Proceedings of the ASME 2020
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
IDETC/CIE2020
August 16-19, 2020, St. Louis, MO, USA

IDETC2020-22711

ARCHITECTING THE COOPERATIVE 3D PRINTING SYSTEM

Laxmi Poudel, Lucas Galvan Marques, Robert Austin Williams, Zachary Hyden, Pablo Guerra, Oliver Luke Fowler, Stephen Joe Moquin, Zhenghui Sha[1], Wenchao Zhou[1]

Department of Mechanical Engineering
University of Arkansas, Fayetteville, AR 72701, USA

ABSTRACT

Cooperative 3D printing (C3DP) is a novel approach to additive manufacturing, where multiple mobile 3D printing robots work together cooperatively to print the desired part. At the core of C3DP lies the chunk-based printing strategy. This strategy splits the desired part into smaller chunks, and then the chunks are assigned and scheduled to be printed by individual printing robots. In our previous work, we presented various hardware and software components of C3DP, such as mobile 3D printers, chunk-based slicing, scheduling, and simulation. In this study, we present a fully integrated and functional C3DP platform with all necessary components, including chunker, slicer, scheduler, printing robots, build floor, and outline how they work in unison from a system-level perspective. To realize C3DP, new developments of both hardware and software are presented, including new chunking approaches, scalable scheduler for multiple robots, SCARA-based printing robots, a mobile platform for transporting printing robots, modular floor tiles, and a charging station for the mobile platform. Finally, we demonstrate the capability of the system using two case studies. In these demonstrations, a CAD model of a part is fed to the chunker, divided into smaller chunks, passed to the scheduler, and assigned and scheduled to be printed by the scheduler with a given number of robots. The slicer generates G-code for each of the chunks and combines G-code into one file for each robot. The simulator then uses the G-code generated by the slicer to generate animations for visualization purposes.

Keywords: Cooperative 3D printing, Swarm 3D Printing, Chunking, Multi-robot 3D printing

1. INTRODUCTION

While 3D printing has been making steady progress with increasing printing capability and decreasing cost, its further adoption has been limited by its lack of scalability in terms of both printing size and speed [1]. Researchers and academics have proposed different approaches to overcome these issues. Most of the efforts focus on increasing the size of the 3D printer itself, such as BAAM [2] and the large 3D printing system from the University of Maine [3]. This approach faces inherent challenges as the increase of printer size leads to a nonlinear increase of the printer cost, poses higher requirements of the accuracy of the motion systems (e.g., it is challenging to maintain a variation of less than 100 um in Z movement while the printhead moves across several meters in XY direction), and reduces the printing resolution. Some other approaches employ multiple printing extruders simultaneously to shorten the print time, such as the Autodesk's project Escher [4, 5]. Although this approach can speed up the process, the major limitation still rests in the scalability because there is a limit on the number of nozzles that can fit within the printer frame, which has since been discontinued by Autodesk.

Cooperative 3D printing is a novel approach to 3D printing that utilizes multiple mobile 3D printing robots to print a largescale part. In C3DP, a part is first divided into smaller chunks using a sloped-based chunking strategy and these chunks are assigned to individual robots for printing. C3DP does not require any post-processing, such as gluing the chunks, surface finishing, and etcetera[2]. Once the chunks are assigned, multiple robots will work together to print the part in less time compared to conventional 3D printers. Since printing is not constrained by a "box", C3DP can theoretically print parts as large as the factory floor size allows. Moreover, since printing is kept local and the

[1] Corresponding authors: zsha@uark.edu or zhouw@uark.edu
[2] The demonstration of the cooperative 3D printing method using two robots: https://www.youtube.com/watch?v=Ruw145U0Lpc

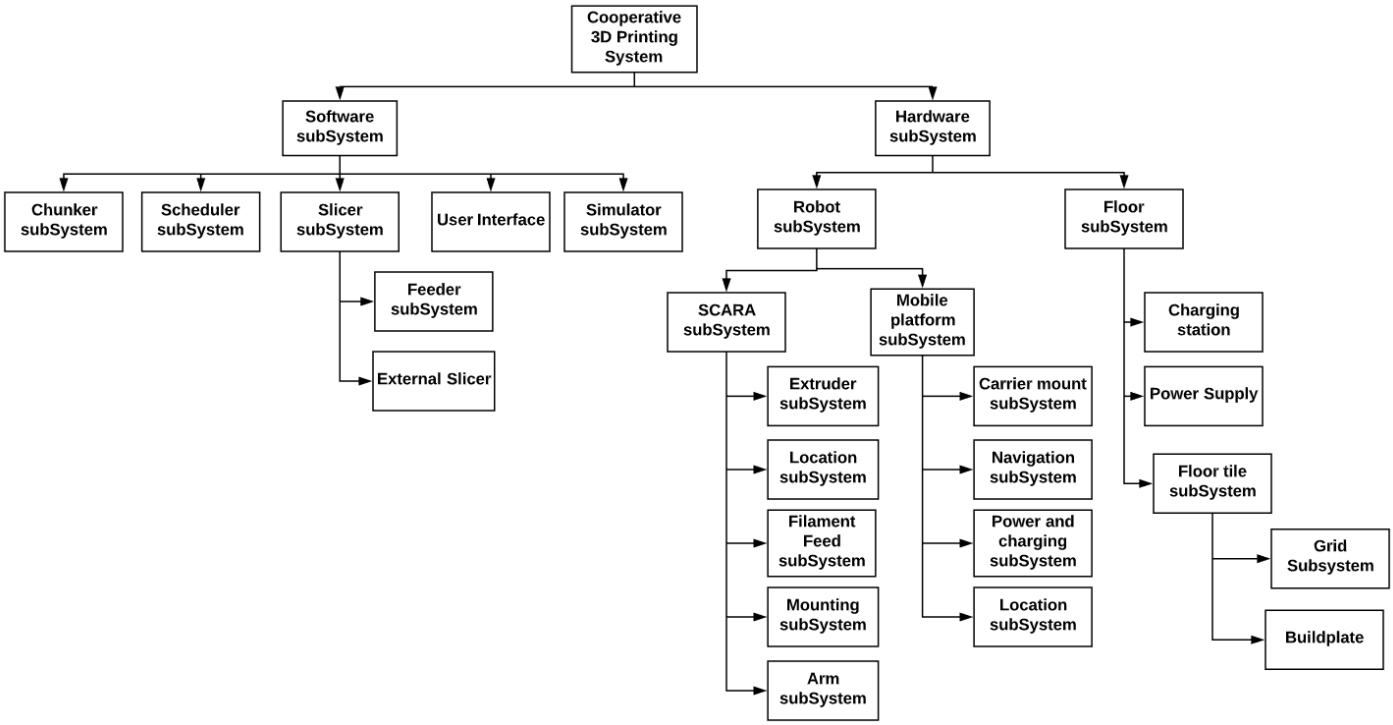

Figure 1. *The Physical system architecture of Cooperative 3D printing system*

nozzle size is regular, the print quality of C3DP can be as good as conventional desktop 3D printers.

In our previous studies, we have established experimental and theoretical foundations on which C3DP can be implemented. In terms of software, we have developed a chunk-based slicer and the impact of chunk-based printing on the mechanical strength of parts [6] [7]. While there exists many slicing software such a Cura [8], Slic3r [9], Skeinforge [10], they lack slicing capabilities to support multi-robot C3DP. Also, we have developed a scalable scheduling strategy for printing with multiple robots [11]. In that study, a framework is introduced to validate newly generated print schedules against geometric constraints for collision-free print. The valid schedules, described as a Directed Dependency Tree (DDT), can be then evaluated for estimating the total print time [12]. In terms of hardware, we demonstrated the first generation of mobile printer design in [13], consisting of four components: the mobile platform, the Z-stage, the main circuit, and the wireless communication system. This design was important in demonstrating the viability of mobile printing but was prone to positioning errors as the printer moved back and forth during printing. Thus, to avoid positioning errors and to make the entire hardware more robust, we have made a significant improvement to the previous design. The details of the new hardware design are presented in **Section 3**.

This paper presents the overall architecture of the C3DP system and our recent progress of integrating all the software and hardware development achieved so far, as shown in **Figure 1**. In this paper, we present how these components communicate with each other and what role each component plays as a part of the entire system. The main contributions of the study are summarized below:

1. The integration of different components of software and hardware that form a system for realizing cooperative 3D printing.
2. An enhanced C3DP software system that offers more chunking options, better slicing operations based on the open-source slicing engine, and a more robust simulator that can demonstrate new chunking strategies.
3. An upgraded C3DP hardware system which includes the new SCARA-arm printer as well as the build floor.

The paper is organized as follows. In **Section 2**, the architecture of cooperative 3D printing is presented focusing on the software components and how they are inter-connected. The new generation of the hardware platform is presented in **Section 3**. In **Section 4**, an explanation of the system (both hardware and the software) is provided along with the validation and implementation of the system. This includes validation using simulation and actual implementation of C3DP using a case study. Finally, **Section 5** includes the conclusion and future works about the project.

2. ARCHITECTURE OF THE C3DP SIMULATION

To realize C3DP, an integrated software system is required that can incorporate the process of chunking, scheduling, slicing, simulating, and printing all in one. However, there is no such system that has all these features for multi-robot C3DP. To

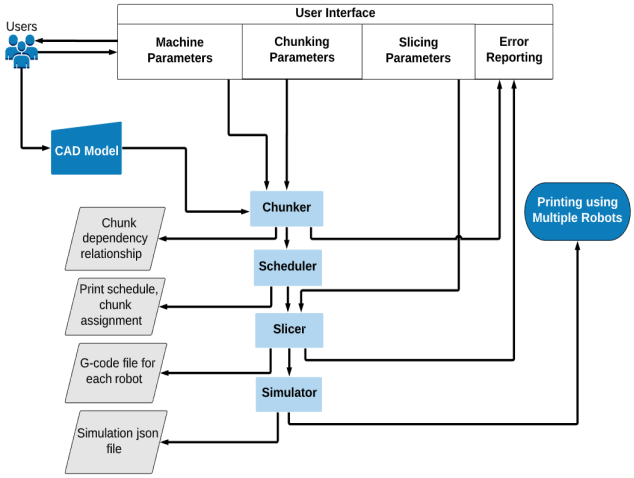

Figure 2: *Process flow in Cooperative 3D printing from a high level perspective*

address this gap, we have built a system that has integrated all these capabilities, and its architecture is presented in **Figure 2**. Once a user uploads a CAD model of the desired object, the part is first chunked by the chunker using the chunking parameters specified by the user. Then, the scheduler generates a print schedule of how the chunks will be printed in sequence and in parallel by considering chunk dependencies as well as the available resources, such as the number of robots available. Chunk assignment is done along with the print schedule in this step. After that, the slicer generates G-code for each of these chunks, and the simulator uses the generated G-code to animate the print schedule. In the following sections, we describe the methods and algorithms that enable each of these individual subsystems.

2.1 Chunker

The chunker takes the CAD model of the desired part as input and splits the CAD model into smaller chunks based on the parameters set by the user. In the current version of the chunker, the sloped-surface chunking strategy [7] is the only chunking strategy available, though, other chunking strategies such as striping method, concurrent printing, etc., are available in other collaborative printing platforms. Striping method follows a similar approach to the printing method adopted by Autodesk for project Escher [4] where each printhead is responsible for printing certain portions of the individual layer. The length of the path allocated to each extruder changes every layer such that the location of the gap (i.e., the spot at which one extruder stops and another begins) changes at every layer. Similarly, concurrent printing [5] assigns each G-code line to different printheads and checks for what lines can be printed together with the available number of printheads without colliding with each other. However, further studies of these chunking strategies need to be done, to ensure the mechanical properties are comparable to that of the traditional printed part, before implementing those chunking strategies in the C3DP system.

Once the desired object is uploaded to the system, the user can see an interface that might look like Figure 3. This UI helps the user choose different options for chunking parameters. The yellow rectangular bar highlights different parameters that the user can change. This includes: 1) build depth (the depth of chunk in the y-direction), 2) slope angle for chunking the part, 3) printhead depth of the robot (how far can the robot reach), 4) number of available robots. If the user does not want to specify the number of robots, the system can suggest the optimal number of robots based on the dimension of the part and the dimension of the robot. Similarly, the system also checks the value inputted by the user to make sure no constraints are violated. An example of such constraint could be the maximum value set for build depth or

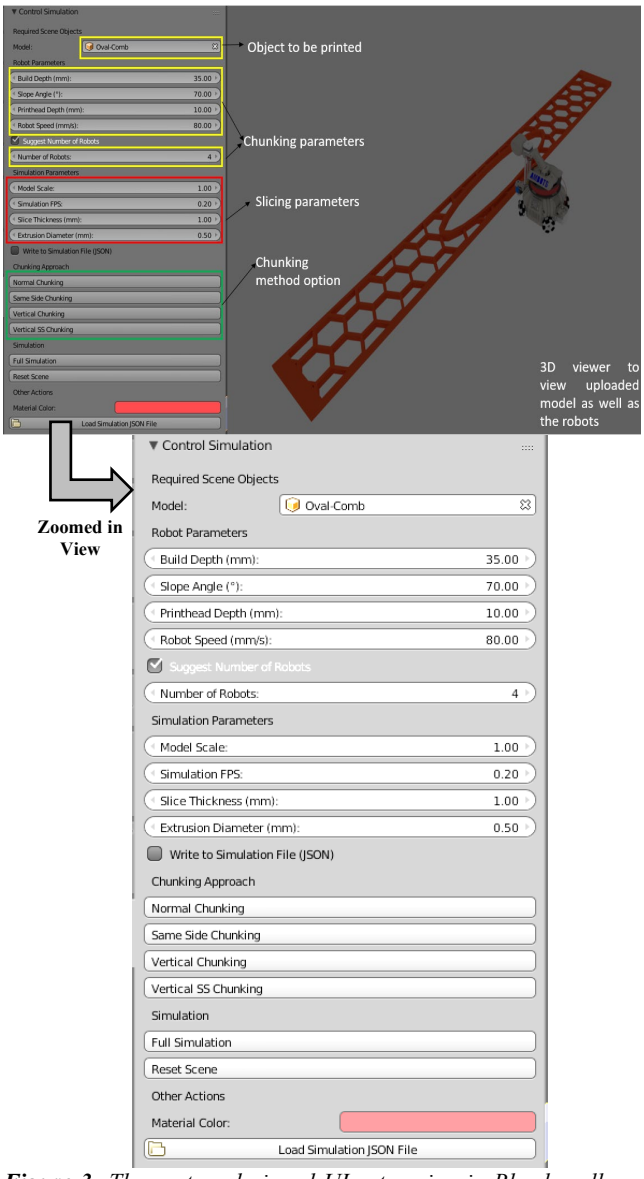

Figure 3*: The custom-designed UI extesnsion in Blender allows user to choose different chunking parameters as well as the slicing*

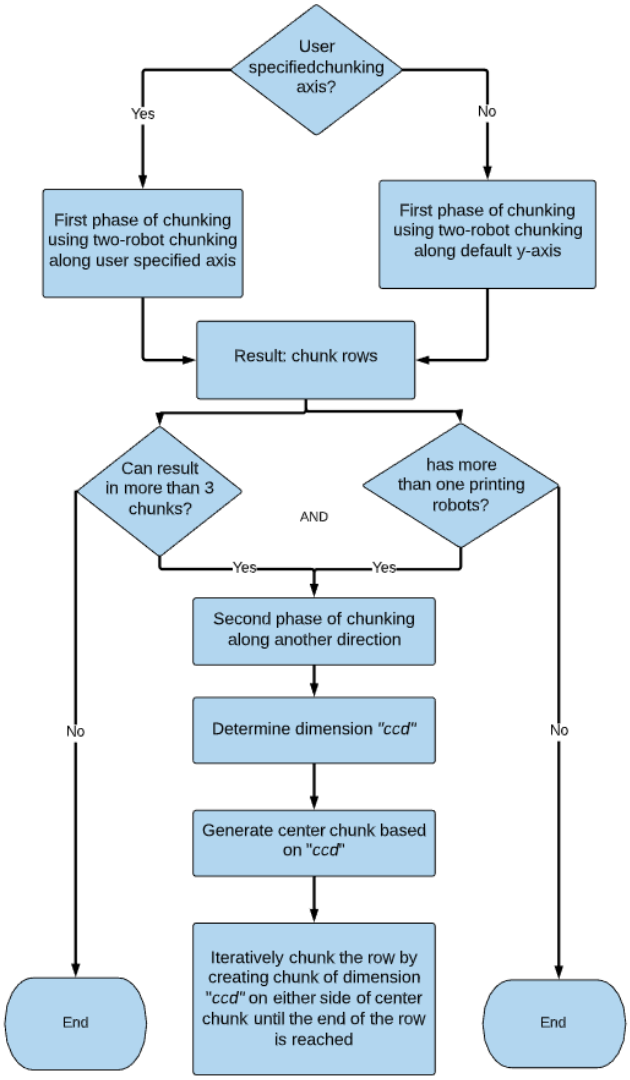

***Figure 4**:* The *flowchart demonstrating the divide and conquer chunking method*

printhead depth that cannot be exceeded to avoid unrealistic scenarios. In such a case, the system outputs error to let the user know that the constraint has been violated and the value needs to be altered. A user can also pick a different type of sloped-surface chunking strategy to chunk the object. More detailed information regarding selecting chunking parameters is presented in our previous study [6], which includes slope determination, chunking plane determination, etc. An enhanced version of the chunker in this paper includes chunking capability for multiple robots.

In this study, we present three additional chunking options. Each of the chunking options has its benefits for a specific type of situation. Below are some of the guidelines for choosing a proper chunking option based on resource availability, space availability, and the dimension of the desired part.

1. If the user only has two robots available for printing, previously developed two-robot sloped surface strategy is to be used.
2. If the user has more than two printing robots, the printing space is unlimited and, the dimension of the desired part is large in only two out of three dimensions, then divide and conquer strategy would provide the best outcome.
3. On the other hand, if the user faces the scenario 2, but the part is large in all three-dimension, vertical chunking with divide and conquer would provide the best outcome.
4. If the user has limited printing space, where the printer can only be placed on one side of the part, such a scenario would call for the same side chunking. For example, if the part starts at 0 and expands only in the positive x-direction and no robots can be placed beyond 0. In such situations, same-side chunking must be used.

Having outlined chunking options, we now describe what each of these different chunking options entails.

2.1.1 Divide and Conquer

This chunking option extends the two-robot chunking to multi-robot chunking, where the chunker first centers the object and then splits the object into multiple chunks along one direction with the sloped-surface using a bisecting algorithm as shown in **Figure 4**. In this strategy, the first phase of chunking takes place along the axis specified by the user. If the user does not specify the axis, the chunking is done along the default y-axis. If the second phase of chunking is required, which depends on the number of available robots and dimensions of the resulting chunk row, it takes place along the axis that is different from the axis of first phase chunking. The center chunk dimension or ccd depends on the reach of the robot and the slope angle chosen by the user. The dimension is calculated using the slope angle of the chunk and the reach of the printing robot as presented in **Equation (1).** Chunking planes are created on each side of the centerline such that they are ccd apart from the center chunk. The chunks are created by bisecting the part at these chunking planes. This process is iterative and takes place on both sides until the entire part is chunked as shown in **Figure 5**.

$$ccd = R + \tan\theta \times z_{height}, \quad (1)$$

where ccd is the center chunk dimension, R is the maximum reach of the robot, θ is the slope angle and z_{height} is the height of part in the z-direction.

The resulting number of chunks depends on the number of available robots and the desired size of the chunk and the size of the part itself. The width of the chunk must be large enough so that the hardware of two printers can fit while two robots are printing the chunks that are on either side of the chunk. Thus, the width of the chunk is

$$Chunk\_width = max(W_r, \frac{W_p}{num\_robots}), \quad (2)$$

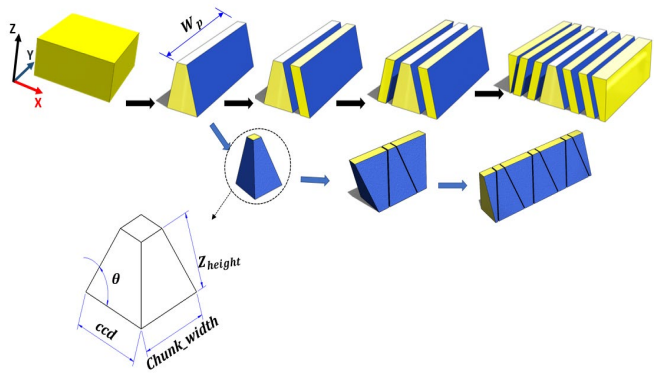

Figure 5: *Chunking process of a rectangular bar. First, the chunking takes place along direction (x-direction). Center chunk is created first and moves out in both direction in iterative manner to split the part further until the end of the part. Second phase chunking, shown in bottom, (only center shown for clarity). First, a center chunk is created. The row is then iteratively chunked along the longitudinal direction until the end of the chunk row is reached*

where W_r is twice the width of the hardware of the printing robot, W_p is the width of the part along a specified direction and, num_robots is the number of available robots for printing. Once the width of the chunk and number of chunks are determined, the chunking plane is created by iteratively spacing $chunk\_width$ from the centerline of the chunk row. The angle of the plane is alternated at every chunking plane as shown in **Figure 5** (using a dotted arrow) i.e., if a chunking plane is created at 45° first, the next chunking plane is created at 135° and so forth. Doing so allows us to create chunks with alternating slopes, which allow multiple robots to print simultaneously in each row

2.1.2 Same side chunking

Same side chunking applies to scenarios where the printing arms cannot be mounted on either side of the center row chunk. The same-side chunking uses a similar chunking method as described in the divide and conquer approach. The only difference lies in the location of origin chunk. The origin chunk is created at the center of the part along the axis in divide and conquer whereas, it is created at the end of the part in the same side chunking. Once the origin chunk is created, the chunking plane is iteratively shifted in one direction rather than two directions. The second phase of chunking takes place in the same manner as the divide and conquer approach. **Figure 6** presents the origin chunk in the same side chunking

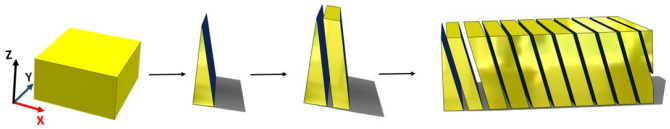

Figure 6: *Demonstration of how same-side chunking takes place in Chunker (The space between the chunks is shown for demonstration purpose only. No such space exists in reality)*

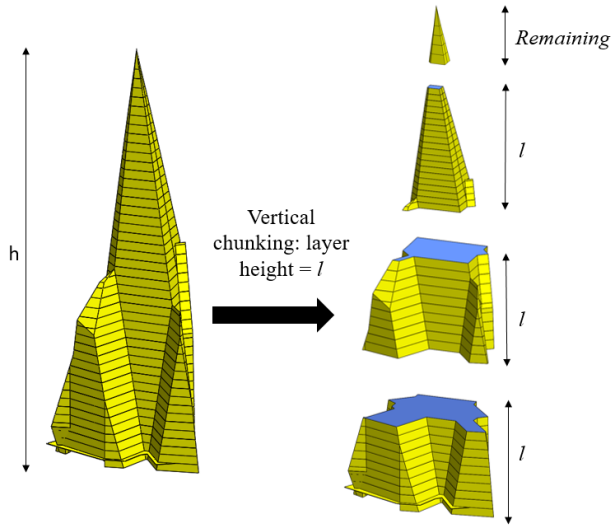

Figure 7*: Demonstration of vertical chunking in a tall cathedral that requires chunking in z-direction. The height of the entire object is h, which is divided into total of four vertical layers based on the layer height l*

and the iteration of the chunking plane along one direction until the end of the part is reached.

2.1.3 Vertical chunking

The above two chunking options allow us to chunk a part in the XY plane. But if the desired part is taller than the printer, the aforementioned chunking strategies would not sufficiently handle such printing scenarios. In such situations, we might need to chunk the part in the vertical direction as well. Allowing such chunking in vertical direction gives the user the option to divide a part that is much taller than the printing robot and print one layer at a time. To reach the subsequent layers that are taller than the printer, either the platform of the robot has to be raised or some sort of spacer can be designed that can hold the printing robot stable while printing. The logistics for such options are to be developed in the future.

In vertical chunking, once the object is uploaded, if the object is taller than 300 mm, the chunker automatically chunks the part in the vertical direction. Though this value is based on the largest z-height the current generation of the robots can reach, the user will have the option to change this default value as long as the inputted value does not exceed the maximum reach of the robot's reach i.e., they can use smaller value but nothing larger than $300mm$. Based on the input value for the layer height, the software will automatically calculate the number of vertical layers using **Equation (3)**. Each of the layers can be further divided into smaller chunks based on the input and the machine parameters using either of the chunking strategies outlined previously. As the chunks are created, the dependencies between chunks are generated. That means the printer should not print a layer above without printing the bottom layer first. The vertical chunking is demonstrated in **Figure 7**, where an architecture building of

height h, is divided into multiple vertical layers of height, l as defined by the user.

$$Num.of\ vertical\ layers\ =\ \frac{h}{l} \qquad (3)$$

2.2 Scheduler

Once the chunks are generated, the chunks are to be assigned to individual robots and scheduled for printing based on their dependency relations. The dependency relation is generated since adjacent chunks share the same sloped surfaces and the chunks with overhanging slope cannot be printed before their adjacent chunks that support the overhanging slopes, as shown in **Figure 8**.

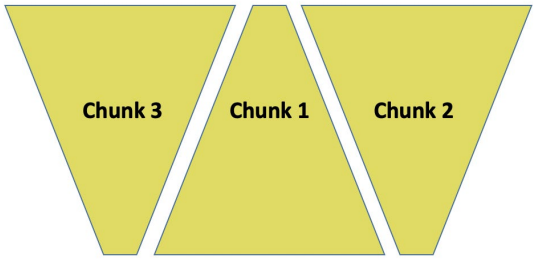

Figure 8: *Adjacent chunks with sloped surface. Chunk 1 has to be printed prior to printing chunk 2 and chunk 3 otherwise the print nozzle of the printer will collide with the overhang of chunk 2 and chunk 3*

The detail of how the chunk assignment takes place and how the dependent relationship is used for scheduling is presented in our previous study [6]. In this section, we discuss how the theoretical information such as scheduling strategy, presented in the previous paper is encoded into the software to achieve collision-free printing. For two-robot printing, a simple strategy is adopted where the origin chunk is assigned along with all the chunks on the left side to one robot and remaining chunks on the right side to the second robot [6]. As the chunking becomes more complicated, the complexity of the chunk assignment increases. The approach for the chunk assignment is presented in **Figure 9**. For simplification, all the robots are divided into two groups randomly, then the assignment approach used in the two-robot printing scenario is implemented. After that, the chunks in each group are assigned among the robots in their respective groups. If C_c represents center row chunks (example: chunk1 in **Figure 8**), C_L represents left row chunks (example: chunk3 in **Figure 8**) and C_R right row chunks (example: chunk2 in **Figure 8**) and the total available robots are randomly divided into two groups, G_A, and G_B. Then, all the chunks in C_c and C_L and assigned to G_A and rest to G_B. The assignment of chunks to a group is done based on human heuristics and might not be optimal.

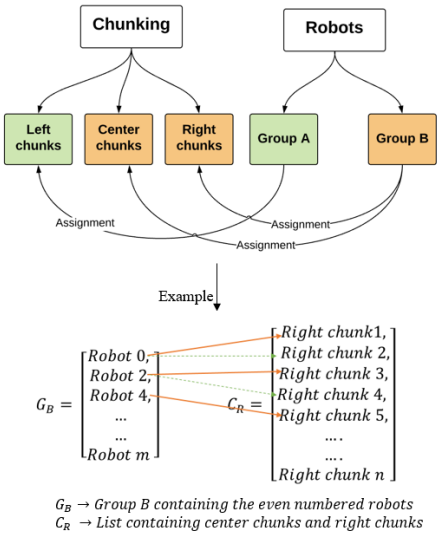

Figure 9 *Chunk Assignment of the robots in scaled-chunker method*

$$C_c = \begin{bmatrix} Center\ chunk1, Center\ chunk2, \\ Center\ chunk3, \ldots\ldots, Center\ chunk\ n \end{bmatrix}$$

$$C_L = [Left\ chunk1, Left\ chunk\ 2\ \ldots\ldots Left\ chunk\ n\ ]$$
$$C_R = [Right\ chunk1, Right\ chunk\ 2\ \ldots\ldots Right\ chunk\ n\ ]$$

$$G_A = [Robot\ 1, Robot3, \ldots\ldots Robot\ m-1\ ]$$
$$G_B = [Robot\ 0, Robot\ 2, \ldots\ldots Robot\ m\ ]$$

$G_A \rightarrow \{C_C, C_L\}$ (Group A is assigned to center chunks and left chunks)

$G_B \rightarrow \{C_R\}$ (Group B is assigned to right chunks)

Once the chunk assignment is completed, a print schedule is generated based on the chunk dependency following the rules below:

- The chunks with no dependencies are printed first.
- Once those chunks are printed, the chunks with already printed chunks as dependencies are chosen for printing. This process iterates till all the chunks are printed.

2.3 Slicer

In our previous study [6], we developed a simple custom-designed slicer that could generate toolpaths for all the printing robots based on the chunk assignment and schedule. To handle more complicated geometries with higher efficiency and robustness, we have decided to utilize a professional slicer (e.g., open-source slicer CuraEngine is chosen for this paper). However, none of the professional slicers support the collaboration of multiple robots. Fundamentally, for two robots to collaborate, they must be able to align in space and time. The spatial alignment is enabled by the positioning mechanisms embedded in our hardware platform as described in Section 3, the temporal alignment is realized with a pair of custom G-code command: "WAIT", "NOTIFY", which are used by robots to tell each other via wireless communication when they should pause and when they may proceed to execute the next line of G-code. A "WAIT" command is inserted at the beginning of any chunk that has chunk

dependencies, i.e., another chunk has to be printed before printing the current chunk. A "NOTIFY" command, on the other hand, is added at the end of the G-code of the chunk on which another chunk depends. This allows the robot to notify another robot that the dependency has been satisfied and it can go ahead with printing the chunk.

To utilize a professional slicer for cooperative 3D printing, we developed a *Feeder* system. The Feeder system imports individual STL models of the chunks output by Chunker one at a time to generate the G-code of each chunk using the professional slicer based on the slicer settings, such as infill density, print speed, layer height, etc. Once the G-code files for all the chunks assigned to an individual robot are generated, additional transitional information, such as telling robots move from one chunk to next chunk, or "WAIT" until another robot finishes printing certain chunk, or "NOTIFY" another robot and proceed with executing their next G-code line, are added between the G-code files of chunks based on the schedule output by the scheduler that is then integrated into one G-code file and sent to the robot. The entire process is demonstrated in **Figure 10**.

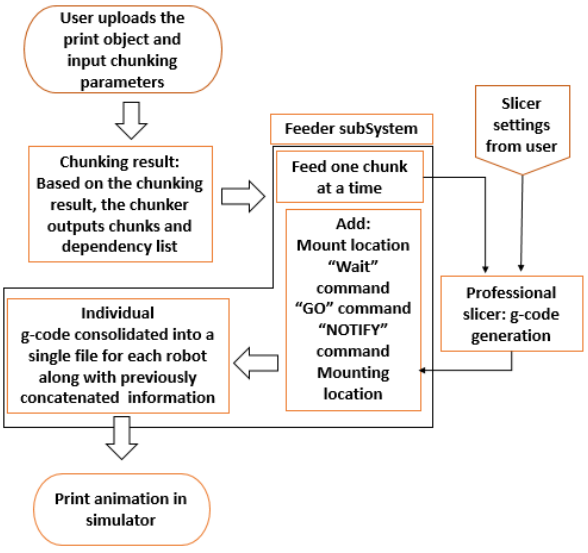

***Figure 10**. Once the part is uploaded and chunked, the feeder subsystem feeds one chunk at a time and send it to the slicer. Once the G-code is created, the other relevant information is added to the G-code. The G-code of all chunks of a robot are consolidated into one single file and used for visualization*

2.4 Simulator

The simulator presents a visualization of the G-code obtained after slicing. It animates the printing process and provides a visual aid of how the printing unfolds and how the final product looks. In our previous work, we developed a simulator based on a Blender environment to visualize such animation. The same simulator is used to provide a demonstration in this study as well. To generate visualization, frames are generated at the rate of 30 frames per second, i.e., one frame represents 1/30 of a second. Users can choose to change the number of frames per second in UI. Increasing the frame per second (fps) would result in better animation resolution but would require more computational resources and decreasing the fps would result in coarser animation but would be computationally extensive. Thus, the user can choose to either increase or decrease fps based on their desired outcome. A more detailed explanation of how the simulator works can be found in [6].

3. HARDWARE PLATFORM DEVELOPMENT

The hardware platform has gone through major updates since the previously presented first generation [13]. This current hardware platform consists of four main components:

1. An immobile SCARA 3D printer for printing.
2. A mobile platform that transports the SCARA printer from one location to another after the completion of a chunk printing.
3. A modularized floor tile system that assists the navigation of the mobile platform, allows SCARA printers to mount, and provides power supply and a charging station for the mobile platform.
4. A wireless network that coordinates the wireless communication between SCARA printers and mobile platforms.

In a typical printing process, the assembled G-code files from the *Feeder* system are sent to their assigned SCARA printers respectively, which will execute their assigned G-code file line by line. To begin printing, a custom G-code command "MOVE" calls the mobile platform to transport the SCARA printer to the print location on the floor. Once the mobile platform reaches the location of the SCARA printer, it notifies the printer to mount onto the platform and unmount from the floor tile. Once mounting and unmounting is complete, the platform carrying the printer maneuvers to its destination. While the printer is mounted onto the mobile platform, it is powered using the battery pack installed on the mobile platform. Once the destination is reached, the platform notifies the printer to start mounting into the floor and unmount from the mobile platform. After this, the SCARA printer will start to execute the rest of the G-code, until it hits a "WAIT" command. The "WAIT" command is inserted at the beginning of the chunk to notify the printer that to print the current chunk, its dependency has to be printed first. Once the dependency chunk is printed by another SCARA printer, it sends another custom G-code command "NOTIFY" to the printer that has been waiting for the chunk dependency to be printed. This notifies the SCARA printer to start printing the next chunk. To do so, the "MOVE" command is used again to call the mobile platform to the current location so that it can be transported to the mount location for next chunk printing. This process continues until all the chunks are printed. To simplify the matter, currently, only one SCARA printer can request the mobile platform at a time. If multiple printers request transportation, first in first out approach is used to pick the printer.

3.1 SCARA Printer

With the wide adoption of robotic arms in modern factories, we developed a robotic arm for C3DP for an easier transition in the future. In this study, we developed a SCARA (Selective Compliance Assembly Robot Arm) 3D printer for filament extrusion as shown in **Figure 11**.

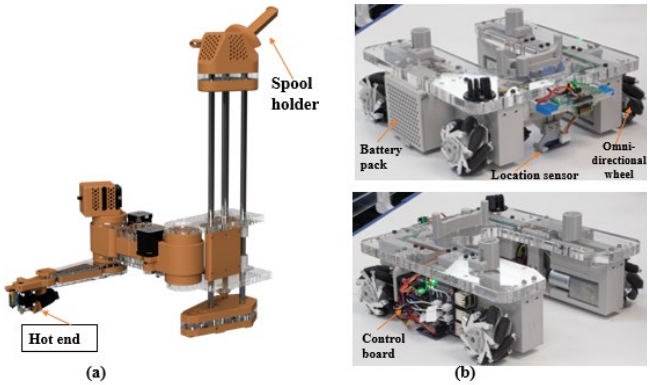

Figure 11. *(a) The updated 3D printer, equipped with scara-arm to increase the reach of the robot and mitigating the issues of the previous generation of mobile robot (b) The new mobile platform that transports scara arm from one print location to another*

The main functionalities of the SCARA printer are:

1. *Print the assigned chunks*: The SCARA printer is equipped with a single extruder and uses 1.75mm filament to print assigned chunks. The print specifications are presented in Table 1. The printer is equipped with three mounting leads that screw into the nuts installed in the mounting holes in the floor tile. While the printer is printing, it is mounted using those lead screws to prevent any vibration or movement. These lead screws also work as a connector to supply power to the printer from the power source installed in the floor tile. Such a locking mechanism allows the printer to minimize vibration as well as tipping over, which is a concern especially when the arm is fully extended.
2. *Communication with the mobile platform*: The SCARA printer is immobile but communicates with the mobile platform wirelessly if it needs to move from one location to another. The printers have two additional mounting pegs that are used to mount onto the mobile platform for secure transportation. These mounts also act as connectors to supply power from the battery pack of the mobile platform to printer during transportation. This along with the power source installed in the floor tile source ensures that the SCARA printers always have at least one source of power.

The maximum reach of the SCARA is much larger than the previous generation of printing robots. It can reach from $50\ mm$ to $350\ mm$, i.e., it cannot print anything that is closer than $50\ mm$ to its mounting location and can reach $350\ mm$ when it is fully extended. Currently, the auto-calibration system is being worked on using an auto bed leveling sensor. This calibration system will use a four-point calibration system, one for each corner of the build plate.

Table 1. *Technical parameters of the new 3D printing SCARA printer*

XY reach	50mm-350mm
Max Z-height	300 mm
Filament Feed	Bowden, 1.75mm
Nozzle	Single extruder
Maximum Temperature	295° C
Hot End	Single extruder
X/Y Motion	2 axis SCARA
Z motion	300 mm guide motion driven by a lead screw
Layer resolution	10 µm
Print Speed	50 mm/s
Print repeatability	5 µm
Power Input	build floor, battery-pack via a mobile platform
Power consumption	78.62 W
Software	compatible with open source software
Connectivity	Wireless

3.2 Mobile Platform

Because the printing robots spend most of the time printing and little time moving between chunks, we separate the mobile platform from the SCARA printer. This new design (see **Figure 11(b)**), on one hand, enhances the stability of the SCARA printers in printing, and on the other hand, reduces the overall cost by allowing multiple SCARA printers to share mobile platforms.

The main functionalities of the mobile platform are:

1. *Navigate on the floor*: The mobile platform is equipped with mecanum wheels due to its omnidirectional property for maneuverability. To maneuver from one location to another accurately, the platform is also equipped with infrared sensors. These sensors are used to align the mobile platform with the navigation lines on the floor. In addition to the sensors, it is also equipped with a camera that can utilize computer vision to verify the positional accuracy. The camera read the barcode on the floor tile to get accurate positional information and adds a layer of position verification. The barcode ensures that the platform moves to the desired location, instead of a different location by mistake. If that happens, there will be a discrepancy between the the positional information provided by the barcode and the actual destination that is assigned to the mobile platform, raising a warning flag. Thus, this information in conjunction with the positional movement that is tracked by the encoder in the

motors of the mobile platform is used to rectify any positional discrepancies that might occur due to some unforeseen mishaps.

2. *Carry the SCARA printer and power supply*: The mobile platform is responsible for transporting the printing arm from one location to another. While doing so it needs to ensure that the arm is safely transported. To do that, it is equipped with two mounting pegs with holes in the center on the top. Once the SCARA printer's location is reached, the printer is notified wirelessly, and the printer starts lowering the leads to mounting into the holes of the platform. These holes also act as a female connector that connects with the male connector of the printer to supply power to the printer so that the printer can mount and unmount while it is not in connection with the floor tile. Once the printer is securely mounted, it unmounts from the floor tile and is transported to its next location. Once the location is reached, the above-described process takes place in reverse order until the printer is securely mounted into the floor.
3. *Autonomous charging*: The mobile platform is equipped with a rechargeable battery pack that has a life cycle of 2 hours of continuous use and idle time of 6 hours. But if it is not being used for more than 5 minutes, it automatically retreats to a charging station located in one of the corners on the build floor.

Once a printing arm receives G-code instruction, it sends a signal to the mobile platform. The mobile platform will move to the location where the printing arm is at currently and locks into the arm. At this point, the arm disengages from the build floor and the mobile platform carries the printing arm to the next print location as instructed in the G-code. Once it reaches the new location, the arm starts engaging with the build floor using the mounting holes and the mobile platform unlocks itself from the arm and moves back either to its charging location or the location of another print arm. **Table 2** provides the specifications of the mobile platform.

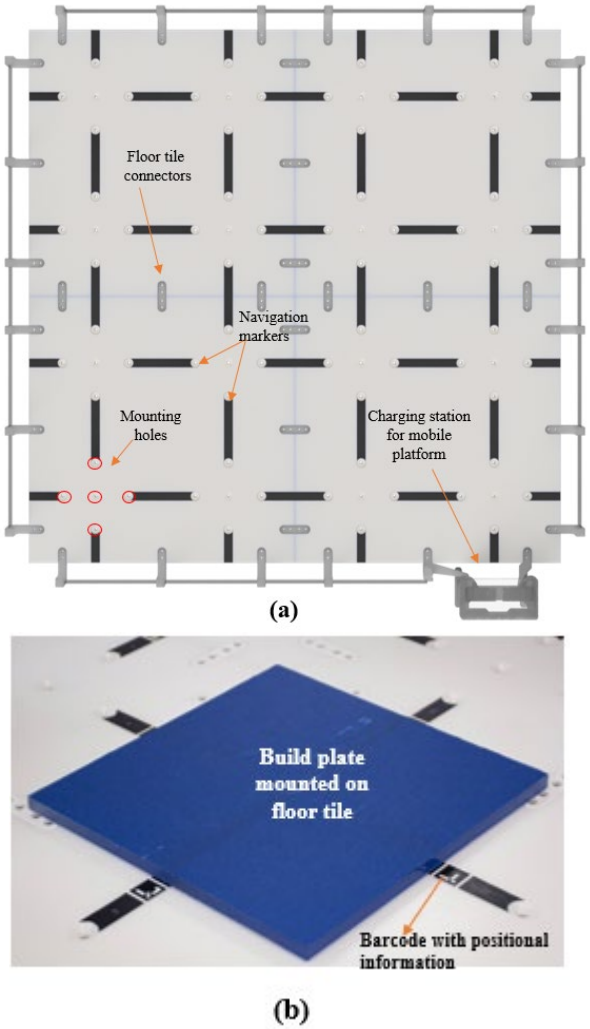

Figure 12: *(a) Floor tile with build plate. (b) Build plate*

Table 2. *Specifications of the mobile platform*

Wheels	Mecanum wheels, omnidirectional
Travel Speed	60 mm/s
Navigation	Infrared sensors, camera
Communication	Wireless communication
Power Source	Battery pack, charging station
Battery life	2 hours use, 6 hours idle
Power Consumption	78 Watts
Product dimension	25 cm×33 cm×9 cm

3.3 Floor Tile

A modularized floor tile system is developed to provide a base for the C3DP platform. It consists of modularized floor tiles and connectors between tiles for scaling to a size as large as the factory size. Each floor tile is 600 mm × 600 mm with 4 mounting locations spaced at 300 mm apart in both X and Y directions, as illustrated in **Figure 12** The modular floor tiles have a slot on each side which is used to align two-floor tiles together. A connector that sits flush in this slot can be used to attach floor tiles with high accuracy. The floor tile system provides the following functionalities:

1. *Mounting and powering the SCARA 3D printers*: Each of the four mounting locations consists of five mounting holes. Nuts are installed inside these mounting holes so that the lead screws in the SCARA arm can mount into them. To auto-correct the slight misalignment sometimes due to tolerances stacking, each mounting hole has countersink on the top surface. Besides, the floor tile system is equipped with electrified stainless-steel conductive strips of alternating polarity on the bottom side of the floor tile. Once the leads of the SCARA printer mount into the mounting holes, they come

in contact with the conductive strips, which transfers power to the printing arm. A rectifier circuit is used to sort the polarity of the current to ensure that current flows are in the correct direction regardless of the polarity of active strips. This powers the SCARA printer during the entirety of the printing and allows us to avoid the use of wires for power.

2. *Assist navigation of mobile platform*: The floor tile is equipped with navigation lines that are of black colored markers to contrast with the floor tile. These navigation lines are followed by the infrared sensors to maneuver from one mount location to another. In addition to having navigation guides, the floor tile also includes an embedded barcode system to provide positional information. This barcode includes positional information such as to what row and column do it belong. This is used by the camera installed on the mobile platform to get positional awareness.
3. *Provide a charging station for the mobile platform*: The floor tile also is equipped with a charging station that is mounted onto the edge of the tile. This charging station resembles a cart storage area outside a supermarket, where a mobile platform can park itself into. Once the platform is aligned, a connection is formed between the charging station and the platform initiating the charging of the platform.
4. *Provide a base to mount build plate*: Build plate is a 300 mm × 300 mm modular square block where a part is printed on. It is mounted on the build floor with a base that has four conical pegs resembling countersink tools. Having such pegs helps align the build plates at the correct location and also prevents the movement of build plate in the XY plane.

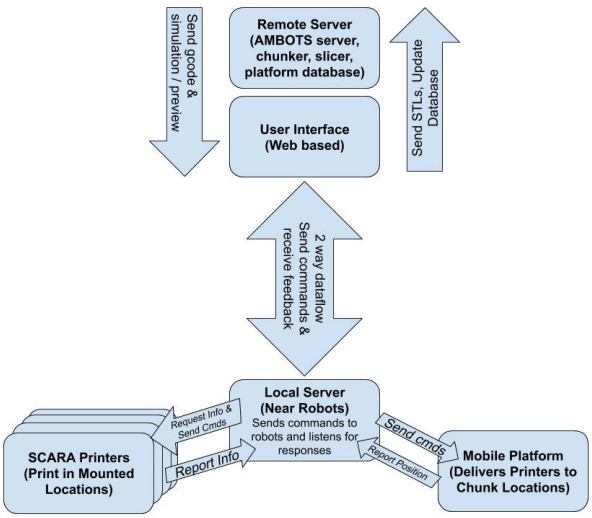

Figure 13. *The information flow in Cooperative 3D printing system between different components*

3.4 Wireless Network

The communication between the different components of the hardware takes place via a wireless communication network. Both the SCARA printer and the mobile platform are equipped with a wireless communicator. Both of these components are connected to a local server via the wireless network. The local server collects all the pertinent information required for planning and coordination from different components of the system to implement cooperative 3D printing. This local server takes information from the software components such as G-code information for each SCARA printer, which is then transmitted to the individual arms wirelessly. This flow of information between the different components of the C3DP is presented in **Figure 13**.

4. VALIDATION OF THE SYSTEM

To validate and demonstrate the capability of the developed C3DP system, we have conducted two case studies with two objects. The first object has relatively simpler geometry but is largely aiming to demonstrate the system's scalability. The second object is to demonstrate the capability of the software in handling more complicated geometries.

Object I: A bar with honeycomb design

The first print object is a rectangular bar with a honeycomb internal structure. Its dimension is $844mm \times 90mm \times 12mm$ (see **Figure 14**). The chunking parameters used are presented in **Table 3**.

Figure 14. *The 3D model of rectangular bar with honeycomb design along with specified dimension used for C3DP print simulation using 6 robots*

Table 3 *important chunking and slicing information used for first chunking and then slicing the rectangular bar*

Parameters	Value (units)
Number of robots	6
Slope Angle	70 degrees
Build depth	35 mm
Printhead depth	30 mm
Chunking Option	Divide and conquer

In total 18 chunks are obtained from chunking, and the chunks are shown in **Figure 15 (a).** The Chunker also outputs the chunk dependency in JSON format, which is then used to generate a print schedule with a chunk assignment. Both print schedules and chunk assignments are outputted in JSON file by Scheduler. The slicer then generates G-code for one chunk at a time, adds other pertinent information related to the position and dependency. All the chunks assigned to the same robot are processed together and a single G-code file is outputted. This G-code file contains printing commands as well as "WAIT" and "NOTIFY" command to implement dependency between chunks. **Figure 15 (b)** shows the different views of the part obtained after slicing. The different color of the chunks does not represent the actual color of the chunks to be printed but is used to differentiate a chunk from adjacent chunks.

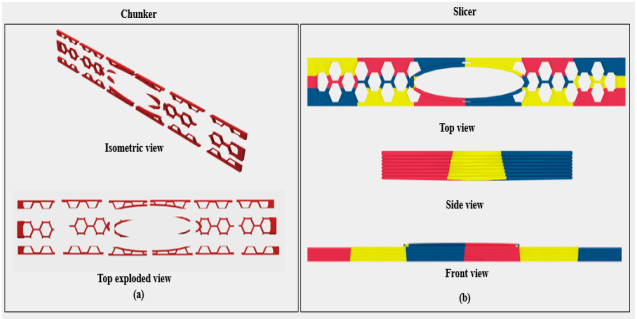

Figure 15. *The output of the Chunker and Slicer (a) Output of the Chunker containing 18 chunks (top view (bottom), Isometric view (top)) (b) Output of the Slicer (top view, side view, lateral view*

Once the G-code files for all the robots are created, the printing is simulated. The Simulator visualizes the print sequence and the entire print process. The screenshots of the simulation at different time steps are presented in **Figure 16**. First, the chunks that do not have any dependencies are printed (three chunks in the center row as shown in **Figure 16(a)).** Once those are complete, the same robots move over to print the gap chunks as shown in **Figure 16(b).** After finishing the center row, half the robot moves to one side of the row and the other half moves to another half and starts working on the chunks as shown in **Figure 16(c)** and **(d)** until the part is completed.

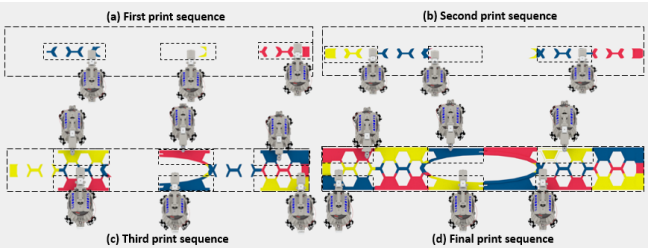

Figure 16. *Snapshots of multi-robot printing at different time stamps. The small dashed boxes represent chunks that are printed during the print sequence and the larger rectangular shows the outline of the entire part.*

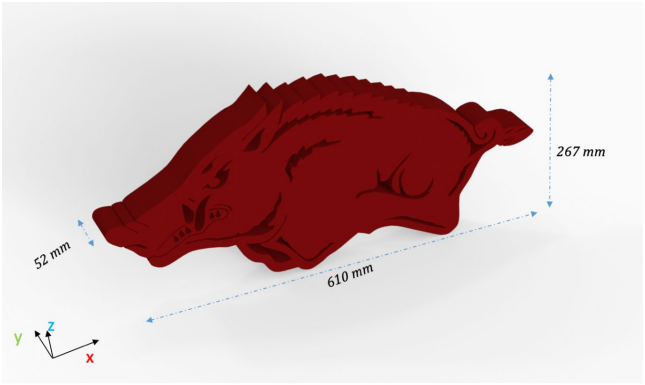

Figure 17. *The Razorback cutout with specified dimension used for C3DP print simulation using 6 robots*

Object II: A Razorback Cutout

Figure 18. *Chunking output and slicing output for*

The second object is a Razorback cutout shown in **Figure 17**. It has a dimension of $610mm \times 267mm \times 52mm$. The chunking and slicing parameter used in this case study in listed **Table 4**. The output of the Chunker for the Razorback cutout is presented in **Figure 18**. The chunking parameters resulted in *24* chunks. Once the chunking is complete, the schedule will be generated based on the chunk dependencies. The Slicer generates a G-code of different chunks to be printed individual robot, one at a time and outputs the G-code files. The output of slicer is presented in **Figure 18 (b),** and the snapshots of the simulation are presented in **Figure 19**.

Table 4 important chunking and slicing information used for first chunking and then slicing the Razorback cutout

Parameters	Value (units)
Number of robots	4
Slope Angle	70(degrees)
Build depth	70 mm
Printhead depth	55 mm
Chunking Option	Divide and conquer

5. CONCLUSION AND FUTURE WORK

In this paper, we presented an integrated C3DP platform with all the necessary software and hardware components. The software component includes five critical subsystems. A chunker divides a part into small chunks with different options, such as divide and conquer, same side chunking, and vertical chunking.

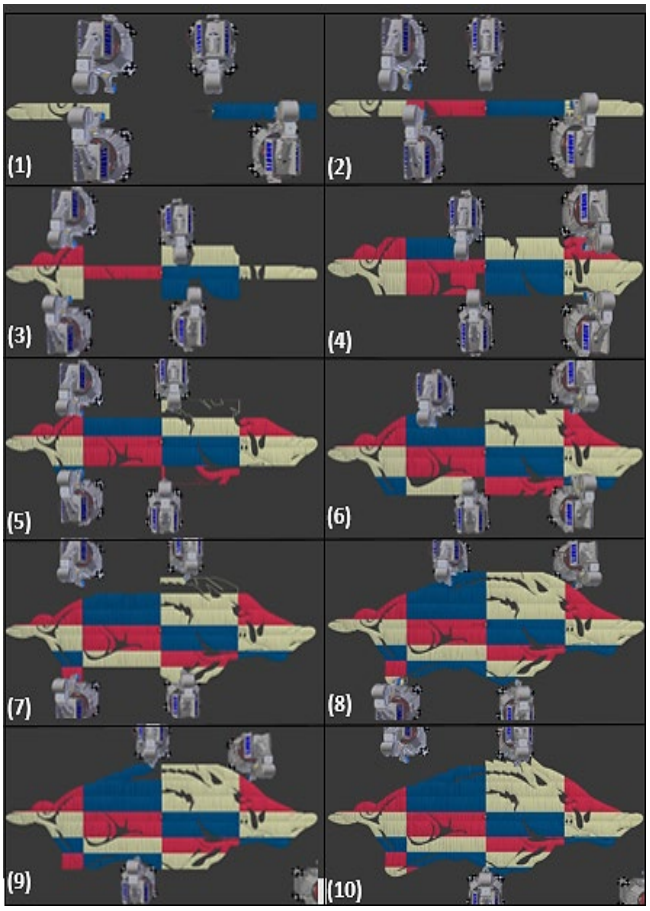

Figure 19: Snapshots of multi-robot print simulation at different time steps.

Users can choose the most appropriate strategy based on parts' geometries and their preferences. The scheduler takes the dependencies between the chunks, which is generated during the chunking process, to generate print schedules. The scheduler also automatically assigns chunks based on the number of available robots. The current C3DP system adopted a more robust slicer, called Cura, and we develop a feeder system to adapt it to the C3DP environment. The same feeder adds pertinent information that includes but not limited to mounting location for a printer, "WAIT" and "NOTIFY" command to enforce the print schedule, etc. The simulator generates an amination based on the generated G-code.

The hardware includes a new SCARA printer and a separate mobile platform for printer transportation. Also, a new modular floor tile is developed that serves as a power source for the SCARA printer and provides positional information to the mobile platform. Finally, to validate the system, two case studies are conducted: a rectangular bar and a Razorback cutout, where all the five components of the software system are tested and verified. The major contributions of this study are in three aspects:

1. This is the first time a fully integrated and functional C3DP platform is presented, which makes the promise of autonomous manufacturing using multiple mobile robots for manufacturing more tangible.
2. We developed new chunking strategies that can be used to subdivide a part into smaller chunks and utilizes multiple mobile robots to print them cooperatively.
3. We successfully adapted a conventional slicer into the C3DP environment.

The demonstration presented in this paper provides a blueprint for how the swarm manufacturing can move forward. Although the paper mainly focuses on 3D printing, the application of the approach can easily be expanded to other manufacturing approaches such as machining, welding, laser cutting. Moreover, heterogeneous tasks can be realized by our system using a swarm of heterogeneous robots equipped with different manufacturing capabilities. Such an approach has the potential to revolutionize the manufacturing industry and propel it to the next level required for sustainability. The actual implementation of C3DP using the hardware platform is underway and will be reported soon in our future studies.

6. ACKNOWLEDGMENT

This project is supported by the National Science Foundation (NSF) Division of IIP through Grant #1914249 and the commercialization fund through The Office of Vice Chancellor for Economic Development from the University of Arkansas.

REFERENCES

[1] J. Go, S. N. Schiffres, A. G. Stevens, and A. J. Hart, "Rate limits of additive manufacturing by fused filament fabrication and guidelines for high-throughput system design," *Addit. Manuf.*, vol. 16, pp. 1–11, 2017.

[2] L. J. Love, "Utility of Big Area Additive Manufacturing (BAAM) For The Rapid Manufacture of Customized Electric Vehicles." United States. Dept. of Energy. Office of Energy Efficiency and Renewable Energy ;, Washington, D.C., 2015.

[3] D. Kr and V. Artemov, "(12) Patent Application Publication (10) Pub. No.: US 2012/0184582 A1," vol. 1, no. 19, 2012.

[4] Autodesk, *Project Escher*. 2016.

[5] Y. Jin, H. A. Pierson, and H. Liao, "Toolpath allocation and scheduling for concurrent fused filament fabrication with multiple extruders," *IISE Trans.*, vol. 51, no. 2, pp. 192–208, 2019.

[6] J. McPherson and W. Zhou, "A Chunk-based Slicer for Cooperative 3D Printing," *From Rapid Prototyp. J.*, vol. 24, no. 9, pp. 1436–1446, 2018.

[7] L. Poudel, Z. Sha, and W. Zhou, "Mechanical strength of chunk-based printed parts for cooperative 3D printing," in *Procedia Manufacturing*, 2018, vol. 26, pp. 962–972.

[8] Ultimaker, "Cura 3D printing slicing software," 2012.

[9] A. Ranellucci, "Reprap/Slic3r and the future of 3D printing," *Canessa, E., Fonda, C., Zennaro, ed.,"Low-cost 3D Print. Sci. Educ. Sustain. Dev.*, pp. 75–82, 2013.

[10] S. Lefebvre and L. N. Grand-Est, “IceSL: A GPU accelerated CSG modeler and slicer,” in *18th European Forum on Additive Manufacturing (AEFA’13)*, 2013.
[11] L. Poudel, C. Bair, J. McPherson, Z. Sha, and W. Zhou, “A Heuristic Based Scaling Strategy For Coperative 3D Printing,” *ASME. J. Comput. Inf. Sci. Eng.*, 2019.
[12] L. Poudel, Z. Sha, and W. Zhou, “Computational Design of Scheduling Strategies for Multi-Robot Cooperative 3D Printing,” *Idetc/Cie*. Anahiem, CA, 2019.
[13] L. G. Marques, R. A. Williams, and W. Zhou, “A Mobile 3D Printer for Cooperative 3D Printing.”