The Thesis Committee for Cole Mensch
Certifies that this is the approved version of the following Thesis:

Physical Validation and Monitoring of Cooperative 3D Printing

APPROVED BY
SUPERVISING COMMITTEE:

Zhenghui Sha, Supervisor

Junmin Wang

Richard Crawford

Physical Validation and Monitoring of Cooperative 3D Printing

by

Cole Mensch

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Mechanical Engineering

The University of Texas at Austin

May 2024

Acknowledgements

There are many people I have to thank for getting me to this point in my academic career, but I will start by thanking my research advisor Dr. Zhenghui Sha. He has been an incredible guide through my two years of graduate studies, and I definitely wouldn’t be where I am today without his feedback and support. I also have Dr. Sha to thank for introducing me to the many amazing people in the SiDi lab, all of whom I’ve had the pleasure of becoming friends with over the last year and a half. In addition, I’d like to thank Dr. Wenchao Zhou from the University of Arkansas for consistently providing thoughtful feedback and assistance with my research. I must also thank Dr. Richard Crawford and Dr. Junmin Wang for agreeing to evaluate my work as members of my supervising committee.

My family has been incredibly supportive of my academic journey throughout my life, and I’d be remiss if I didn’t take a moment to thank them as well. Specifically, I must thank my parents, Heather Dayton and Stephen Mensch, for consistently encouraging me to follow my dreams and strive for greatness. I would also like to thank my brothers, Bradyn, Nolan, and Heath, as well as my sister, Harlow, for always being there for me and pushing me to do my best.

Lastly, I’d like to express my gratitude to the University of Texas for an amazing six years at the Forty Acres. My years here have been some of the best of my life, and the friendships and memories that I have made in the Cockrell School of Engineering and the Texas Ice Hockey team will last forever. I am a better, smarter, more capable person now than when I was initially accepted into the University of Texas back in 2018, and for that I am immensely grateful. Finally, I would like to thank the National Science Foundation (NSF) for their generous support through Grant #2112009 via AMBOTS Inc, and the

Walker Department of Mechanical Engineering for supporting my education via teaching assistant opportunities.

Abstract

Physical Validation and Monitoring of Cooperative 3D Printing

Cole Mensch, M.S. in Mechanical Engineering

The University of Texas at Austin, 2024

Supervisor: Zhenghui Sha

Swarm Manufacturing (SM) is a novel manufacturing paradigm that utilizes a swarm of mobile robots to complete manufacturing tasks in a factory setting. Cooperative 3D Printing (C3DP) is a rudimentary form of SM that utilizes mobile 3D printing robots working in tandem to print large objects. Several parts of the C3DP pipeline have been explored in previous work, such as geometric partitioning, scheduling, path planning, and placement. While this pipeline has been thoroughly developed, gaps still exist regarding the validation and monitoring of various steps along the process in order to ensure the computational pipeline can be operated safely in the real world. This thesis proposes a physical validation study and process monitoring integration for the emerging additive manufacturing (AM) technology of cooperative 3D printing (C3DP).

To address these research gaps, we pose a few research questions that are explored over the course of this thesis: 1) Can the existing placement and scheduling algorithms be physically validated? If so, to what extent can they be validated and what changes should be made to these algorithms to improve their functionality when being physically implemented? 2) How can the C3DP process be monitored to further improve its implementation and account for uncertainties and errors in manufacturing?

To answer the first question, this thesis presents a physical validation study of the previously developed placement optimization algorithm. This is done using two test cases which are analyzed to determine what changes must be made to the algorithm to make it better represent the physical C3DP platform. The second question will be addressed with a thorough review of various process monitoring techniques for both object tracking and 3D printing error detection. The latter will be expanded further, with both 2D and 3D vision techniques being tested and integrated to detect errors in parts manufactured using C3DP.

Between these two studies, numerous gaps in the C3DP process will be filled in, specifically relating to the autonomy of the physical system. In doing so, this thesis expands on the concept of SM, and brings more flexibility to the manufacturing field as a whole.

Table of Contents

List of Tables 10

List of Figures 11

Chapter 1: Introduction 14

1.1 Motivation 14

1.2 Cooperative 3D Printing 16

1.2.1 Hardware 17

1.2.2 Software 20

1.3 Research Questions 25

1.4 Thesis Roadmap 26

Chapter 2: Physical Validation of Job Placement Optimizer for C3DP 27

2.1 Background and Motivation 27

2.2 Methodology and Test Cases 28

2.2.1 Test Case: Small Box 30

2.2.2 Test Case: 3DBenchy 31

2.3 Results 32

2.4 Discussion 35

2.4.1 Movement Actions 36

2.4.2 Printing Actions 38

2.5 Future Work 40

Chapter 3: Process Monitoring for C3DP 42

3.1 Background and Literature Review 42

3.1.1 System Level Monitoring 42

3.1.2 Task Level Monitoring 44

3.2 3D Vision 46

3.2.1 Methodology 48

3.2.2 Results 51

3.2.3 Discussion 54

3.3 2D Vision 56

3.3.1 Methodology 57

3.3.2 Preliminary Results 61

3.3.3 Discussion and Future Work 64

Chapter 4: Conclusions and Future Work 67

Appendix 70

All Publications and Posters Published, Submitted, and Planned 70

References 71

List of Tables

Table 1: Technical Specifications of SCARA Printing Robots [36] 17

Table 2: Technical Specifications of Mobile Transporter Robots [36] 18

Table 3: Total Makespans of Examined Test Cases 34

List of Figures

Figure 1: Swarm manufacturing vision [9]. 15
Figure 2: The Cooperative 3D Printing platform [11]. (i) SCARA based 3D printing platform, (ii) Mobile transporter, and (iii) Mountable build plate. 19
Figure 3: Flow chart of the C3DP process. 21
Figure 4: Sloped chunking and slicing [17]. Chunk 2 cannot be printed until chunk 1 has been printed. 23
Figure 5: Chunking of a tall box object [11]. 24
Figure 6: Flow chart of physical validation process. 29
Figure 7: Small box test case. (a) Full model, (b) Chunk configuration for all jobs (they are identical) with print times and dependencies, notated with red arrows. 31
Figure 8: 3DBenchy test case. (a) Full model, (b-d) Chunk configurations for job 0-2 respectively with print times and dependencies, notated with red arrows. 32
Figure 9: Genetic algorithm placement results. 33
Figure 10: Initial C3DP testing setup. 34
Figure 11: Assembled model of 3DBenchy test tase (orange: Printer 1, black: Printer 2, gray: Printer 3). 35
Figure 12: Movement time testing results. Left: Small box, Right: 3DBenchy. 38
Figure 13: Print time testing results. Left: Small box, Right: 3DBenchy. 40
Figure 14: 3-Robot cooperative print of castle demonstrating common FDM printing errors. 45
Figure 15: Intel Realsense D435 depth camera [42]. 49

Figure 16: Intel Realsense calibration. Left: Full calibration setup, Top Right: Object detection with a simple box. Bottom Right: Object detection of complex shape. 50
Figure 17: Azure Kinect DK. Left: Camera [45], Right: Experimental calibration setup. 51
Figure 18: Test case model of UT Tower top in orange PLA filament. 52
Figure 19: Pointcloud of UT Tower top. Left: Intel Realsense D435, Middle/Right: Azure Kinect. 53
Figure 20: Meshes generated with CON model. Left: D435, Right: Azure Kinect. 54
Figure 21: Mounts for use in object tracking. Left: Overhead camera mount, Right: ArUco marker nozzle mount. 58
Figure 22: Side profile camera mount. 60
Figure 23: Flow chart of process monitoring framework. 61
Figure 24: Overhead camera feed during nozzle tracking. 62
Figure 25: Detection of stringing in 3D printed parts with different filament colors and backgrounds. 63
Figure 26: Masks of 3DBenchy. Left: Mask generated from camera image, Middle: Mask generated from correct STL file, Right: Mask generated from altered STL file. 64

Chapter 1: Introduction

1.1 Motivation

The low cost and high adaptability of additive manufacturing (AM) make it an ideal tool for rapid prototyping and iterative design. It has specifically garnered attention in medical [1] and aerospace applications [2] due to its ability to build complex, highly customized parts. However, a persistent drawback of traditional AM systems is their lack of scalability, both regarding manufacturing speed [3] as well as the size of objects that can be printed [4]. Efforts have been made to upscale the commonly used gantry or robot arm techniques, such as big area additive manufacturing (BAAM) [5] or using multiple nozzles in tandem [6]. However, drawbacks persist with these technologies regarding print resolution and flexibility [7], two issues that must be addressed for AM to be capable of competing with traditional manufacturing methods.

One proposed solution to address these gaps is Swarm Manufacturing (SM). This is a new paradigm for distributed manufacturing, in which a factory employs a swarm of mobile robots to complete manufacturing tasks. These robots can be customized to fulfill different roles within the factory, allowing for flexibility as the factory can be easily reconfigured to meet any changing requirements such as product dimensions, production capacity, or material selection [8]. The concept of swarm manufacturing can be traced back to nature, as many insects such as bees and ants exhibit swarm-like behavior as they cooperate to collect resources and build hives and colonies. Figure 1 provides a visualization of this swarm manufacturing concept.

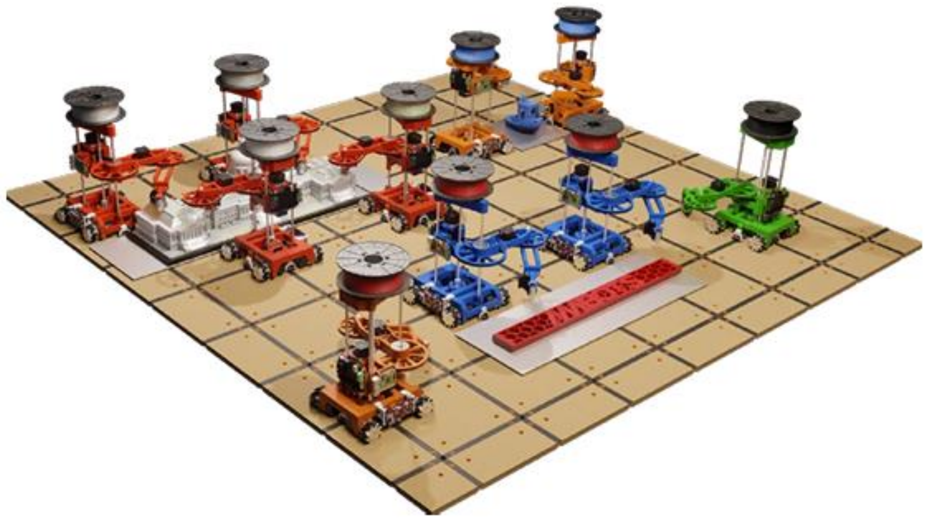


Figure 1: Swarm manufacturing vision [9].

Swarm manufacturing allows for adaptability and efficiency that is not possible in a traditional factory setting. This is because robots can be designed for specialized tasks, adding new capabilities to the factory, or speeding up the manufacturing process through collaboration. At a very low-level, robots designed for fused deposition modeling (FDM) 3D printing can be equipped with different filaments allowing them to print objects of varying colors and material properties. Moving up a level, robots can be designed for 3D printing techniques that require unique extrusion methods, such as resin, gel, or even wire-arc printing to allow for metal parts to be additively manufactured within the swarm factory. Lastly, at a high-level robots can be designed to perform completely unique manufacturing tasks, such as pick-and-place, laser sintering, or subtractive manufacturing such as drilling or CNC for final shaping and surface finishing. This gives the user complete flexibility to manufacture almost anything within a single cooperative workspace, without having to worry about reconfiguring the factory or manually retooling the robots when changing the manufacturing parameters.

The end goal of this research is to lay the groundwork for the swarm manufacturing paradigm. We envision a decentralized, highly flexible, generalized factory that is capable of manufacturing products at a rate similar to that of modern specialized production facilities with centralized assembly lines. However, while it is a promising alternative to modern manufacturing paradigms, the swarm manufacturing concept has yet to be fully realized. The work presented in this thesis is a small, but important contribution towards the realization of this vision. Rather than try to tackle the complexity of swarm manufacturing directly, this research focuses instead on cooperative 3D printing (C3DP), a primitive form of swarm manufacturing that will be expanded upon in the next section.

1.2 Cooperative 3D Printing

Cooperative 3D printing (C3DP) is an emerging AM technology that utilizes a swarm of mobile 3D printing robots working in tandem to manufacture large-scale parts at high speeds without compromising printing resolution. It has been in development for many years, and with many advancements in the technology over the course of that time. This section will provide a summary of the progress to this point.

1.2.1 Hardware

The current iteration of the C3DP system, developed by AMBOTS Inc., consists of a factory floor made of modular floor tiles upon which printing robots and build plates can be placed at discrete locations in one-foot intervals [8]. The printing robots are a unique Selective Compliance Articulated Robot Arm (SCARA) design that mounts securely to and draws power directly from the factory floor [10] for any print action. These printers can be mounted in any of the four cardinal directions at discrete mounting points on the floor, which are spaced at intervals of 300 mm (roughly one foot). Custom-built mobile transporter robots can be used to unmount and move these printers around the factory floor. The transporters are capable of moving in all four cardinal directions, and also rotating in place to allow printers to be placed in any location and orientation on the floor. The parameters of the SCARA printers and mobile transporters are shown in Table 1 and Table 2, respectively [8].

Table 1: Technical Specifications of SCARA Printing Robots [36]

XY reach	50 mm–350 mm
Max *Z*-height	300 mm
Filament feed	Bowden, 1.75 mm
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 lead screw

Layer resolution	10 μm
Print speed	50 mm/s
Print repeatability	5 μm
Power input	Build floor, battery pack via mobile platform
Power consumption	20 W
Software	Compatible with open-source software
Connectivity	Wireless

Table 2: Technical Specifications of Mobile Transporter Robots [36]

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 h use, 6 h idle
Power consumption	20 W
Product dimension	25 cm × 33 cm × 9 cm

Another important component to the C3DP system is the floor tiles. As stated before, these tiles are modular, allowing complete flexibility to set them up in any sized factory or workspace, with the option to easily expand the floor if more space becomes available. This floor is also powered, with conductive strips that allow the printers to operate wirelessly, which is key to allowing the printers to move around the factory floor unencumbered by power cables [10]. To enable this movement, the floor is lined with black strips that assist the mobile transporters in navigating from one mounting point to the next.

ArUco markers are placed at the end of these strips, which can be read by a camera mounted to the rear of each transporter to verify its position and orientation. Lastly, a build plate can be mounted at each mounting point instead of a printer, with the mounting hole used for alignment while magnets on the bottom of the build plate lock it in place. These build plates are made of acrylic with a specialized top layer to allow for cold adhesion, which is necessary because the lack of a heated bed can cause adhesion issues during printing. Each of these individual components, as well as an example floor setup can be seen in Figure 2.

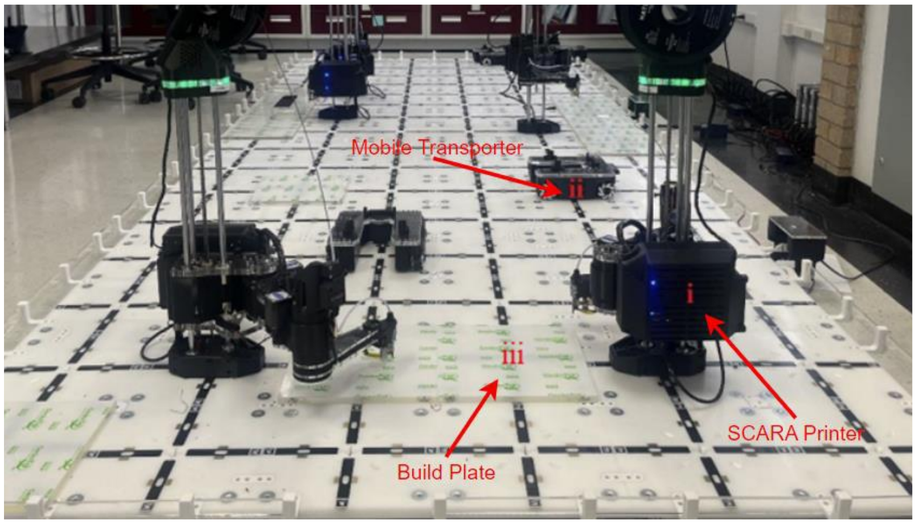


Figure 2: The Cooperative 3D Printing platform [11]. (i) SCARA based 3D printing platform, (ii) Mobile transporter, and (iii) Mountable build plate.

The final hardware component of the C3DP system is the control hub. This consists of a raspberry pi connected to all the robots via WiFi to communicate with their onboard Duet chips in order to receive information and send commands. This control hub allows the user to control the C3DP system, as a job can be sent to the control hub, and from there it will send out commands to each available robot on the network. To this end, most of the software development on the C3DP system is focused on the control hub, increasing its functionality and improving existing processes to allow it to instruct the robots in the most efficient way possible.

1.2.2 Software

There are multiple steps required to transform an input STL file into a completed print in the C3DP workflow. Before we can explain them, however, it is necessary to introduce some important terminology.

- *Project*: This is the input of C3DP, in the form of an STL file that the user wants printed. It can also be considered the desired output of the process.
- *Job*: This is an object on the factory floor that one or more robots work on. A job can be generated as one of the Z-Chunks from the Z-Chunking algorithm if the project is too tall for a single robot. Alternatively, if the project is shorter in the Z-direction than the maximum print height of the robots, the job is the entirety of the *project* directly placed on the factory floor.
- *Chunk*: This is a singular partition of the job that can be printed by one robot from one location on the factory floor.

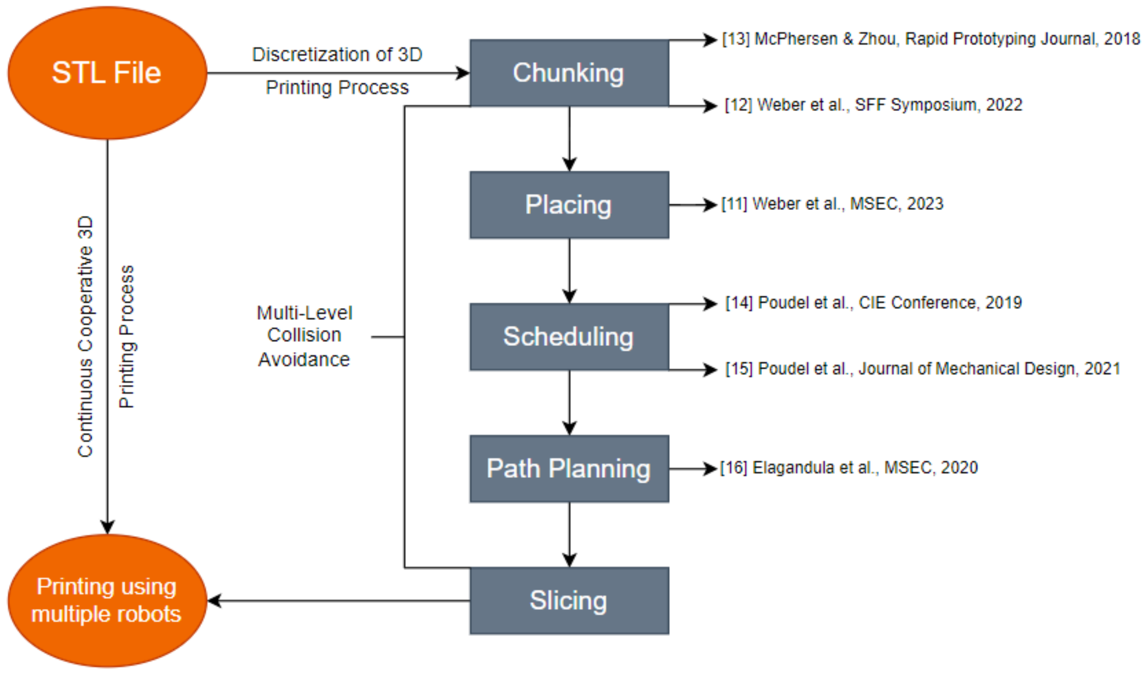

STL File
Discretization of 3D Printing Process
Chunking
[13] McPhersen & Zhou, Rapid Prototyping Journal, 2018
[12] Weber et al., SFF Symposium, 2022
Placing
[11] Weber et al., MSEC, 2023
Continuous Cooperative 3D Printing Process
Multi-Level Collision Avoidance
Scheduling
[14] Poudel et al., CIE Conference, 2019
[15] Poudel et al., Journal of Mechanical Design, 2021
Path Planning
[16] Elagandula et al., MSEC, 2020
Printing using multiple robots
Slicing

Figure 3: Flow chart of the C3DP process.

The key steps of the C3DP process required to turn a project into a completed print are shown in Figure 3 and will now be described.

- *Chunking*: Geometrically partition an STL file into chunks printable by individual robots.
- *Placing*: Locate jobs on the factory floor for printing and assembly.
- *Scheduling*: Determine in what order robots will complete tasks and print specific chunks.
- *Path Planning*: Dictate how robots move between tasks.
- *Slicing:* Determine how a robot will move its toolhead to print a chunk.

Algorithms for Z-Chunking, XY-Chunking, Scheduling, and Path Planning have been introduced in our previous work [12-16]. The next section explains these concepts in further detail.

The first step is chunking. This is the process of breaking down a single CAD model (e.g., an STL file) into smaller pieces, or “chunks”, that can then be printed by a single printing robot. By breaking an object down into chunks, multiple robots can work on the individual parts of a larger object at the same time without interfering with one another. This chunking process is further broken down into the Z and XY directions. Z-chunking is the process of turning a tall object into multiple shorter objects that can each be printed as a distinct job in the C3DP process. This is necessary because the printing robots have a maximum height they can reach while mounted, and any part taller than this limit cannot be printed without chunking. Previously, we developed software to automate the Z-chunking process while generating assembly geometry to facilitate easy assembly after printing [12].

The XY-chunking process occurs after Z-chunking and provides a method for breaking up the individual jobs into printable chunks. The limit on the size of these XY-chunks is the dimensions of the build plate, which, as mentioned previously, is 300mm×300mm. Each chunk is connected to its neighbors using sloped interfaces, allowing the current chunk to be printed on top of the previous chunk. This allows adjacent chunks to form a mechanical bond, eliminating the need for assembly after the printing process, as shown in Figure 4 [17].

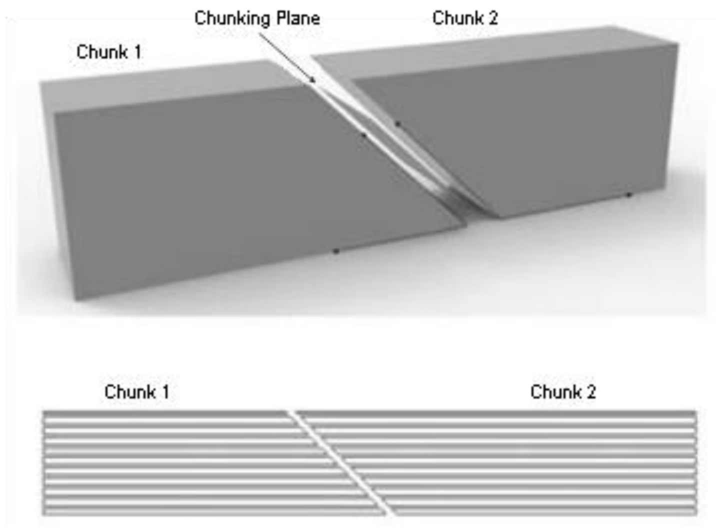
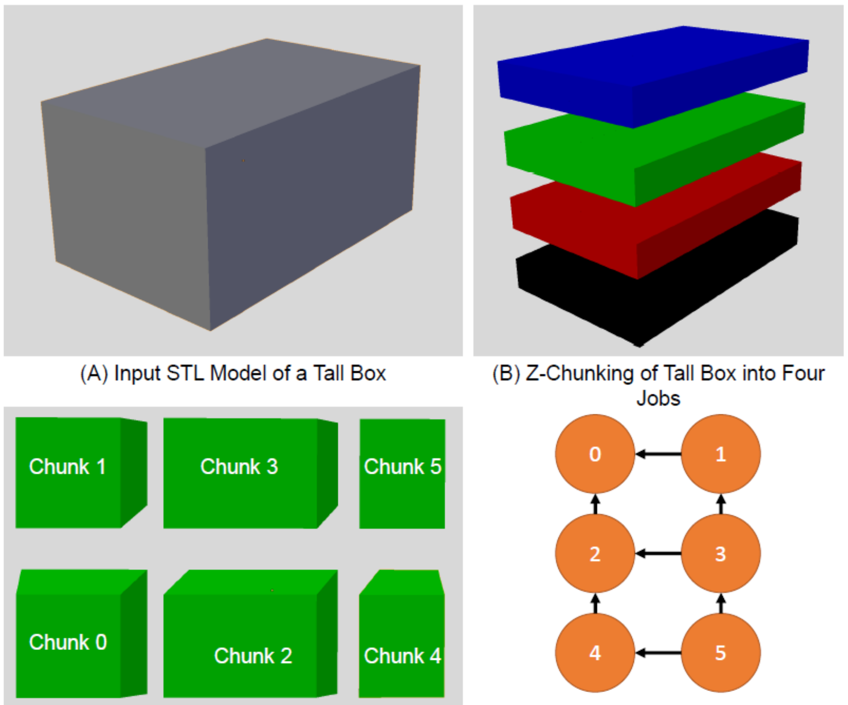


Figure 4: Sloped chunking and slicing [17]. Chunk 2 cannot be printed until chunk 1 has been printed.

One common question for XY-chunking is the quality and mechanical strength at the interfaces of these chunked parts. While an in-depth study would be beneficial, preliminary testing has shown that carefully controlling variables such as the interface slope, number of shell layers, and amount of overlap can result in parts of equal or higher strength than non-chunked parts given specific loading conditions [17]. Another option currently being developed is a layer-wise cooperation strategy that could provide even greater interfacial strength while also allowing adjacent chunks to be printed simultaneously [18].

The chunks resulting from the XY-chunking process need to be printed in a certain order due to the dependency between them. Specifically, any part with an overhang (or negative slope) must be printed after the adjacent part (with positive slope) that it overlaps with. Using this logic, a dependency tree can be created showing the required print order for each part. An example of the full chunking process can be seen in Figure 5.



(A) Input STL Model of a Tall Box

(B) Z-Chunking of Tall Box into Four Jobs

(C) XY Chunking of One Job into Six Chunks

(D) Chunk Dependency Tree

Figure 5: Chunking of a tall box object [11].

A further step, scheduling, has been explored to determine in what order chunks should be printed by minimizing the makespan while satisfying the chunk dependencies generated by the sloped surface chunking method [13, 14]. Additionally, an algorithm has been developed to enable collision-free paths for robots moving between printing locations [15] for both single-job and multi-job prints. Lastly, most of the previous studies have placed jobs on the factory floor arbitrarily, with no consideration given to placement. To remedy this, an algorithm has been developed to optimize job placement for C3DP, linking the geometric partitioning algorithms to the scheduling and path planning algorithms [11]. However, this placement optimization algorithm was not initially physically validated with the AMBOTS hardware.

1.3 Research Questions

While the C3DP framework has experienced many advancements over the years, there is still much work to be done to ensure that the system can function autonomously. After all we are still far from the end goal of a hybrid manufacturing system that can operate autonomously with very little human oversight. To this end, the following research questions have guided my work:

1) Can the existing placement and scheduling algorithms be physically validated to ensure that they accurately reflect the real C3DP system? If so, to what extent can they be validated and what changes should be made to these algorithms to improve their functionality when being physically implemented?
2) How can the C3DP process be monitored to further improve its implementation and account for uncertainties and errors in manufacturing? A major problem with the existing C3DP algorithm is its severe lack of feedback. It works well when everything is running smoothly, but when errors arise, the system has no way of detecting and correcting these issues.

1.4 Thesis Roadmap

The following sections will focus on my research, and the efforts I have made to address these research questions. Chapter 2 will present a physical validation of the existing placement algorithm for C3DP, with a focus on addressing any issues the algorithm might have with being applied to the real C3DP system. Chapter 3 will discuss our preliminary efforts to incorporate process monitoring techniques to the C3DP platform, with the end goal of allowing the system to autonomously detect and correct any errors with printing or moving. This work is still ongoing, but it is a key step in enabling a fully autonomous swarm manufacturing factory. Lastly, in Chapter 4 I will present my closing thoughts on this research, discuss avenues for potential future work, and discuss the impacts of this study in a broader engineering context.

Chapter 2: Physical Validation of Job Placement Optimizer for C3DP

2.1 Background and Motivation

In our previous work, we developed an approach to optimize the placement of printing jobs in the C3DP system using a genetic algorithm [11]. The algorithm starts by generating a random population of placements (each member is a placement of the chunks on the floor), which are then evaluated using a scheduler that dynamically assigns chunks to each available printer based on their relative position on the floor and spot in the dependency tree [14, 15]. This scheduler then uses a basic A* algorithm for path planning [19], which determines the locations the robots will occupy at each time interval during movement and utilizes a conflict-based search algorithm to generate optimal conflict-free paths if multiple robots are moving simultaneously [20]. Using these methods, the total makespan of each member of the population is evaluated by taking the highest combined print and move time of the printers. Finally, the genetic algorithm generates a new population based on the fittest members of the previous generation until it converges to an optimal placement. In our previous study, we tuned the parameters of the genetic algorithm [21] to maximize the speed of convergence to a quality solution. These parameters were determined to be a 40% mutation chance and 10% crossover chance, with 30% of elite populations carried over to the next generation and 30% new populations generated. This leaves 40% of the population to be generated from crossover and mutation via the previous generation. These parameters were carried over into this study, with the results from the genetic algorithm being used to optimally place each job on the build floor.

While our previous work has demonstrated the capability and effectiveness of the proposed placement algorithm within the C3DP pipeline computationally, a physical experimental study is an important link yet to be accomplished for research validation and verification. To address this gap, we aim to physically validate the algorithms and methods that make up the full C3DP process, specifically regarding the placing, scheduling, and path planning by running various test cases on the physical C3DP system. Such a physical validation study is of great importance and value since the C3DP system introduces several additional complexities that are not accounted for in the placement algorithm, such as printer orientation, the independence of printers and transporters, and various additional actions that the printers must perform. By performing this physical validation, we hope to determine what additional factors need to be considered in the optimization algorithm in order to make the C3DP framework more robust against various uncertainties in the actual manufacturing process and thus improve print efficiency.

2.2 Methodology and Test Cases

Based on the C3DP system and the framework presented in Figure 3, we run two unique test cases to validate the placement optimization algorithm in different scenarios. These test cases vary the geometric complexity of the object in question to determine how it would influence the performance and outcomes of the C3DP system, specifically in relation to job placement optimization. The steps for running each of the test cases are shown in Figure 6.

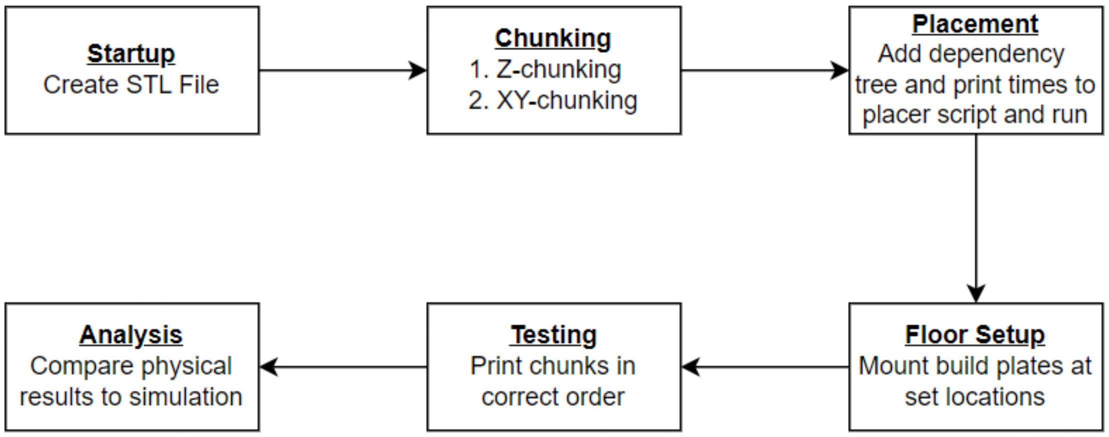

Startup
Create STL File
Chunking
1. Z-chunking
2. XY-chunking
Placement
Add dependency tree and print times to placer script and run
Floor Setup
Mount build plates at set locations
Testing
Print chunks in correct order
Analysis
Compare physical results to simulation

Figure 6: Flow chart of physical validation process.

The physical validation process starts by first modelling the object to be printed and saving it as an STL file. Second, the chunking process starts by Z-chunking the model into a desired number of layers, or jobs, followed by XY-chunking each layer based on the desired length and width. Third, the placement optimization algorithm must be executed, which requires inputting the dependency tree and print times for each chunk. Print settings have been developed specifically for use with the SCARA printers, with a layer height of 0.4 mm, a grid pattern infill with a density of 20%, and a 50 mm/s printing speed. These print settings are used with the Ultimaker Cura slicer to generate g-code and estimate the print time of each chunk. Fourth, build plates are set up at the location of each chunk on the build floor, as determined by the algorithm. Fifth, transporters are instructed to move the printers to the desired build plate based on the results from the scheduler. Finally, the total makespan of the physical system is compared to that produced by the scheduling and path planning algorithm.

2.2.1 Test Case: Small Box

For the first physically validated test case, we decided to use an object with simple geometry. This “small box” test case has dimensions of 60mm×20mm×30mm. These dimensions were chosen to ensure that once the box was fully chunked, each individual chunk would have identical dimensions. This was done to reduce the computational complexity of the placement algorithm since, in this case, each chunk should finish printing at roughly the same time, and each printer should move directly onto its neighbor. To this end, the overhang that is typically present on parts chunked in the XY-plane to ensure mechanical bonding between adjacent parts has been removed for this test. It is worth noting that the removal of this sloped boundary does not impact the objective of this study, as the primary focus is to validate the placement algorithm, and not to verify the mechanical integrity of parts chunked in the XY-direction, which has been done previously. As a result, each of the printed chunks has dimensions 20mm×20mm×10mm. These chunks are clearly very small compared to the dimensions of the printers (300mm×300mm×3000mm), but this was chosen to expedite the printing process and ensure that the tests could be finished in a reasonable time frame with the available equipment. Also, while the lack of an overhang means there are technically no dependencies, an origin chunk was still chosen to ensure each printer started at the same chunk within its respective print. The small box test case is shown in Figure 7.

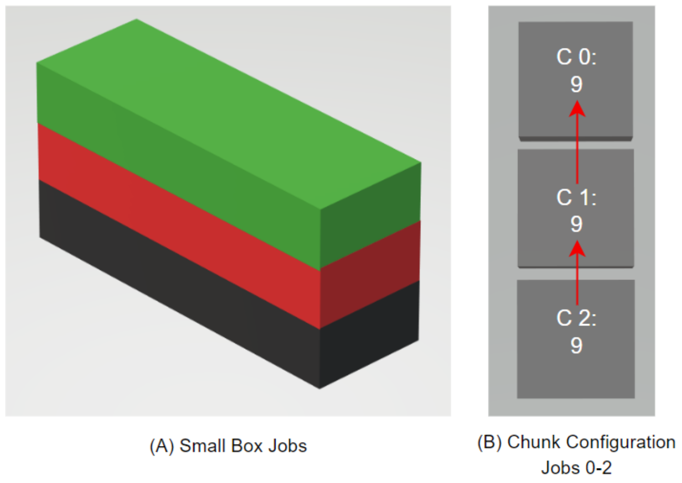


Figure 7: Small box test case. (a) Full model, (b) Chunk configuration for all jobs (they are identical) with print times and dependencies, notated with red arrows.

2.2.2 Test Case: 3DBenchy

The next test case uses a downsized version of the popular 3DBenchy model [22], which was also used as a test case in the previous paper. Before, this model was used to verify the algorithm for any case with a rectangular footprint, not only objects with strictly rectangular jobs, such as the small box test case. For the physical test presented here, the model will be used to validate the algorithm in the case where the chunks being printed are not identical, and the individual jobs have a varying number of chunks. The dimensions of the 3DBenchy model used in this test case are 200mm×103mm×160mm. Due to the non-uniformity of this model, the first layer has three chunks, the second layer has four chunks, and the third layer has two chunks. Also, for this test case the chunking strategy was changed in order to provide a more interesting dependency tree, since the standard chunking strategy would provide very little scheduling flexibility. This test case and its dependency tree are shown in Figure 8.

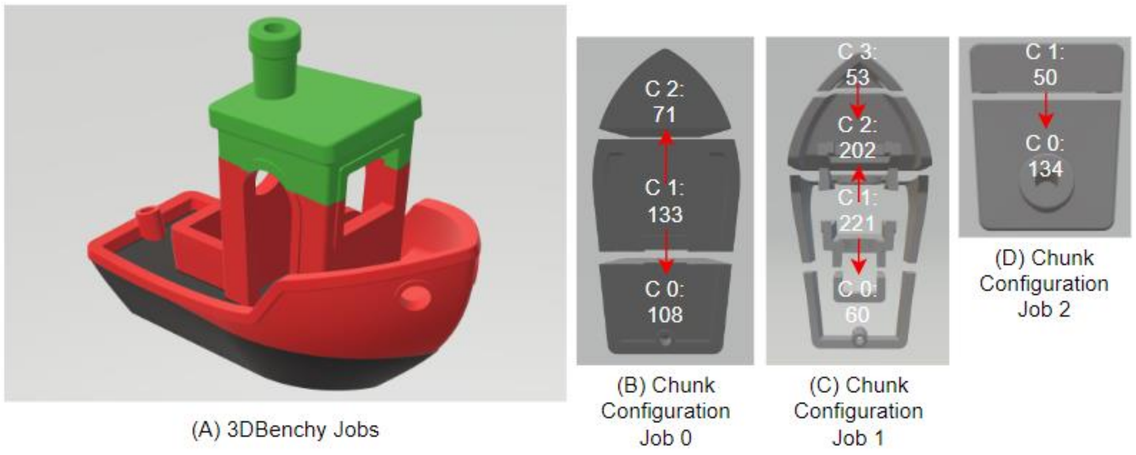

C 2: 71
C 1: 133
C 0: 108
C 3: 53
C 2: 202
C 1: 221
C 0: 60
C 1: 50
C 0: 134
(A) 3DBenchy Jobs
(B) Chunk Configuration Job 0
(C) Chunk Configuration Job 1
(D) Chunk Configuration Job 2

Figure 8: 3DBenchy test case. (a) Full model, (b-d) Chunk configurations for job 0-2 respectively with print times and dependencies, notated with red arrows.

2.3 Results

The results from the placement algorithm for the small box and 3DBenchy physical validation tests are shown in Figure 9. These tests are done with a floor size of 4×3 floor tiles (half the size of our full test bed), with three robots starting at (2, 0), (4, 0), and (6, 0). In these diagrams, the printing robots are represented by gray squares and numbered from one to three. The chunks are represented by different colored squares to differentiate between jobs and are labeled with two numbers. The first number represents the robot that is tasked with printing that chunk, while the second number represents where the chunk lies in that robot's schedule. For example, a chunk labeled "2, 3" will be the third chunk printed by robot two. Lastly, the arrows show the path the robot is expected to take to move from one location to another. This is determined by the path planning algorithm described previously, wherein a robot will attempt to find the shortest, unobstructed path available when moving. A good example of this is presented with printer 3 in the 3DBenchy test case. When attempting to move from chunk 1 to chunk 2, the shortest path is obstructed by printer 2. As a result, the robot must find a new path around the job it is currently working on to reach its desired destination.

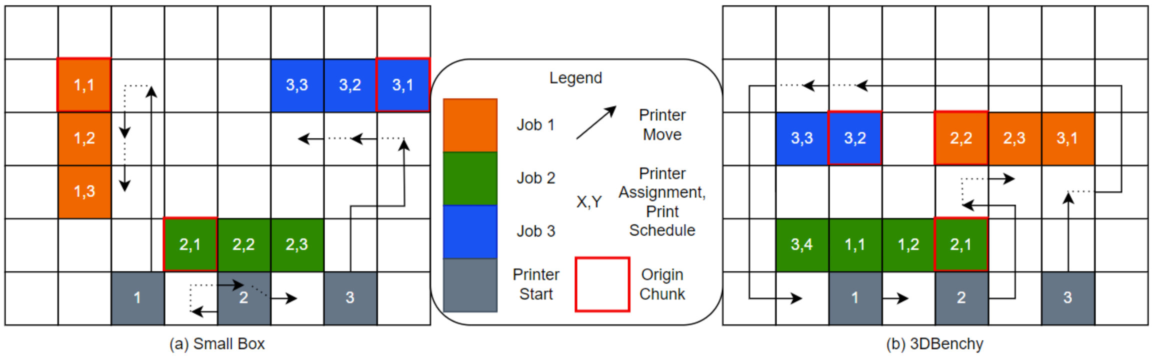


Figure 9: Genetic algorithm placement results.

Next, the physical setup of the C3DP system for the small box and 3DBenchy test cases are shown in Figure 10. The total time for each test case to be completed represents the total makespan of the object with the C3DP system. These makespans are shown in Table 3 for the estimated and experimental C3DP cases, along with the estimated print times of each object without the use of C3DP.

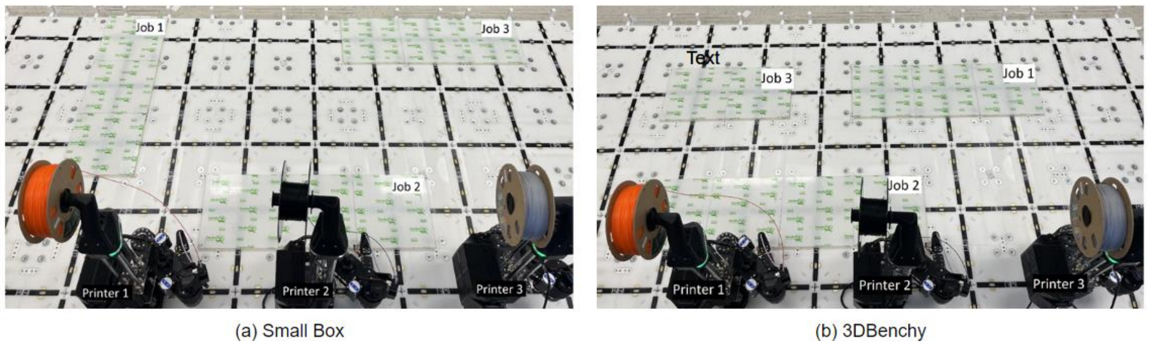


Figure 10: Initial C3DP testing setup.

Table 3: Total Makespans of Examined Test Cases

Test Case	Small Box	3DBenchy
Estimated C3DP Makespan (mins)	30.6	423.6
Experimental C3DP Makespan (mins)	43.0	503.0
Estimated Non-C3DP Makespan (mins)	46	893
Percentage Error (Estimated vs. Experimental)	41%	19%
Percentage Error (Estimated C3DP vs. Non-C3DP)	50%	111%

This table shows one of the benefits of the C3DP system, scalability. The makespan of each test case without the division of labor provided by C3DP is significantly larger than when C3DP is used, and this difference is increased as the total makespan gets larger, as demonstrated by the percentage error between the estimated C3DP and non-C3DP makespans for each test case. Figure 11 shows the fully assembled 3Dbenchy model manufactured using C3DP, and effectively demonstrates the division of labor utilized in the process. However, the estimated results for C3DP do not perfectly match the experimental results obtained using the physical C3DP system, as shown by the percentage error between the estimated and experimental tests. The smaller test case provides a greater error, but for both, the experimental results show a greater makespan than what is predicted by the algorithm. The reasons for these differences will be discussed now.

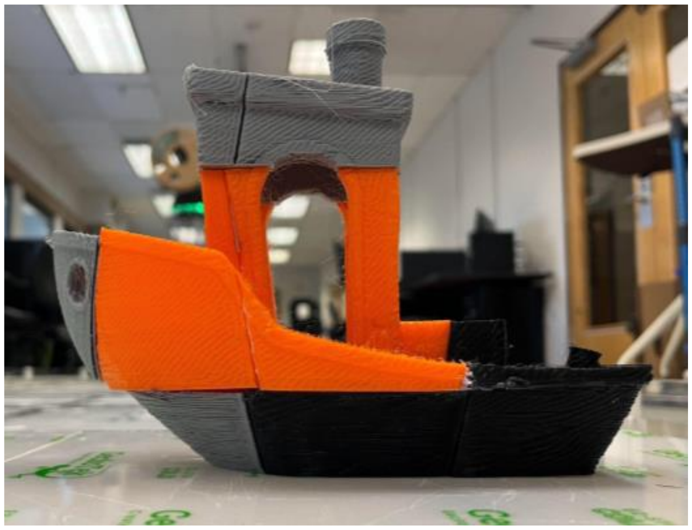


Figure 11: Assembled model of 3DBenchy test tase (orange: Printer 1, black: Printer 2, gray: Printer 3).

2.4 Discussion

As shown in Figures 12 and 13, the actions of each robot can be neatly broken down into two primary types: movement and printing. Each discrete movement and print are shown with both estimated and experimental results, allowing each action to be easily compared. As such, the following discussion will focus on each of these actions and how they can be better incorporated to improve the robustness of the job placement algorithm.

2.4.1 Movement Actions

The algorithm handles movement in a very basic manner. When a robot needs to move, it can go to any adjacent square so long as it is unoccupied, and each move takes exactly 36 seconds (0.6 minutes) regardless of direction. This move time was determined by pre-calibration of the system. Also, the orientation of the robot is not accounted for, so as long as the robot is next to its target chunk, the move action is complete.

In the physical system, movement is a much more complicated process. Not only is orientation important (both in terms of movement speed and printer direction), but there are additional actions that must be taken and requirements that must be met to complete a move. First, in the physical system, the transporters and printers are two separate objects. As such, for a move action to start, the printer must unmount from the floor and onto the transporter. The same is true at the end of a move action, with the printer having to mount to the floor and release the transporter. The transporter also must drive out from under the printer, creating another requirement that there must be an empty tile behind the printer wherever it is to be mounted. Lastly, since the transporters run off an internal power supply, they must return to their charging dock when not actively in use to maintain battery life. This creates an additional requirement that the transporter must always have a path back to its charging dock when not in use.

Comparing the results for estimated movement times to those in the physical system leads to a few key observations: First, for single moves, the estimated results are nearly identical to the experimental tests. This makes sense, as single moves were used to determine the move time that was input to the algorithm. However, the experimental times were consistently shorter for longer chain moves than the estimated times from the algorithm. This is because there is a significant time delay between sending a command to the transporter and the transporter performing that command. Once the command is received, however, the transporter can perform a long chain of moves without having to communicate with the control hub again, decreasing the time of consecutive moves. Another observation is seen in the small box test with the first move of robots 1 and 3. These moves should take the same amount of time according to the algorithm, and yet the experimental results are over 20 seconds off. This is because, as mentioned previously, the algorithm does not account for orientation, while in the physical system, the final orientation of the printer is very important. In this case, printer 1 had to rotate to face the desired chunk while printer 3 did not, resulting in a longer move time. Lastly, it must be noted that the move times shown in Figure 12 do not account for mounting and unmounting, which must be completed before and after every move, adding approximately 1 minute to each time.

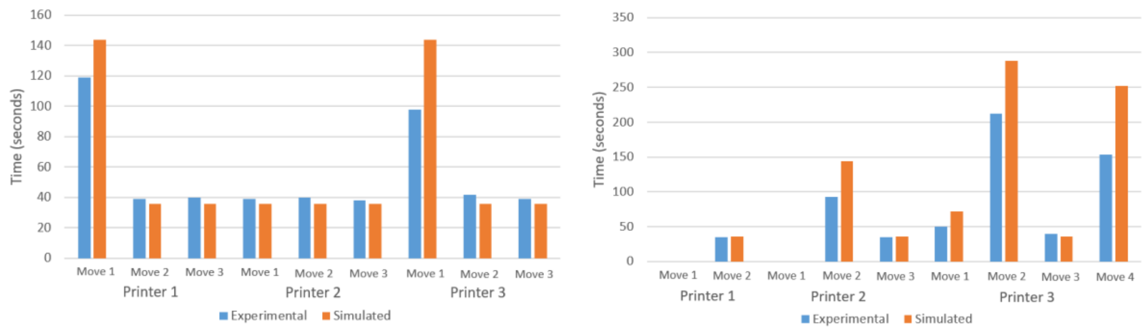


Figure 12: Movement time testing results. Left: Small box, Right: 3DBenchy.

2.4.2 Printing Actions

The print times in the algorithm are also very simple, using the time estimate for each chunk from the slicing software. There are a few additional actions related to printing that the physical system must perform, but the estimate from the algorithm does not account for. First, the printer must complete a homing sequence during which it moves in each direction until it contacts a mechanical limit switch. This process is completed before and after every print, which takes about 40 seconds and is included in the total print time for the experimental results. Second, the nozzle needs time to reach the desired printing temperature of 240°C, which takes about 100 seconds and is also included in the total print time for the experimental results. Lastly, the printer should be calibrated after every move to ensure high print quality on the new build plate. This can take anywhere from 2 to 20 minutes, depending on the size of the object being printed and desired accuracy of the calibration. While calibrating at every location is technically not necessary, neglecting it will result in printing errors unless the build plates are perfectly identical, which is difficult to verify. Regardless, while a calibration was done before each print, due to the variability of viable calibration times it is not included in the experimental results.

When comparing the estimated and experimental results, it is readily apparent that the experimental print times are consistently larger than the estimated times, even with the homing and heating times removed. The percentage error between the two varies from around 5% up to over 20% in the most extreme cases. This is at least partially to do with the fact that the SCARA printers being used for these tests are not registered in the Cura software, and as such, there is a difference in how long it takes the printer to execute certain G-code commands. It also seems to be related to chunk complexity, as chunks with simple geometries tend to produce less error than those with more complex geometries. For example, in the 3Dbenchy test, the second print completed by printer 2 took about 21% longer to complete than the slicer estimate. This makes sense when looking at the chunk, as it is composed of tall thin walls that are loosely connected, and an internal geometry that is completely removed from the rest of the print. However, this is not a complete explanation, as even the simple chunks from the small box test case took about 12% longer to complete on average than the slicer predicted. Regardless of the reason, this error cannot be easily accounted for in the placement algorithm and will require custom slicing software to correct.

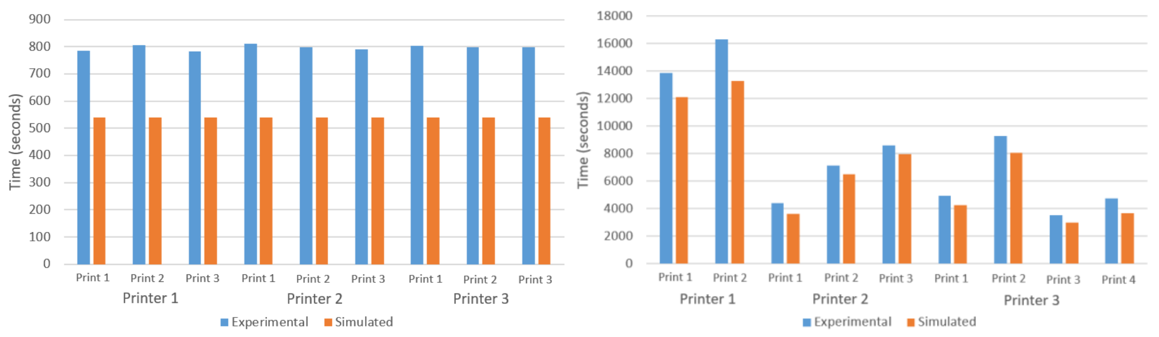


Figure 13: Print time testing results. Left: Small box, Right: 3DBenchy.

2.5 Future Work

There are a number of potential avenues for improvement that require further exploration. First, changes could be made to refine the optimization algorithm itself to improve runtime efficiency and provide more consistent results. One of these changes would be to rewrite the scheduling algorithm. In this study, we utilized a dynamic scheduler to assign tasks to the robots in real-time as they become available. This is computationally inexpensive but is not guaranteed to give a global optimum, especially as the size of the workspace and number of tasks increases. A schedule optimization algorithm could provide better results, assuming the increased computational complexity can be accounted for by improved hardware or other software improvements.

Another avenue of improvement to the optimization algorithm would be adding the capability to optimize for additional variables. Currently, the algorithm only works to optimize the total makespan of the project. This can be improved by allowing the algorithm to optimize for variables such as combined makespan (total build time of all robots, not just the final robot to finish), energy/material efficiency, or specific task priority.

Lastly, there are changes that must be made to allow this placement optimization algorithm to be effectively integrated into the physical C3DP system. Many of these were discussed in the previous section, but some are easier to fix than others. First, there are many simple additions that can be made to account for things like the response time of the transporters, mounting and unmounting of the printers, homing, and nozzle heating. Even orientation falls in this category, as it simply requires adding an orientation variable and adding rotation commands. However, other fixes, such as accounting for the printer and transporter as independent entities or fixing the print time estimate on the slicer are more complicated changes that will require sweeping changes to the existing algorithm. We hope to address both issues in our future work, but that is outside the scope of this study.

Chapter 3: Process Monitoring for C3DP

3.1 Background and Literature Review

While the C3DP pipeline is mostly complete, another important tool that must be implemented in order to facilitate the full autonomy of the system is process monitoring. As of right now the control hub is able to send commands and receive information from the robots, but this is all very surface level. Printers can send data such as nozzle temperature, expected nozzle position, and print status, but none of it can be externally verified automatically. As such, the real nozzle position could vary from the expected position due to poor calibration, or worse yet a collision, and a print could fail with no way for the printer to know on its own. The only external monitoring the system does is with the cameras mounted to the rear of the transporters, which check their position using ArUco markers on the floor. This can help track the robots and avoid collisions, but it does not provide an adequate failsafe for the various errors that can occur during movement and printing. Because of this, two levels of process monitoring have been identified for implementation into the C3DP platform: system level and task level.

3.1.1 System Level Monitoring

System-level monitoring aims to look at the entire C3DP system holistically, tracking different robots on the platform to assist with path planning and collision avoidance. One method of doing this would be to build a digital twin of the manufacturing environment, allowing any action to be simulated before being executed by the physical system. Methods for automatically building digital twins of manufacturing spaces have been developed using object recognition methods with a scan of the shop floor [23]. Such approaches have already been tested to great effect on traditional robot assembly lines, with machine learning enabled manufacturing systems capable of self-monitoring [24]. This is enabled by significant advances in object detection technology in recent years, specifically within the realms of real-time image segmentation and object localization [25]. Also, due to the mobile nature of the C3DP system, the robots will have to be accurately detected and tracked during movement anywhere on the factory floor. Fortunately, there is existing literature on using multiple cameras to track objects in settings where occlusions are a frequent issue, such as a manufacturing floor [26].

In addition, a system-level monitoring tool will be necessary for path planning and collision avoidance when we expand to a hybrid swarm manufacturing framework, with many different mobile robots moving around and completing tasks on a single manufacturing floor. However, such a holistic monitoring tool is not necessary for the existing C3DP system, which features discrete coordinates that are simple to determine, allowing the robots to be easily tracked. As such, the remainder of this study will focus on task-level process monitoring, focusing on the primary task the current system engages in, i.e., FDM 3D printing.

3.1.2 Task Level Monitoring

Process monitoring for additive manufacturing is currently a growing field of research because one of the major drawbacks of the process is the inconsistency of parts produced. This has been a heavily discussed area for metal AM specifically, where parts for use in applications such as aerospace must meet strict quality standards that are difficult to verify with additively manufactured parts. As such, thorough process monitoring and control techniques have been researched to guarantee that parts manufactured through this method have minimal defects and meet these high standards of quality [27]. Industrial providers have even begun developing and selling solutions for this issue, such as EOS Smart Monitoring, which is capable of monitoring the build and correcting parameters such as laser power to ensure the final part meets the standards set during pre-processing [28].

This level of quality assurance is typically unnecessary for FDM printing because the parts are typically not used in such demanding environments. However, the process has a larger failure rate than most other manufacturing processes, at around 20% for unskilled users [29]. While the C3DP robots have set printing parameters specific to the platform and filament used, there are still opportunities for failure. We have noted many of these in our own testing, with extrusion issues, poor bed calibration, warping, and stringing being common problems encountered during test prints. An example of some of these failures is shown in Figure 14, which features some common defects such as layer splitting and interfacial gaps in a cooperative test print of a castle as a result of poor extrusion and bed calibration. These are common enough issues with FDM printing and typically are not a major problem since the material cost and print times are kept low. However, if FDM printing is going to be incorporated in an automated factory setting, there needs to be a system in place to ensure that prints are completed accurately and with no major errors. Otherwise, a single print failure can lead to other failures in adjacent printers, and eventually cascade into a total failure of the entire factory.

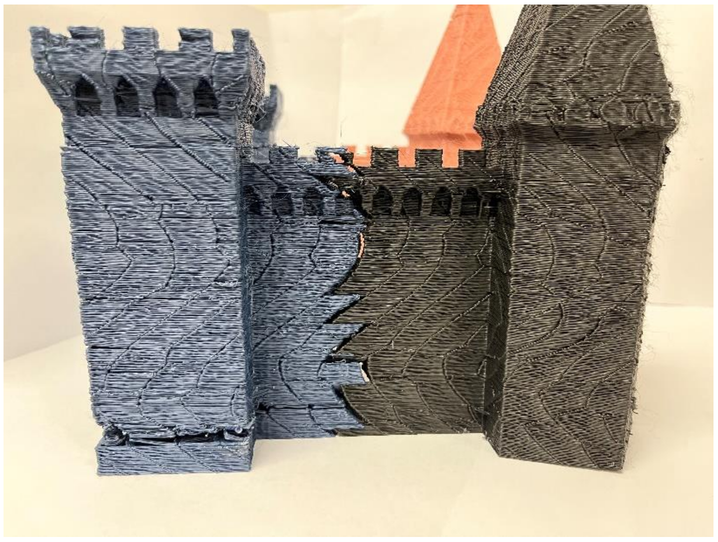


Figure 14: 3-Robot cooperative print of castle demonstrating common FDM printing errors.

In-situ process monitoring of FDM prints is another area that has received a lot of research [30] that can be built upon to provide an effective solution to the above FDM failures. Many different methods have been explored for this purpose, from 2D and 3D vision techniques to temperature, vibration, and acoustics monitoring [31]. However, due to the relatively low cost of FDM 3D printing compared to other AM techniques, there is a desire to keep the cost of monitoring low so as to not outpace the cost of the printer itself.

As such, most of these methods utilize cheap equipment with computationally efficient algorithms that can be run effectively on most standard computers.

This study will explore the integration of a number of these process monitoring techniques in the C3DP platform. The remaining sections discuss the existing 3D vision techniques, how we attempt to adapt them for use in C3DP, and the difficulties this poses. This is followed by a discussion of the 2D vision techniques relevant to the C3DP platform, specifically focused on nozzle tracking, warping, and stringing and how they are integrated into a holistic monitoring system for a single print. Lastly, our future plans for this project, as well as the immediate next steps that are currently being worked on, are presented.

3.2 3D Vision

3D vision is an interesting technique for process monitoring. On the one hand, 3D scanning is a highly accurate, non-destructive technique for reverse engineering 3D models of physical parts [32], which should make it ideal for comparing a printed geometry to the ideal geometry via an STL or G-code file. On the other hand, many techniques struggle with accuracy due to scans being noisy, highly sensitive to lighting conditions, or simply taking a long time to gather data.

One such technique is laser profilometry (or interferometry), which can provide a highly accurate surface topology, but is limited by being time-consuming and requiring precise control of the laser [33, 34]. A simpler method is using two 2D cameras in a stereo depth configuration, which is similar to human depth perception in that the depth of an object is triangulated by knowing the distance between the two cameras and determining the difference of each point between the two captured images. The benefits of this method are its low cost and ease-of-use, but it is highly susceptible to changing lighting conditions. Even under ideal lighting, it is generally much lower resolution than other 3D scanning methods [35,36].

Another option guided by a similar principle is using multiple cameras with structured light to generate a pointcloud of the printed object. This method utilizes the stereo depth principle, but also incorporates a laser projecting a fringe pattern on the object to help eliminate the effects of lighting conditions and provide high-accuracy scans for a relatively low cost, so long as the fringe pattern is projected perpendicular to the layer direction [37, 38, 39]. This is the principle most 3D scanners are based on. The primary drawback of using structured light is that the fringe pattern must be projected on the part during scanning, which makes it very hard to scan entire objects in real-time, as the scanner needs to be moved around the object during scanning. It is also impossible to use multiple scanners from different angles, as overlapping fringe patterns would cause errors in the scan data.

The final 3D vision technique we looked at is time of flight (TOF) sensors, which emit a pulse of near-infrared light and measure the time it takes for this light to get reflected back to the sensor to determine the distance of objects in its field of view. The benefits of this technology are quite simple: providing good accuracy at a high frame rate while being low-cost and relatively lightweight. However, this technology is highly affected by environmental conditions such as lighting, object color and material, and distance [40, 41]. Lastly, using multiple TOF sensors looking at the same point can cause issues, as the light pulses can interfere with each other and produce errors in the scan data.

3.2.1 Methodology

To effectively monitor the entire print, we need to ensure that the computer vision system can see all four sides of the build plate as well as the top of the print. We also need to be able to take data in real-time to detect errors as they occur. This immediately eliminates a few options. First, profilometry is out due to the time-intensive nature of the process, and the fact that it requires precise control of the laser to effectively scan the part surface. A robotic arm or a linear rail system could be used to precisely move the laser around the part to get a full surface profile during printing, but this would increase the cost and complexity of the system beyond what is reasonable for this study. For similar reasons, structured light scanning was also eliminated. This was a promising option, but the need for multiple laser projectors and cameras alternating to cover every side of the print was deemed too complex and costly, and having a single moving scanner would incur similar issues as the profilometer. Because of this, the two different 3D imaging techniques adopted were stereo depth and TOF sensing.

Stereo depth was tested first. The benefits of this method were simple, as it provided a low-cost, easy-to-use method of generating a full 3D reconstruction of the object via pointclouds using only a few camera perspectives. Rather than use multiple 2D cameras synchronized for stereo depth, we used an Intel Realsense D435 stereo depth camera [42]. This provided many benefits, such as known specs, preconfigured calibration, and prewritten scripts for operating the camera via the Intel Realsense SDK 2.0, Intel's open-source developer toolkit [43]. This camera is shown in Figure 15.

Figure 15: Intel Realsense D435 depth camera [42].

Using three D435 cameras mounted at roughly 120-degree intervals, a complete 3D reconstruction of a print can be generated by stitching the individual pointclouds from each camera together. To do this, the transformation matrix of the cameras had to be determined. This was done using CV2, an OpenCV library for Python [44], which can estimate the pose of each camera using a printed checkerboard pattern attached to a flat surface in the camera's field of view. This allowed us to generate the transformation matrix of each camera for the purpose of pointcloud stitching, while also determining the plane of the build plate where the object would be printed. This was useful as it allowed the object itself to be detected, isolating the pointcloud of the object from its environment to easily reconstruct the object. This camera setup, along with an example of the object detection script, is shown in Figure 16.

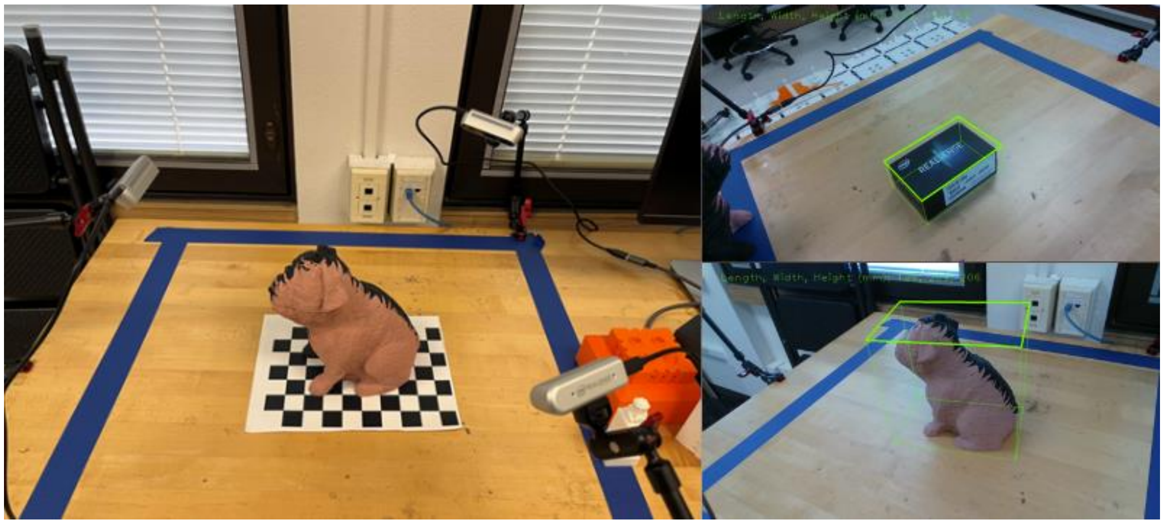


Figure 16: Intel Realsense calibration. Left: Full calibration setup, Top Right: Object detection with a simple box. Bottom Right: Object detection of complex shape.

The other method that we tested was time-of-flight imaging using a Microsoft Azure Kinect DK. This camera features a 1 MP depth sensor, along with sync ports to allow multiple cameras to operate in the same space without the TOF pulses intercepting each other and causing errors in the scan data [45]. Microsoft also hosts multiple SDKs to assist with different applications [46], and other open-source repositories exist as well, such as LiveScan3D, a software package that syncs multiple Azure Kinect depth streams and generates a pointcloud of an object in real-time [47]. This repository made generating the pointcloud of an object viewed by the Kinect cameras very simple, and as such, it was used with a similar camera setup to the D435, but with this program replacing the pointcloud generation process. The Kinect camera and experimental setup are shown in Figure 17.

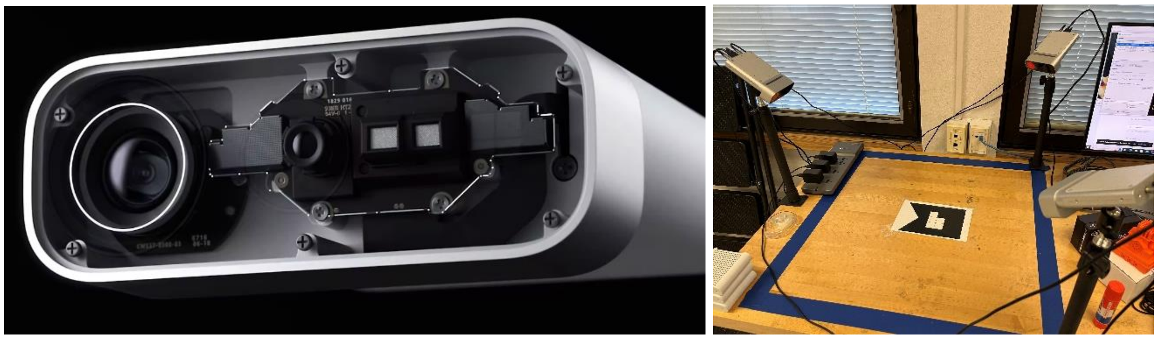


Figure 17: Azure Kinect DK. Left: Camera [45], Right: Experimental calibration setup.

Once the pointclouds were gathered and stitched together, the final step was to transform the complete pointcloud into a mesh that could be compared to the STL file of the printed object. This was originally done using the "Compute Normals for Point Sets" command in MeshLab [48], but there was too much noise in the pointcloud, and as such, the mesh it generated was very rough. To correct this, we utilized the convolutional occupancy network (CON) model to generate our pointclouds, as it is designed to reconstruct "complex geometry from noisy point clouds and low-resolution voxel representations" [49]. The results of this process are shown in the next section.

3.2.2 Results

Each camera was tested on a 3D-printed model of the top portion of the UT tower. This test case was printed in orange PLA filament and is shown in Figure 18. This specific test case was chosen as it has a relatively simple rectangular geometry but also includes some fine features that act as a good benchmark for the capability of each 3D imaging method. This way, the process can be graded both on reconstructing the general geometry of the part while also analyzing how accurately it can recognize and display fine features.

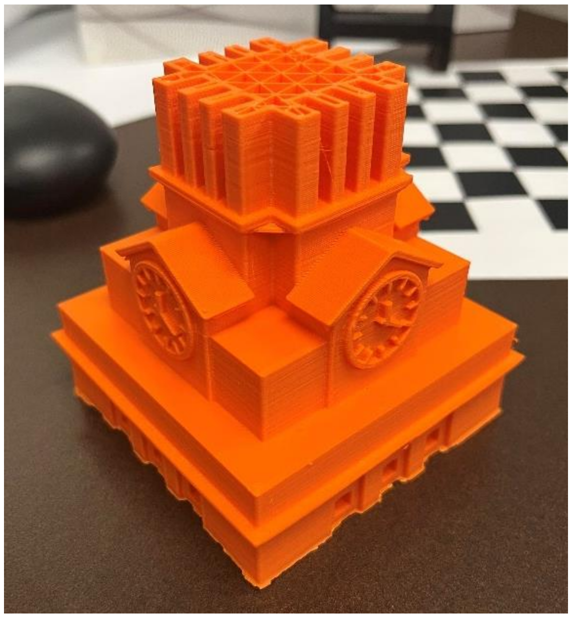


Figure 18: Test case model of UT Tower top in orange PLA filament.

The first step in the testing process, as mentioned previously, is to generate a pointcloud of the full part using each of the above methods. These pointclouds are seen in Figure 19. There are a few key points to note in these pointclouds. First, both cameras are clearly poorly calibrated, which can be seen in the overlapping points not properly aligning with one another. This is more noticeable in the pointcloud generated by the Azure Kinect, as the overhead view shows that one camera is clearly slightly rotated with respect to the other two. Another important feature is the random, disconnected pixels generated by both cameras. They show in the Realsense pointcloud as sets of points jutting off from the main geometry, while in the Kinect pointcloud, there are flying pixels spraying away from the base. Lastly, a final note to make is how poorly these pointclouds mirror the reference geometry, which can be better seen in the meshes.

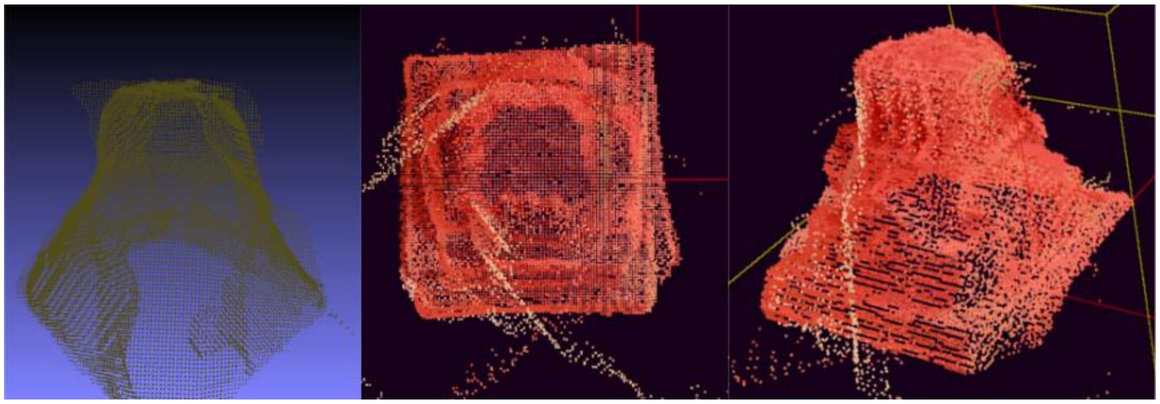


Figure 19: Pointcloud of UT Tower top. Left: Intel Realsense D435, Middle/Right: Azure Kinect.

The meshes generated by these pointclouds using the CON model can be seen in Figure 20. The most apparent feature of these models is their complete lack of both detail, and even general geometric resemblance to the test part. The mesh from the D435 specifically lacks any discernible features aside from the vaguely rectangular base and top, and even this is a stretch. The Azure Kinect mesh does slightly better, showing the general shape with the discrete “steps,” as well as the sloped geometry on top of the clock. However, this mesh still provides no fine detail while also failing to accurately capture the full geometry of the part.

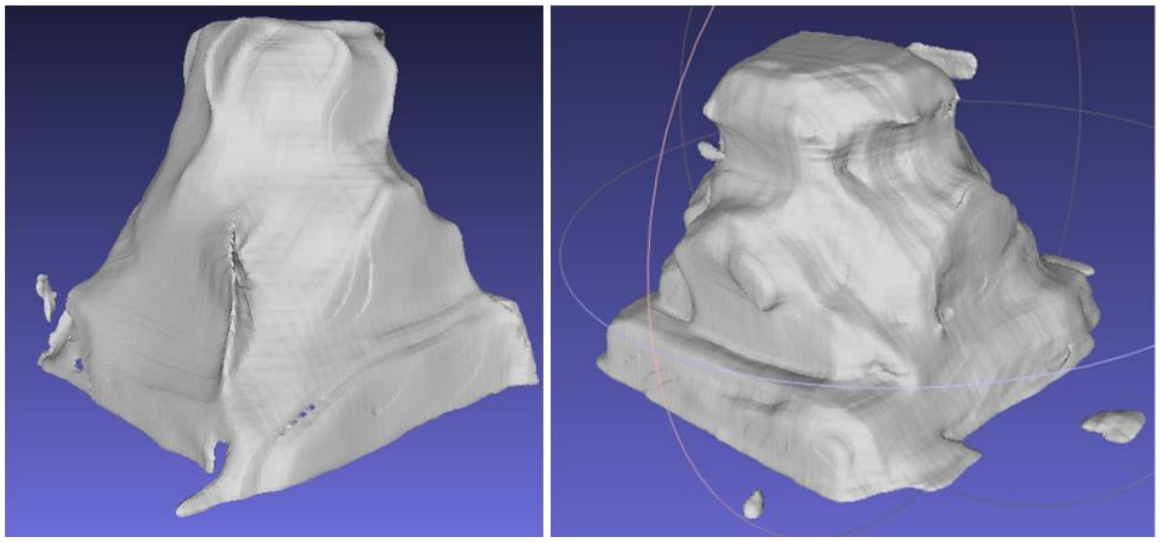


Figure 20: Meshes generated with CON model. Left: D435, Right: Azure Kinect.

3.2.3 Discussion

As the results in Figure 20 clearly indicate, process monitoring via 3D reconstruction using the cameras from this study is not an effective strategy in its current state. Looking at the hardware specifications for these cameras can give some indication as to why. The Intel Realsense D435 provides a depth accuracy of <2% at 2m, meaning up to 40mm of error is expected during standard use [42]. This might be improved by moving the camera closer to the object, but even if this 2% error was predicted at the smallest acceptable range of 0.3m, an error of up to 6mm would still be expected. This completely negates any potential the system might have of detecting and correcting errors in 3D printing, as even if the system could detect larger failure regions, an error of more than a couple of millimeters in size is typically catastrophic to the printing process. However, even this is unlikely, as an error of 6mm at each point is enough that the general geometry of the print, specifically for smaller parts, would be completely unrecognizable. Similar issues exist with the Azure Kinect camera, which features a random error standard deviation of <17mm, and a typical systematic error of <11mm + 0.1% of the distance [50]. While some changes can be made to the experimental setup to improve the image quality for this camera, such as narrowing the field of view and bringing it closer to capture more points on the object, this is unlikely to improve the results enough to justify using this method in process monitoring. Some of these tweaks have already been tested, and the improvements made have been marginal at best. While we had hoped for better results, the poor quality of these 3D reconstructions was not entirely unexpected. Studies have found that the TOF sensors in the Kinect V2 camera (similar to the Azure Kinect DK we are using) cannot accurately capture small objects with fine-scale features to the same level of detail as other optical techniques such as fringe projection profilometry, a variation of structured light scanning [51].

There are other issues that make the prospect of exploring this method further difficult to justify. For one, the TOF sensor is highly sensitive to color, as discussed previously, which is a major problem for FDM 3D printing, as it features a wide variety of different colored filaments. The orange filament used in this study was one of the better-performing colors we tested, while darker filaments such as black and navy were almost completely undetected by the TOF sensor. Another issue is that additional complexities arise when trying to monitor a 3D print in situ, with the primary problem being occlusion from the nozzle. During the printing, the nozzle consistently covers most of the top of the print, and occasionally part of the sides, from the view of one or all of the cameras. As such, not only will we be dealing with an already inaccurate 3D reconstruction, but it will not even be consistently updated without taking time to stop the print, which takes away from the benefits that could be provided by such a real-time monitoring system. For these reasons, we shifted our focus from the 3D reconstruction of printed parts to the myriad of 2D process monitoring techniques that currently exist for FDM 3D printing.

3.3 2D Vision

2D vision is the most widely explored method for process monitoring of FDM 3D printing. Many different techniques have been explored in this field to detect errors that are commonly seen. For example, deep learning techniques have been used in tandem with a camera focused on the extruder to both detect and correct extrusion errors in real-time [52, 53]. Externally mounted cameras can also be used to view the printed object as it is being built and compare the image to the desired print geometry from the STL file [54]. This can also be used to detect warping, and machine learning models have been developed to not only recognize warping when it occurs, but also correct it at its onset to ensure it does not become catastrophic and cause the part to fail [55]. Similar 2D imaging techniques can be done on a layer-wise level by extracting the geometry of the most recently completed layer from the printed part and comparing it to the same layer in the g-code [56]. However, such a process cannot be done in real-time, as it requires the nozzle to move out of the way of the camera to capture an image of the completed layer. Stringing is another common defect that can be indicative of failures in FDM printed parts. A small amount of stringing is a common occurrence in 3D printing, especially when printing thin walls or cylinders that are spaced apart from one another, but heavy stringing tends to indicate print failure as it occurs when the filament is deposited but does not properly adhere to the part. Fortunately, machine learning methods have been developed to detect stringing in real-time using simple 2D images of the part [57].

3.3.1 Methodology

One of the primary benefits of these 2D computer vision techniques is their ease of use and low cost. As such, any standard RGB camera can be used for this purpose. Our testing was conducted using an Intel Realsense D435 camera, shown in Figure 15 of the previous section. The depth sensing capabilities of this camera are not needed for performing 2D vision monitoring, but the camera was readily available and provides good image quality with a frame resolution of 1920x1080, a sensor resolution of 2MP, and a maximum framerate of 30 FPS [42]. This provides a good baseline level of quality for running all our 2D vision methods.

The first 2D vision method we analyzed was object tracking. By determining the position of the nozzle at each point in time, we can ensure that it follows the correct path set by the g-code and can correct for issues such as motor slippage and external collisions. One such object-tracking method is using machine learning methods such as adaptive correlation filters to recognize and track the nozzle in real-time [58]. While this could be an effective strategy, a simpler alternative exists of simply attaching an ArUco marker to the nozzle mount and using an overhead camera to determine the location of this marker using a marker detection script [59]. Using such a marker, as long as the camera position and distance between the marker and nozzle are known, the nozzle position can be easily extracted from the marker position. To this end, a custom camera mount was developed that attaches to the filament spool of the printer, which was chosen because this position is unaffected by the location of the printer and the nozzle. A custom marker base was also developed to attach to the nozzle mount, providing a consistent marker position relative to the nozzle. Both mounts are shown in Figure 21.

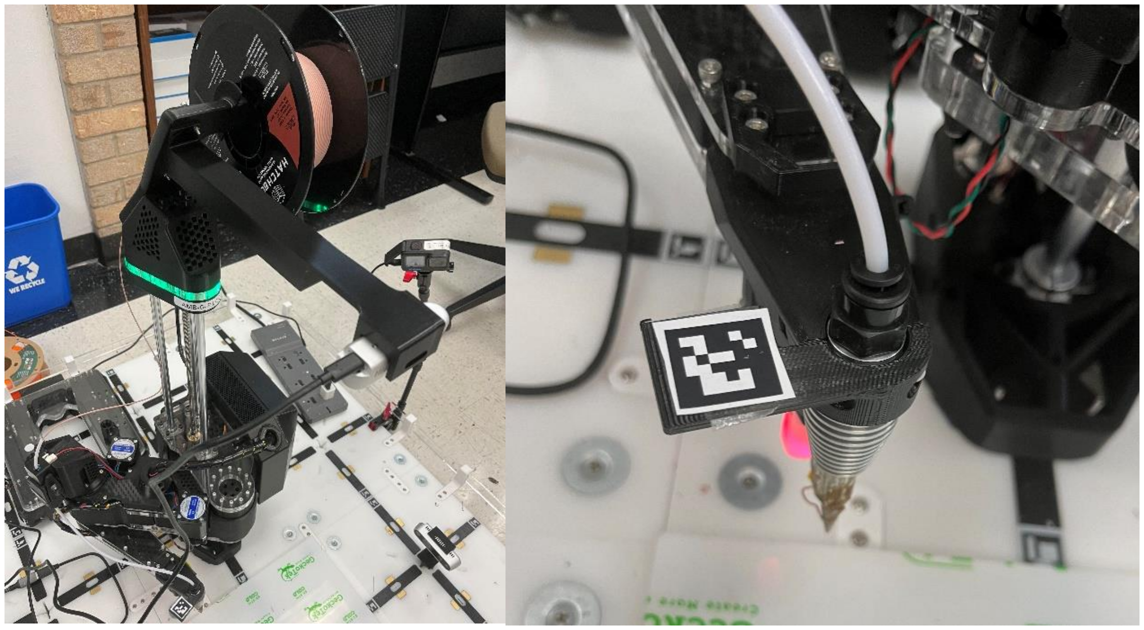


Figure 21: Mounts for use in object tracking. Left: Overhead camera mount, Right: ArUco marker nozzle mount.

The next 2D vision method we looked at was the detection of stringing. As mentioned previously, this is a well-established machine learning method, and its ease of use makes it very attractive for hobbyists who want a very basic monitoring tool for their 3D prints. Obico is an open-source, network-enabled 3D print monitoring tool that allows users to monitor and control their 3D printers remotely using a mobile app, and it uses a similar AI-driven stringing detection algorithm to predict part failure and warn users when such failure occurs [60]. For our stringing detection method, we utilized the Ultralytics YOLOv8 model, which is capable of object detection, classification, and segmentation [61]. This model was trained using a stringing dataset from Roboflow, which features images of hundreds of examples of stringing in 3D prints. We trained the model with this dataset over 40 iterations to ensure that it would be able to consistently detect stringing from images taken in various different 3D printing scenarios. Due to the use of a trained object detection model to recognize stringing, the position of the camera for this monitoring method is mostly arbitrary. The only requirements are that the camera must be able to see the print without obstruction, and the background must provide good contrast against the filament being used. As such, a side profile of the print was desired to ensure that the print could be monitored in real-time without being occluded by the nozzle, while also not confusing the model by having stringing present in front of filament with the same color, both of which could occur in an overhead image. A mount was developed for this purpose that could be slotted into the mounting holes on the AMBOTS floor tiles (Figure 22), allowing a side profile of the print to be captured.

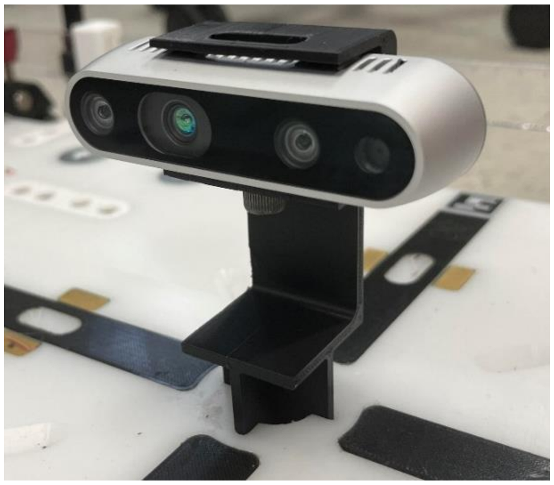


Figure 22: Side profile camera mount.

The final method of interest was the 2D reconstruction of the printed part. This process is done by generating a mask of the part at a certain standardized camera position and comparing this to the rendered mask that can be produced from the STL model at this same orientation [63]. The two masks can then be compared using the structural similarity index method, and a similarity score is generated. This tells the user how well the two masks overlap (a similarity score of 1 is perfect) and therefore how accurately the print reflects the ground truth model. Because this technique requires masks of both the physical print and 3D model in the same pose, a standardized camera position must be determined. Fortunately, the camera mount generated for stringing detection in Figure 22 is perfect for this application, as it mounts directly to the factory floor and, as such, ensures that the camera always sits at a standardized distance and orientation relative to the build plate. The rendered mask from the STL file can then use this information to ensure the pose generated always perfectly matches what the camera will see in the physical system.

These three methods of print monitoring can be combined into a single holistic monitoring framework to detect any possible failure in the print. Object tracking will ensure that the nozzle is always in the right position, stringing detection will ensure that there is good layer adhesion and filament retraction when printers pause, and the 2D image comparison will check for major flaws in the print such as layer shifting and warping. A flow chart of the full monitoring framework is shown in Figure 23.

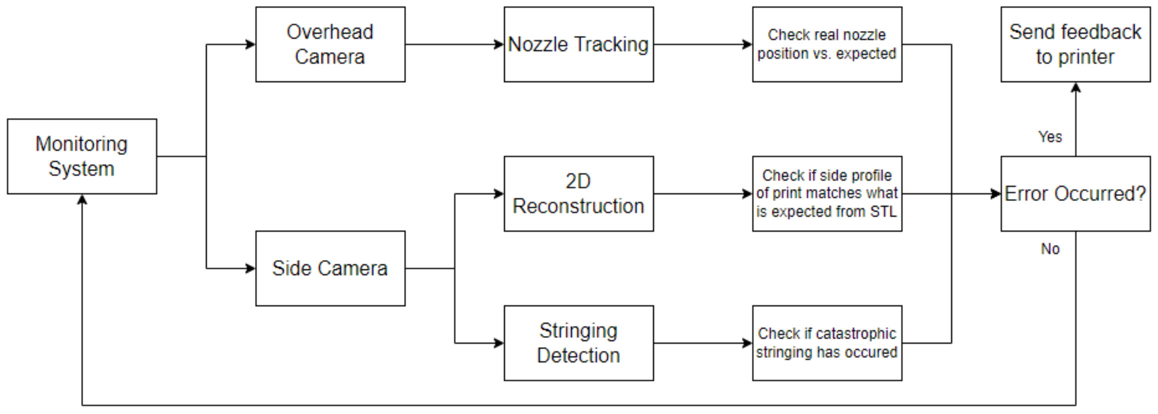

Monitoring System
Overhead Camera
Side Camera
Nozzle Tracking
2D Reconstruction
Stringing Detection
Check real nozzle position vs. expected
Check if side profile of print matches what is expected from STL
Check if catastrophic stringing has occured
Send feedback to printer
Yes
Error Occurred?
No

Figure 23: Flow chart of process monitoring framework.

3.3.2 Preliminary Results

With the testing methodology established, the next step is to verify that each of the proposed process monitoring techniques works effectively. After the techniques are individually validated for the C3DP platform, they will be integrated into the holistic monitoring framework outlined in Figure 23.

The first method we tested was the nozzle tracking using a mounted ArUco marker. The initial results for this study show that the overhead camera can track the ArUco marker effectively, detecting it in over 99% of frames during typical printer movement. This was tested by running a standard square test print and counting both the total number of frames captured by the camera, and how many of those frames the computer was able to detect an ArUco marker. However, we have noted a significant error in the comparison between the nozzle position from the marker and the ground truth position from the printer of over 20mm and increasing during nozzle movement. The overhead camera feed with the marker tracking overlay can be seen in Figure 24.

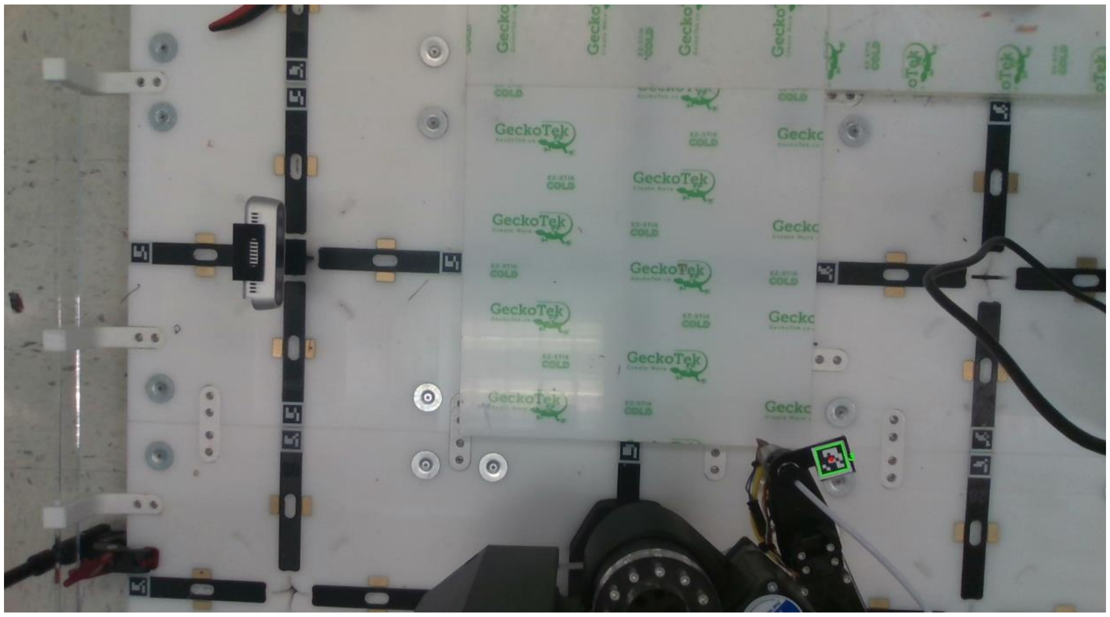


Figure 24: Overhead camera feed during nozzle tracking.

We then evaluated the stringing algorithm using several test parts that were printed with the goal of demonstrating stringing. This was done in a few different settings to ensure that the process was effective regardless of variables such as the background and filament color. A few of these test cases are shown in Figure 25. While this has not yet been tested on a real-time print, the initial results show that the algorithm is highly sensitive to these variables. For example, the algorithm can detect stringing in darker parts with a light background at a very high level of confidence, while lighter colored filaments such as orange and yellow do not provide enough contrast for the algorithm to consistently detect the stringing. On the other hand, the stringing in parts printed with these light filaments can be easily detected against a black background, but the stringing in the black part goes undetected with this background.

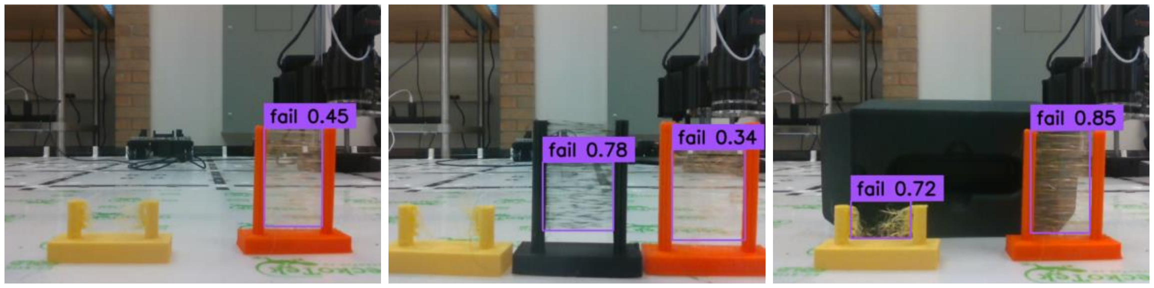


Figure 25: Detection of stringing in 3D printed parts with different filament colors and backgrounds.

The final 2D vision technique tested was comparing the printed geometry to the reference geometry via 2D reconstruction. To do this, the side profile of a printed 3DBenchy model was used to generate an image mask of the part. This was then compared to rendered masks of both the STL file used to print the model and a slightly altered STL file with holes to simulate an error in the final print when compared to the ground truth model. The similarity scores generated are 0.808 for the comparison between the object and the correct STL file and 0.743 for the comparison between the object and the altered STL file. The masks generated for this test are shown in Figure 26.

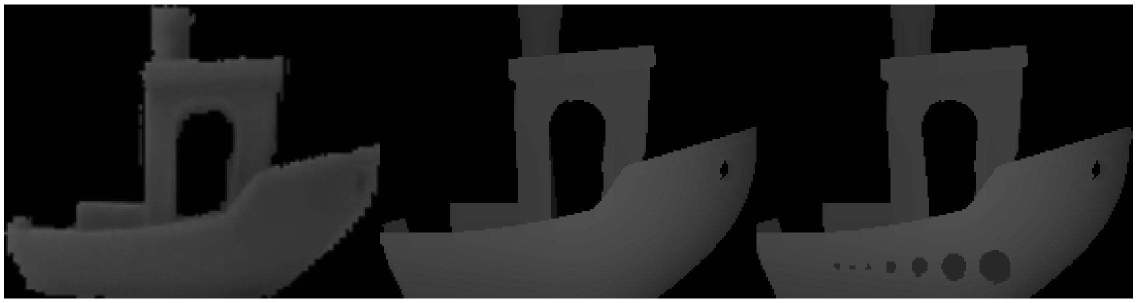


Figure 26: Masks of 3DBenchy. Left: Mask generated from camera image, Middle: Mask generated from correct STL file, Right: Mask generated from altered STL file.

3.3.3 Discussion and Future Work

While the test cases described above are preliminary explorations of their respective monitoring techniques, there is still some valuable information to be discussed. This discussion will focus on the key insights gathered from each of the above tests, as well as the future work that must be done to properly integrate them into the C3DP pipeline.

First, the preliminary nozzle tracking data gives us a few insights that can be used to guide our path forward. For one, while the system is capable of tracking the general position of the nozzle, and thus detecting large shifts or errors, it lacks the accuracy to effectively detect small movements on the scale of a few millimeters due to the error present in the ArUco marker tracking. As such, it cannot be used to effectively automate tasks such as nozzle calibration, which requires sub-millimeter accuracy to guarantee satisfactory results. Also, while the initial design of the overhead camera mount was done to ensure that the exact position of the camera was always a known value, the slight deflection due to the weight of the camera (which can be seen in Figure 24) makes its position difficult to determine. As such, we are exploring an alternative that utilizes ArUco markers on the corners of the build plate to give the camera feed some additional known coordinates, with which it should be able to determine the exact location of the nozzle using trilateration.

Next, the stringing data we gathered also provides us with many interesting possibilities. Not only does the stringing algorithm show the area where stringing is being detected, but it also gives us a percentage value that represents the level of confidence of the model that catastrophic stringing is present. Using this value, the system's response can be tuned to provide different directions based on the severity of the stringing detected. For example, a small amount of stringing can be ignored, while moderate stringing can lead to the printer correcting itself by adjusting the nozzle temperature or printing speed, and high stringing can result in a complete stoppage of the print. The thresholds for these states can also be tuned to ensure that false positives are not a common occurrence while still being sensitive enough to detect catastrophic stringing when it occurs. However, if this method is to be used, the user must be careful that the background and filament color provide enough contrast for stringing to be detected consistently, as the algorithm is highly sensitive to these variables.

Lastly, the key takeaway from the 2D reconstruction data is that more testing is needed to determine acceptable similarity thresholds for 3D-printed parts. The comparison between the physical model and ground truth STL, producing a similarity score of around 0.8, is quite low as they should be nearly identical. On the other hand, the comparison between the physical model and altered STL, producing a similarity score of roughly 0.74, seems too high. The holes in this model are quite large, much larger than typical failures that will be observed during printing, and yet they decrease the similarity score by less than 10%. Additional testing will be needed to determine how susceptible the image mask is to external factors such as lighting and background conditions, and whether these variables have any noticeable impact on the similarity score. It would also be interesting to look at dividing the mask of the print into subregions, which would make the 2D reconstruction more sensitive to local defects and potentially lead to a higher success rate in error detection.

Chapter 4: Conclusions and Future Work

Additive manufacturing has several limitations that must be overcome before it can be seen as a realistic alternative to traditional production line manufacturing practices. Two of the major limitations of modern AM systems are speed and scalability, which are difficult to overcome due to the fact that these parameters are intrinsic to most standard 3D printers, and even if they could be improved, there must be tradeoffs in terms of the cost and quality of parts printed. Swarm manufacturing seeks to provide a more attractive alternative by addressing many of these shortcomings while also enabling the use of other manufacturing techniques in tandem with 3D printing to foster hybrid manufacturing. However, there are a number of unique challenges with the swarm manufacturing process that must be addressed before the technology can be widely adopted. To this end, we have been developing a fully autonomous cooperative 3D printing (C3DP) system to research and validate many swarm manufacturing principles.

In this thesis, I have presented my contribution to the development of C3DP and by proxy swarm manufacturing. My main contributions are:

- A physical validation of the entire C3DP pipeline, with a focus on placement and scheduling algorithms. This both proved that these algorithms can be applied to implement the C3DP concept in the physical world while also discovering various ways they could be improved to better mirror reality.
- The integration of process monitoring techniques for 3D printing error detection to the C3DP platform. This is a requirement for the system to function autonomously,

as it must be able to detect when tasks fail in order to ensure that a single failure does not spiral into the failure of the entire system, i.e., error propagation.

There are plenty of potential avenues for future work that can build on the progress I have presented in this thesis. Many were already presented in Sections 2.5 and 3.3.3, but a few will be discussed here. First, while the C3DP pipeline has been physically validated, a unifying software is still needed to combine the various steps of the C3DP process. As of right now, in order to run a build on the C3DP system, each individual process must be run individually, with the outputs being gathered and manually input into the following piece of the chain. For example, a desired print has to be uploaded to the Z-chunker, then the Z-chunks have to be individually uploaded to the XY-chunker, and finally, the print data of each individual chunk must be inputted to the placement and scheduling programs to generate a final build plan. Combining these disjointed steps is a pivotal step in achieving a fully functional C3DP workflow. This could then be further improved by the software communicating directly with the control hub, starting the printing process as soon as the desired print is uploaded.

Another avenue of future work is integrating the 2D vision systems presented in Section 3.3 into a holistic monitoring framework. As of now, the individual monitoring systems must be run separately, and there is no way of sharing information between them. This could be improved by combining them into a single script and potentially adding some cross-communication, such as a shared print quality metric. This way, each individual system can still be used to detect catastrophic failures while also monitoring the overall quality of a print. Also, implementing a GUI to allow a user to see the camera feeds and easily read the outputs of each monitoring technique would prove very useful for evaluating the effectiveness of the monitoring system.

The work presented in this thesis has improved the C3DP process and addressed various research gaps related to it and swarm manufacturing as a whole. Future work in this field will enable a hybrid manufacturing paradigm dominated by general-purpose factories employing highly flexible manufacturing robots.

Appendix

All Publications and Posters Published, Submitted, and Planned

1. Cole Mensch, Anuj Swaminathan, and Zhenghui Sha. Process Monitoring for Cooperative 3D Printing. In preparation for the Solid Freeform Fabrication Symposium 2024.
2. Cole Mensch, Daniel Weber, Wenchao Zhou, and Zhenghui Sha. Job Placement Optimizer for Cooperative 3D Printing. In preparation for the Journal of Computing and Information Science in Engineering.
3. Mensch, C., Swaminathan, A., & Stone, R. (2024, February 23). Toward Swarm Manufacturing: Developing a Multi-Robot Cooperative Framework for Hybrid Manufacturing Tasks. University of Texas at Austin Mechanical Engineering Annual Research Poster Session, Austin, Texas.
4. Mensch, C., Zhou, W., & Sha, Z. (2023, August 14) Physical Validation of Job Placement Optimization in Cooperative 3D Printing. The 34th Annual International Solid Freeform Fabrication Symposium, Austin, TX. https://utw10945.utweb.utexas.edu/sites/default/files/2023/100%20PhysicalValidationofJobPlacementOptimizationinCooperative3DPrinting.pdf

References

[1] Majeed Farooqi, K. (Ed.). (2018). *Rapid prototyping in cardiac disease: 3D printing the heart*. SPRINGER INTERNATIONAL PU.

[2] 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

[3] 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

[4] Go, J., Schiffres, S. N., Stevens, A. G., & Hart, A. J. (2017). Rate limits of additive manufacturing by fused filament fabrication and guidelines for high-throughput system design. Additive Manufacturing, 16, 1–11. https://doi.org/10.1016/j.addma.2017.03.007

[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] Ali, Md. 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

[7] 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

[8] Poudel, L., Marques, L. G., Williams, R. A., Hyden, Z., Guerra, P., Fowler, O. L., Sha, Z., & Zhou, W. (2022). Toward swarm manufacturing: Architecting a cooperative 3D printing system. *Journal of Manufacturing Science and Engineering*, *144*(8). https://doi.org/10.1115/1.4053681

[9] Elagandula, S., Poudel, L., Zhou, W., & Sha, Z. (2021). Enabling multi-robot cooperative additive manufacturing: Centralized vs. Decentralized Approaches. *Volume 2: 41st Computers and Information in Engineering Conference (CIE)*. https://doi.org/10.1115/detc2021-71343

[10] Currence, J., Morales-Ortega, R., Steck, J., & Zhou, W. (2017). A Floor Power Module for Cooperative 3D Printing. Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium (SFF). https://repositories.lib.utexas.edu/server/api/core/bitstreams/6aa9851e-6fd8-445c-a32b-8063cc1244a4/content

[11] Weber, D., Zhou, W., & Sha, Z. (2023). Job Placement for Cooperative 3D Printing. *Manufacturing Science and Engineering Conference (MSEC).* https://doi.org/10.1115/MSEC2023-104613

[12] Weber, D., Zhou, W., & Sha, Z. (2022). Z-Chunking for Cooperative 3D Printing of Large and Tall Objects. Proceedings of the 33rd Annual International Solid Freeform Fabrication Symposium (SFF). http://dx.doi.org/10.26153/tsw/44190

[13] 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

[14] Poudel, L., Zhou, W., & Sha, Z. (2019). Computational design of scheduling strategies for multi-robot cooperative 3D printing. Volume 1: 39th Computers and Information in Engineering Conference (CIE). https://doi.org/10.1115/detc2019-97640

[15] 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

[16] Elagandula, S., Poudel, L., Sha, Z., & Zhou, W. (2020). Multi-robot path planning for cooperative 3D printing. International Manufacturing Science and Engineering Conference (MSEC). https://doi.org/10.1115/msec2020-8390

[17] 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

[18] Krishnamurthy, V., Poudel, L., Ebert, M., Weber, D. H., Wu, R., Zhou, W., Akleman, E., & Sha, Z. (2022). Layerlock: Layer-wise collision-free multi-robot additive manufacturing using topologically interlocked space-filling shapes. Computer-Aided Design, 152, 103392. https://doi.org/10.1016/j.cad.2022.103392

[19] Cui, S., Wang, H., & Yang, L. (2012). A simulation study of A-star algorithm for robot path planning. 506-509. https://www.researchgate.net/publication/290546219_A_simulation_study_of_A-star_algorithm_for_robot_path_planning

[20] Sharon, G., Stern, R., Felner, Ar., & Sturtevant, N. R. (2015). Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence, 219, 40–66. https://doi.org/10.1016/j.artint.2014.11.006.

[21] Sivanandam, S., Deepa, S. (2008). Genetic Algorithms. In: Introduction to Genetic Algorithms. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-73190-0_2

[22] CreativeTools. (2016). 3DBenchy. GitHub. https://github.com/CreativeTools/3DBenchy/

[23] Sommer, M., Stjepandić, J., Stobrawa, S., & Soden, M. von. (2023). Automated generation of digital twin for a built environment using scan and object

detection as input for production planning. *Journal of Industrial Information Integration*, *33*, 100462. https://doi.org/10.1016/j.jii.2023.100462

[24] Xia, K., Saidy, C., Kirkpatrick, M., Anumbe, N., Sheth, A., & Harik, R. (2021). Towards semantic integration of machine vision systems to aid manufacturing event understanding. *Sensors*, *21*(13), 4276. https://doi.org/10.3390/s21134276

[25] Diwan, T., Anirudh, G., & Tembhurne, J. V. (2022). Object detection using yolo: Challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, *82*(6), 9243–9275. https://doi.org/10.1007/s11042-022-13644-y

[26] Anuj, L., & Krishna, M. T. (2017). Multiple camera based multiple object tracking under occlusion: A survey. *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. https://doi.org/10.1109/icimia.2017.7975652

[27] Usha, S. (2021). In situ monitoring of Metal Additive Manufacturing Process: A Review. *Additive Manufacturing*, 275–299. https://doi.org/10.1016/b978-0-12-822056-6.00007-2

[28] *Eos Smart Monitoring for 3D printing*. EOS. https://www.eos.info/en-us/enablement/software/eos-smart-monitoring

[29] Wittbrodt, B. T., Glover, A. G., Laureto, J., Anzalone, G. C., Oppliger, D., Irwin, J. L., & Pearce, J. M. (2013). Life-Cycle Economic Analysis of distributed manufacturing with open-source 3-D printers. *Mechatronics*, *23*(6), 713–726. https://doi.org/10.1016/j.mechatronics.2013.06.002

[30] Fu, Y., Downey, A., Yuan, L., Pratt, A., & Balogun, Y. (2021). In situ monitoring for fused Filament Fabrication Process: A Review. *Additive Manufacturing*, *38*, 101749. https://doi.org/10.1016/j.addma.2020.101749

[31] Oleff, A., Küster, B., Stonis, M., & Overmeyer, L. (2021). Process monitoring for material extrusion additive manufacturing: A state-of-the-art review. *Progress in Additive Manufacturing*, *6*(4), 705–730. https://doi.org/10.1007/s40964-021-00192-4

[32] Haleem, A., Javaid, M., Singh, R. P., Rab, S., Suman, R., Kumar, L., & Khan, I. H. (2022). Exploring the potential of 3D scanning in Industry 4.0: An overview. *International Journal of Cognitive Computing in Engineering*, *3*, 161–171. https://doi.org/10.1016/j.ijcce.2022.08.003

[33] Borish, M., Post, B. K., Roschli, A., Chesser, P. C., Love, L. J., & Gaul, K. T. (2018). Defect identification and mitigation via visual inspection in large-scale additive manufacturing. *JOM*, *71*(3), 893–899. https://doi.org/10.1007/s11837-018-3220-6

[34] Sitthi-Amorn, P., Ramos, J. E., Wangy, Y., Kwan, J., Lan, J., Wang, W., & Matusik, W. (2015). Multifab: a machine vision assisted platform for multi-material 3D printing. *ACM Transactions on Graphics*, *34*(4), 1–11. https://doi.org/10.1145/2766962

[35] Nuchitprasitchai, S., Roggemann, M., & Pearce, J. M. (2017). Factors effecting real-time optical monitoring of fused filament 3D printing. *Progress in Additive Manufacturing*, *2*(3), 133–149. https://doi.org/10.1007/s40964-017-0027-x

[36] Nuchitprasitchai, S., Roggemann, M., & Pearce, J. (2017a). Three hundred and sixty degree real-time monitoring of 3-D printing using computer analysis of two camera views. *Journal of Manufacturing and Materials Processing*, *1*(1), 2. https://doi.org/10.3390/jmmp1010002

[37] Ye, Z., Liu, C., Tian, W., & Kan, C. (2020). A deep learning approach for the identification of small process shifts in additive manufacturing using 3D point clouds. *Procedia Manufacturing*, *48*, 770–775. https://doi.org/10.1016/j.promfg.2020.05.112

[38] Preissler, M., Zhang, C., Rosenberger, M., & Notni, G. (2018). Approach for process control in additive manufacturing through layer-wise analysis with 3-dimensional Pointcloud information. *2018 Digital Image Computing: Techniques and Applications (DICTA)*. https://doi.org/10.1109/dicta.2018.8615803

[39] Fastowicz, J., Grudziński, M., Tecław, M., & Okarma, K. (2019). Objective 3D printed surface quality assessment based on entropy of depth maps. *Entropy*, *21*(1), 97. https://doi.org/10.3390/e21010097

[40] He, Y., Liang, B., Zou, Y., He, J., & Yang, J. (2017). Depth errors analysis and correction for time-of-flight (TOF) cameras. *Sensors*, *17*(1), 92. https://doi.org/10.3390/s17010092

[41] Niedermayr, D., & Wolfartsberger, J. (2022). Analyzing the potential of a time-of-flight depth sensor for Assembly Assistance. *Procedia Computer Science*, *200*, 648–659. https://doi.org/10.1016/j.procs.2022.01.263

[42] *Depth camera D435*. Intel® RealSenseTM Depth and Tracking Cameras. (2022, December 5). https://www.intelrealsense.com/depth-camera-d435/

[43] maloel. (2023, September 28). *Release Intel® RealsenseTM SDK 2.0 (v2.54.2)*. GitHub. https://github.com/IntelRealSense/librealsense/releases/tag/v2.54.2

[44] *Opencv-python*. PyPI. (2023, December 31). https://pypi.org/project/opencv-python/

[45] *Azure Kinect DK*. Microsoft Azure. https://azure.microsoft.com/en-us/products/kinect-dk

[46] Microsoft. *Azure-Kinect-sensor-SDK: A cross platform (linux and windows) user mode SDK to read data from your Azure Kinect device.* GitHub. https://github.com/Microsoft/Azure-Kinect-Sensor-SDK

[47] Kowalski, M. *LiveScan3D at AzureKinect*. GitHub. https://github.com/MarekKowalski/LiveScan3D/tree/AzureKinect

[48] MeshLab. https://www.meshlab.net/

[49] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., & Geiger, A. (2020). Convolutional Occupancy Networks. *Computer Vision – ECCV 2020*, 523–540. https://doi.org/10.1007/978-3-030-58580-8_31

[50] qm13. (2022, September 2). *Azure Kinect DK Hardware Specifications*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/kinect-dk/hardware-specification

[51] Balasubramaniam, B., Li, J., Liu, L., & Li, B. (2023). 3D imaging with fringe projection for food and agricultural applications—a tutorial. *Electronics*, *12*(4), 859. https://doi.org/10.3390/electronics12040859

[52] Brion, D. A., & Pattinson, S. W. (2022). Generalisable 3D printing error detection and correction via multi-head neural networks. *Nature Communications*, *13*(1). https://doi.org/10.1038/s41467-022-31985-y

[53] Jin, Z., Zhang, Z., & Gu, G. X. (2019). Autonomous in-situ correction of fused deposition modeling printers using computer vision and Deep Learning. *Manufacturing Letters*, *22*, 11–15. https://doi.org/10.1016/j.mfglet.2019.09.005

[54] Straub, J. (2015). Initial work on the characterization of Additive Manufacturing (3D printing) using software image analysis. *Machines*, *3*(2), 55–71. https://doi.org/10.3390/machines3020055

[55] Brion, D. A. J., Shen, M., & Pattinson, S. W. (2022). Automated recognition and correction of warp deformation in extrusion additive manufacturing. *Additive Manufacturing*, *56*, 102838. https://doi.org/10.1016/j.addma.2022.102838

[56] Petsiuk, A. L., & Pearce, J. M. (2020). Open source computer vision-based layer-wise 3D printing analysis. *Additive Manufacturing*, *36*, 101473. https://doi.org/10.1016/j.addma.2020.101473

[57] Paraskevoudis, K., Karayannis, P., & Koumoulos, E. P. (2020). Real-time 3D printing remote defect detection (stringing) with Computer Vision and artificial intelligence. *Processes*, *8*(11), 1464. https://doi.org/10.3390/pr8111464

[58] Bolme, D., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2010.5539960

[59] *Detection of ArUco Markers*. OpenCV. https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[60] *Your 3D printer is dumb. make it smart!*. Obico. https://www.obico.io/

[61] Jocher, G. *Ultralytics YOLOv8*. GitHub. https://github.com/ultralytics/ultralytics?tab=readme-ov-file

[62] QA, A. M. (2022, December 29). *3DPrinting Computer Vision Project*. Roboflow. https://universe.roboflow.com/additive-manufacturing-qa/3dprinting

[63] Lyngby, R. A., Wilm, J., Eiriksson, E. R., Nielsen, J. B., Jensen, J. N., Aanæs, H., & Pedersen, D. B. (2017). In-line 3D print failure detection using computer vision. Euspen. https://www.euspen.eu/knowledge-base/AM17133.pdf