Z-Chunking for Cooperative 3D Printing of Large and Tall Objects

Daniel H. Weber*, Wenchao Zhou†, and Zhenghui Sha*

*Walker Department of Mechanical Engineering, The University of Texas at Austin
†Department of Mechanical Engineering, University of Arkansas
Corresponding Author: zsha@austin.utexas.edu

Abstract

Cooperative 3D Printing (C3DP) is an emerging technology designed to address the size and printing speed limitations of conventional gantry-based 3D printers. To print large-scale objects, C3DP divides a job into chunks to be printed by a swarm of mobile robots. Previously, we developed a Chunker algorithm to partition jobs into printable parts in the XY direction, which theoretically enables the printing of objects of unlimited size in XY dimensions. However, print size is limited in the Z direction due to the physical constraints of the printer. In this paper, we introduce the first working strategy and rules of Z-Chunking for C3DP, such as where and how to place chunk boundaries along the Z direction and alignment geometries for easy post-assembly. Additional challenges of interfacing with XY chunking and facilitating re-assembly of the job are also considered. We conduct two case studies on objects of varying geometric complexity (e.g., simple solids vs. hollow structures) in which the object is chunked, printed, and assembled.

Keywords: Cooperative 3D Printing, Swarm Manufacturing, Geometric Partitioning

Introduction

The emergence of Additive Manufacturing (AM) technologies over the past decades has led to many advancements in the field of rapid prototyping and low-quantity manufacturing [1]. However, issues exist in current 3D printing technologies regarding print speed [2] and size [3]. While solutions to each of these exist or are in development, such as using multiple nozzles as described by Ali et al. [4] or the Big Area Additive Manufacturing (BAAM) system described by Roschli et al. [5], each of them has their own drawbacks such as cost, scalability, or print resolution. The fundamental question still remains: how do we enable 3D printers to print large parts quickly without compromising on print quality?

Cooperative 3D Printing (C3DP) is an emerging technology that utilizes a swarm of mobile robots that move around and work together on an open factory floor. One of the necessary methods to enable C3DP is to chunk large jobs into smaller pieces to be printed by a swarm of mobile 3D printing robots in parallel. An example of C3DP is shown in Figure 1. The current iteration consists of two types of robots: the SCARA printers and the mobile platforms. The SCARA printers can be placed in any of the four cardinal directions at discrete locations on the factory floor. They are fixed in place at these locations and draw power directly from the floor. They can be moved between locations automatically by the mobile platforms, which supply battery power to the

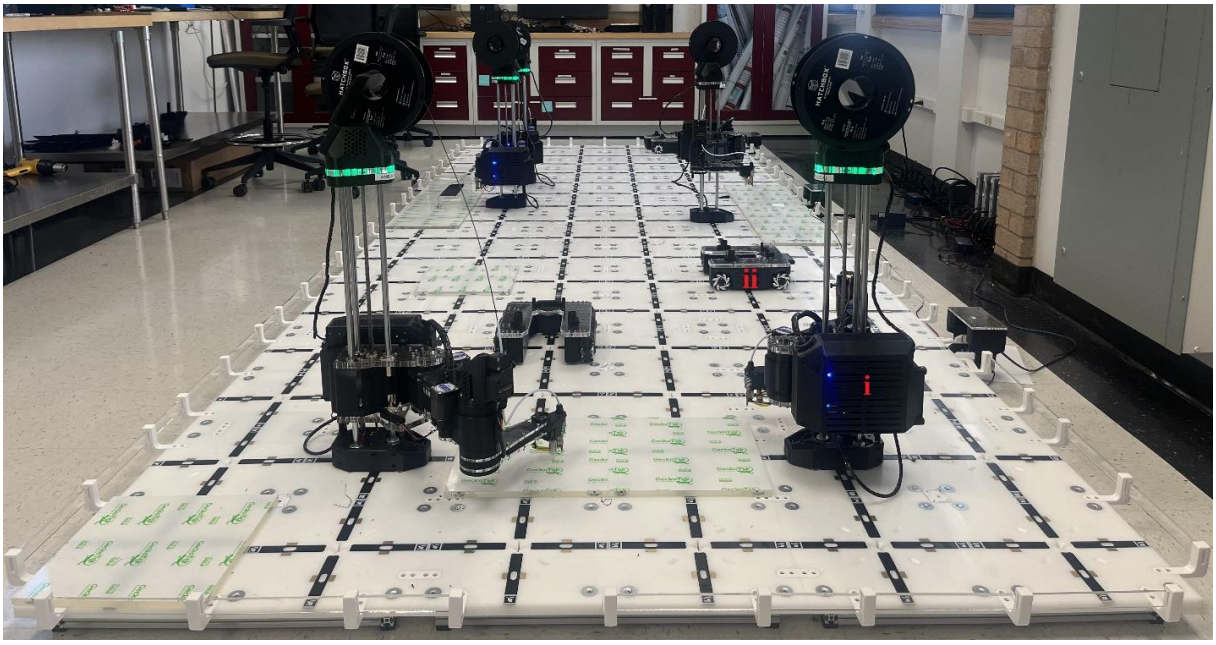


Figure 1: The current iteration of the C3DP system including: i) the SCARA printer and ii) the mobile platform.

printers during transportation. Other tool heads such as a pick-and-place head or a pellet extrusion printer are planned or in progress based on the SCARA printer design.

There are two fundamental directions of work in this setup: 1) the XY direction, which is the direction the robots can move in and has a theoretically infinite build size given a large enough factory floor, and 2) the Z direction, which is unique because it is the direction parts are printed in and has a limited size due to the physical characteristics of the 3D printing robots. The current iteration of the SCARA printer with a Z-limit of 265mm is shown in Figure 2.

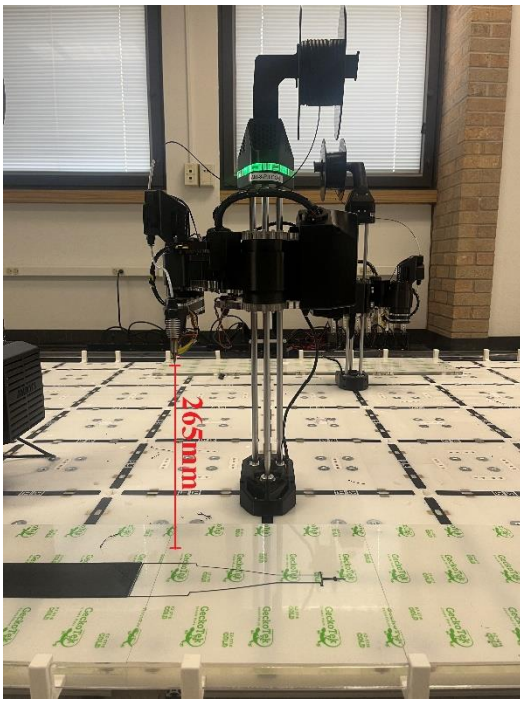
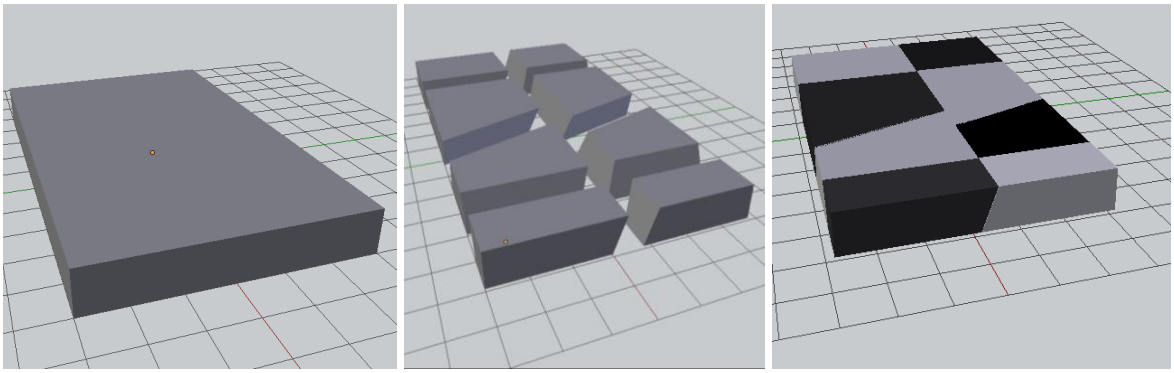


Figure 2: The current iteration of the SCARA printer with a Z-limit of 265mm

XY chunking, as shown in Figure 3, is a well-established process developed in our previous work [6]. Work by Poudel et al. found that the mechanical strength of parts printed by this strategy can be made as good as traditional, single print job parts [7]. Scheduling and path planning [8-10] have also previously been explored, leading to a fully functional process for the printing of XY large pieces.

Z-Chunking, on the other hand, has not been previously demonstrated in C3DP. Z-Chunking is unique because the SCARA robots cannot move up to print a new chunk due to challenges with moving and powering the robot, as they can sideways. We cannot directly print the chunks bonded as one large print as with XY Chunking or more traditional 3D printing; therefore, we must instead print the Z-



a) b) c)

Figure 3: A large rectangular box chunked into smaller, printable XY sections. a) shows the part while b) shows the resulting chunks in an exploded view. c) shows the assembled box after printing, color coded for chunk visibility. Notice the sloped surface chunk interface to prevent collisions between printers and the adjacent chunks.

Chunks separately and assemble them. Because of this, Z-Chunking is not only a partitioning and printing problem but also an assembly problem. The main purpose of this work is to introduce a solution that solves all three of these problems, with a focus on the partitioning and assembly because those are unique to Z-Chunking.

This paper is organized into the following sections. First, we present related work as well as concepts that we used while implementing Z-chunking. We then discuss the rules for Z-chunking that must be followed to enable tall parts to be printed in C3DP. Next, we present our current implementation of these rules consisting of 1) Determining appropriate chunking locations and 2) Adding Assembly Geometry (AG) to facilitate re-assembly. This implementation aims to generate as few chunks as possible and determine the size and location of AGs to facilitate printing and re-assembly. The primary constraint we encountered was geometric (i.e., the chunks must all fit within the Z-limit of the printer). This approach allows us to generate a feasible Z-Chunking strategy to enable C3DP of large jobs in all three dimensions. We follow this with two case studies to demonstrate the feasibility of our algorithm. Finally, we discuss the variables that can be changed within the framework of the Z-chunking rules to optimize the results for different use cases before presenting the directions of future work.

Related Work

A naturally related topic to this work is the topic of geometric partitioning, particularly as it relates to 3D printing. The work presented by Luo et al. [11] focuses on dividing a print job into smaller sections by iterative bisection until all parts are printable, given the volume constraints of the printer. Hao et al. [12] introduced an approach to partitioning a large print job based on surface features to reduce the complexity of the chunks. However, these have limited application in C3DP because partitions from these methods are often in irregular shapes that do not take full advantage of the unlimited XY space of the system. In the current C3DP framework, parts are only limited in size in the vertical direction, so chunking parts in this direction is most useful.

Because we determined this to be relevant to an assembly problem, we were interested in methods of maximizing assembly efficiency. Much research has been done in the field of Design

for Assembly (DFA) to create generalized concepts to facilitate the cheap and efficient assembly of manufactured components. Otto and Wood [13] summarize these DFA concepts in Table 1.

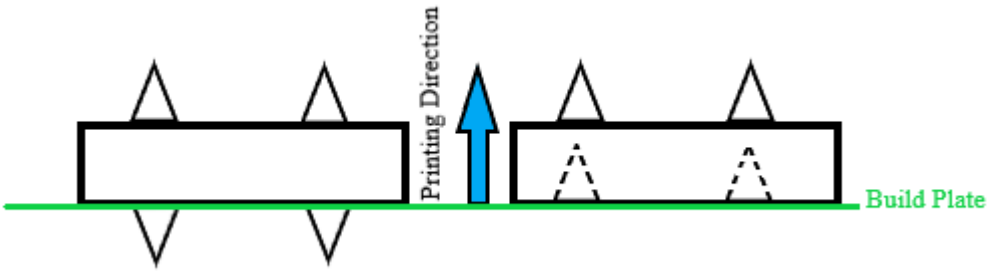
Table 1: Design for Assembly Concepts

	Concepts
1	Minimize part count by incorporating multiple functions into single parts.
2	Modularize multiple parts in single subassemblies.
3	Assembly in open space, not in confined spaces. Never bury important components.
4	Make parts to identify how to orient them for insertion.
5	Standardize to reduce part variety.
6	Maximize part symmetry.
7	Design in geometric or weight polar properties if nonsymmetric.
8	Eliminate tangly parts.
9	Color code parts that are different but shaped similarly.
10	Prevent nesting of parts.
11	Provide orientating features on nonsymmetries.
12	Design the mating features for easy insertion.
13	Provide alignment features.
14	Insert new parts into an assembly from above.
15	Insert from the same direction or very few. Never require the assembly to be turned over.
16	Eliminate fasteners
17	Place fasteners away from obstructions
18	Deep channels should be sufficiently wide to provide access to fastening tools.
19	Providing flats for uniform fastening and fastening case
20	Proper spacing ensures allowances for a fastening tool

One of the most related works [14] applies DFA concepts described to partition 3D objects into smaller objects using a lattice of orthogonal chunking planes while considering common issues in 3D printing such as distortion, feature size, and need for support material. The main limitation of this paper to C3DP is that it discusses partitioning in 3D, and we are only interested in Z-direction partitioning. However, the system of creating a large set of regular, smaller divisions, then additively combining them to create feasible prints, was adapted for use in our work.

Other works have also sought to adapt Design for Manufacturing principles for use with additive manufacturing technologies, also known as Design for Additive Manufacturing (DfAM). [15] provides an explanation of the necessity of DfAM as well as the design opportunities of AM, such as high internal geometry complexity or adaptability to many materials. Work by Steuben et al. [16] provides insight into design considerations for the Fused Filament Fabrication (FFF), which is the current process in use by our C3DP system. Specific topics include constraints related to part orientation and geometry and material considerations.

The last related area of research is that of adding assembly geometry to the part allowing the chunks to be assembled. We based our approach on the work conducted in [17] with some modifications. For example, as shown in Figure 4, in their work, male and female assembly geometry could be placed on both sides of the chunk interface. In our case, we constrain that a



a) b)

Figure 4: A demonstration of the side dependent placement of AG. a) shows AG with the male piece on both sides of the chunk as can be found in [17]. This would mean the base of the chunk cannot be printed directly on the build plate. To address this either the AG cannot be printed or the whole chunk must be raised and extensive support structure used. b) shows our method where all AG face upwards and the female AG are placed recessed in the base of the chunk.

male chunk must be added to the top of the lower chunk while a female chunk must be added to the bottom of the upper chunk to allow chunks to be printed in a fixed orientation without the need for a support structure.

Rules for Z-Chunking

This section of the paper introduces and explains the rules for Z-chunking of large and tall objects for C3DP. These rules are designed to be general enough to function with future versions of this system while allowing large jobs to be printed on the current system. The goal is to narrow the design space to facilitate user-determined optimizations such as part strength or job appearance.

Table 2: Rules for Z-Chunking

	Rules
1	Chunks must be printable by C3DP
2	Generate a minimum number of disjoint chunks
3	Generate assembly geometry to join chunks
4	Chunking is deterministic

Chunks must be printable by C3DP

The first rule is the most fundamental and main motivation for this project: All jobs exported from the algorithm must be printable. This means that each chunk must physically fit within the build volume of the printer, as shown in Figure 3, and that all features of the part should follow general 3D printing guidelines as described by [18]. We also want to ensure that the Z-Chunking algorithm does not interfere with the other aspects of C3DP, such as XY Chunking or assembly. In this work, we only focus on the Z-limit of the build volume, as shown in Figure 5, because it is considered to be infinite in the XY direction.

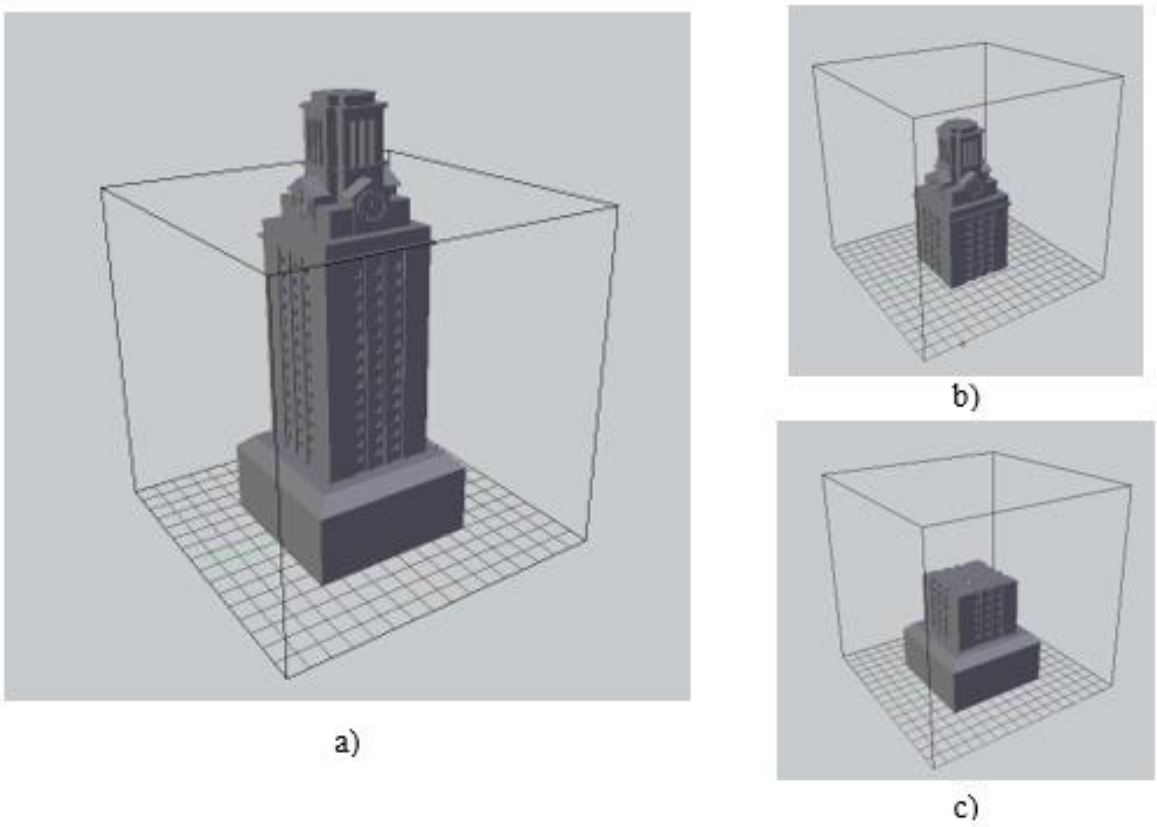


Figure 5: a) A model of The Tower at UT Austin which is taller than the bounding print volume. b) and c) The top and bottom of the same model after Z-Chunking. Both chunks are now printable.

This rule ensures that a wide variety of jobs can be printed by C3DP, comprised of Z-Chunking, XY chunking, and part assembly, among other features. The open build and assembly space of C3DP will naturally lead to the fulfillment of the DFA concept 3. The fundamental layer-by-layer, vertical approach to 3D printing, as well as the design of the pick-and-place robots in the existing C3DP strategy, are related to DFA concepts 14 and 15. This is also loosely related to concepts 6, 8, and 10. This is because the Z-Chunking strategy should deliver a part that interacts with the XY chunking algorithm. The XY chunking algorithm is most effective when used on large parts in the XY dimension with regular top and bottom surfaces.

Generate a Minimum Number of Disjoint Chunks

This rule is related to DFA concepts 1 and 2. There are some differences between the DFA concept and this rule because our process starts by breaking down the part to be re-assembly. However, the basic premise remains that fewer parts lead to a lower assembly complexity and cost. In this work, minimizing the number of disjoint chunks means having as few vertical layers and disjoint XY sections, which would require active assembly as opposed to being bonded together, as is the case with connected XY sections as possible. An example of this can be seen in Figure 6, where it is clear that if chunked at Chunking Plane 1, the result is one large interface, while if chunked at Chunking Plane 2, the result is two smaller interfaces, the former being preferred. It should be noted at this point that our primary goal is to have as few chunks as possible to minimize

the complexity of re-assembly. We are not concerned at this stage with creating chunks of equal print time because printer utilization should be optimized on a global scale for C3DP.

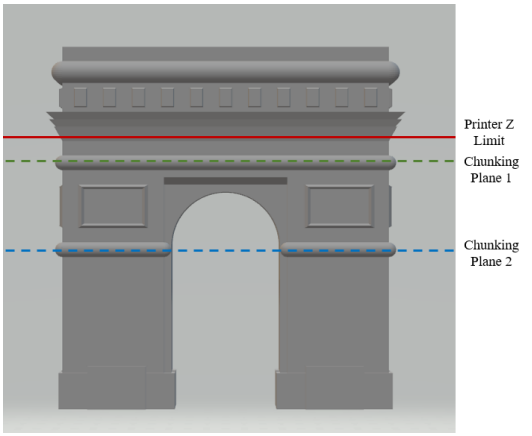


Figure 6: A model of the Arc de Triomphe shown to be too tall for the SCARA printers to print. Two different Chunking Planes are shown as examples of where the job could be broken into two printable parts.

Generate Assembly Geometry

This rule is related to the DFA concepts 12, 13, and 16. The purpose of generating Assembly Geometry (AG) is to enable a simple assembly. For our case, this means using as few unique robots as possible with as few consumables, such as fasteners, as possible. This is because more different types of robots and consumables increase scheduling and path planning complexity and reduce usable space on the factory floor. This is also beneficial to the implementation of the previous rule because fasteners would add to the overall part count of the assembly. An AG is any added geometry that helps to constrain the assembled parts in at least one direction. Examples of types of AG include hole-peg, snap-fit, and dovetail. An example of a hole-peg style AG used in our algorithm is shown in Figure 7.

Notable constraints for the AGs are that they should be present at every interface, they should be fully enclosed by the volume of the surrounding parts (i.e., not change the surface geometry of the job), and they should follow the 3D printing guidelines as described above. This latter point is especially relevant to the fit and tolerances of these parts.

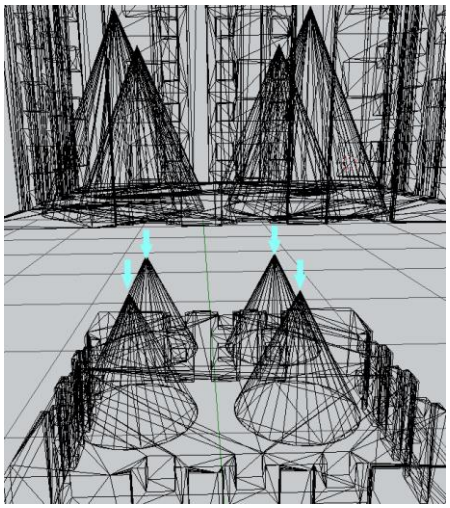


Figure 7: Skeleton view of a hole and peg style Assembly Geometry. Notice that the assembly is constrained in X and Y direction by this type of AG but not in the Z direction.

Chunking is deterministic

This means that the Z-Chunking algorithm should produce consistent, repeatable results given the same inputs. This is related to DFA concept 5 for standardized parts or the common mass manufacturing concept of ‘interchangeable parts.’ Examples of this rule in practice include: the same AG size and locations and the same layer split locations across different iterations.

Unused DFA Concepts

It is obvious that many of the rules we developed are based on the DFA concepts from Table 1; however, some were not applicable to the z-chunking of C3DP. Because we are not designing the base part, only chunking a given part, concept 7 is irrelevant. C3DP utilizes a central control for all of the robots and parts being printed on the factory floor; therefore, concepts 4, 9, and 11 are unnecessary as the computer will track orientation and control assembly. Also related to this, we do not standardize AG configuration across layers as might be suggested by concepts 5

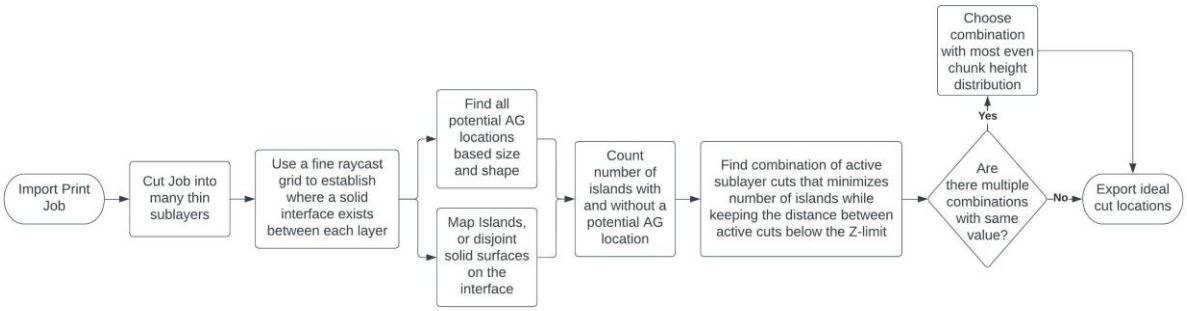


Figure 8: Flow chart for finding the ideal chunking locations

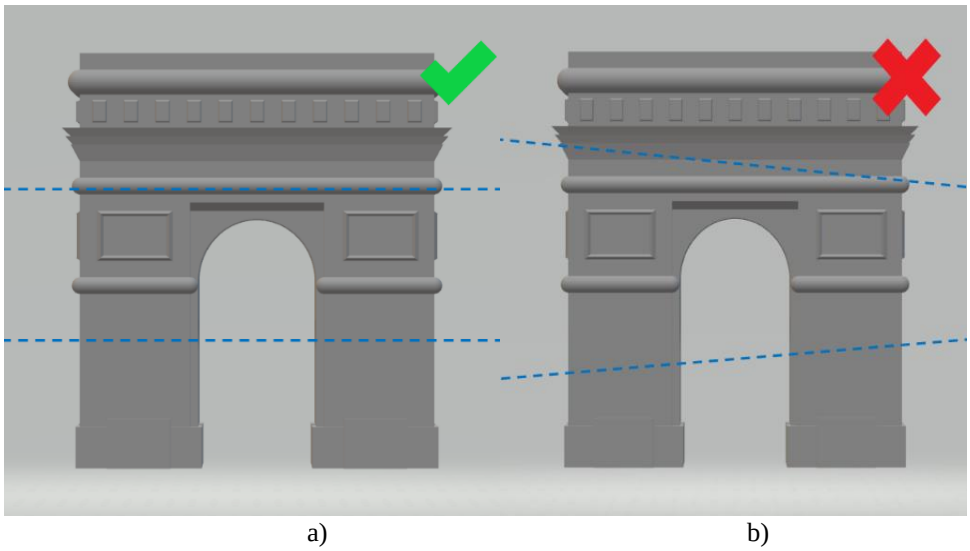
and 11. That is not to say that we specifically enforce consistent AG configuration at different layer interfaces, just that we do not constrain ourselves to unique AGs between layers, whether that be size or type. Lastly, because of the previously discussed aversion to using fasteners, concepts 17-20 are unnecessary.

Implementation

This section discusses the algorithm that carries out the Z-Chunking and how it ensures that all of the rules discussed above are followed. As described in the introduction, there are two main steps to this: 1) Determine the best chunking location, and 2) Add Assembly Geometry (AG) to facilitate re-assembly.

Determine the best chunking location

The algorithm for determining the best chunking locations follows the flow chart shown in Figure 8. We constrain all splits, or cuts, to be flat and horizontal. This is because, as discussed above, the mechanical strength of parts chunked in the XY direction with our 'sloped surface'



a) b)

Figure 9: Arc de Triomphe model shown a) chunked correctly with parallel horizontal cuts and b) incorrectly with nonparallel non-horizontal cuts

strategy has been explored and shown to be able to achieve similar strength of singular parts. However, we have no explored cases where there are slanted vertical cuts to go along with the sloped surface XY cuts. Additionally, allowing non-horizontal cuts would lead to upper layers either requiring print support or rotation, which is discouraged by the DFA concept 15. For these reasons, and to enable an easy, consistent assembly of chunks (DFA concept 5), we constrain our Z-chunking algorithm to only output flat layers with horizontal cuts, as shown in Figure 9.

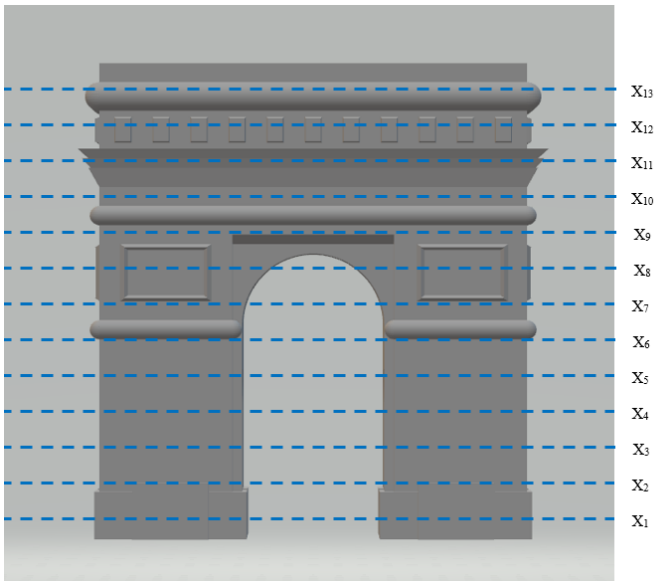


Figure 10: An example of the Arc De Triomphe model split into 14 sublayers. 13 cuts are shown marked X_1-X_{13}

The algorithm starts by dividing the job evenly into many small sublayers, as shown in Figure 10. It then uses a fine grid of raycasts to find the interface between each of these sublayers and its neighbors. Raycast means that a ray is projected directly down onto the top of the interface. If it intersects the object at the layer height, we record that XY coordinate as a success. The raycasting method is adopted instead of directly evaluating the geometry due to its inexpensive computational cost, as displayed in the runtime section, its scalability, and interactability with different AG types. In the current algorithm, we use a raycast grid with a spacing of 1mm that begins and ends at the extremities of the bounding box of the whole job. This level of detail was sufficient in our testing but can be easily increased. At each interface, the algorithm determines how many disjointed pieces, or islands, there are. An example of this process can be seen in Figure 11.

We also check to see if each island is suitable for at least one AG of minimum size; we discuss this process in detail in the next section. We then minimize an objective function that counts the number of layers based on each of the cuts to create the sublayers being either active or inactive. The objective function has the following form:

$$\min\left(\sum_{i=1}^{n} x_i * (y_i + k * z_i)\right), \text{ Subject to: } \sum_{i=1}^{n} x_i \geq J/H \text{ and } \sum_{i=1}^{L} x_i \geq 1,$$

where n is the number of sublayer cuts, x_i is the status of the ith cut (0 for inactive, 1 for active), y_i is the number of islands on the ith cut, z_i is the number of islands that cannot have an AG on the ith cut, k is a weighting factor >> 1 to ensure that AGs are generated on each island if possible, J is the total height of the job, H is the printer Z limit, and L is the number of cuts that fit within the Z limit or $H/(\frac{J}{n})$ rounded up to the nearest integer. The reason we have a weighting factor and the number of islands unsuitable for AG is so that the algorithm will attempt to follow the rule noted in the next section, that AG should be added to each island, but in case the user supplies a model where this is not possible, the algorithm can still supply a result.

At this stage, we use a relatively small number of sublayers (11) because we conduct an exhaustive search of all combinations of inactive and active sublayer cuts. For example, for 11

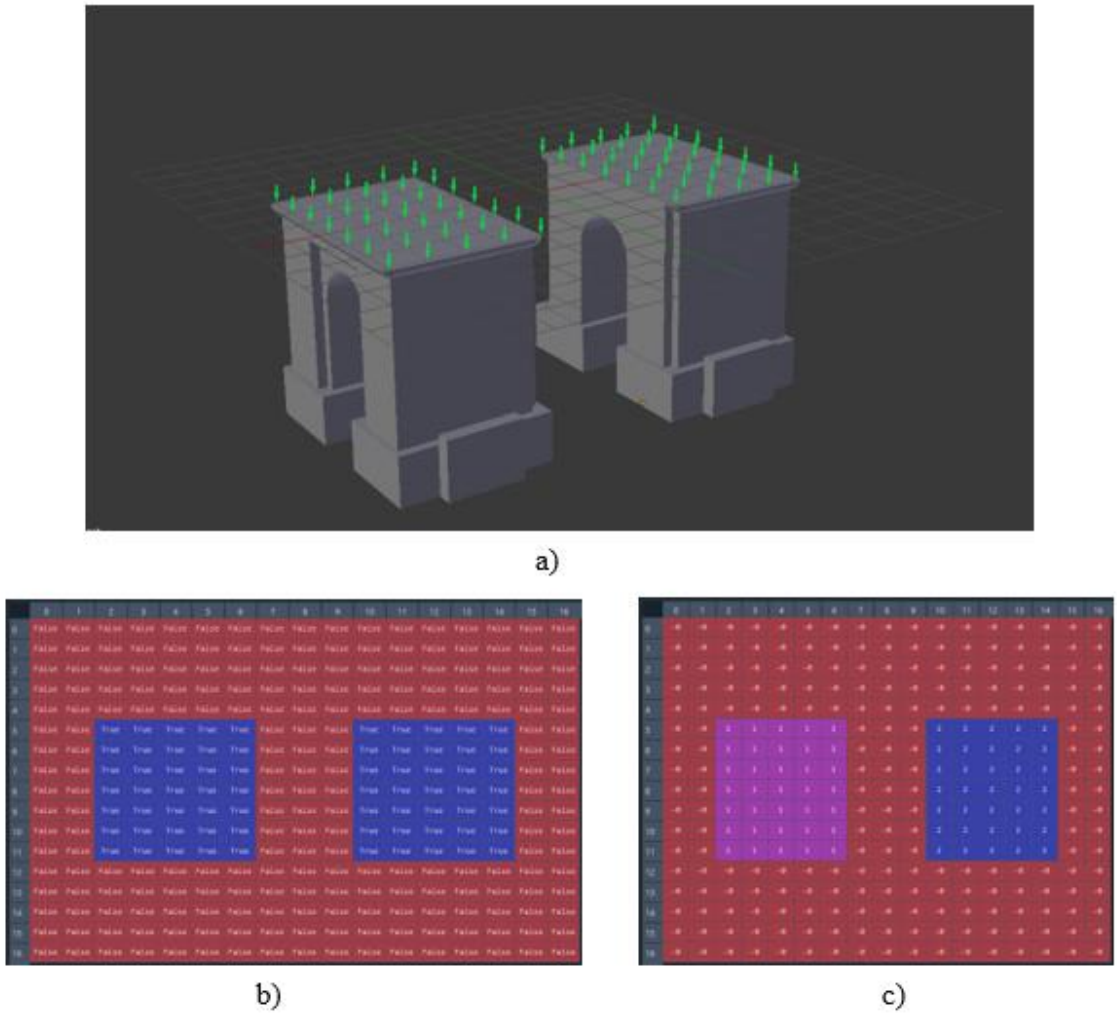


Figure 11: a) an example 17x17 raycast grid shown on the Arc de Triomphe model cut at Chunking Plane 2. Green arrows indicate success at that XY coordinate and show the direction of ray projection. b) shows the results of the raycast grid of the same example. c) shows the island map based on this raycast grid

sublayers, there are 10 cuts (or interfaces) that can either be active or inactive. If a cut is designated as active, that means the job will be bisected at that plane; otherwise, that cutting plane is ignored when the final chunking is performed. The first combination is when they are all inactive, the second is just the first cut active, the third is just the second cut active and the last combination is when all of the cuts are active. The number of combinations can be represented by 2^n, where n is the number of cuts (or the number of sublayers minus 1). The value, 11 sublayers, was chosen because we found it gave good enough results for the case study while not significantly slowing the runtime of the algorithm. Runtimes will be further covered in greater detail in the discussion section. In the future, the precision could be improved by increasing the number of sublayers and using an integer programming optimization strategy; however, the method as described above ensures the following:

1. A minimum number of disjointed chunks are generated.
2. All chunks fall within the height z-limit of the SCARA printer.
3. Z-Chunking enables XY chunking and assembly by C3DP.

Add Assembly Geometry to facilitate re-assembly

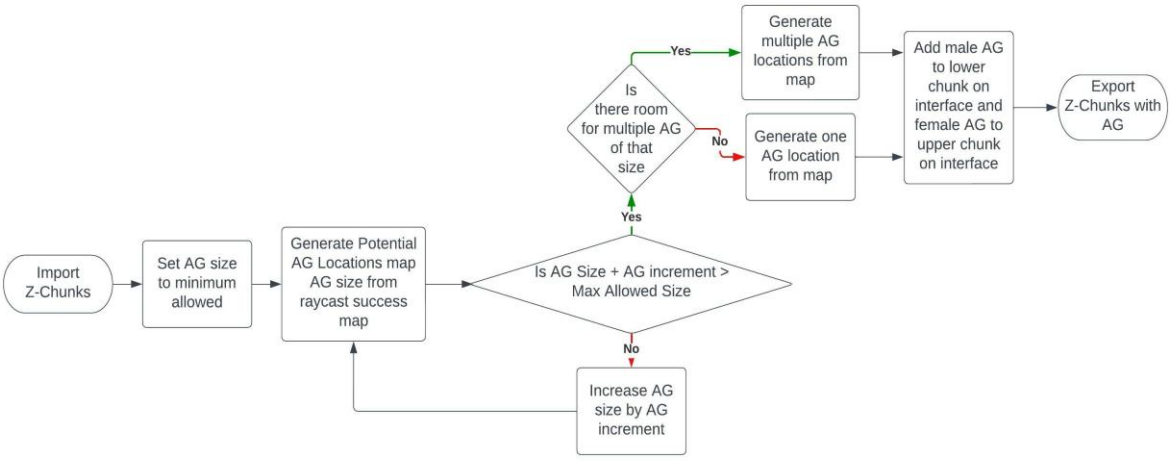
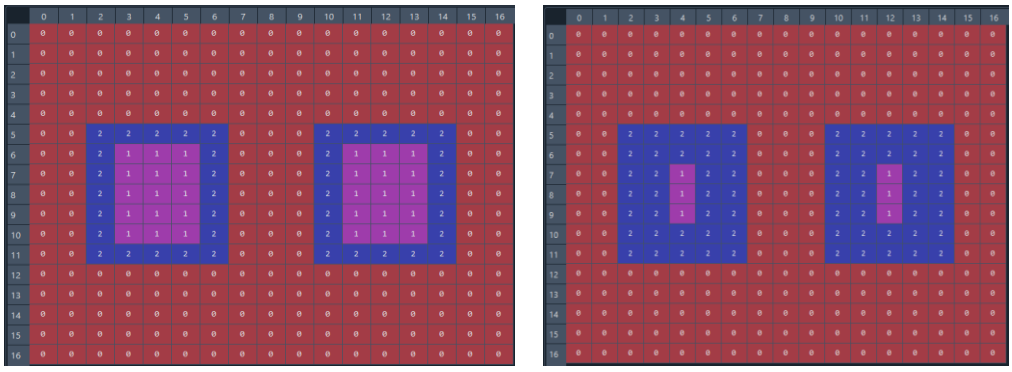


Figure 12: Flowchart for adding Alignment Geometry

Once chunking locations are determined, the algorithm cuts at those locations, and the algorithm places AG on each of the chunk interfaces, following the flowchart in Figure 12. First, we expand the raycast results at those locations past the success map shown in Figure 11 b) and the island map shown in Figure 11 c) to the potential AG locations map shown in Figure 13. We do this by checking the neighboring grid points around a location within a radius corresponding to the desired AG size. If all are valid interface points, the center point is a valid AG location. It should be noted that because the AG locations must be valid for both the male and female, we actually use the radius of the female AG which is multiplied by a fit multiplier, in our case 15%, versus the male AG.



a) b)

Figure 13: The map of all potential AG locations on the interface. '1' means this location is suitable for an AG of at least minimum size based on a radius of connected successful locations from Figure 12 b). '2' means that this location is connected to a potential AG location, therefore notating that it is not a disjoint island. For a) a radius of 1 grid location was used, and for b) a radius of 2 grid locations was used.

As discussed above, there is a minimum AG size based on printer accuracy. The algorithm will iteratively use larger and larger AGs until there is no island that can accommodate an AG of

that size or the maximum AG size, as specified by the user, has been reached. The current step size is an increase of 5mm in radius each iteration. The algorithm tracks AG size and location on a per-island basis to ensure the most appropriately sized AG for each island, as shown in Figure 14.

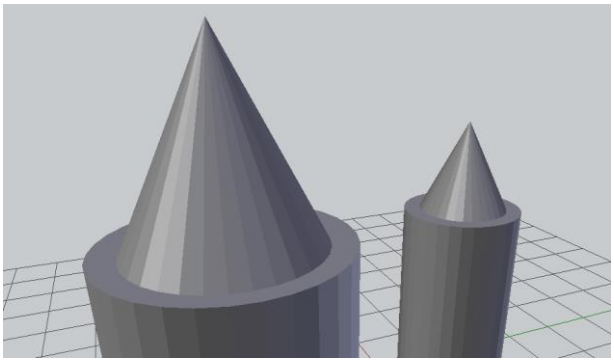


Figure 14: A demonstration of differently sized AG on the same layer. Each AG conforms to the size of the island that it is on.

Additionally, if there is room for more AG of the maximum size for a given island on that island, multiple will be placed. This is done by sequentially filling in the area within a set spacing distance around an AG with non-AG spaces, as shown in Figure 15.

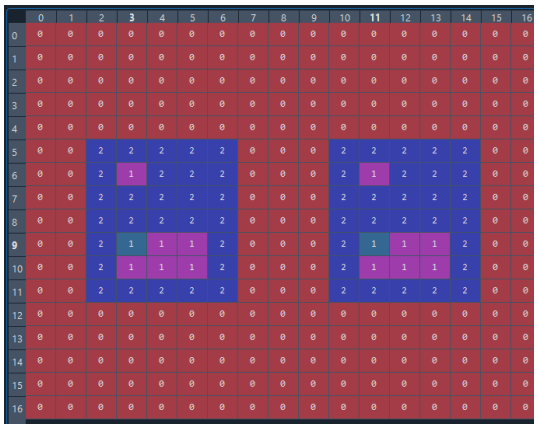


Figure 15: Placement strategy when multiple AG can be placed on one island. The grid spaces (3,6) and (11,6) are the first AG of a particular size on each island. A spacing of 2 grid points between AG was used, represented with '2's in the grid map. The algorithm would continue by placing the next AG at the highlighted spaces of (3,9) and (11,9) and repeat if there were room for any more.

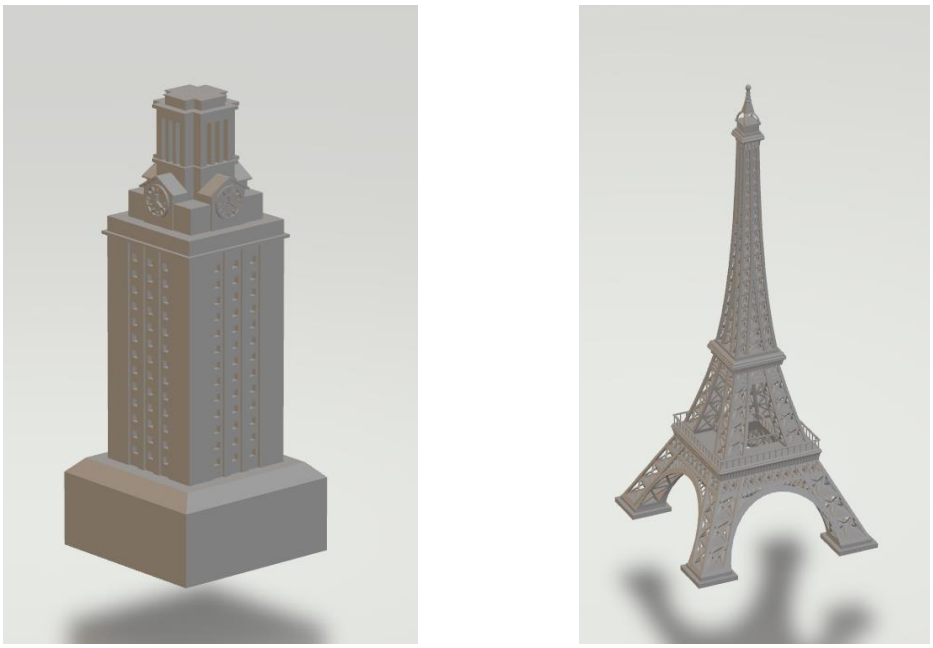
As mentioned above, in the current iteration of the algorithm, all of the AG have the same shape but can have a different scale. By default, the global maximum height for AG is set at 10% of the Z limit of the printer. This is done so that the algorithm can check the sublayers within this height above to ensure that there is a solid volume to fit in and that no part of the AG will be showing from the outside of the job. This data is compiled into the success data from the raycast for that layer. For example, if on one layer, the raycast detects a successful interface but not in the layer above, the algorithm will return a value of "False" for the interface check at this point. To ensure printability, we only use a hole and peg style for the AG, as shown in Figure 7, and set a minimum AG size based on the printer's accuracy. Additionally, we introduce a fit multiplier to ensure adequate clearance between the hole and the peg. We set this clearance at 15% based on several quick printing trials. The algorithm for adding Alignment Geometry ensures that:

1. All added AGs are printable because of the minimum size constraint
2. AGs are only added to solid surface locations
3. AGs are added strictly internally
4. AGs are aligned across interfaces and join the part together after assembly

Together, these two parts of the algorithm ensure that all four of the rules are followed and that feasible prints are generated.

Results and Discussion

This section will detail STL file and physical print results from applying the Z-Chunking rules, implemented as described above, on two case studies, The University of Texas Tower (UT Tower) and the Eiffel Tower, as shown in Figure 16. These two were chosen as test cases because they are both tall objects that, when appropriately scaled, require Z-chunking. The UT Tower is a relatively simple structure consisting mainly of progressively narrower boxes with some shallow surface adornment. The Eiffel Tower model, on the other hand, while having the same general shape, is much more challenging because it consists of many small lattices at many layers across its height. For the Eiffel tower, the main challenge is finding the correct locations to place the cuts for Z-chunking.



a) b)

Figure 16: Models of a) the UT Tower and b) Eiffel Tower before chunking.

We evaluated the success of our model test cases on several criteria related to the rules for Z-Chunks. Those criteria were: do all of the chunks fit within the printer bounds, and are fully enclosed AG generated at every interface?

In our first test, we set the height of the UT Tower to be 350mm and the height of the Eiffel Tower to be 530mm, which would require at least two vertical chunks given the printer Z limit of 265mm. The algorithm produces two chunks for the UT Tower but three for the Eiffel Tower. The location of the cuts, as well as the configuration of the AG, was different for each, as shown in Figure 17. The height of each chunk was: 245.45mm and 226.55mm for the UT tower and 241mm, 146.5mm, and 146.5mm for the Eiffel Tower from top to bottom, respectively, all within the height limit of the printers.

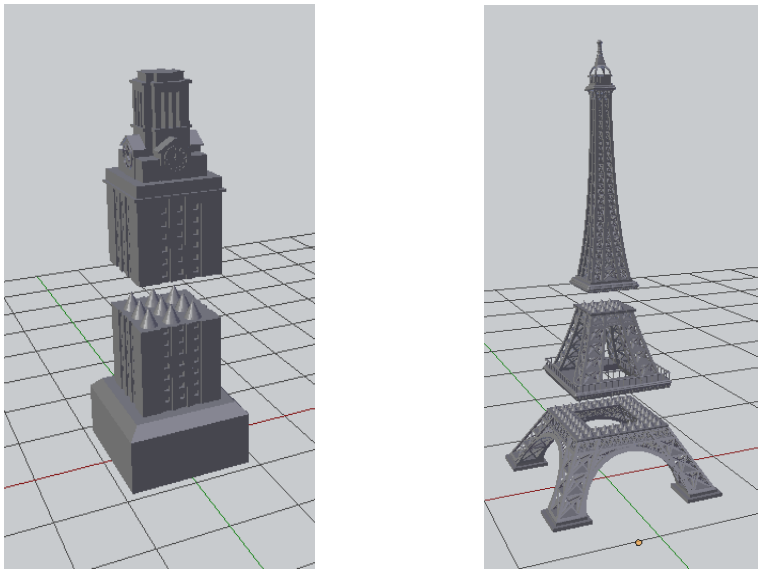


Figure 17: Three Layer chunked models of the UT Tower (left) and Eiffel Tower (right).

For the UT Tower, the first observation is that the total height of the chunks adds up to 472mm, not the original height of 450mm. This is because of the additional height from the AG on the lower layer. The height of the bottom without AG is 204.55, which now adds up to the expected 450mm. Other results are that the AG had a height of 22mm found using the default values discussed previously.

For the Eiffel Tower, the first peculiarity is that three chunks instead of two were generated. This is because there are two locations, the two platforms in the middle, that allow for just one interface across instead of the many small lattices which are unsuitable for AG. For this model, the AG ended up at 2mm tall, which we had to reduce the AG increment to .5mm/iteration to achieve. This is very close to the minimum AG size because the solid areas of the tower are very thin, and we were not exactly in the center of them because of the resolution of our sublayers. These models show that we can create Z-Chunks that are geometrically printable and have automatically generated Assembly Geometry. It should also be noted that the difference in size between the two tests is because the height of the Eiffel Tower had to be adjusted so that the cutting locations lined up with the solid portions. This highlights the need for more cutting locations in the algorithm, which is infeasible with the current structure as we will discuss later in the runtime section.

To further prove that the chunks are printable and that the Z-chunking does not interfere with the C3DP workflow, we physically printed the more complex of the two cases, the Eiffel Tower. For the purposes of print time, the physical prints were scaled down to be 300mm tall, with the printer Z-limit set to 177mm. We scaled them down this way instead of just scaling down the STL file for all of the layers to ensure that none of the rules, especially pertaining to printability of small added features, were broken. Images of the print can be seen as Figure 18. Because the AG size for the print was so close to the minimum allowed by the SCARA printer, some light post-processing was needed on and around the AG to enable assembly.

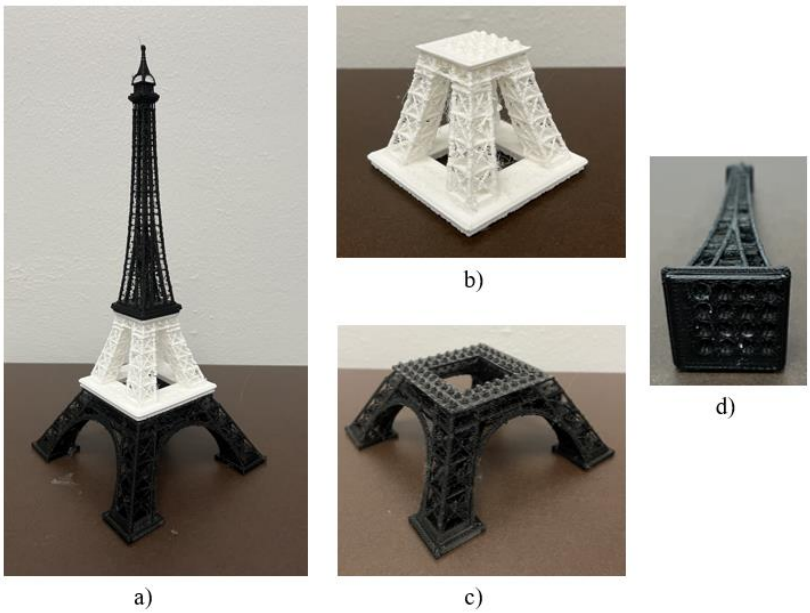


Figure 18: a) shows the assembled model. b) shows the middle chunk. c) shows the bottom chunk. d) shows the base of the upper chunk with the female AG.

One important factor in determining the feasibility of this algorithm is the impact on runtime for the chunking process. The following data was collected on a computer running Windows 10 Enterprise on an Intel i9-11900 with 32 GB of RAM at 3200 MHz. The object used for trial was the UT Tower model shown above

Table 3: Runtime Data

Scenario	5 Trial Average Time to Run (s)
Base Algorithm: cuts job into 3 even height chunks	.53
Base with Raycast 1mm density	.53
Base with Raycast .001mm density	.53
Base with AG generation	6.71
10 potential cuts without AG generation	2.43
20 potential cuts without AG generation	29.16
25 potential cuts without AG generation	988.12

Table 3 shows that the Raycast method is computationally inexpensive and scalable. It also shows that the opposite is true about the method for finding the best combination of active and inactive sublayers. The brute force search of all combinations increases exponentially in runtime when compared to the increase in number of potential cuts.

A more efficient method of finding an optimal configuration would allow for the number of sublayers to me greatly increased. However, we do expect that for sublayers that are very small (ex. Less than the layer height) there will be greatly diminishing returns. We have not been able to test this theory because for objects of the size of this test cases, we would need to scale up the number of sublayers by about a factor of 100.

Additionally, we do not take layer height into account in this algorithm because we are purely concerned with chunking, not slicing, in this paper. Slicing is handled after Z-chunking by dedicated slicers for each chunk.

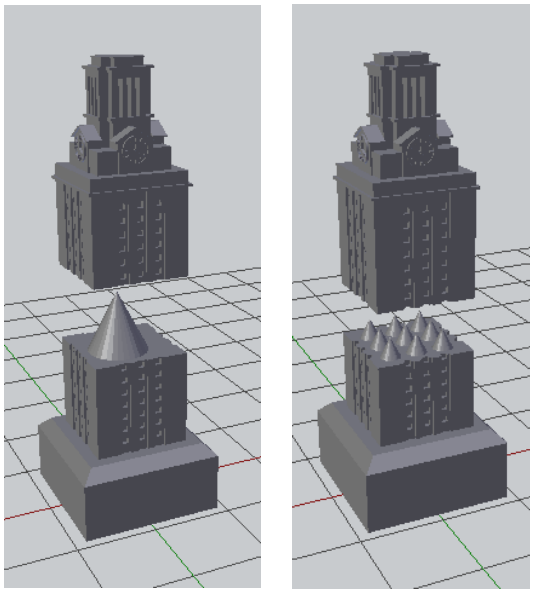
Conclusion and Future Work

In this paper, we have shown that by applying the five rules for Z-Chunking, based on DFA and related principles, we can generate feasible chunking configurations. We would like to highlight that these rules and our current algorithm do not guarantee that the chunking strategy is optimal, only that it will return feasible results. Case studies were presented to demonstrate that these chunks are printable in C3DP as described in our previous work. In the future, the algorithm could be expanded to allow users to specify different desired performance metrics for different use cases within the framework of the rules for Z-Chunking. Some examples of this include:

1. Change structural strength of assembled job by adjusting:
    - Type of AG used
    - Maximum AG size and spacing
    - Weight chunk location selection process to maximize interface
2. Optimize work distribution by printers
    - Weight chunk location selection process to equalize print volume, which is directly related to print time
3. Alter aesthetics
    - Weight chunk location selection process to be at transitions between features to help hide the seams

None of these variables have been accounted for in the current iteration of the code because they are subjective and difficult to quantify and automate. However, at this stage, these can be manually controlled and adjusted. Some examples created by manually adjusting the chunking parameters are shown in Figure 19.

In the future, we plan to expand the Z-chunking capability to include a larger variety of Alignment Geometry types and shapes and to work towards the optimal chunking strategy. The optimal chunking strategy would need to tie into and be interdependent with the scheduling and path planning algorithms and would require the development of a robust solution to determine where to place the different layers on the factory floor. This will help to facilitate the growth of the AMBOTS printing platform and will progress towards the goal of a robust network of decentralized factories that can print anything from any material at any time.



a) b)

Figure 19: Examples of modifications to the Z-Chunking strategy within the rules for Z-Chunking. a) shows the results if the AG size is allowed to increase past the default 10% of the height to 40mm. b) shows a more even volume distribution where the chunk split is lower on the model. In this case the top chunk is the maximum of 265mm tall.

Acknowledgments

This project is supported by the National Science Foundation (NSF) through Grant #2112009.

References

[1] Horn, T. J., & Harrysson, O. L. (2012). Overview of current additive manufacturing technologies and selected applications. *Science Progress*, *95*(3), 255–282. https://doi.org/10.3184/003685012x13420984463047

[2] Shusteff, M., Browar, A. E., Kelly, B. E., Henriksson, J., Weisgraber, T. H., Panas, R. M., Fang, N. X., & Spadaccini, C. M. (2017). One-step volumetric additive manufacturing of complex polymer structures. *Science Advances*, *3*(12). https://doi.org/10.1126/sciadv.aao5496

[3] Al Jassmi, H., Al Najjar, F., & Mourad, A.-H. I. (2018). Large-scale 3D printing: The way forward. *IOP Conference Series: Materials Science and Engineering*, *324*, 012088. https://doi.org/10.1088/1757-899x/324/1/012088

[4] Ali, M. H., Mir-Nasiri, N., & Ko, W. L. (2015). Multi-nozzle extrusion system for 3D printer and its control mechanism. *The International Journal of Advanced Manufacturing Technology*, *86*(1-4), 999–1010. https://doi.org/10.1007/s00170-015-8205-9

[5] Roschli, A., Gaul, K. T., Boulger, A. M., Post, B. K., Chesser, P. C., Love, L. J., Blue, F., & Borish, M. (2019). Designing for big area additive manufacturing. *Additive Manufacturing*, *25*, 275–285. https://doi.org/10.1016/j.addma.2018.11.006

[6] McPherson, J., & Zhou, W. (2018). A chunk-based Slicer for cooperative 3D printing. *Rapid Prototyping Journal*, *24*(9), 1436–1446. https://doi.org/10.1108/rpj-07-2017-0150

[7] Poudel, L., Sha, Z., & Zhou, W. (2018). Mechanical strength of chunk-based printed parts for cooperative 3D printing. *Procedia Manufacturing*, *26*, 962–972. https://doi.org/10.1016/j.promfg.2018.07.123

[8] Poudel, L., Blair, C., McPherson, J., Sha, Z., & Zhou, W. (2020). A heuristic scaling strategy for multi-robot cooperative three-dimensional printing. *Journal of Computing and Information Science in Engineering*, *20*(4). https://doi.org/10.1115/1.4045143

[9] Poudel, L., Zhou, W., & Sha, Z. (2021). Resource-constrained scheduling for multi-robot cooperative three-dimensional printing. *Journal of Mechanical Design*, *143*(7). https://doi.org/10.1115/1.4050380

[10] Poudel, L., Zhou, W., & Sha, Z. (2020). A generative approach for scheduling multi-robot cooperative three-dimensional printing. *Journal of Computing and Information Science in Engineering*, *20*(6). https://doi.org/10.1115/1.4047261

[11] Luo, L., Baran, I., Rusinkiewicz, S., & Matusik, W. (2012). Chopper: partitioning models into 3D-printable parts. *ACM Transactions on Graphics*, *31*(6), 1–9. https://doi.org/10.1145/2366145.2366148

[12] Hao, J., Fang, L., & Williams, R. E. (2011). An efficient curvature-based partitioning of large-scale STL models. *Rapid Prototyping Journal*, *17*(2), 116–127. https://doi.org/10.1108/13552541111113862

[13] Otto, K. N., & Wood, K. L. (2001). In *Product design: Techniques in reverse engineering and new product development* (pp. 1–2). essay, Pearson Custom Pub.

[14] Medellín, H., Lim, T., Corney, J., Ritchie, J. M., & Davies, J. B. (2006). Automatic subdivision and refinement of large components for rapid prototyping production. *Journal of Computing and Information Science in Engineering*, *7*(3), 249–258. https://doi.org/10.1115/1.2753162

[15] Thompson, M. K., Moroni, G., Vaneker, T., Fadel, G., Campbell, R. I., Gibson, I., Bernard, A., Schulz, J., Graf, P., Ahuja, B., & Martina, F. (2016). Design for Additive Manufacturing: Trends, opportunities, considerations, and constraints. *CIRP Annals*, *65*(2), 737–760. https://doi.org/10.1016/j.cirp.2016.05.004

[16] Steuben, J., Van Bossuyt, D. L., & Turner, C. (2015). Design for fused filament fabrication additive manufacturing. *Volume 4: 20th Design for Manufacturing and the Life Cycle*

Conference; 9th International Conference on Micro- and Nanosystems. https://doi.org/10.1115/detc2015-46355

[17] Chan, C. K., & Tan, S. T. (2003). Generating Assembly features onto Split Solid Models. *Computer-Aided Design*, *35*(14), 1315–1336. https://doi.org/10.1016/s0010-4485(03)00062-9

[18] Redwood, B., Schöffer Filemon, Garrett, B., & Fadell, T. (2020). *The 3D Printing Handbook Technologies, design and applications*. 3D Hubs.