Analysis and Comparison of Computationally-generated 3D Puzzle and Non-Puzzle Interlocking Algorithms

Dominic Ngoetjana
NGWKGA001
University of Cape Town

ABSTRACT

Computationally-generated and interlocking 3D puzzles are a fascinating concept—particularly in how there are various methods that attempt to as efficiently, accurately, and speedily as possible generate a variety of 3D puzzles that are not only challenging but incite more creation and excitement for the type of puzzles that are offered. We describe and compare the various methods and algorithms used to computationally generate these 3D objects, along with what these algorithms have in common and how they compare in being computationally efficient with regards to their input model processing, output, and interlocking mechanisms.

CCS Concepts

• **Mathematics of computing** → **Information Theory;** • **Theory of computation** → **Design and analysis of algorithms** → **Mathematical optimization** → Continuous optimization; • **Theory of Computation** → **Theory and algorithms for application domains** → **Algorithmic game theory and mechanism design** → Algorithmic game theory

Keywords

Interlocking Puzzle(s); 3D Printing; 3D Models; Computationally Generated; Approach;

1. INTRODUCTION

3D puzzles are generally nontrivial geometric problems that challenge our ingenuity. They have been longstanding stimulating recreational artifacts, where the task is to put together the puzzle pieces to form a meaningful 3D shape [10]. The same applies to non-puzzle 3D objects that not only closely emulate puzzles but also real-world objects that may be intended for actual real-world use, such as furniture.

We focus on computationally-generated 3D puzzles and non-puzzles. This refers to objects that are (in the general case) presented as input to a computational program as a 3D triangle mesh or voxelized representation. These objects are then processed by a given algorithm that determines how to split the object into pieces that interlock. This object is then either reconstructed using external materials (such as wood), or 3D printed; after which, the object can be dis/assembled and is interlocking.

Interlocking assemblies have a long history in the design of puzzles, furniture, architecture, and other complex geometric structures. The key defining property of interlocking assemblies is that all component parts are immobilized by their geometric arrangement, preventing the assembly from falling apart [16].

Contribution. Our purpose here is to compare various approaches and algorithms that generate interlocking puzzles. The contribution that this makes will be in isolating approaches for specific purposes, and with particular attention to key characteristics of these approaches. This is not meant to present a new, subjectively better, or faster method, but rather provide an analysis of existing and documented methods.

2. OVERVIEW

The overall process is a simple one. First, a brief background and minor analysis of each approach (for 3D puzzles and non-puzzles respectively) will be given. Second, a tabled comparison of the 3D puzzle approaches listed below will follow. Then, a conclusion to wrap it up.

For the sake of uniform terminology and meaning, several term meanings will be clarified. The term *piece*, unless otherwise stated, refers to a component of a whole 3D object, that is extracted and used for dis/assembly in conjunction with other *pieces*, and may not necessarily lead to an interlocking state on its own. The term *segment*, unless otherwise stated, refers to a subset of a whole 3D object, formed by several adjoining pieces that belong to said object. The term *layer*, unless otherwise stated, refers to a subset of a whole 3D object, formed by several adjoining segments/pieces that belong to said object, where each piece/segment shares an approximate vertical location in the object.

3. ALGORITHMIC APPROACHES

The set of algorithms to be compared are those specifically referring to the computational generation of 3D interlocking puzzle and non-puzzle objects (as per the following subsections 3.1 and 3.2). Each subsection consists of a variety of categories of algorithms based on the type of approach used to generate said objects. This section provides a background on the represented paper, the algorithm used, and a preliminary analysis of its effectiveness and use.

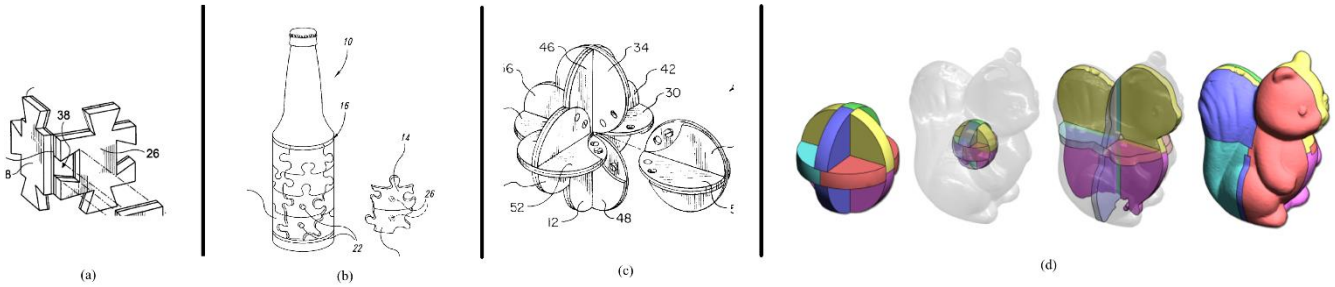


Figure 1: (a) dovetail-type joints; (b) non-planar clipping; (c) disc connected by pins; (d) extruded canonical 6-piece burr; (source: [2,13,9,14])

3.1 3D Puzzles

Puzzles provide an engaging challenge and remain a new and enjoyable experience so long as the way(s) to solve them remain undetermined or unclear; once techniques to solve them are determined, there tends to be a sense of disinterest in said puzzles that arises. A solution to this problem is found in interlocking puzzles—which provide both a challenge and a unique experience. With the advent of 3D printing, various types of 3D models can now be printed; combine this with techniques to segment the models into interlocking pieces and a variety of different puzzles can be generated and printed for renewed challenges. There are various techniques that have been adopted for engaging in this process of breaking objects into interlocking pieces. This subsection explores a few categories of these techniques.

3.1.1 3D Jigsaw Puzzle-esque

Background. In the early stages (and times) of the creation of interlocking puzzles, computational generation was not existent, thus puzzles were merely produced but not yet artificially generated for editing on computer systems. Some earlier examples of these types of puzzles include 3D jigsaw-esque puzzles; these are puzzles composed of pieces that resemble 2D jigsaw puzzles, only they contain larger depths and depending on the approach, the pieces may be planar or non-planar, i.e., flat.

Approach. Though at a high level the core procedure for compiling these types of puzzles is somewhat similar, the main distinction is in how the mechanics work to not only connect the pieces but also ensure that they interlock. One such method is one accomplished with two types of puzzle elements—one that can be folded into a corner configuration through a hinge that unites the planar segments of that element, and another that is generally planar, and (much like the first) does not have a specific design but rather differs per piece. The two element types interlock through a dovetail-type joint [2]; Figure 1 (a) depicts this element configuration. This method is ideally suited for flat structures (or objects built out of wall-like elements) such as house models. Taking this type of method further by using non-planar elements and different interlocking mechanics, this concept evolves further to emulate more than fundamentally flat structures. An example of this is one that uses a support structure as a basis for a known object, with the non-planar elements clipping onto the support structure to form a resembling outer surface of the known object, and held together by release pins in the interior of the support structure, such as in [13] and shown in Figure 1 (b). An optional addition to this approach includes forming a release mechanism inside the support structure configured to release at least one of the pieces on the exterior of the object.

Inspection. This approach proved sufficient to reproduce/emulate real-world objects and of various shapes and sizes. It succeeds in being generally applicable. One of its main downsides is that each support structure will have a finite and predetermined set of assemblable objects. It appears to have a low difficulty in terms of assembly, so it fails in providing a complex challenge as a puzzle. This shortfall means that it does not have longevity in terms of replayability and enjoyment. The main downside is, of course, that it is not computationally-generated and therefore cannot provide a variety of interlocking puzzles.

3.1.2 Disc and Extrusion

Background. Similarly to the aforementioned approach, this category of 3D puzzles also has some roots in physically-produced objects composed of likewise pieces. Before computability was applied to this type of interlocking puzzle creation, there existed one in which discs were used to interlock and form larger objects. This concept was later adapted to computation and enhanced with the process of extrusion.

Approach. The earlier method adopted for this type of puzzle made use of specifically and purposefully-designed discs. A "disc" within the context of [9] is described as a puzzle piece which includes a first flat side surface, a second flat side surface opposite the first side surface, and an outer circumferential surface between the first side surface and the second side surface around the circumference of each puzzle piece. Each of the puzzle pieces further includes a radial slot extending from approximately the center point along the radius to and through the outer edge and the outer surface. The disc, as shown in Figure 1 (c), has a singular design and each piece is able to interlock with another, albeit with the assistance of pins that are slotted into the two apertures on the disc. The puzzle is considered complete/solved once the total set of pieces are joined to form a sphere-like structure. With the power of computation added, later on, a similar (but not necessarily linked) concept was applied using a construct made up of disc-like segments, called a canonical six-piece burr puzzle (or knot). This burr puzzle acts as a basis for even larger puzzles composed of just of one of these, or a multitude. In the base case, this is achieved when the outer pieces of one knot are extruded using an anisotropic (axial) scaling until they go beyond the 3D model. The next step is to apply a CSG (Constructive Solid Geometry) intersection between the extruded six pieces and the given 3D model (i.e., isolate the intersecting parts of the model and extruded pieces) to produce the puzzle "skeleton" as shown in Figure 1 (d) [14]. The general case is when several knots are joined in a network in order to emulate the desired output object. Both cases produce an object that is perfectly interlocking and consists of one key piece.

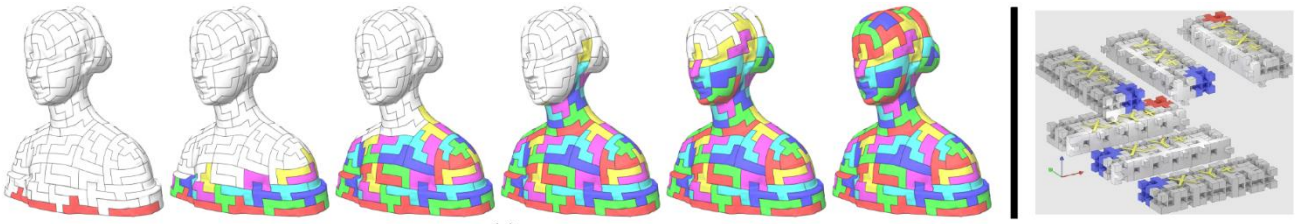


Figure 2: (a) 3D polyomino pieces layered bottom to top; (b) interlocking blocks layered bottom to top; (source: [5,15])

Inspection. The former method, not being computationally implemented, fails to be generally applicable. It is limited to one type of final structure that can be assembled using its pieces. It appears to have a low assembly difficulty and it lacks longevity due to its absence of variety. The latter method, with its computational implementation, is generally applicable. In using multiple knots it is able to contort its extrusion and conform to match a variety of objects. This variety enables it to add complexity to the types of interlocking puzzles produced and therefore provide a suitable challenge. It also has longevity attributed to its variety.

3.1.3 Layers

Background. This approach is one in which a realised object is built up layer by layer till completion. Each layer is built from the bottom up, as it would naturally be to assemble a real-world object with that type of layout of segments. Each layer may consist of several segments joined together. Each segment is composed of several pieces.

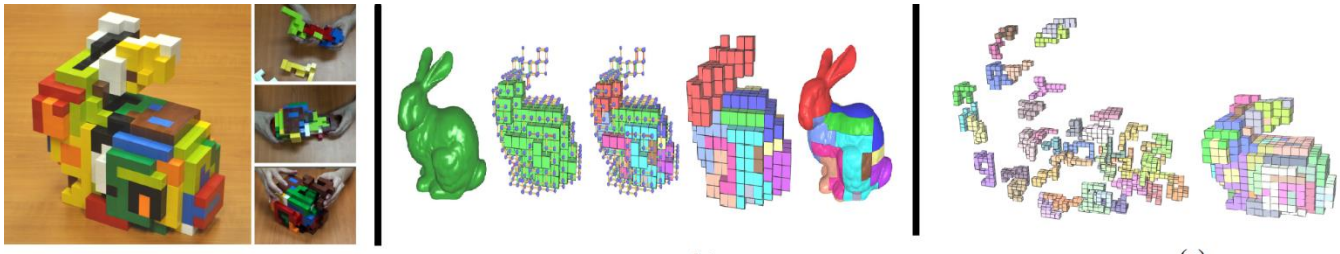
Approach. The first type of method that implements this kind of approach is one that uses polyominoes as basic building blocks for constructing the shapes used in the puzzle assembly. A polyomino is a generalization of a domino constructed by connecting *n* squares edge-to-edge instead of the two squares of an ordinary domino [5]. This method involves three main steps; The initial step is that it applies quad-based surface parametrization to the input solid, and tiles the parametrized surface with polyominoes. This means that a joint sequence of quads are mapped onto the entirety of the input 3D model such that each quad is flattenable to a square, which is achieved with the assistance of a dual graph over the surface, where each node corresponds to a quad region on the parametrized surface, and then a variety of polyomino shapes are generated and mapped to these quads. In the next step, the method constructs a nonintersecting offset surface inside the input solid and shapes the puzzle pieces to fit inside a thick shell volume (which is structured similarly to the input solid). The build-up process' consistency is assured with the use of a dependency graph—which describes the building-order frequency among the puzzles; in essence, the implication is that puzzle piece 1 depends on piece 2 when piece 1 is placed into a partially-completed puzzle model which already contains piece 2. Finally, the method develops a family of associated techniques for precisely constructing the geometry of individual puzzle pieces, including the ring-based ordering scheme, the motion space analysis technique, and the tab and blank construction method—all detailed in [5]. The end result is an interlocking puzzle, buildable from the bottom up. Another type of method that follows this approach is one that uses a voxelized model, with each voxel split into 8 equal-size blocks. To construct the model, the method first builds long flat chains of the squares (named segments) by connecting joints (male and female connectors on the blocks that permit movement in a certain direction depending on the type of block) of previous squares to new squares. Then the next step is to build layers, where a layer is a set of blocks with the same *z*-coordinate. Each layer is composed of segments, where each segment prevents movement of the previous segment. Then for each layer, the method determines the last block assembled, orders segments, determines the last block of each segment, determines assembly directions, and then ensures that there are joints between adjoining layers, segments, and blocks in every segment. The last step is to match every block with predefined blocks, and finally assemble the model using predefined blocks. This process is outlined in [15]. Both method types use a layered approach in that pieces have to be assembled from the bottom up, meaning each piece depends on the piece(s) below it when the assembling the puzzle. Figure 2 depicts this type of assembly system.

Inspection. The two methods of the layered approach mentioned above are sufficiently generally applicable in terms of the variety of objects that can be produced and assembled having used that approach as a basis. However, only the first method outputs an assembly of a model that still matches the original shape and structure of the original input model but still emulates a real object, whereas the second method both inputs and outputs a voxelized object, with no attempts to match any particular real-world geometry of the represented object. The first method appears fairly straight-forward in terms of assembly difficulty, whereas the second method appears to be a bit more complex, due to the various shapes and extrusions of the pieces. Though different in assembly difficulty, both methods have longevity since they vary in the types of objects that can work with.

3.1.4 Voxelisation

Background. This last approach in the 3D puzzle section highlights methods that rely on the voxelization process in order to produce pieces that interlock when assembled. Three methods will be highlighted—with the second and third building off the first.

Approach. The first method takes in a general voxelized shape as input and iteratively extracts puzzle pieces from it to generate the puzzle. First is the key piece, then the next piece adjacent to that, and so on. The method ensures that every three consecutive pieces interlock, and therefore ensuring that the whole puzzle interlocks without having to exhaustively verify this. As mentioned, the first step is to isolate and extract the key piece, which is accomplished by first picking candidate seed voxels—that is, voxels that are on the outer surface of the model and that are unobstructed above them, then calculate voxel accessibility, ensure blocking and



(a) (b) (c)

Figure 3: (a) recursive interlocking bunny; (b) voxelized bunny with surface; (c) voxelized bunny with disassembled pieces; (source: [10,12,16])

mobility in a given direction, and finally expand the key piece (select more adjoining voxels to balance the size of the puzzle pieces) and confirm that the voxels next to the key in a set can be visited or in the remaining set of pieces that exclude the next piece to be removed after the key. This process is similarly applied to the remainder of the voxels, with exceptions to this detailed in [10]. The end result is a voxelized interlocking puzzle that can be assembled and disassembled, with one piece as the key. The resulting pieces of this method can only be assembled in a specific order. The second method draws on concepts from the first, but still offers its own procedure, and outputs an object that matches the geometry of the input object, along with ensuring that this object can be 3D printed (by partitioning it into 3D parts that can be assembled) without concerns of its size. The first step in this method is to take in a 3D watertight (well-defined exterior boundary and closed volume) model as input, then place a voxel grid in the 3D object space and slightly adjust the voxel size to reduce the number of undesired partial voxels. Then, the input model is voxelized. Partial voxels with tiny fragments are avoided by slightly deforming the model geometry locally. Next, the connection strength between neighboring (partial) voxels is measured and a shape connection graph is built by analyzing the local shapes [12]. Then, construct an initial set of interlocking 3D parts from the internal voxels, and attach the remaining partial voxels one by one to the constructed interlocking parts without breaking the fulfilled requirements (structural soundness and strong connection). Lastly, the method constructs the geometry of the final 3D parts by using CSG intersection (mentioned earlier) between the voxelized parts and the input mesh. The third method in this approach also draws on the concepts and techniques in the first method. To start off with, the input expected is the final shape of the assembly, from which the component parts are either constructed from scratch as in [10] or explicitly initialized. The computational process for creating an interlocking assembly starts with the full input model, then iteratively splits off successive parts for disassembly. At each iteration, it first identifies a set of suitable blocking relations to be generated between the current assembly and the new part such that the interlocking property is maintained. Then it searches for the part geometry that satisfies these blocking relations. The selection of a new part is guided by a ranking function that takes into account certain geometric properties, e.g., part size, or other requirements, e.g., on part fabrication. The search space is then explored in a tree traversal process that uses automatic backtracking when no interlocking solution could be found in a specific iteration. The method implementation includes a user interface to interactively explore different options for part decomposition, allowing the user to overwrite the generic ranking function for part selection. The end result is a perfectly interlocking voxelized puzzle [16]. The third method has a major difference between it and the first—that being that while the first relies only on the previous piece to immobilize the current piece, the third relies on all previous pieces to immobilize the current and any other remaining pieces. All three methods fundamentally rely on the voxelization process in order to segment the input model into interlocking pieces. Figure 3 gives an idea of these methods.

Inspection. All three methods are generally applicable, i.e., can generate a wide variety of interlocking objects. The difficulty of assembling any of these depends entirely on the builder's competency and the model, size, and number of pieces of the relevant puzzle. All of the methods have longevity, attributed to the variety in the type of puzzles that can be generated.

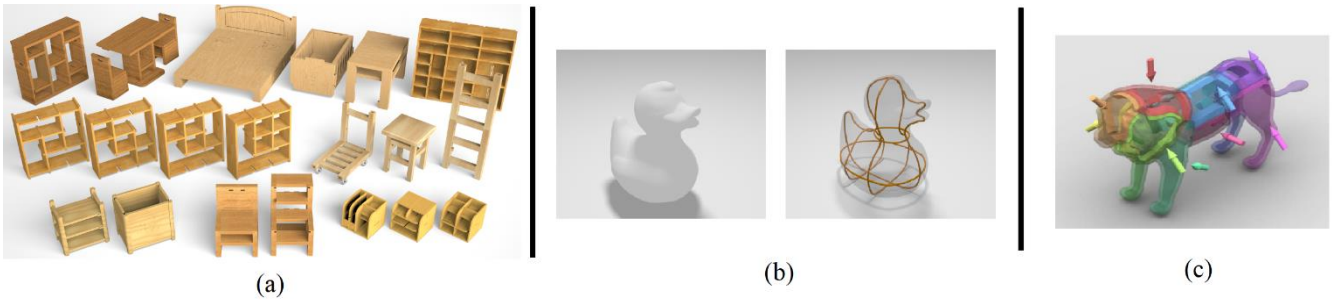
3.2 3D Non-Puzzles

Computationally-generated objects go beyond just 3D puzzles—they span to whatever the algorithm/approach applied can reproduce. This can be anything from toys to furniture, general, and novelty items all alike. The possibilities are endless. The objects visualized with these methods generally inform and feed into new ways of creating objects that are self-reliant in terms of assembly, i.e., don't require external objects to keep them held together. This is particularly true for those with interlocking mechanisms. This subsection explores a few approaches used in pursuit of these mechanisms. Perhaps, to some extent, these approaches could potentially be applicable to 3D puzzles—but this requires some intel.

3.2.1 Furniture

Background. Furniture typically consists of assemblies of elongated and planar parts that are connected together by glue, nails, hinges, screws, or other means that do not encourage disassembly and re-assembly [3], so new approaches were developed that would rid the process of these additional means, i.e., through interlocking mechanisms. There are various methods for this purpose but three will be highlighted here.

Approach. The first method presents a computational solution to support the design of a network of interlocking joints that form a globally-interlocking furniture assembly. The key idea is to break the furniture complex into an overlapping set of small groups, where the parts in each group are immobilized by a local key, and adjacent groups are further locked with dependencies. The



(a) (b) (c)

Figure 4: (a) reconfigurable furniture; (b) duck formed from wire structure; (c) lion made of shell pieces; (source: [11,8,7])

dependency among the groups saves the effort of exploring the immobilization of every subset of parts in the assembly, thus allowing the intensive interlocking computation to be localized within each small group [3]. The second method presents an interactive tool for designing intrinsic joints. Users draw the visual appearance of the joints on the surface of an input furniture model as groups of two-dimensional regions that must belong to the same part. The method automatically partitions the furniture model into a set of solid 3D parts that conform to the user-specified 2D regions and assemble into the furniture. If the input does not merit assemblable solid 3D parts, then the method reports the failure and suggests options for redesigning the 2D surface regions so that they are assemblable. Similarly, if any parts in the resulting assembly are unstable, then the method suggests where additional 2D regions should be drawn to better interlock the parts and improve stability. To perform this stability analysis, a novel variational static analysis method that addresses the shortcomings of the equilibrium method for this task is introduced. Specifically, this method correctly detects sliding instabilities and reports the locations and directions of sliding and hinging failures [4]. The third method presents computational methods as tools to assist the design and construction of reconfigurable assemblies (i.e., consists of a common set of parts that can be assembled into different forms for use in different situations), typically for furniture. There are three key contributions to this work. First, the method presents the compatible decomposition as a weakly-constrained dissection problem, and derive its solution based on a dynamic bipartite (consisting of two parts) graph to construct parts across multiple forms; particularly, the method optimizes the parts reuse and preserve the geometric semantics. Second, the method develops a joint connection graph to model the solution space of reconfigurable assemblies with part and joint compatibility across different forms. Third, the method formulates the backward interlocking and multi-key interlocking models, with which it iteratively plans the joints consistently over multiple forms [11]. An example of furniture is shown in Figure 4 (a).

Inspection. The first method performs well for its intended purpose but its interlocking mechanism isn't suited to puzzles, similarly to the third method, due to their strictly purposed parts and inconvenience of disassembly. The second method, however, can be reconstructed as a puzzle thanks to its variety of parts that can be assembled and disassembled with more ease, and that interlock.

3.2.2 General

Background. Unlike the previous approach (to some extent) there are approaches that were designed to be generally applicable, i.e., generate or guide in creating a variety of different objects. Two methods within this approach will be highlighted here.

Approach. The first method is one that presents a software environment intended to support the fluid interactive design of reconfigurables, featuring tools that identify, visualize, monitor and resolve infeasible configurations [1]. This method relies on users to use the tools for the creation and editing of reconfigurables, and aids in that process; although it's largely autonomous. The second method that falls within the *General* approach is one that allows the computation of aesthetically pleasing structures that are structurally stable, efficiently fabricatable with a 2D wire bending machine, and assemblable without the need of additional connectors. Starting from a set of planar contours provided by the user, this method automatically tests for the feasibility of a design, determines a discrete ordering of wires at intersection points, and optimizes for the rest shape of the individual wires to maximize structural stability under frictional contact [8]. An example of a generally applicable method is shown in Figure 4 (b).

Inspection. Both of the methods are generally applicable and specialize in creating/designing a wide range of different objects. Unfortunately, though, neither is suited for creating puzzles—interlocking ones included as the first simply contorts to reconfigure itself into a predefined configuration, and doesn't necessarily assemble and disassemble with ease, or would be suitable to do so recreationally, and the second would be too complex and not suitably designed for recreational use as well.

3.2.3 Novelty

Background. In addition to the specialized and purposed approaches stated earlier, there also exist approaches that merely facilitate in creating/designing novelty objects, i.e., cheap or unusual objects. Two such methods will be highlighted here.

Approach. One such method in this approach is one that presents a computational system to design an interlocking structure of a partitioned shell model, which uses only male and female connectors to lock shell pieces in the assembled configuration. Given a mesh segmentation input, this system automatically finds an optimal installation plan specifying both the installation order and the installation directions of the pieces, and then builds the models of the shell pieces using optimized shell thickness and connector sizes. To find the optimal installation plan, simulation-based and data-driven metrics are developed and incorporated into

an optimal plan search algorithm with fast pruning and local optimization strategies. The whole system is automatic, except for the shape design of the key piece [7]. The second method is one that presents an interactive tool for designing physical surfaces made from flexible interlocking quadrilateral elements of a single size and shape. With the element shape fixed, the design task becomes one of finding a discrete structure—i.e., element connectivity and binary orientations—that leads to the desired geometry. In order to address this challenging problem of combinatorial geometry, the method proposes a forward modeling tool that allows the user to interactively explore the space of feasible designs. Paralleling principles from conventional modeling software, this approach leverages a library of base shapes that can be instantiated, combined, and extended using two fundamental operations: merging and extrusion [6]. An example of a novelty object is shown in Figure 4 (c).

Inspection. Both of the methods are generally applicable and specialize in creating/designing a wide range of different objects, just as the methods in the previously-mentioned approach. From a design standpoint, both methods would be suited for creating/designing puzzles, although the first method wouldn't offer a high difficulty due to its ease of assembly.

4. ALGORITHM COMPARISON

This section compares some key attributes/characteristics of algorithms, in an attempt to find one or a few that are optimal in generating interlocking puzzles, and perhaps finding one that subjectively does it best.

Ref	Computational Speed	Variability of Output	Model Size Limit	Quality of Output Puzzle	Level of Difficulty of Implementation	Interlocking Mechanism Difficulty
[2]	N/A	Medium—purposed for houses/castles, but can be repurposed for more	Depends on the size of the object's materials	Not computationally-generated, but the output is flat along each surface and interlocks without fault	N/A	Low
[13]	N/A	High—with non-planar pieces, more and different types of objects can be made	Depends on the size of the object's materials	Not computationally-generated, but the output interlocks without fault, although it consists of too many additional items to ensure interlocking	N/A	Medium
[9]	N/A	Low—Output always likened to a sphere. No variability	Each piece has an approximate radius of 38 cm, the diameter of the assembled object = 38*4 = 152	Not computationally-generated, but the output interlocks without fault. Can only be likened to a sphere and no other object	N/A	Low
[14]	Model: bunny #pieces: 16 Time (s): 163	High—but limited to 3D shapes with parts that aren't too flat or narrow	64 knots. 240 pieces	High—output is wooden, but still resembles source object	Medium	Medium
[5]	Model: bunny #pieces: 258 Time (s): 0.224	High—generates a wide variety of objects, but limited to models that don't have thin or highly curved parts	325 pieces	High—completely resembles source object	Medium	Low
[15]	Model: alpaca #pieces: 400 Time (s): 0.15	High	13104 pieces	Low—resembles source object but has no outer surface, and is blocky and non-smooth	Medium	Medium
[10]	Model: bunny #pieces: 10 Time (s): 378.6	High—generates voxelized variations of different objects, but limited by object pieces that rotate	1250 pieces	Low—resembles source object but the output is still a voxelized representation	Medium	Low

	#voxels: 20010					
[12]	Model: bunny #pieces: 16 Time (s): 43 #voxels: 536	High—generates a wide variety of objects, but limited to models that aren't hollow with thin boundaries	20 pieces	High—completely resembles source object	Medium	Low
[16]	Model: bunny #pieces: 10 Time (s): 43.8	High—generates voxelized variations of different objects	1500 pieces	Low—resembles source object but the output is still a voxelized representation	High	Low

Table 1: 3D Puzzle Algorithm Comparison

5. CONCLUSION

Interlocking is an intriguing but complex mechanical state, where assembled component pieces appear to lock one another. Yet, the puzzle can be disassembled through certain sequences of moves starting from the key(s) [10]. We have peered at various approaches to solving this problem and analyzed a few key characteristics that show resemblance or difference among them here. For general comparison purposes, a model of a bunny (or as close a representation as can be found) is used across all 3D puzzle approaches.

As can be seen from the comparisons in Table 1, the following apply:

Speed. Pertaining to computational speed, [15] is able to generate more pieces with the least amount of time.

Output variability. The majority of approaches have high variability.

Model size limit. [15] has the largest number of recorded pieces for its respective assembly.

Quality of output puzzle. [2,13,9] are not computationally-generated, the rest are, but only [14,5,12] are of high quality.

Implementation difficulty. [16] has shown to be the most difficult, while [10] has shown to be the least difficult.

Interlocking mechanism difficulty. On average, the difficulty is low. In this case, the higher the difficulty, the better it is for dis/assembly—as people generally want this to be a challenge. Approaches [13,14,15] accomplish this best.

Non-puzzles. On the basis of the *inspections* listed above for the described and analyzed non-puzzle approaches, there are three methods that could potentially be applied to puzzle generation—given time for testing. The first is [4]—where based on its ability to create a variety of interlocking parts that can be assembled and disassembled with more ease, paves way for the possibility of venturing into 3D puzzles, except that no computation times are provided and so this is only conjecture. The second prominent method is [7]—where similarly to [4], provides variety and ease of assembly; in addition, a squirrel model composed of 11 pieces takes 130.7 seconds to generate, which is significantly slower than the fastest 3D puzzle approaches but still makes it a choice among them. The last is [6]—where likewise, there's ease of assembly and variety of output; in addition, a bunny model composed of 188 pieces takes 160 seconds to generate, which is faster than a few of the 3D puzzle approaches, so it is a choice as well.

Overall. On the basis of Table 1 and minor analysis of each approach in subsections 3.1 and 3.2, it appears that [15], although generating models without surfaces for resembling source models, is the best/most efficient among the various approaches investigated here, based on the specific characteristics in Table 1.

6. REFERENCES

[1] AKASH, G., ALEC, J., AND EITAN, G. 2016. Computational Design of Reconfigurables. ACM Trans. on Graph. (SIGGRAPH) 35, 4 (2016). Article No. 90.

[2] BENOIT, P. AND GAREAU, D. (2000). *Three-dimensional Puzzle*. 6,086,067.

[3] FU, C.W., SONG, P., YAN, X., YANG, L.W., JARAYAMAN, P.K., AND COHEN-OR, D. 2015. Computational interlocking furniture assembly. ACM Trans. Graph. 34, 4, Article 91 (July 2015), 11 pages.

[4] JIAXIAN, Y., DANNY, K., YOTAM, G., AND MANEESH, A. 2017. Interactive Design and Stability Analysis of Decorative Joinery for Furniture. ACM Trans. on Graph. 36, 2 (2017). Article No. 20.

[5] LO, K.-Y., FU, C.-W., AND LI, H. 2009. 3D Polyomino puzzle. ACM Tran. on Graphics (SIGGRAPH Asia) 28, 5. Article 157.

[6] MÉLINA, S., STELIAN, C., EITAN, G., AND BERHNHARD, T. 2015. Interactive Surface Design with Interlocking Elements. ACM Trans. Graph. (SIGGRAPH Asia) 34, 6 (2015). Article No. 224.

[7] MIAOJUN, Y., ZHILI, C., WEIWEI, X., AND HUAMIN, W. 2017a. Modeling, Evaluation and Optimization of Interlocking Shell Pieces. Comp. Graph. Forum 36, 7 (2017), 1–13.

[8] MIGUEL E., LEPOUTRE M., BICKEL B.: Computational design of stable planar-rod structures. *ACM Transactions on Graphics 35*, 4 (July 2016), 86:1–86:11.

[9] MILLER, JR., J. (1998). *Three Dimensional Interlocking Puzzle*. 5,762,336.

[10] PENG, S., CHI-WING, F., AND DANIEL, C. 2012. Recursive Interlocking Puzzles. ACM Trans. on Graph. (SIGGRAPH Asia) 31, 6 (2012). Article No. 128.

[11] PENG, S., CHI-WING, F., YUEMING, J., HONGFEI, X., LIGANG, L., PHENG-ANN, H., AND DANIEL, C. 2017. Reconfigurable Interlocking Furniture. ACM Trans. On Graph. (SIGGRAPH Asia) 36, 6 (2017). Article No. 174.

[12] PENG, S., ZHONGQI, F., LIGANG, L., AND CHI-WING, F. 2015. Printing 3D Objects with Interlocking Parts. Comp. Aided Geom. Des. 35-36 (2015), 137–148.

[13] SIMMONS, T. (2006). *Three-dimensional Puzzle*. US 7,021,625 B2.

[14] XIN, S.-Q., LAI, C.-F., FU, C.-W., WONG, T.-T., HE, Y., AND COHEN-OR, D. 2011. Making burr puzzles from 3D models. ACM Tran. on Graphics (SIGGRAPH) 30, 4. Article 97.

[15] YINAN, Z., AND DEVIN, B. 2016. Interlocking Structure Assembly with Voxels. In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems. 2173–2180.

[16] ZIQI, W., PENG, S., AND MARK, P. "DESIA: A General Framework for Designing Interlocking Assemblies". In: ACM Transactions on Graphics (SIGGRAPH Asia) 37.6 (2018). Article No. 191.