Algorithms to Generate Interlocking Three-Dimensional Puzzles

Nkosi Gumede
CSC4003W Literature Review
University of Cape Town, 2019
gmdnko003@myuct.ac.za

ABSTRACT

This paper takes a look at various pieces of literature on the topic of interlocking printable three-dimensional (3D) puzzles. Interlocking 3D puzzles lock solid pieces together to form an object. The puzzle pieces must be assembled in a specific order and the final object must be disassembled by the reverse of that order. Similar to the way a stack works, the last piece in the puzzle must be the first piece taken out. Known approaches to generate interlocking 3D puzzles take too long. We will further discuss the history and relevance of this problem in relation to algorithms in computer science.

General Flow of Work:

1) Get a triangle mesh object. 2) Voxelize the object. 3) Apply a relevant puzzle generating algorithm. 4) Generate multiple voxel pieces. 5) Triangulate the voxel pieces. 6) Add a triangulated outer surface (optional). 7) 3D-print the puzzle pieces.

CR Concepts

- Computer-aided design - Usage of computer applications to guide the design of a project
- 3D printing - Triangularization of elements depicted in the third dimension for the purpose of printing
- Interlocking 3D puzzle algorithms - Optimization of application and computation using multi-core computer architectures

Keywords

Interlocking puzzles; 3D printing; Computational geometry.

1. INTRODUCTION AND MOTIVATION

This literature review focuses on algorithms to generate puzzles with the following two attributes: 1) Interlockable (assembly and disassembly are required to solve the puzzle). 2) 3-Dimensional (the third dimension should constrain all generated puzzle pieces). Finding a valid interlocking puzzle given an enclosing volume for a target shape is a computational geometry problem that has already been solved by Song et al [14]. However, their solution requires a blocky outer surface aligned on a regular grid that can take up to 10 hours to generate. Our research project will explore speeding up the computation by using multiple CPUs in a cluster and exploiting GPU co-processors. We aim to generalize the algorithm to allow a detailed non-voxelized outer surface that more closely resembles the target shape. The final design will be realized on a 3D printer. This will require a large amount of geometric post-processing.

2. ALGORITHMS OF PREVIOUS APPROACHES

Song et al. [14] define interlocking puzzles as an assembly of puzzle pieces (with at least three pieces) whereby there is only one movable puzzle piece; all other puzzle pieces are immobilized relative to one another. Their research publication can be regarded as the most relevant to date with respect to algorithms for generating 3D interlockable puzzles using voxels. A voxel can be described as a volume element – a 3D pixel/ cube with all six sides being of equal length and breadth.

In oversimplifying the problem at hand we can re-imagine the solution proposed by Song et al [14]. Each one of the six sides represents a direction in which a voxel can move in or out of the puzzle – up, down, left, right, forward or backward. We can call these directions the six degrees of freedom. To create an interlocking structure, the final piece in the puzzle should have only one degree of freedom; that is, it should be allowed to move in only one direction either into or out of the puzzle. The final piece should also block all n degrees of freedom of the pieces already in the puzzle. By applying this method recursively, we can create a puzzle using 3 or more puzzle pieces.

Below, we discuss previous approaches to generate interlocking puzzles. Although our focus is on 3D puzzles, we imagine that 2-Dimensional (2D) puzzles are also relevant as we can represent them in the third dimension by constraining them on the z-axis. The subsections are split by the approach each algorithm used to generate interlocking puzzles.

2.1. Exhaustive Search

The exhaustive search approach describes the naive method of generating puzzle pieces. The set of valid puzzle pieces are generated iteratively by going through the set of all possibilities. This approach was commonplace in the early days of puzzle generation. Due to its inefficiency, exhaustive searching has since been replaced by more efficient approaches in recent times. Little is known about the origins of interlocking puzzles. Many believe that they were invented by the Chinese, who used wood to build earthquake-resistant homes without nails. Edwin Wyatt, author of

Puzzles in Wood, coined the term 'burr' to describe puzzles which resemble a burr seed.

In 1997, IBM Research [4] launched the burr puzzles site. IBM Research defines burr puzzles as "consisting of at least three rods intersecting at right angles". Burr puzzle algorithms attempted to discover new interlocking puzzles based on exhaustive search. There are various types of three-piece and six-piece burr puzzles. The burr puzzles site extensively covers Bill Cutler's contributions to burr puzzles. Cutler [2] also covers previous attempts to discover new interlocking puzzles based on exhaustive search. Bill Cutler introduced a large number of six-piece interlocking structures. He shows that there 314 ways to assemble 25 notchable pieces to make a six-piece burr puzzle by assessing all permutations through exhaustively searching for puzzle pieces which meet the criteria for interlocking. Bill Cutler [2] outlines that there are two approaches for constructing all assemblies via a computer program: 1) The program determines every set of six pieces and possible assemblies, then checks for a solution. 2) The program constructs each assembly of six pieces, analyzing it immediately. Approach 2 was selected as it saves time.

Rover [11] created the Burr Tools computer software to help solve certain puzzle designs by trial-and-error (an exhaustive search approach). The program supports "square or dice shaped units, spheres, prisms with an equilateral triangle as base or 2 grids that use tetrahedra". The program allows users to construct puzzle pieces and target shapes visually. Users can then use the puzzle solver to check for the number of assemblies, solutions and the time taken to complete a generated puzzle.

2.2. Construction Approach

The construction approach refers to a method of generating interlocking puzzle pieces recursively from a source object. This can be done by dividing the source object into its constituent puzzle pieces (top-down approach) or by creating new puzzle pieces from the source object (bottom-up approach). Song et al. [14] explore a recursive constructive approach for devising new interlocking geometries that directly guarantee the validity of the interlocking instead of exhaustive testing it. The algorithm takes a voxelized shape (S) as input and iteratively extracts pieces P1, P2, …, Pn. Rn (the remaining part of S) is the last piece ie. $S \rightarrow [P1, \ldots, Pn, Rn]$. The algorithm requires local interlocking among Pi, Pi+1, and the remaining part which is denoted as Ri+1. The idea of the algorithm is to pick a seed voxel, ensure its blocking/ mobility, and expand the key parts recursively. Wang et al. [16] present a general framework for designing interlocking structures which use Direction Blocking Graphs and analysis tools. They use an algorithm outlined in section 4 to design an interlocking assembly from a given shape by generating parts P1,…, Pn, Rn. Much like Song et al. [14] first, they generate the key (only movable part) and then the parts Pi and Ri (where i>1). The main difference between the two approaches is the design of the graphs used to ensure interlocking – all previous parts are used as opposed to using only the most recently added part.

Song et al. [15] delve further into the matter of generating interlocking puzzle pieces while also focusing on their ability to be 3D printed. They take a watertight surface mesh as input and place a 3D grid within the object's space to voxelize it. They deform the geometry locally to cater to fragments. Neighbouring voxel shape connection strength is calculated by analysis local shapes and a graph is built. The important features of the model are realized on saliency graph. The construction of puzzle pieces is guided by the graphs, reassigning boundary voxels (for aesthetics) and constructive solid geometry (CSG) intersection. Interlocking does not necessarily have to be produced with voxelized pieces. This is demonstrated by Lau et al. [9] who converted furniture models into parts and connectors using lexical and structural analysis. Their algorithm uses lexical analysis to identify primitive shapes (processed as tokens). Then, structural analysis to generate fabricatable parts and connectors automatically. The output of their algorithm can also generate user manuals on how to assemble furniture from constituent components via the bottom-up approach.

2.3. Curve Matching Approach

The mapping approach refers to a method of matching puzzle pieces by assessing the curves on their edges. This information is then processed as input to solve puzzles according to the given algorithm. Curve matching normally employs a bottom-up approach, assembling target shapes from puzzle pieces.

Kong and Kimia [8] developed algorithms to solve 2D/3D puzzles using curve matching. First, they "define an affinity measure for a pair of pieces in two stages, one based on a coarse-scale representation of curves and one based on a fine-scale elastic curve matching method". The algorithm eliminates pieces with overlapping boundaries and forms a rank-ordered list of pairs. The puzzle solution is a "recursive grouping of triples using a best-first search strategy". In the case of 3D curve matching, a laser is used to scan 3D fragments and the 2D method of curve matching is applied over the z-axis. Sagiroglu and Ercil [12] have also developed an algorithm for 2D/3D fragment assembly. Their algorithm uses visual information such as "texture, colour, and continuity of lines" in addition to geometrical information. They generate the best assembly of a puzzle by optimizing the affinity (or suitability) value. They use an algorithm with parameters such as texture and colour to predict the region extending a puzzle piece and use that prediction to find a true neighbouring puzzle piece. The solution maximizes the correlation between predictions and actual puzzle pieces. Their algorithm is outlined at the end of section 4 as follows: "1) Place the pieces on board B. 2) Find the best transform for a randomly selected piece t using the presented method. 3) If there exists a piece that can be moved, go to step 2. 4) Select a piece and transform it. 5) Go to step 2, until the puzzle is uniquely assembled even if all pieces are tried in step 4."

In 1988, Wolfson et al. [17] solved the assembly of large 2D jigsaw puzzles using curve matching and combinatorial optimization techniques. Each puzzle piece is photographed and immediately fed into the algorithm which assembles an apictorial (without a picture) puzzle as the puzzle piece come. The algorithm uses only boundary data to match neighbouring puzzle pieces. After preprocessing, data is parsed into equations which conduct local matching. The puzzle assembly algorithm first assembles the frame and then uses it a starting point for the assembly of the entire puzzle. In 2002, Goldberg et al. [3] also worked on algorithms to assemble 2D apictorial jigsaw puzzles by boundary data alone. Their algorithm could solve more complex

puzzles with a maximum of 200 pieces by applying new techniques such as "robust fiducial points, highest-confidence-first search, and frequent global re-optimization of partial solutions". They found that the fiducial point comparison approach of global matching (rejected by Wolfson et al. [17]) was significantly faster and more robust to scanning noise. Their algorithm works by identifying the inflexion points, ellipse centres, and tangent points on a tab (indent or outdent) and minimizing the distance between pairs of corresponding points.

2.4. Geometric Design Approach

Stewart Coffin is widely regarded as the world's best designer of interlocking puzzles. In 1990, he produced a book [1], which described interlocking cubes. His book motivated the study of geometric mechanics producing interlocking. The geometric design approach refers to using lines and curves generated by a mathematical equation to generate puzzle pieces. Lo et al. [10] used the geometric design approach to generate 3D polyomino puzzles. They created shell-based 3D puzzles with polyominoes as the component shape of the puzzle pieces. The algorithm works as follows; they first apply quad-based surface parametrization to the input solid and tile the parametrized surface with polyominoes. Then, construct a tiled surface inside the parameterized shape to fit inside a thick shell volume. Finally, use associated geometric techniques to construct puzzle pieces (polyominoes generating via Procedure 1 outlined in section 2), including the ring-based ordering scheme, the motion-space analysis technique, and the tab and blank construction method. The final puzzle must be buildable, and maintainable.

Xin et al. [5] also explored the governing mechanics of interlocking puzzles using geometric methods. They replicated and connected pre-defined six-piece burr structures to create larger interlocking puzzles from 3D models. First, they embed a network of knots into a given 3D model. Then, the 3D model is split according to its geometry. The method requires one to first manually design and construct a grid-based graph inside a given 3D shape. The method can put a six-piece burr puzzle at each grid point and connect them to guarantee the interlocking. Their paper describes both a single-knot and multi-know burr puzzle. Zhang and Balkcom [18] explore a solution to assemble voxelized interlocking structures using joints (pairs of male-female connectors on adjacent voxels) to guarantee interlocking and assembly order. The algorithm breaks models into layers and sequentially builds layers using block types to restrict the mobility of puzzle pieces.

3. DISCUSSION OF PREVIOUS APPROACHES

This section discusses the important characteristics of the 3D interlocking puzzle algorithms according to our survey of known interlockable structures. We will compare and contrast the algorithms in terms of applicability and efficiency; that is, how relevant the algorithm is to 3D printing and how many computational operations (or how much time in seconds) it may take to generate the puzzle pieces after the original object (which may be represented as a triangular mesh) has been served into the algorithm as input.

3.1. Exhaustive Search

The analyses of Bill Cutler's algorithms [4] are categorized and presented as follows:

Name	Size	Author	Date	Runtime
Notchable, Solid "JRM"	314 solutions	Bill Cutler & Tom O'Beirne (manual & computer)	Fall 1974	Not available
General, Solid "CSIAM"	119 979 solutions	Bill Cutler (computer) & Arthur Cross (random manual checking)	Winter 1975	5 minutes on IBM mainframe
Notchable, Holey "NOTC"	13 354 991 assemblies	Bill Cutler (computer)	26 March 1987 – 6 June 1987	2 months on PC AT
General, Holey "HB6"	35 657 131 235 assemblies	Bill Cutler & others (computer)	October 1987 to 4 August 1990	The equivalent of 62.5 years on PC AT (approximately 2.5 years more recently)

On average, Rover [11] computes assemblies and dis-assemblies in a mere matter of seconds. It is important to note that the runtime of exhaustive search algorithms is highly dependent on the size of the problem at hand. Exhaustive search algorithms for generating puzzles normally run at least n operations (to generate or assemble each puzzle piece), where n is the number of elements the algorithm has to process. Since the Burr Tools software is not open-source, it was not possible to assess its algorithm in detail.

3.2. Construction Approach

The Song et al. [14] interlocking structure is decentralized and thus can adapt more flexibly to complex 3D shapes and topologies. It generates recursive interlocking shapes with different Ns (number of voxels) and Ks (number of puzzle pieces). Their method works with a large variation of Ns and m (average puzzle piece size). Time complexity depends on a combination of N, K, m and the target shape. Wang et al. [16] tests for global interlocking in polynomial time complexity. Their approach generates puzzle pieces much faster than the previous

Song et al. and is more generally applicable. The primary disadvantage is the ease of implementation. The latest Song et al. [15] uses more or less the same approach with additional aesthetic and 3D printing requirements. Their largest structure (11 x 16 x 11 volume resolution consisting of 20 pieces and 44 973 model vertices) took a decent 150 seconds to generate via C++ on a 3.4GHz CPU with 8GB memory. Lau et al. [9] tested their constructive approach of generating 3D furniture models with a less than 100% success rate. They claim that adding new production rules to cater for the unsuccessful attempts is effective in producing a 100% success rate. The average runtime was less than 1 second. Voxelization occupied the majority of the runtime.

3.3. Curve Matching Approach

Kong and Kimia [8] apply their algorithm to the problem of reassembling pieces of a broken ceramic tile. The algorithm contains a for loop in a triply nested if-statement which can be expected to compute at least Big-O of n operations. The Sagiroglu and Ercil [12] method's time complexity is similar to that of Kong and Kimia. They assert that the main drawback is that multiple solutions to the problem are possible depending on how the puzzle pieces are placed on the board. The additional parameters and complex equations make it likely to be a relatively slow curve matching approach. The time complexity of the algorithm is highly dependent on the number of puzzle pieces.

Wolfson et al. [17] state that the time complexity of their algorithm is "O(max(K^3, K^2.m log m) + O(N^2 x KBEST x m log m), where N is the total number of puzzle pieces, K is the number of frame pieces". O(m) is the number of sample points on a boundary curve of a piece. On a Sun Ultra-60 workstation, the Goldberg et al. [3] algorithm took approximately 3 minutes and 20 minutes to solve 100-piece and 204-piece puzzles respectively. This can be expected to run in a fraction of the time on a modern computer.

3.4. Geometric Design Approach

Lo et al. [10] used a procedure called Procedure 1 to assess the performance of their polyomino tiling algorithm. They used a table to show the performance on three different 3D models using a Pentium-4 PC with 3GHz CPU and 512MB memory. The table showed the time taken to build six different 3D puzzles. Their largest polyomino, consisting of 325 tiles, took only 0.29 seconds to generate. The Xin et al. [5] method was used to assemble various burr puzzles with a varying number of knots and puzzle pieces. The MegaBox puzzle model took the longest (1405 seconds) to assemble given 64 knots (4 x 4 x 4) and 240 pieces. The time taken is affected by the nature of the problem at hand as well as the number of pieces in the puzzle. Zhang and Balkcom [18] implemented their algorithm in Python and ran it on a computer with 3.4GHz CPU and 8GB memory. Their most complex model (a house consisting of 13104 blocks and 40 layers) was designed in only 633 milliseconds. Runtime was linear relative to the number of voxels.

4. CONCLUSIONS

Since the advent of 3D printing in the 1980s, little research has been done with respect to algorithms for generating interlocking 3D puzzles. We have explored a few known pieces of research which delve into the topic of algorithms which generate interlocking 2D and 3D. Below we assess these algorithms according to various criteria (in order of appearance):

Algorithm Name and Author(s)	Speed of puzzle generation	Variability of shape generation	Ease of implementation
IBM Research (1997)	Very slow (~ 2.5 years maximum)	Burr puzzles	Fairly simple
Cutler (2007)	Very slow (~ 2.5 years maximum)	Six-piece burr puzzles	Fairly simple
Rover (2011)	Fast	Cubes, spheres, equilateral triangles, tetrahedra	Not applicable (the algorithm is unavailable)
Song et al. (2012)	Slow (534.43 minutes to generate a bunny when K=150)	Solid interlocking voxelized structures	Average difficulty
Wang et al. (2017)	Average	Wide range of interlocking assemblies	Fairly difficult
Song et al. (2015)	Average	Solid interlocking voxelized structures	Fairly difficult
Lau et al. (2011)	Extremely fast (~ > 1 second)	Parts and connectors	Very difficult
Kong and Kimia (2001)	Average	Applies only to the re-assembly of broken puzzles	Fairly difficult
Sagiroglu and Ercil (2006)	Average	Applies only to the re-assembly of broken puzzles	Difficult
Wolfson et al. (1988)	Slower than average (Big-O of n^3 maximum)	2D apictorial jigsaw puzzles	Difficult
Goldberg et al. (2002)	Average	2D apictorial jigsaw puzzles	Difficult
Lo et al. (2009)	Fairly fast	Quad-based 3D model surfaces	Difficult
Xin et al. (2011)	Average	Six-piece orthogonal burr puzzles with a custom 3D model outer surface	Fairly simple

Zhang and Balkcom (2016)	Very fast (in a matter of milliseconds)	General voxelized interlocking structures	Difficult

In assessing the above-mentioned algorithms, it is important to take note of the date on which they were published. The vast majority of them were published prior to 2016, during a time when Moore's Law still applied. For example, IBM's PC-AT Model 2 (manufactured in the 1980s) had only 512kB of RAM, one floppy disk unit and one hard disk. Modern computers use multicore architectures and parallelism to achieve speedup. It is only fair to compare the performance of these algorithms on the same modern computer to get an accurate representation of how they perform relative to each other.

We will proceed to employ the algorithm proposed by Song et al. [14] during the implementation phase of the project. This is due to its ease of implementation and applicability in solving the problem of creating 3D printable recursively interlocking puzzles. Song et al. [15] also present an algorithm worthy of consideration. Despite being more complex, their algorithm caters for aesthetically-pleasing design and 3D printing. Their algorithm is arguably the most suitable to date although Zhang and Balkcom's [18] algorithm outperforms most of the others. Kilian et al. [7] proposed a method to construct surfaces with carved folding by minimizing the bending energy. Skouras et al. [13] presented an algorithm for designing interactive surfaces from "flexible interlocking quadrilateral elements of a single size and shape". Should time permit, we will implement such algorithms to add an outer surface to our voxelized output shape prior to triangularization for 3D printing.

5. REFERENCES.

[1] COFFIN, S. T. 1990. The Puzzling World of Polyhedral Dissections. Oxford University Press.

[2] CUTLER, B. 2007. A Computer Analysis of All 6-Piece Burrs. Available online: http://billcutlerpuzzles.com/docs/CA6PB/index.html. [Accessed 23 April 2019]

[3] GOLDBERG, D., MALON, C., AND BERN, M. 2002. A global approach to automatic solution of jigsaw puzzles. In Proceedings of the Eighteenth Annual ACM Symposium on Computational Geometry, 82–87.

[4] IBM RESEARCH, 1997. The burr puzzles site. Available online: http://www.research.ibm.com/BurrPuzzles/. [Accessed 23 April 2019]

[5] 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.

[6] SLOCUM, J. 2001. Mechanical puzzles their history and their challenge. In Katonah Museum of Art. The art of the puzzle: astounding and confounding

[7] KILIAN, M., FLÖERY, S., CHEN, Z., MITRA, N. J., SHEFFER, A., AND POTTMANN, H. 2008. Curved folding. ACM Tran. on Graphics (SIGGRAPH) 27, 3.

[8] KONG, W., AND KIMIA, B. B. 2001. On solving 2D and 3D puzzles using curve matching. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, 583–590.

[9] LAU, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2011. Converting 3d furniture models to fabricatable parts and connectors. ACM Tran. on Graphics (SIGGRAPH) 30, 4. Article 85.

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

[11] RÖVER, A. 2011. Burr tools. Available online: http://burrtools.sourceforge.net/. [Accessed 23 April 2019]

[12] SAGIROGLU, M., AND ERCIL, A. 2006. A texture based matching approach for automated assembly of puzzles. In 18th International Conference on Pattern Recognition, vol. 3, 1036–1041.

[13] SKOURAS, M., COROS, S., GRINSPUN, E., AND THOMASZEWSKI, B. 2015. Interactive surface design with interlocking elements. ACM Transactions on Graphics, vol. 34, issue 6, Article 224.

[14] SONG. P., FU. C., AND COHEN-OR. D. 2012. Recursive Interlocking Puzzles. ACM Transactions on Graphics, Vol. 31, No. 6, Article 128.

[15] SONG, P., FU, Z., LIU, L., AND FU, C. 2015. Printing 3D objects with interlocking parts. Computer Aided Geometric Design, Vol. 35 Issue C, 137-148.

[16] WANG, Z., SONG, P., AND PAULY, M. 2018. DESIA: a general framework for designing interlocking assemblies. ACM Transactions on Graphics, Vol. 37 Issue 6, No. 191.

[17] WOLFSON, H., SCHONBERG, E., KALVIN, A., AND LAMDAN, Y. 1988. Solving jigsaw puzzles by computer. Annals of Operations Research.

[18] ZHANG, Y., AND BALKCOM, D. 2016. Interlocking structure assembly with voxels. 2173-2180. 10.1109/IROS.2016.7759341.