

TOOLPATH PLANNING FOR MULTIPLE BUILD POINTS USING K-MEANS CLUSTERING

Breanne Crockett and Michael Borish

Manufacturing Science Division, Oak Ridge National Laboratory, Knoxville, TN 37932

Abstract

Traditional 3D printers deposit material at one build point at a time, often resulting in long print times. To reduce print time, 3D printers could increase throughput with parallel construction at multiple build points. The primary challenge in path planning for parallel construction is dividing an object between the build points. The object should be divided such that the workload is balanced, and the individual build areas are discrete. This work proposes a variation of k-means clustering for object division. The algorithm considers coordinate position and geometric area as an indicator of workload. This method is demonstrated on several test models to compare workload across the number of build points.

Introduction

Additive manufacturing (AM), also referred to as 3D printing, is a construction process where material is iteratively deposited to create a three-dimensional object. This process offers several advantages over other manufacturing methods, namely the ability to build complex objects as a single part and the ability to create unique parts in small quantities with comparatively low cost. Traditional 3D printing deposits material at one location at a time, therefore employing a single build point. The addition of multiple build points could provide several advantages over a single build point. Multiple build points could increase material throughput and decrease construction time, enabling larger and more complicated objects to be printed. Furthermore, multiple build points could support printing with multiple materials and allow for the construction of objects with unique mechanical properties. Multiple build points could also be foundational in creating a system with multiple construction processes, as in a system with one additive build point and one subtractive build point. This work focuses on the toolpath planning required for multi-build point systems with the goal of decreasing construction time.

Toolpath planning for single-build point AM begins with a CAD model of the object to be constructed. Typically, the model is transformed into a triangulated mesh, sliced, and then converted into gcode instructions for the machine. Slicing is the process of cutting the object into layers and creating paths from two-dimensional cross-sections. To accommodate multiple build points, the slicing process must also include steps for dividing the object among the build points. Object division is a complex optimization problem with many parameters, and machine learning could be a potential solution. More specifically, clustering algorithms offer a means to group input data by similar features. Clustering is also effective with smaller datasets and less computationally expensive compared to other machine learning methods, making it a reasonable addition to existing slicing workflows.

Related Work

The issue of decreasing build time has been investigated before. The research space can be divided into two categories: parallel printing and cooperative printing. Parallel printing is the process of segmenting an input mesh into smaller parts that are individually sliced and printed concurrently on distinct machines. Chen, Kuo-Wei, et al. proposed a method to segment an input mesh by generating a support skeleton for the model and partitioning the shell so that each partition can be assembled by attaching it to the skeleton [5]. Dinar introduced a series of heuristic rules to consider when partitioning a volume to maintain the advantages offered by AM [6]. Later, Li, Poozhikala, and Dinar suggested a segmentation algorithm where the largest inscribed cube or extended cuboid serves as the skeleton and the six faces of the skeleton are used as cutting planes to divide the input mesh [9]. While these methods effectively decrease print time, segmenting the input mesh changes the mechanical properties of the resulting object and is only considered in Dinar's work [6]. Additionally, parallel printing reintroduces an assembly step, negating a key advantage of AM.

In contrast to parallel printing, cooperative 3D printing employs a single system with multiple build points. Cooperative, also referred to as collaborative, printing implies the coordinated use of many motion-independent robots. The term is also occasionally used to describe gantry systems with multiple independent extruders. We use the term "multi-build point" to unambiguously refer to systems comprised of many individual robots or a single machine that is capable of depositing material at multiple locations simultaneously. Two commercial examples of multi-build point systems have been reported. Cronus by Titan Robotics is a gantry system with five print heads that was released in 2017 as the first product to employ software developments achieved in Autodesk's Project Escher [2]. SiSpis by Siemens is a collection of mobile arachnid-inspired robots each with an extruder that was unveiled in 2016 [16].

In addition to the commercial products, there is active research into multi-build point systems. Much of the work follows the traditional layer-by-layer toolpath planning process, and then divides the pathing for each layer among the build points. Leite et al. propose a hypothetical machine that includes multiple gantries and is extendable along both X and Y axis [8]. For the proposed machine, Leite et al. employed Monte Carlo simulations and heuristics to divide a closed contour along axis-aligned boundaries. Hongyao, Lingnan, and Jun discussed a system with multiple, fixed location robotic arms and suggested a method to post-process gcode instructions for collaborative printing with four robotic arms [7]. Zhang et al. investigated a similar system with the distinction that two mobile robotic arms were used [19]. Zhang et al. first addressed the problem of placing the robotic arms and then performed path planning. Nguyen-Van and Gwak introduced a cable-driven parallel robot with two-nozzles mounted on the same print head [12]. For the cable-driven parallel robot, a single closed contour is printed by first planning the motion of the print head and then the motion of each nozzle relative to the print head. The problem of dividing a single contour among build points is well discussed, but none of the aforementioned work elaborates on how to divide a layer with multiple closed contours. The algorithm we propose addresses the problem of dividing multiple closed contours between the build points. Additionally, the results of the proposed algorithm are not limited to axis-aligned boundaries as in the work of Leite et al [8].

As an alternative to slicing an object and then dividing its toolpath, McPherson and Zhou proposed a method to “chunk” an object into 3D pieces and then slice each chunk into layers [11]. In their initial implementation, all chunks were printed by a single, mobile build point. Poudel et al. expanded that work by introducing a heuristic to divide chunks among multiple build points [13]. Poudel, Zhou, and Sha have since introduced a computational scheduling strategy for scheduling chunks in a multi-build point system, although it did not perform as well as the heuristic approach [14]. While this method works well for their machine implementation, the proposed algorithm aims to provide a general solution to the division problem that can be used for a variety of machine implementations.

The method proposed in this work employs a common unsupervised machine learning (ML) algorithm, therefore it is worth mentioning the body of work related to the use of ML in additive manufacturing. A recent and comprehensive review of existing ML applications in AM by C. Wang et al. notes that ML has been used for parameter optimization, in-process defect monitoring, pre-manufacturing planning, and product quality control [17]. For example, Caggiano et al. demonstrate the use of a deep convolutional neural network for defect detection in selective laser melting systems [4]. Similarly, T. Wang et al propose a neural network for real-time control of printing parameters to mitigate defects in liquid metal jet printing [18]. While ML has been used to correct or update tool pathing, the use of ML in the initial path planning steps, as we propose here, has not been reported.

Layer Division: An Application of K-means Clustering

We introduce the application of k-means clustering with the addition of a balancing step to divide a part between multiple build points. The presented algorithm is built on the ORNL Slicer 2.0 architecture [3]. ORNL Slicer 2.0 is an ongoing research project that provides basic slicing capabilities and integrates numerous cutting edge slicing techniques. The general slicing workflow follows 4 steps: (1) input a triangulated mesh, (2) cut the mesh into a series of 2D cross-sections, (3) generate paths from cross-sections, (4) convert paths into machine instructions. The proposed algorithm operates on a set of islands, or closed polygons, generated by cross sectioning after step 2. The division between multiple build points is performed before path generation and conversion to gcode in steps 3 and 4. This allows for the customization of parameters that may be specific to a build point. Additionally, dividing the part geometry before pathing and gcode generation enables the separation of the division problem from the collision problem. The separation means that the division algorithm could be applied to a variety of systems, and that the machine-specific constraints to prevent collisions can be handled later in the path planning process. Both the ability to customize tuning parameters and prevent collisions as a separate phase would be advantageous for system with heterogeneous build points.

The goal of layer division is to create non-overlapping print areas of equal workload. Layer division is performed in two phases: initial division and rebalancing. To create non-overlapping print areas, islands to be printed by the same build point must be spatially near each other. To compare the location of islands, only the centroid of the island is considered. The centroid is found as the average of all points defining the polygon. To balance workload between build points, a geometric indicator of work must be defined. If the balancing process occurred after gcode generation, total printing length could be used as an exact measure of work. However, the proposed algorithm performs division before path generation and instead uses geometric area as a proxy for work. This is most accurate when the part is being printed with

infill, or some densifying structure. If the model was to be printed as a hollow shell, perimeter length would be a more appropriate indicator of work.

The initial division focuses only on the location of the polygons. The goal is to divide the polygons into natural spatial groups, making clustering an ideal fit. The k-means method segments the set of input points by finding a set of k centers and assigning each point to its nearest center. The standard method for computing centers is Lloyd's algorithm, also commonly referred to as the k-means algorithm [10]. Lloyd's algorithm computes the centers by randomly selecting initial centers then iteratively assigning points to clusters and recomputing centers until convergence. Arthur and Vassilvitskii introduced k-means++, a variation of Lloyd's algorithm that intelligently selects initial centers [1]. K-means++ is employed in this algorithm because it often runs faster and produces better results. Additionally, an implementation was readily available via the SciKit Learn python package [15]. Figure 1 depicts the process of calculating polygon centroids and then clustering those centroids.

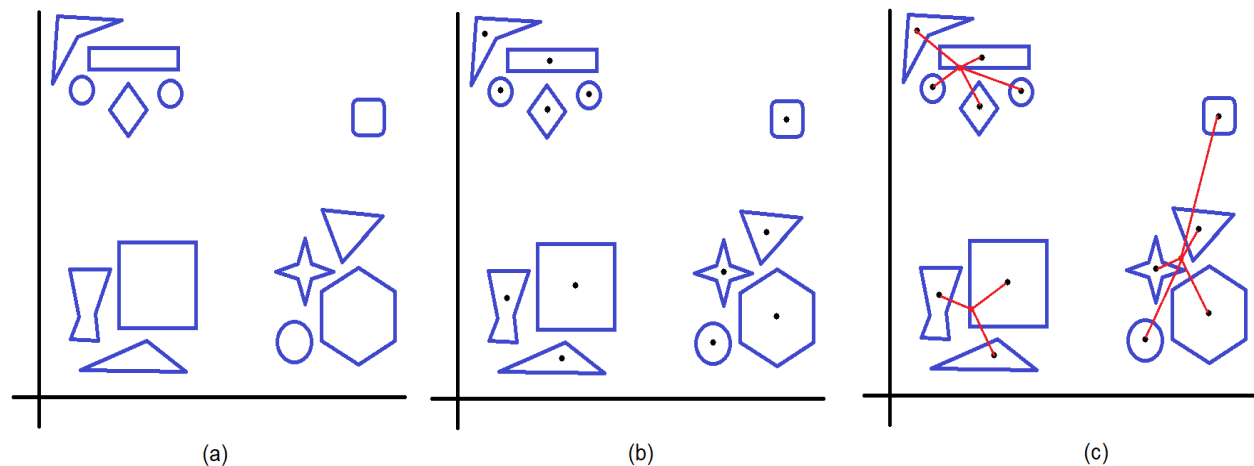


Figure 1: K-means Clustering on a Set of Polygons

(a) Input polygons, (b) Polygons with their centroids, and (c) Results of clustering for $k=3$ with a line drawn from each input point to its nearest cluster center

Once the polygons are divided into initial groups by k-means clustering, the rebalancing phase aims to correct any workload imbalances in the results. This is achieved by transferring polygons from the cluster with the highest workload to another, such that the maximum amount of work done by one cluster is reduced. A rule is imposed such that a polygon must be transferred to an adjacent cluster, so that the clusters continue to define non-overlapping build areas. Clusters are adjacent if the cells of the Voronoi diagram created from cluster centers share an edge. Figure 2 depicts an initial clustering and its corresponding Voronoi diagram. In this example, a polygon could be transferred from Cluster 3 (red) to Cluster 2 (green), but not from Cluster 3 (red) to Cluster 1 (purple).

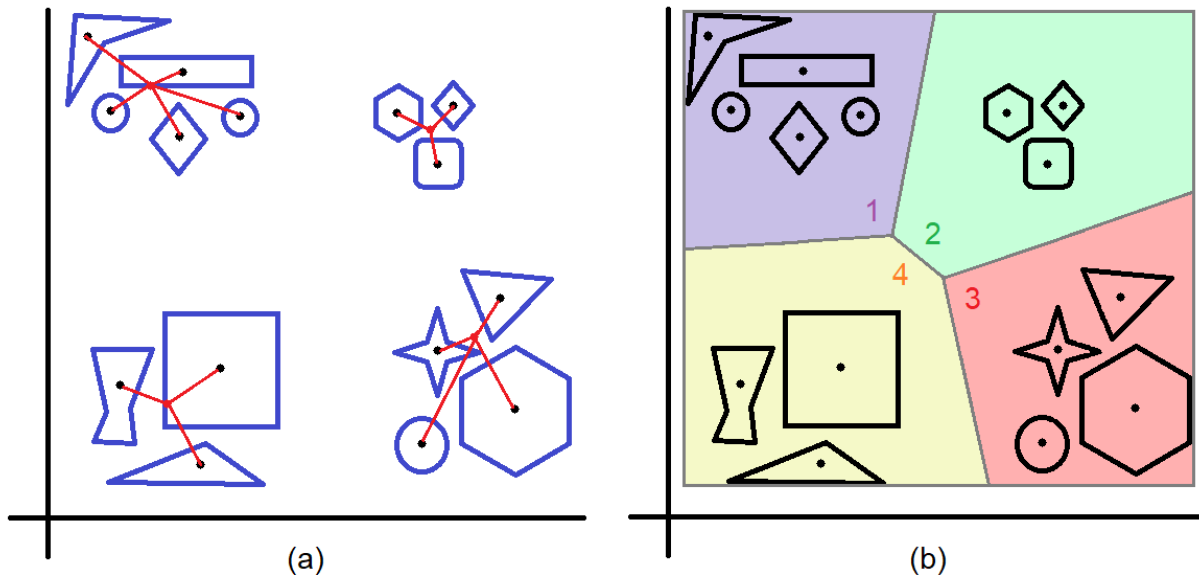


Figure 2: Determination of Adjacent Clusters

(a) An initial clustering for $k=4$ with a line drawn from each input point to its nearest cluster center; (b) the Voronoi diagram created from the centers of each cluster

In determining which polygon should be transferred, the set of clusters adjacent to the cluster with the largest workload are considered potential destinations. The transfer distance is calculated as the distance from the polygon to be transferred to the center of the destination cluster, as pictured in Figure 3. For each potential destination, a polygon is selected as the transfer candidate if its transfer does not increase the workload of the destination cluster beyond the current maximum workload and it has the shortest transfer distance of any polygon currently in the largest cluster. The list of transfer candidates is reduced to the best transfer option by selecting the candidate with the shortest transfer distance. A transfer is made by re-assigning the polygon to the new cluster and updating the workloads of the clusters. This transfer process is performed iteratively, until there are no more acceptable transfers from the largest cluster. A summary of the layer division algorithm is provided in Table 1.

Layer Division Algorithm
<ol style="list-style-type: none"> 1. Given a set of polygons, generate initial clusters with kmeans++ 2. Compute Voronoi diagram to determine which clusters are adjacent 3. Until no more transfers can be made: <ol style="list-style-type: none"> i. Find cluster with largest workload ii. Find all clusters adjacent to largest cluster iii. For each adjacent cluster: <ol style="list-style-type: none"> a. Determine if any polygons in largest cluster meet transfer requirements b. Add to list of transfer candidates iv. Select the best transfer from the list of candidates v. Make the transfer.

Table 1: Summary of Layer Division Algorithm

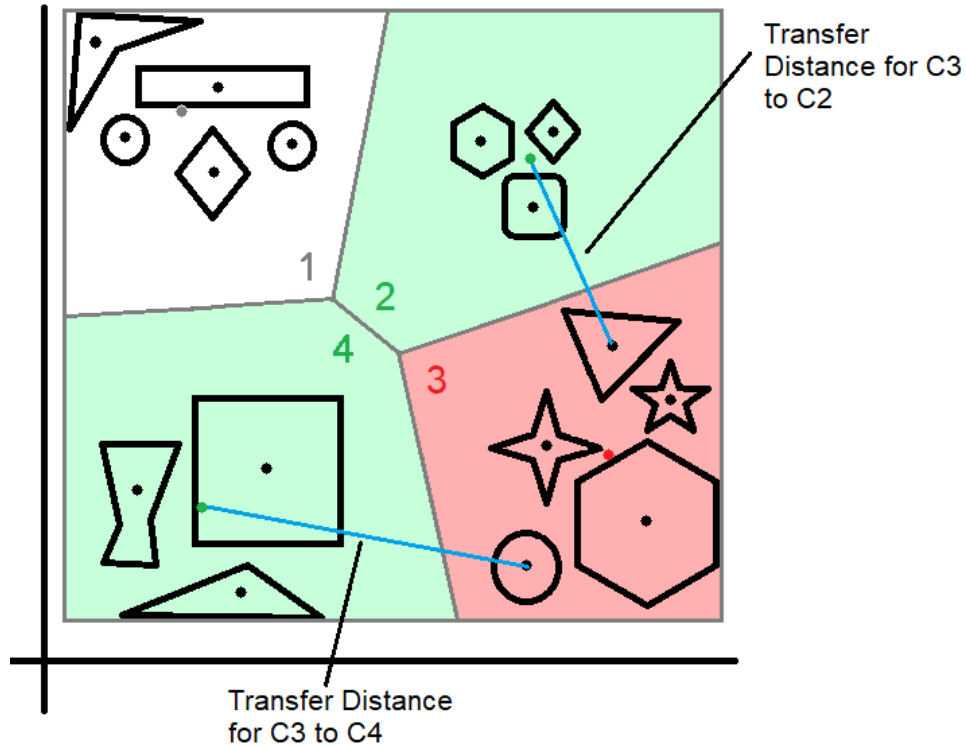


Figure 3: Transfer Requirements

Cluster 3 has the largest workload. The shortest transfer distances for each adjacent cluster are shown. The better of the two transfer options is the triangle to cluster 2.

The division algorithm is applied to every layer during the slicing process. In large objects with many layers, this operation could become time consuming, particularly if the initialization of k-means takes many iterations to converge. This can be avoided by employing a fundamental assumption of additive manufacturing: the geometry between sequential layers does not change significantly, so that there is minimal overhang and need for support structure. This concept is illustrated in Figure 4 where the cross-sections between layers 9 and 10 are similar, even though layers 10, 20, and 30 are notably different. Since changes between consecutive layers are likely to be insignificant, the resulting centroids from clustering one layer can be used as the initialization for the next layer. This would reduce the number of iterations until convergence during the clustering phase and therefore reduce computation time.

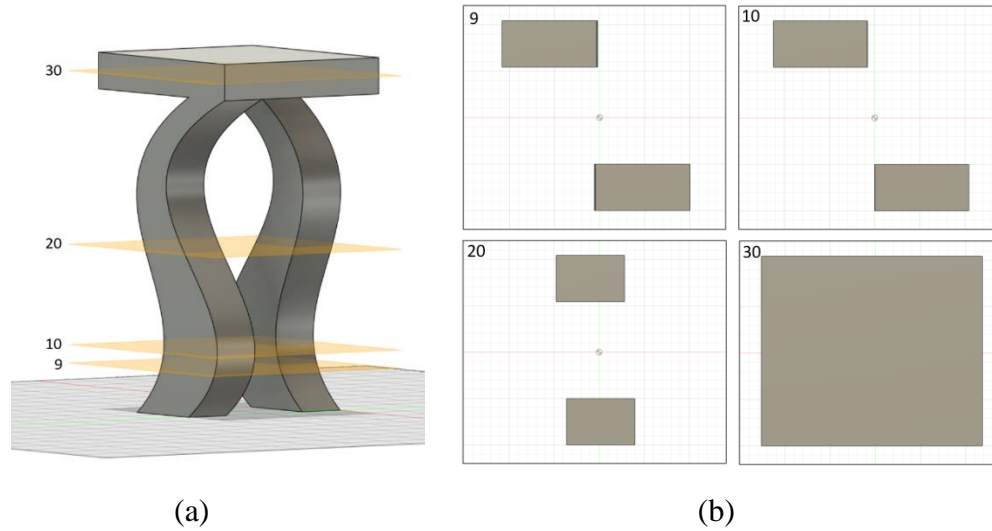


Figure 4: Model with Changing Cross Sections

(a) 3D model with planes indicating layer ‘slices’. (b) The results of cross-sectioning with the depicted planes.

Results

Several example models that were sliced and divided for multiple build points are presented. The original 3D model and a single cross-sectional layer is included for each example. Then, the results of dividing the cross-sectional layer are depicted. We consider only one cross-section of the presented examples because every layer is identical. In practice, cross-sections often change incrementally between layers, but this variation does not alter the efficacy of the layer division algorithm on a layer-by-layer basis. Each sample illustrates that the proposed algorithm produces non-overlapping print areas of approximately equal size.

Example 1: 16 Identical Cubes

The first example model is made of 16 identical cubes aligned in a four-by-four grid. This simple use case exemplifies how the proposed algorithm behaves when printing many, tightly packed duplicates of the same object. Figure 5 shows the 3D model and a single layer cross-section. The results of the proposed algorithm are illustrated in Figure 6. Each island is drawn with a color corresponding to its build point. For example, in Figure 6(a) the top two rows of islands are red and assigned to the first build point, and the bottom two rows of islands are blue, indicating their assignment to a second build point. We can quantitatively assess the success of the algorithm by comparing the percentage of the workload assigned to each nozzle. Table 2 summarizes the workload division corresponding to each scenario in Figure 6. As noted in the table, the division for 2 and 4 build points resulted in perfectly even workload distribution. Additionally, the division for 3 and 5 build points was as close to evenly distributed as possible without segmenting an island. For each of the test number of build points, the division results in non-overlapping print areas.

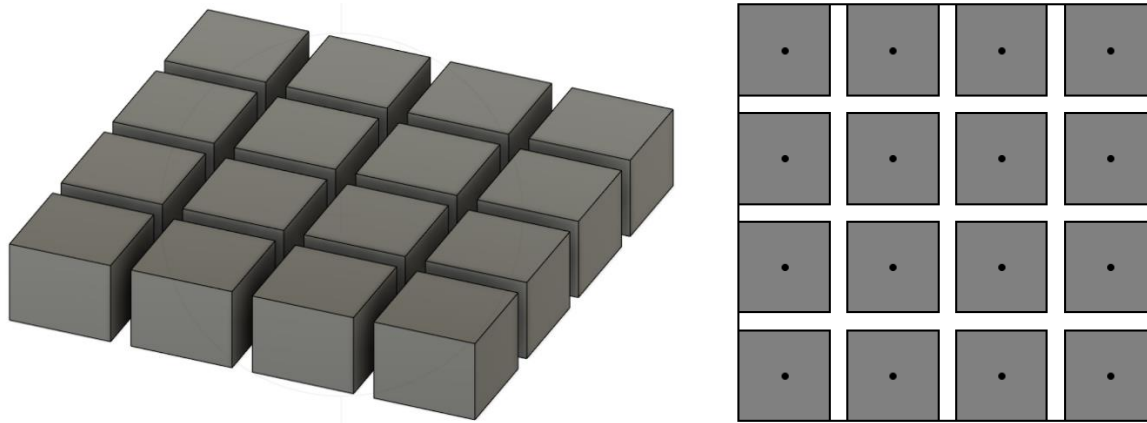
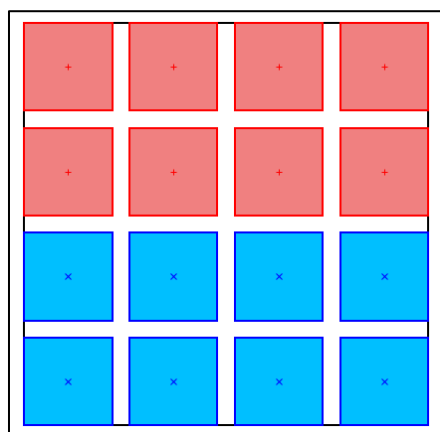
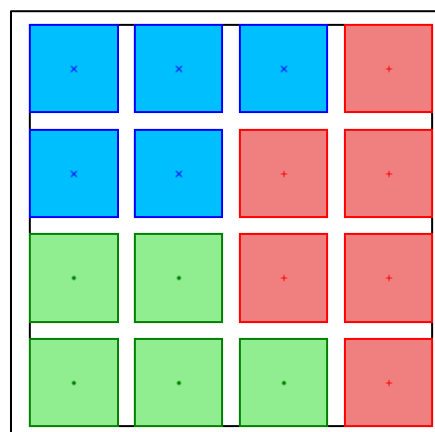


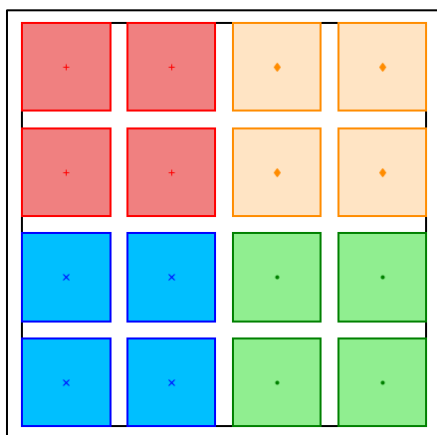
Figure 5: Input model and Single Layer Cross-Section for 16 Identical Cubes



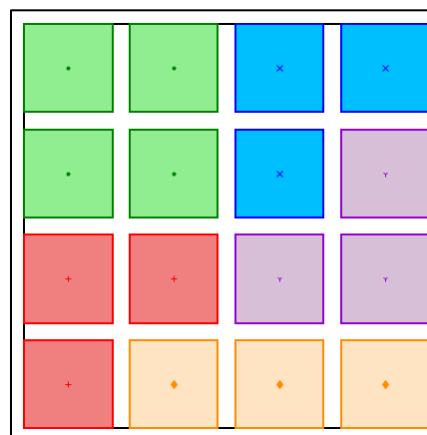
(a) 2 Build Points



(b) 3 Build Points



(c) 4 Build Points



(d) 5 Build Points

Figure 6: Layer Division Result for 2, 3, 4, and 5 build points

Build Points	Cluster #	Color	% of Total Area
2	1	Red	50.00
	2	Blue	50.00
3	1	Red	37.50
	2	Blue	31.25
	3	Green	31.25
4	1	Red	25.00
	2	Blue	25.00
	3	Green	25.00
	4	Orange	25.00
5	1	Red	18.75
	2	Blue	18.75
	3	Green	25.00
	4	Orange	18.75
	5	Purple	18.75

Table 2: Workload Division Results for 16 Identical Cubes

Example 2: 28 Heterogeneous Boxes

This example model contains 28 boxes of varying sizes as illustrated in Figure 7. This example demonstrates the algorithms behavior for islands of differing size, as would be the case in real-world prints. Just as the previous example, Figure 8 illustrates the division for 2, 3, 4, and 5 build points by coloring an island according to its grouping. Additionally, Table 3 shows the percentage of the total workload for each build point. In this example, the algorithm produced near ideal results for 2, 3, and 5 build points. In all cases, the division resulted in discrete printing areas for each build point.

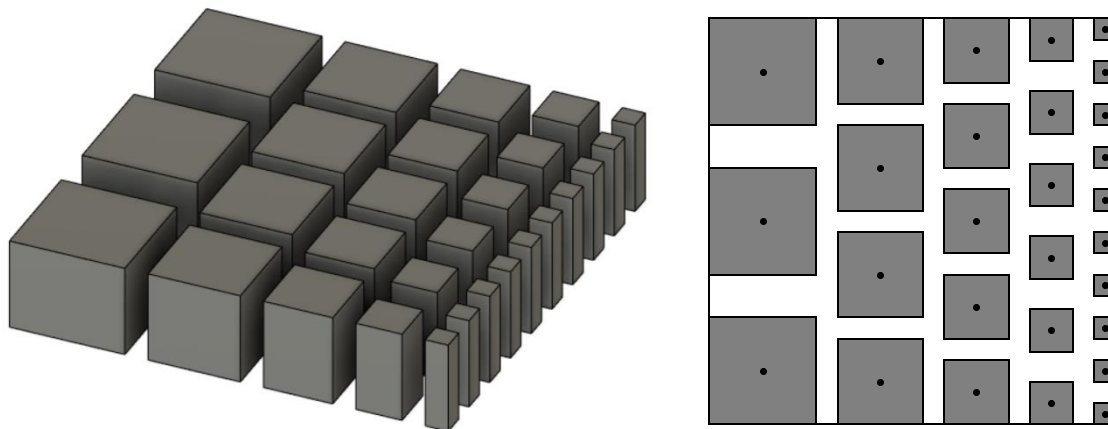


Figure 7: Input model and Single Layer Cross-Section for 28 Boxes

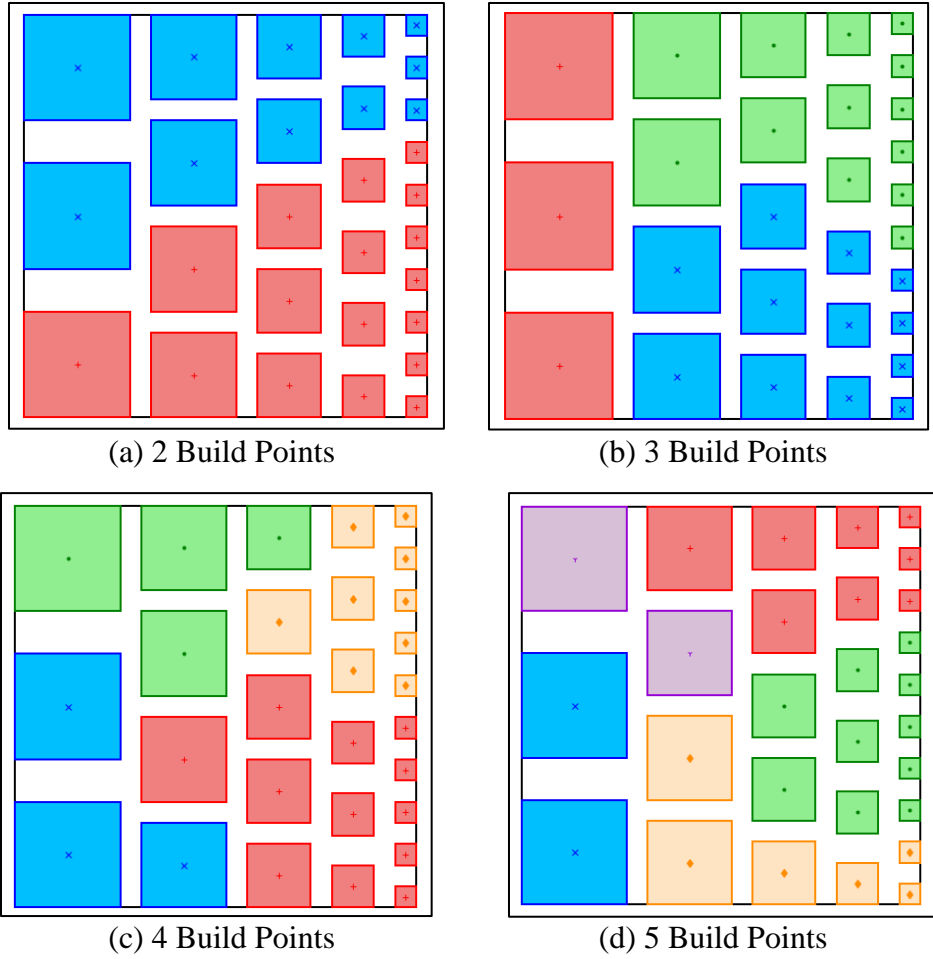


Figure 8: Layer Division Result for 2, 3, 4, and 5 build points for 28 boxes model

Build Points	Cluster #	Color	% of Total Area
2	1	Red	49.08
	2	Blue	50.92
3	1	Red	34.40
	2	Blue	34.40
	3	Green	31.19
4	1	Red	27.52
	2	Blue	30.28
	3	Green	30.28
	4	Orange	11.93
5	1	Red	20.64
	2	Blue	22.94
	3	Green	16.06
	4	Orange	21.56
	5	Purple	18.81

Table 3: Workload Division Results for 28 Heterogeneous Boxes

Example 3: Assorted Prisms

Figure 8 depicts the final example model. This model includes 17 prisms of varying shapes and sizes. As in the previous examples, Figure 10 shows the division for 2, 3, 4, and 5 build points by coloring an island according to its grouping, and Table 4 includes the percentage of the total workload for each build point. The division results for 5 build points in Figure 10(d) demonstrate a limitation of dividing along island boundaries. The circles in the blue and purple groups have a much larger area than the other islands, therefore placing an upper bound on the even distribution of workload. If the mechanical requirements of the part allowed, this limitation could be mitigated by dividing the island.

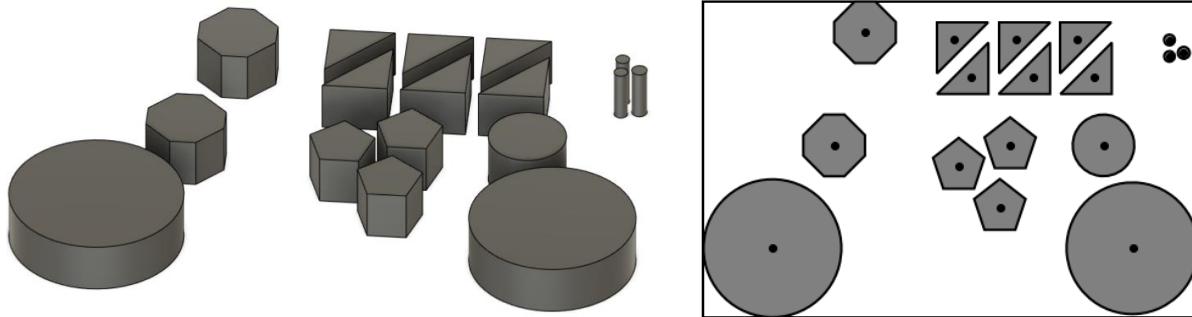


Figure 9: Input model and Single Layer Cross-Section for Assorted Prisms

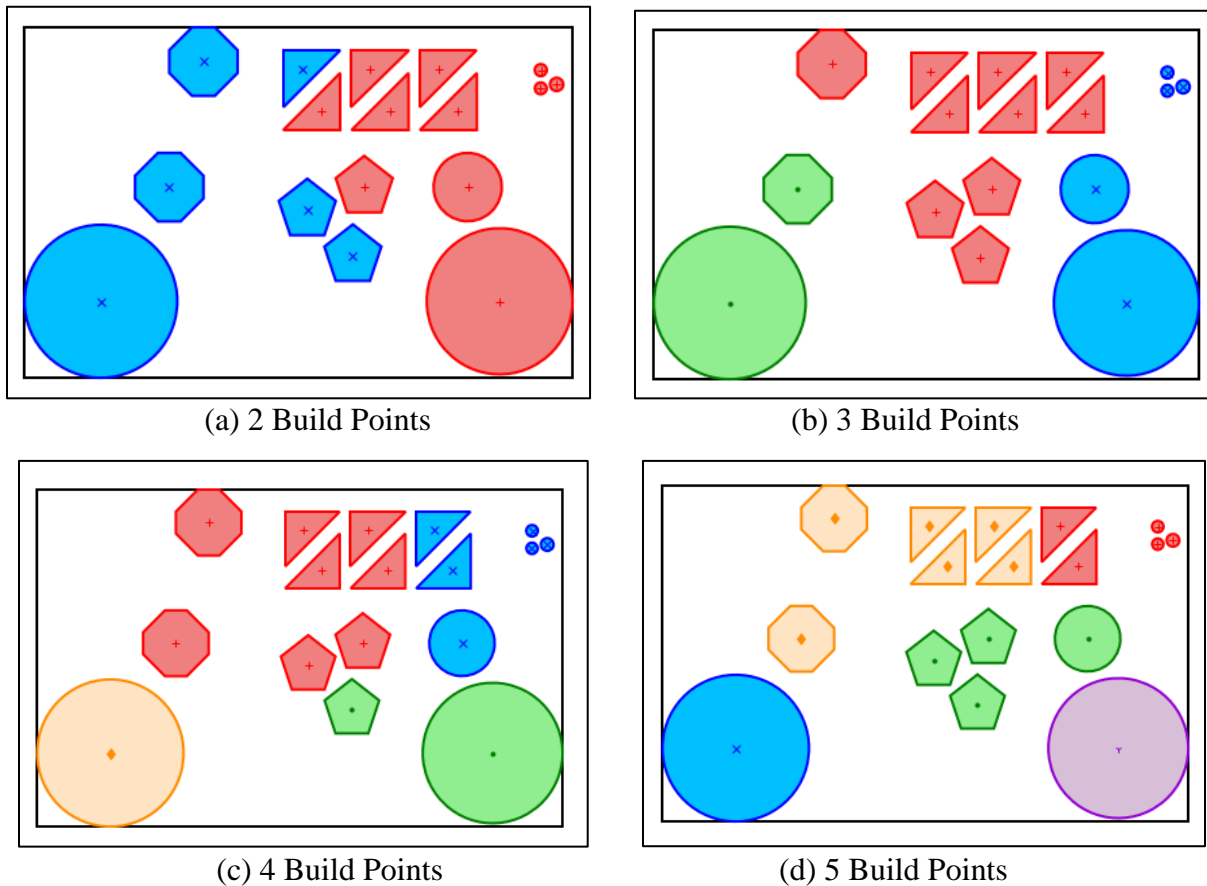


Figure 10: Layer Division Result for 2, 3, 4, and 5 build points for Assorted Prisms

Build Points	Cluster #	Color	% of Total Area
2	1	Red	49.52
	2	Blue	50.48
3	1	Red	31.86
	2	Blue	33.06
	3	Green	35.08
4	1	Red	28.98
	2	Blue	11.58
	3	Green	30.16
	4	Orange	29.27
5	1	Red	5.76
	2	Blue	29.27
	3	Green	16.32
	4	Orange	21.99
	5	Purple	26.66

Table 4: Workload Division Results for Assorted

Conclusion and Future Work

This work presents a layer division algorithm to divide an object among multiple build points. The algorithm employs clustering and a rebalancing phase to equally distribute workload while producing discrete, non-overlapping print areas. While non-overlapping print areas minimize the chance of build points colliding, it is not sufficient to avoid collisions. There must be an additional phase in toolpath planning to detect and mitigate collisions. Hongyao, Lingnan, and Jun proposed a means of avoiding collisions by labeling the area around shared boundaries as an “interference zone” and enforcing a rule that only one build point may be printing in an interference zone at one time [7]. Their method could follow the proposed algorithm in the toolpath planning process. To enable path planning that is independent of machine constraints, collision prevention needs to be performed in real-time by the machine. Perhaps further investigation could expand the “interference zone” rule to behave similar to a mutex lock in computing.

Additionally, there could be more work on the input to the clustering algorithm. The inclusion of more parameters could allow clustering to inform the path planning for heterogeneous build points. For example, the build material for an object could be used as input for clustering in a multi-material system. Similarly, a heterogeneous system could overcome the limitation of dividing along island boundaries, particularly if one build point is faster than another. Lastly, there could be more research into the use of the cluster centroids in machine setup. For systems that employ multiple robotic arms [7, 19], the cluster centroids could inform the placement location for the robots.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, Office of Advanced Manufacturing, under contract number DE-AC05-00OR22725.

References

1. Arthur, David, and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Stanford, 2006.
2. "Autodesk Project Escher Finds First Commercial Release in Cronus". <https://www.engineering.com/story/autodesk-project-escher-finds-first-commercial-release-in-cronus>. Accessed June 2022.
3. Borish, Michael, and Alex Roschli. "ORNL Slicer 2.0: Towards a New Slicing Paradigm." 2021 International Solid Freeform Fabrication Symposium. University of Texas at Austin, 2021.
4. Caggiano, Alessandra, et al. "Machine learning-based image processing for on-line defect recognition in additive manufacturing." CIRP annals 68.1 (2019): 451-454.
5. Chen, Kuo-Wei, et al. "Parallel 3D printing based on skeletal remeshing." ACM SIGGRAPH 2016 Posters. 2016. 1-2.
6. Dinar, Mahmoud. "Parallelized Additive Manufacturing of Variably Partitioned Volumes for Large Scale 3D Printing With Localized Quality." International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Vol. 83983. American Society of Mechanical Engineers, 2020.
7. Hongyao, Shen, Pan Lingnan, and Qian Jun. "Research on large-scale additive manufacturing based on multi-robot collaboration technology." Additive Manufacturing 30 (2019): 100906.
8. Leite, M., et al. "Multiple collaborative printing heads in FDM: The issues in process planning." 2018 International Solid Freeform Fabrication Symposium. University of Texas at Austin, 2018.
9. Li, Wilson, Thomas Poozhikala, and Mahmoud Dinar. "An Algorithm for Partitioning Objects Into a Cube Skeleton and Segmented Shell Covers for Parallelized Additive Manufacturing." International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Vol. 85376. American Society of Mechanical Engineers, 2021.
10. Lloyd, Stuart. "Least squares quantization in PCM." IEEE transactions on information theory 28.2 (1982): 129-137.
11. McPherson, Jace, and Wenchao Zhou. "A chunk-based slicer for cooperative 3D printing." Rapid Prototyping Journal (2018).
12. Nguyen-Van, Sy, and Kwan-Woong Gwak. "A two-nozzle cable-driven parallel robot for 3D printing building construction: path optimization and vibration analysis." The International Journal of Advanced Manufacturing Technology 120.5 (2022): 3325-3338.
13. Poudel, Laxmi, et al. "A heuristic scaling strategy for multi-robot cooperative three-dimensional printing." Journal of Computing and Information Science in Engineering 20.4 (2020).
14. Poudel, Laxmi, Wenchao Zhou, and Zhenghui Sha. "Computational Design of Scheduling Strategies for Multi-Robot Cooperative 3D Printing." International Design Engineering

- Technical Conferences and Computers and Information in Engineering Conference. Vol. 59179. American Society of Mechanical Engineers, 2019.
15. [Scikit-learn: Machine Learning in Python](#), Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
 16. "Siemens team unveils 3D printing spider-bots".
<https://www.theengineer.co.uk/content/news/siemens-team-unveils-3d-printing-spider-bots>.
Accessed June 2022.
 17. Wang, Chengcheng, et al. "Machine learning in additive manufacturing: State-of-the-art and perspectives." Additive Manufacturing 36 (2020): 101538.
 18. Wang, Tianjiao, et al. "In-situ droplet inspection and closed-loop control system using machine learning for liquid metal jet printing." Journal of manufacturing systems 47 (2018): 83-92
 19. Zhang, Xu, et al. "Large-scale 3D printing by a team of mobile robots." Automation in Construction 95 (2018): 98-106.