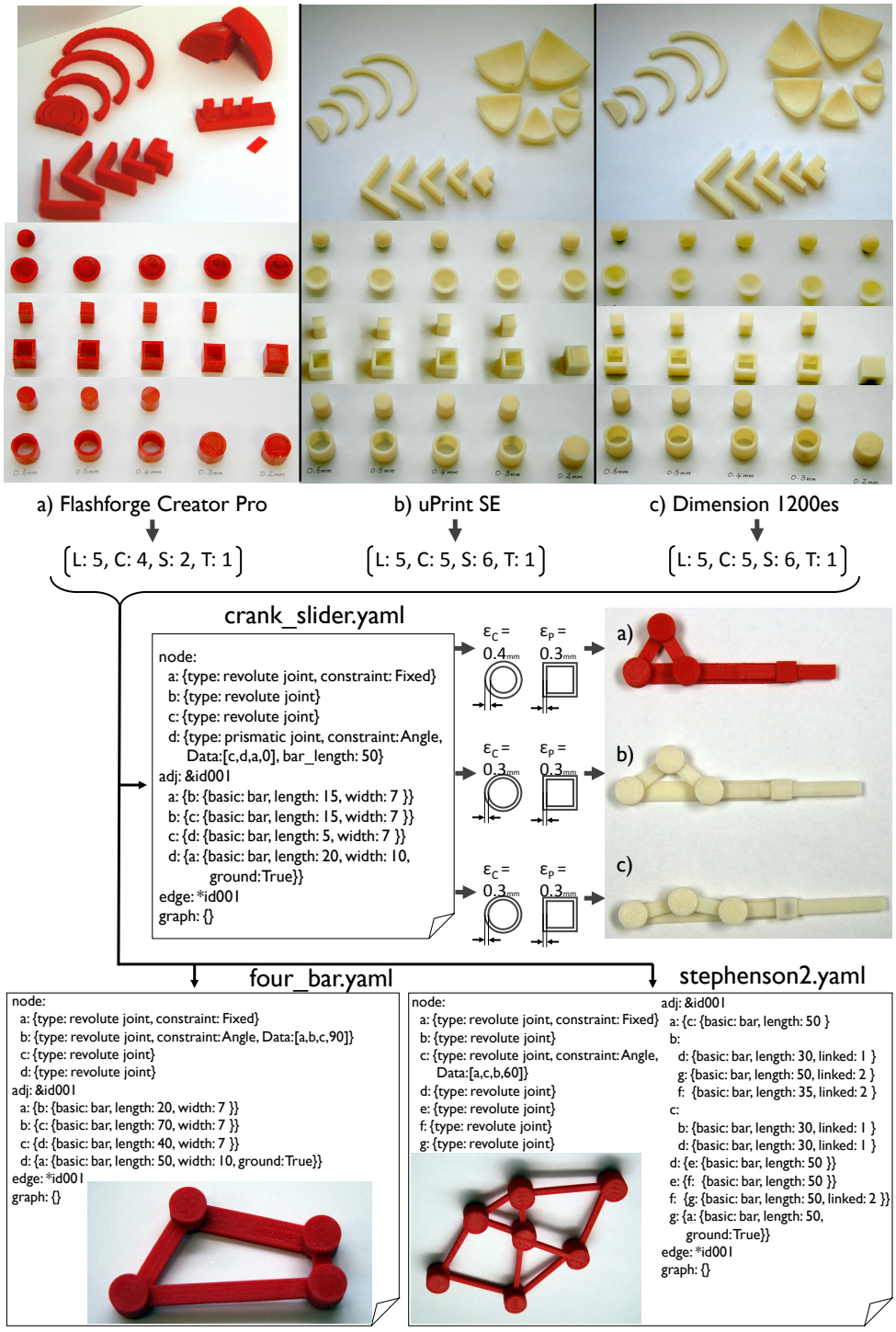

Fig. 6: **Top**: Our manufacturing test-case accurately emulates the performance of joints across three printers: a consumer-grade FDM machine (Flashforge Creator Pro), and two professional-grade FDM machines (Stratasys uPrintSE and Dimension 1200es). Across all tests, the joint clearances decrease monotonically from 0.6mm to 0.2mm. **Middle**: Our algorithms modify the user-specified geometry to match those achievable clearances. The right-hand side shows the same mechanisms, but printed on different printers and then manually positioned so that each mechanism is in a different point in its motion path. **Bottom**: the YAML input file and printed mechanism for a Four-Bar linkage and Stephenson Type-II linkage.

should consistently replicate across different machines and accuracy levels.

To demonstrate internal validity, we printed a series of prismatic, cylindrical, and spherical joints on our consumer-grade Flashforge Creator Dual Extrusion 3D printer using Acrylonitrile Butadiene Styrene (ABS) and High Impact Polystrene (HIPS) as the dissolvable support material. We recorded whether the performance of the joints mirrored the

test case. Each joint used clearances equivalent to the test case. As shown in Fig. 6a, the joints mirrored the test piece exactly as predicted: Clearance values that fused in the test piece, likewise fused in the joints.

To demonstrate external validity, we repeated the above test on two additional machines: the Stratasys Dimension 1200es and uPrint SE. These professional-grade machines cost one to two orders of magnitude more than standard consumer-grade printers, and one would expect their performance on both the test case and printed joints to improve commensurately. The Dimension 1200es and uPrint SE use a proprietary ABS blend as the main material, but different proprietary soluble support materials.

As expected, both professional-grade printers achieved smaller clearances compared to the Flashforge tests (Fig. 6). Moreover, the respective test cases mirrored the mechanical joint tests exactly across all printers. Our test-case provides more information than could simply be obtained using the manufacturers specifications: the fact that the Stratasys machines can produce tighter spherical and prismatic tolerances than circular tolerances cannot be readily discerned from the manufacturers specification sheets.

While this nesting approach should scale up or down to different resolution sizes and build volumes (provided the clearance sizes are adjusted appropriately), we have yet to verify that claim on machines with an accuracy greater than that of the Dimension 1200es. The test piece only works with removable support (the joints our library generates also require that capability). This test case only needs to be run once upon machine installation, and can be used to recalibrate our algorithm if the machine's performance degrades over time. The test case only accounts for systematic machine error and does not account for one-off errors in printing that occur due to mis-prints (*e.g.*, when supporting structures peel and warp due to an non-level build plate). In these cases, joints that would otherwise print fine will be warped and may not move as intended.

4 Physical Case Studies

We provide several physical case studies that demonstrate how a user input file translates to a physical mechanism: 1) a four bar linkage, 2) a crank-slider linkage, and 3) a six bar linkage. We implemented the framework in Python, using FreeCAD (as a wrapper for the OpenCasCADe CAD kernel) to generate the STL mesh files. We printed all parts on a sub-$1,500 FlashForge Creator Dual Extrusion 3D printer using Acrylonitrile Butadiene Styrene (ABS) and High Impact Polystrene (HIPS) as the dissolvable support material. Python code used the user-provided YAML files to generate all examples. Those who wish to replicate our results can download the open-source library, manufacturing test cases, and example STL files from the paper's companion website.[1]

All mechanisms can be parametrically controlled by the user to increase link lengths, *etc.* However, all joint and linkage placement is performed automatically, and the joint clearances for the revolute, prismatic, and spherical joints are calculated via the one-time calibration test in Fig. 6. Figure 6 demonstrated how a the user input file maps to the underlying mechanism graph and the resultant 3D printed mechanism. They also demonstrate how the design representation handles user-specified constraints, such as different joint lengths or types.

The printed examples mirror the results of our test case, except when the printer mis-prints its material. For example, when parts fail to adhere to the build plate securely or when extrusion nozzles accidentally scrape and move support filaments warping the part.

5 Discussion and Limitations

The above methods come with several limitations that future research could address: 1) design validation and optimization, 2) user interface design, 3) additional kinematic pairs, materials, & machine types, 4) algorithmic improvements to expand the feasible domain of mechanism designs, and 5) improvements to the manufacturing test-case.

5.1 Design Validation and Optimization

Presently, our algorithm does not perform static, dynamic, kinematic, or interference testing of the virtual model. Past research efforts in this area could be combined with our framework to achieve this [27, 28] by incorporating it into either the initial geometric solution (Fig. 4.2-3) or as an additional objective function during the MILP optimization (Fig. 4.5). Likewise, on the manufacturing side, one could use our physical test cases as a user-friendly input to analytical error-checking techniques [47, 48], identifying machine specifications by aggregating across multiple users and machines. A similar approach might be used for capturing the strengths of various print materials, and using those results to modify the geometry, as in [43].

5.2 User Interface Design

Presently, our algorithm relies on a set of YAML configuration files as input. Future work could provide more intuitive user interfaces that simplify how novices design and print working mechanisms. One possible direction could use 2D or 3D sketch [54, 30] or drag-and-drop interfaces [37] to automatically generate the input files. We intend to implement such an interface and conduct formal user studies with novice and expert participants in the future. Presently our representation provides basic geometric connectivity, but does not yet encode functional constraints, such as movement paths, which could be used to further guide mechanism generation.

5.3 Expanded Kinematic Pairs, Materials, and Machines

Our algorithms operate on lower-order kinematic pairs (*e.g.*, revolute joints, prismatic joints, spherical joints, *etc.*).

[1] `https://ideal.umd.edu/mechprocessor`

However, mechanisms will combine these lower-order pairs into higher-order pairs to save space or improve mechanism performance. Higher-order pairs would require improvements to the graph generation, geometric constraint solver, and interface optimization (Fig. 4.2, 3, & 5), to account for collapsed nodes and interfaces in the mechanism graph.

Our test-case and mechanisms require removable support material, given the internal joint complexity. While this is not a large barrier to entry (dual-extrusion FDM machines are presently available for under $1,500), the vast-majority of consumer-grade FDM machines only support single extrusion. Future research could explore algorithms for automatically segmenting the mechanism graph at joints in such a way that a single extruded parts might snap or assemble together into a larger mechanism. Expanding our techniques to account for mixed or functionally-graded materials [41] is another open challenge.

5.4 Expanding Feasible Mechanism Domains

Certain mechanisms represent future research avenues: 1) particular configurations of over- and under-constrained geometry, and 2) infeasible interface selections. Our algorithm terminates and requests a revision from the user if the user specifies inconsistent, over-constrained, or under-constrained geometry. This occurs if the default lengths, angles, *etc.* do not produce a feasible solution. This might be resolved if the algorithm had access to additional information: if it could simulate a mechanism's dynamic or kinematic performance, it could break ties between candidate constraints to produce reasonable motion. The trade-off between following user specifications and correcting faulty designs remains an open question. Interface optimization may also fail if the allowable part geometry is too rigidly defined (*i.e.*, it cannot adjust the heights of various joints or connections to find a feasible solution). In this case, we require the user to relax possible constraints on a part (such as the height) to open up feasible regions. However, future work could explore automated methods to sequentially relax these component constraints subject to some penalty.

5.5 Improving the Manufacturing Test-Case

In long joints with tight clearances dissolving the support material may take longer than a user is willing to wait, even if the joints would eventually separate. Figure 2 demonstrates one possible solution: a spherical joint uses a cage-type socket, rather than a fully enclosed socket, so that the solution more easily dissolves the supports.

The examples in Section 4 used polymer dual-extrusion FDM processes. While we do not expect our techniques to differ substantially for other processes or material types, future research could explore different manufacturing methods, such as Direct Metal Laser Sintering. Likewise, specific printer CAM settings (*i.e.*, how the machine transforms the STL file into tool paths) affect the test-case results: if settings such as infill size or how it shells an STL file differ between the test-case and the printed mechanism, one would expect the test-case to lose accuracy. One can avoid this by printing a mechanism with the same print settings used to print the test-case.

6 Conclusions

This paper presents methods for fabricating printable mechanisms from a user-specified graph and basic dimensions. We calibrate these methods to a specific machine's capabilities using a nesting-doll test-case that does not require measurement equipment or prior knowledge of a machine's tolerances and specifications. We demonstrate this capability on several examples: a four bar linkage, a crank-slider linkage, and a six-bar linkage. All examples are entirely programmatically generated.

Our framework accomplishes this through algorithms that leverage 1) an object-oriented design representation that allows experts and novices alike to describe mechanisms of differing complexity, cascading back to defaults when needed; 2) a series of constraint solvers that calculate mechanism positions, select components, and optimize how those components attach to one another (Fig. 4); and 3) a printable test-case that captures machine variation in planar, revolute, and spherical joint clearances, as well as minimum part thickness. Through simple counting, a user can capture appropriate joint tolerances, which our algorithm uses to modify the geometry. To verify the test case, we compare the test-case to printed mechanical joints across three printers of varying quality: a Flashforge Creator Pro, a Stratasys Dimension 1200es, and a Stratasys uPrint SE. In all cases, the joints match test-cases exactly. The entire library and source-code is available for experimental replication or extension by future researchers.[2]

The utility of our contributions are several fold:

1. Novices can now prototype complex, moving mechanisms without needing familiarity with the full capabilities of a CAD system or measurement equipment;
2. Mechanisms and Robotics researchers can use or modify our library to quickly test and print mechanisms or platforms on various AM machines;
3. Our AM test-case provides an intuitive way to approximate joint clearances required for mechanisms across different AM size scales and without requiring any measuring equipment.

All of the above can be performed on equipment costing less than $1,500, lowering the barrier-to-entry for printable mechanisms and robotics. It also opens up many opportunities for educational initiatives that can use our framework and library for STEM education or outreach.

The "Maker Movement" and recent interest in Additive Manufacturing has placed renewed interest on incorporating novices into the design of new devices. Society needs tools not just for the mass consumption of AM goods, but also those that empower novices to create new ideas. Our framework steps in that direction, providing a means for those

[2] `https://ideal.umd.edu/mechprocessor`

without extensive mechanical knowledge to contribute and identify as makers; to turn consumers into designers.

Acknowledgements

We thank the reviewers for their helpful suggestions which improved the manuscript. We also thank the Department of Mechanical Engineering at the University of Maryland for contributing the AM machines used to conduct the tests.

References

[1] Dougherty, D., 2012. "The Maker Movement". *Innovations: Technology, Governance, Globalization,* **7**(3), July, pp. 11–14.

[2] Tsai, L.-W., 2010. *Mechanism design: enumeration of kinematic structures according to function*. CRC press.

[3] McCarthy, J. M., 2000. *Geometric design of linkages*, Vol. 11. Springer Science & Business Media.

[4] Mruthyunjaya, T., 2003. "Kinematic structure of mechanisms revisited". *Mechanism and Machine Theory,* **38**(4), pp. 279–320.

[5] Moon, Y.-M., and Kota, S., 2002. "Automated synthesis of mechanisms using dual-vector algebra". *Mechanism and Machine Theory,* **37**(2), pp. 143–166.

[6] Kota, S., and Chiou, S.-J., 1992. "Conceptual design of mechanisms based on computational synthesis and simulation of kinematic building blocks". *Research in Engineering Design,* **4**(2), pp. 75–87.

[7] Chiou, S.-J., and Sridhar, K., 1999. "Automated conceptual design of mechanisms". *Mechanism and machine theory,* **34**(3), pp. 467–495.

[8] Sunkari, R. P., and Schmidt, L. C., 2006. "Structural synthesis of planar kinematic chains by adapting a mckay-type algorithm". *Mechanism and Machine Theory,* **41**(9), pp. 1021 – 1030.

[9] Hoeltzel, D. A., and Chieng, W.-H., 1990. "Knowledge-based approaches for the creative synthesis of mechanisms". *Computer-aided design,* **22**(1), pp. 57–67.

[10] Chakrabarti, A., and Bligh, T. P., 1994. "An approach to functional synthesis of solutions in mechanical conceptual design. part i: Introduction and knowledge representation". *Research in Engineering Design,* **6**(3), pp. 127–141.

[11] Chakrabarti, A., and Bligh, T. P., 1996. "An approach to functional synthesis of mechanical design concepts: theory, applications, and emerging research issues". *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing,* **10**(04), pp. 313–331.

[12] Li, C., Tan, S., and Chan, K., 1996. "A qualitative and heuristic approach to the conceptual design of mechanisms". *Engineering Applications of Artificial Intelligence,* **9**(1), pp. 17–32.

[13] Wang, Y.-X., and Yan, H.-S., 2002. "Computerized rules-based regeneration method for conceptual design of mechanisms". *Mechanism and Machine Theory,* **37**(9), pp. 833–849.

[14] Radhakrishnan, P., and Campbell, M. I., 2012. "An automated kinematic analysis tool for computationally synthesizing planar mechanisms". In ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1553–1562.

[15] Han, Y.-H., and Lee, K., 2006. "A case-based framework for reuse of previous design concepts in conceptual synthesis of mechanisms". *Computers in Industry,* **57**(4), pp. 305–318.

[16] Zhang, L., Wang, D., and Dai, J. S., 2008. "Biological modeling and evolution based synthesis of metamorphic mechanisms". *Journal of Mechanical Design,* **130**(7), p. 072303.

[17] Lipson, H., and Pollack, J. B., 2000. "Automatic design and manufacture of robotic lifeforms". *Nature,* **406**(6799), pp. 974–978.

[18] Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N. V., and Wood, K. L., 2011. "Computer-Based design synthesis research: An overview". *Journal of Computing and Information Science in Engineering,* **11**(2), pp. 021003+.

[19] Helms, B., Shea, K., and Hoisl, F., 2009. "A framework for computational design synthesis based on graph-grammars and function-behavior-structure". In ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 841–851.

[20] Radhakrishnan, P., and Campbell, M. I., 2011. "A graph grammar based scheme for generating and evaluating planar mechanisms". In *Design Computing and Cognition '10*. Springer, pp. 663–679.

[21] Schmidt, L. C., Shetty, H., and Chase, S. C., 2000. "A graph grammar approach for structure synthesis of mechanisms". *Journal of Mechanical Design,* **122**(4), pp. 371–376.

[22] Münzer, C., Helms, B., and Shea, K., 2013. "Automatically transforming Object-Oriented Graph-Based representations into boolean satisfiability problems for computational design synthesis". *Journal of Mechanical Design,* **135**(10), July, pp. 101001+.

[23] Mavroidis, C., DeLaurentis, K. J., Won, J., and Alam, M., 2001. "Fabrication of non-assembly mechanisms and robotic systems using rapid prototyping". *Journal of Mechanical Design,* **123**(4), pp. 516–524.

[24] Lipson, H., Moon, F. C., Hai, J., and Paventi, C., 2005. "3-d printing the history of mechanisms". *Journal of Mechanical Design,* **127**(5), pp. 1029–1033.

[25] Mehta, A. M., DelPreto, J., Shaya, B., and Rus, D., 2014. "Cogeneration of mechanical, electrical, and software designs for printable robots from structural specifications". In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, IEEE, pp. 2892–2897.

[26] Mehta, A., DelPreto, J., and Rus, D., 2015. "Integrated codesign of printable robots". *Journal of Mechanisms and Robotics,* **7**(2), Feb., pp. 021015+.

[27] Coros, S., Thomaszewski, B., Noris, G., Sueda, S., Forberg, M., Sumner, R. W., Matusik, W., and Bickel, B., 2013. "Computational design of mechanical characters". *ACM Transactions on Graphics (TOG),* **32**(4), p. 83.

[28] Thomaszewski, B., Coros, S., Gauge, D., Megaro, V., Grinspun, E., and Gross, M., 2014. "Computational design of linkage-based characters". *ACM Transactions on Graphics (TOG),* **33**(4), p. 64.

[29] Mota, C., 2011. "The rise of personal fabrication". In Proceedings of the 8th ACM conference on Creativity and cognition, ACM, pp. 279–288.

[30] Fuge, M., Yumer, M. E., Orbay, G., and Kara, L. B., 2012. "Conceptual design and modification of freeform surfaces using dual shape representations in augmented reality environments". *Computer-Aided Design,* **44**(10), pp. 1020 – 1032. Fundamentals of Next Generation CAD/E Systems.

[31] Kara, L. B., D'Eramo, C. M., and Shimada, K., 2006. "Pen-based styling design of 3d geometry using concept sketches and template models". In Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling, SPM '06, ACM, pp. 149–160.

[32] Murugappan, S., Piya, C., Ramani, K., et al., 2012. "Handy-potter: Rapid 3d shape exploration through natural hand motions". In ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 19–28.

[33] Design Engineering Lab. Design repository. `http://design.engr.oregonstate.edu/repo`. Accessed: 2015-04-09.

[34] de Weck, O. L., 2012. "Feasibility of a 5x speedup in system development due to meta design". In ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1105–1110.

[35] Ceylan, D., Li, W., Mitra, N. J., Agrawala, M., and Pauly, M., 2013. "Designing and fabricating mechanical automata from mocap sequences.". *ACM Transactions on Graphics (TOG),* **32**(6), p. 186.

[36] Megaro, V., Thomaszewski, B., Gauge, D., Grinspun, E., Coros, S., and Gross, M., 2014. "ChaCra: An interactive design system for rapid character crafting". In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association.

[37] Schulz, A., Shamir, A., Levin, D. I., Sitthi-Amorn, P., and Matusik, W., 2014. "Design and fabrication by example". *ACM Transactions on Graphics (TOG),* **33**(4), p. 62.

[38] Lau, M., Ohgawara, A., Mitani, J., and Igarashi, T., 2011. "Converting 3d furniture models to fabricatable parts and connectors". In ACM Transactions on Graphics (TOG), Vol. 30, ACM.

[39] Kalogerakis, E., Chaudhuri, S., Koller, D., and Koltun, V., 2012. "A probabilistic model for component-based shape synthesis". *ACM Transactions on Graphics (TOG),* **31**(4), p. 55.

[40] Ma, R. R., Belter, J. T., and Dollar, A. M., 2015. "Hybrid deposition manufacturing: Design strategies for multimaterial mechanisms via Three-Dimensional printing and material deposition". *Journal of Mechanisms and Robotics,* **7**(2), Feb., pp. 021002+.

[41] Skouras, M., Thomaszewski, B., Coros, S., Bickel, B., and Gross, M., 2013. "Computational design of actuated deformable characters". *ACM Transactions on Graphics (TOG),* **32**(4), p. 82.

[42] Calì, J., Calian, D. A., Amati, C., Kleinberger, R., Steed, A., Kautz, J., and Weyrich, T., 2012. "3d-printing of non-assembly, articulated models". *ACM Transactions on Graphics (TOG),* **31**(6), p. 130.

[43] Stava, O., Vanek, J., Benes, B., Carr, N., and Měch, R., 2012. "Stress relief: improving structural strength of 3d printable objects". *ACM Transactions on Graphics (TOG),* **31**(4), p. 48.

[44] Bochmann, L., Bayley, C., Helu, M., Transchel, R., Wegener, K., and Dornfeld, D., 2015. "Understanding error generation in fused deposition modeling". *Surface Topography: Metrology and Properties,* **3**(1), p. 014002.

[45] Clemon, L., Sudradjat, A., Jaquez, M., Krishna, A., Rammah, M., and Dornfeld, D., 2013. "Precision and energy usage for additive manufacturing". In ASME 2013 International Mechanical Engineering Congress and Exposition. Volume 2A: Advanced Manufacturing, ASME, pp. V02AT02A015+.

[46] Hildebrand, K., Bickel, B., and Alexa, M., 2013. "Orthogonal slicing for additive manufacturing". *Computers & Graphics,* **37**(6), pp. 669–675.

[47] Nelaturi, S., Kim, W., and Kurtoglu, T., 2015. "Manufacturability feedback and model correction for additive manufacturing". *Journal of Manufacturing Science and Engineering,* **137**(2), Feb., pp. 021015+.

[48] Rajagopalan, S., and Cutkosky, M., 2003. "Error analysis for the in-situ fabrication of mechanisms". *Journal of mechanical design,* **125**(4), pp. 809–822.

[49] Liang, V.-C., and Paredis, C. J., 2004. "A port ontology for conceptual design of systems". *Journal of Computing and Information Science in Engineering,* **4**(3), pp. 206–217.

[50] Ben-Kiki, O., Evans, C., and Ingerson, B., 2005. YAML Ain't Markup Language (YAML) Version 1.1. Tech. rep., yaml.org.

[51] van der Meiden, H. A., and Bronsvoort, W. F., 2010. "A non-rigid cluster rewriting approach to solve systems of 3d geometric constraints". *Computer-Aided Design,* **42**(1), pp. 36 – 49. Advances in Geometric Modelling and Processing.

[52] Mitchell, S., OSullivan, M., and Dunning, I., 2011. PuLP: a linear programming toolkit for python. Tech. rep., The University of Auckland, Auckland, New

Zealand.

[53] Milgram, R. J., and Trinkle, J. C., 2004. "The geometry of configuration spaces for closed chains in two and three dimensions". *Homology, Homotopy and Applications,* **6**(1), pp. 237–267.

[54] Eicholtz, M., and Kara, L. B., 2015. "Intermodal image-based recognition of planar kinematic mechanisms". *Journal of Visual Languages & Computing,* **27**(0), pp. 38 – 48. Distributed Multimedia Systems DMS2014 Part II.