Powder Bed Metal Printer

FINAL PROJECT REPORT

sddec21-11
Timothy Bigelow
Colin Firth
Chris Johannsen
Aaron Martin
Tary Todd
Addison Ulrick
Dale Young
sddec21-11@iastate.edu
https://sddec21-11.sd.ece.iastate.edu

Revised: December 8, 2021

Executive Summary

Development Standards & Practices Used

We will be working with 2 standards. The slicer works with the STL files, an ASCII text file format produced by most 3D CAD software defined in the ISO / ASTM52915 - 20 standard.

The other standard that we will be handling is the gcode file standard defined in the ISO 6983 standard.

Summary of Requirements

- Functional Requirements
    - Refactor and convert source code from C# to Matlab
    - Be able to print 3 cm^3 metal blocks
    - Adjustable laser scan line spacing
    - Adjustable voxel count per box
    - Create a defect at a specific location
    - Extend capabilities of existing/new sensors
- Nonfunctional Requirements
    - The printer will print at the appropriate speed.
    - The printer will be located in a secure lab.
    - The laser will operate under safe conditions.
    - The printer will function reliably.

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

Dr. Timothy Bigelow, Associate Professor of Electrical and Computer Engineering at Iowa State University, has served as the faculty advisor for this project. Professor Bigelow has and continues to provide excellent guidance, technical advice, and other assistance to this project. Professor Bigelow has also provided the necessary funding for this project. His assistance is greatly appreciated and needed in order to complete this senior design project. Additionally, Franklin Sullivan is an undergraduate student that assisted Dr. Bigelow last semester by adding helpful comments to the last senior design group's code.

1.2 Problem and Project Statement

Problem Statement:

The general problem is that Professor Bigelow and his graduate students are having issues debugging and modifying the C# implementation of software for their 3D Printer that they use to create metal objects for nondestructive evaluation

Solution Statement:

The general solution is to rewrite the code using MATLAB, which is software that the users are familiar with and conduct tests to make sure that the software works properly.

1.3 Operational Environment

The end product will be used inside a lab in the Applied Sciences II building. The metal prints will be done in a sealed container filled with Nitrogen to prevent oxidation. Due to the possibility of excess nitrogen venting into the lab, training will be required to prevent suffocation by the operators of the 3D printer.

1.4 Requirements

Functional Requirements:

- Refactor and convert source code from C# to Matlab
- Be able to print 3 cm^3 metal blocks
- Adjustable laser scan line spacing
- Adjustable voxel count per box
- Create a defect at a specific location for nondestructive evaluation testing
- Extend capabilities of existing/new sensors

Non-functional Requirements:

- The printer will print at the appropriate speed.
- The printer will be located in a secure lab.
- The laser will operate under safe conditions.
- The printer will function reliably.

Environmental Requirements:

- Metal Printer operates in a nitrogen rich environment
- Metal powder used is hazardous to the lungs
- Must be able to obtain proper respirators for lab work
- Obtain proper training in order to work in the lab

Economic Requirements:

- Most of the funding required for this project has already been used for items such as the metal printer, sensor systems, and other equipment
- May need additional funding for respirators, and other miscellaneous items such as auxiliary sensors

1.5 Intended Users and Uses

The intended use is for Graduate Students and Professors at Iowa State University to operate a metal 3D printer for the purposes of generating metal cubes and other structures.

1.6 Assumptions and Limitations

Assumptions:

- The 3D printer will be able generate small metal cubes
- Only one person will be controlling the 3D printer software at a time

Limitations:

- The 3D printer may not be able to generate more complex structures by the end of the project
- Tests will only be performed once personnel have received sufficient safety training
- Energy consumption is not a limiting factor, but minimizing energy consumption is preferable.

1.7 Expected End Product and Deliverables

One deliverable for this project will be refactored code to control the printer and generate G-Code. The refactored code will be delivered by the end of the Spring 2021 semester, May 7, 2021. The refactored code will consist of MATLAB functions that will function the same as the old C# code or better. The refactored code will also be functional instead of object-oriented.

Another deliverable for this project is the testing of the 3D printer. The 3D Printer tests and code revisions will be completed by the end of the Fall 2021 semester, December 17, 2021. The 3D Printer tests will consist of generating and printing metal cubes and debugging the refactored code. The end result of the testing and debugging will be a working 3D printer that will be easier for Dr. Bigelow and his Graduate students to modify as needed.

2 Project Plan

2.1 Task Decomposition

- Port the object oriented C# code from previous groups to function-oriented Matlab code
    - Slicer program which converts .stl files that represent 3D objects into gcode which represents instructions for the motors of the printer.
        - Slice the object into several layers
        - Slice the layer into several voxels with infill supports
        - Print a gcode line for each point of the layer
    - Printer Control Unit which manages the motors of the printer
        - Read a gcode file, translate the changes into motor movements
        - Write the motor movements to the corresponding COM port
- Build and code usable external sensors using arduino
    - Arduino will be programmed using C for the Arduino
    - Create and test different external sensors
    - Test and debug sensors within the actual printing machine

2.2 Risks And Risk Management/Mitigation

The only real risks that exist in the printing process are the laser blinding eyesight, the room becoming too enriched with nitrogen, and the aluminum powder that is hazardous when inhaled. The laser risk is mitigated by the shield door being correctly placed on the machine before printing is started. The risk created by the nitrogen can be mitigated using sensors that ensure the oxygen level of the room is normal. Air is 70% nitrogen to begin with so it only creates a risk when oxygen is driven from the room. The risk created by the aluminum powder is more severe and may be difficult to deal with because it is required for printing to be completed. The only option is to be cautious when measuring the aluminum powder and ensuring that an excess is not created. As discussed in the functional requirements, being able to create a defect at a specific location is intentional, and therefore would not be classified as a risk. However, defects created unintentionally would need to be mitigated by monitoring the printer's progress and aborting execution when necessary. A review of the software and hardware may need to be taken if such a situation occurs. Whoever operates the printer needs to have the technical knowledge needed to operate the printer to ensure that no user inputs are incorrect and that the printer has been set up properly to begin printing. There are not any procedures or products that would mitigate this risk, rather just precautions and proper steps to be taken.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Port the object oriented C# code from previous groups to function-oriented Matlab code
- Build and code usable external sensors using arduino
- Print metal cube

2.4 Project Timeline/Schedule

Action/Task	Completion Date
Finish porting code	5/7/2021
Be ready to test working sensors	5/7/2021
Finish laser control	12/03/2021
Ensure sensors are working	12/03/2021

Table 1: Project Timeline/Schedule

2.5 Evolution of Project Design Since CPRE 491

One aspect of the design of our project that has changed since the first semester is how we give our Slicer parameters that represent the metal cube. We first decided on using a GUI to implement cube parameters, but since then, we have also added a config file. Now, you can give this file as a parameter to the slicer_ctrl.m function and it will feed the cube parameters to the rest of the slicer functions.

When we started communicating with the laser in CPRE 492, we found that the PC would have to maintain both a serial connection and a TCP connection. Previous iterations had relied on only storing baud rate and serial port within the text file. On top of that, the laser stored commands in uint8 bytes rather than strings, so the text file would have to denote that as well.

Printer Control was redesigned to use TOML files for their readability and support for hexadecimal bytes unlike the JSON file format. Matlab does not have native TOML support as of version r2020b, so an external Add-On was imported to handle parsing the data. That library on it's own had bugs and did not support syntax like hexadecimal bytes, so a fork of the project was made and added to the project.

With the knowledge that printing cubes would no longer be a required deliverable for this semester, efforts were transferred to cleaning and refactoring Printer Control for clarity and maintainability.

2.6 Personnel Effort Requirements

Action/Task	Time required for completion
Port the object oriented C# code from previous groups to function-oriented Matlab code	30-60 hours
Build and code usable external sensors using arduino	20-40 hours
Testing Hours	10-30 hours
Troubleshoot and Debug	10-30 hours

Table 2: Personnel Effort Requirements

2.7 Other Resource Requirements

- External sensors
- Arduino parts

2.8 Financial Requirements

n/a, all equipment is provided to us, and extra funding will be received upon getting the printer in working condition.

3 Design

3.1 Previous Work And Literature

There was a significant amount of research done by the two teams preceding this team on the project and as such we are inheriting much of and are largely constrained by their design decisions. Notably, these decisions entail the exact types of hardware and how the actual 3D printing takes place. The design of the printer came out to be a 2 part system including 1) a roller which deposits metal powder from a supply bed and 2) a laser which then sinters that metal into place. This system is seen in figure 1. The documentation for this work is available from archives of previous semesters or on request.

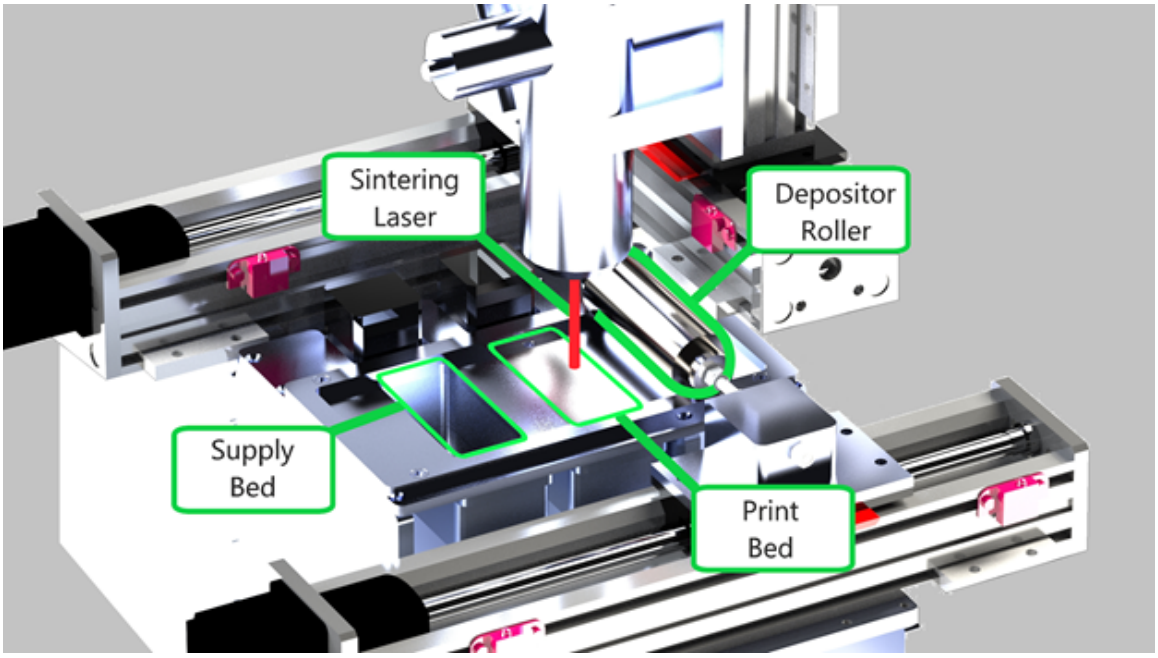

Figure 2: Previously designed Powder Metal Printing System

3.2 Design Thinking

Initially for the "define" stage of development, the team looked at all currently available options for the slicer and printer control programs. Notably, we sought to know if any research groups had already done some work in the area that we may be able to apply to our own system. Due to the unique nature of the hardware available to us, however, the team decided that any available research would not be directly applicable to the system as we seek to create it. In particular, a previous group had looked into a printer owned by CIRAS on campus. While being related, the printer is much faster, larger, and does not move with a 2 axis motor system.

The two main approaches the team developed in the "define" stage of development then were to either a) stick with the previous team's software approach or b) start anew and rewrite the software from scratch in MATLAB. Both approaches had some benefits and drawbacks.

Notably for option a, the team could write software in a much more clean and clear way insofar as we would be the ones who are the original authors and therefore have the best understanding of the code. Further, the previous team's approach appeared to have some issues as far as extensibility that may hold back the approach at the end. In terms of drawbacks, this approach would take significantly more time and effort in how the team would have to look into alternative ways of creating a slicer program from scratch, for example.

As for option b, the team saw the main benefit as being able to work off the previous team's work and not have to rewrite the entire software. This would mean far less work as far as reaching the mandatory goal Dr. Biglow laid out. Reaching the stretch goals may become an issue however, and there is a possibility there was a mistake in the original approach which makes it untenable. After this "define" stage, the team opted for option a.

In the "ideate" stage the team came up with some ideas such as:

- Breaking gcode generation functionality into smaller, more manageable chunks
- Develop a new algorithm for generating infill voxels (to be more in line with requirements)
- Create a more intuitive way for defects to be placed
- Create more options for types of shapes and potential defects

3.3 Proposed Design

The software design of the project consists of 3 main parts: the slicer, the printer control unit, and the sensor system. First, the slicer program divides up a 3D object .stl file into a series of instructions for the printer in a gcode file. Then the printer control unit will read this file line by line, and translate the instructions to commands to send to the motor controllers that move the laser and the printer bed. The sensor system will check that the conditions within the chamber are nominal before any action is conducted.

To print a 3cm cube, the 3D printer can't just print the layers of material that make up the walls of the cube. To provide support, the slicer program must divide up the object into layers with support structures called infills. The image below illustrates the "scanline rasterization" method that the infill pattern must be drawn in.

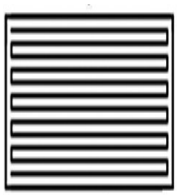

Figure 3: Example voxel

We will be working with 2 standards. The slicer works with the STL files, an ASCII text file format produced by most 3D CAD software defined in the ISO / ASTM52915 - 20 standard. The other standard that we will be handling is the gcode file defined in the ISO 6983 standard. Because of the nature of our hardware, we may have to deviate from the gcode standard, as our printer may need to support actions not specified within the standard.

The printer control unit acts as the brain of the printer. Each instruction in a gcode file maps to certain motor movements. The Velmex Motor Controllers (VXM) which handle the printer bed and laser communicate with the computer via COM ports.

The sensor system is composed of an oxygen, temperature, and pressure sensor connected to an Arduino. This system is crucial because of the specific conditions within the printer's chamber that are necessary for operation. The printer operates in a pressurized, nitrogen filled environment, to prevent the high temperature laser from igniting any oxygen. All three sensors must detect a safe environment before any printer actions are performed.

So far our group has implemented various helper functions for the slicer and printer control units. More specifically, the slicer group has completed gen_voxel(), which generates a tileable square with an infill pattern. The printer control group has completed gcode_reader() which reads in the gcode and maps them to VXM commands. Both have been functionally tested and confirmed to have expected results. The sensor team has acquired the arduino and sensors and is actively working on implementing the sensor functions.

These functions are easily testable as their results can be verified through console log messages. Ideally we want to complete as many parts of the project that can be tested remotely first. In-person testing with the hardware will require setting a meeting at the lab in ASCII, and completing a lab safety certificate.

3.4 Technology Considerations

The team is constrained to the technology made available to the team at the beginning of the project as well as in the requirements. Notably this means the team must use the available VXM system, the Arduino with the oxygen and pressure sensors, lab computer, laser, and other such hardware. In terms of software the team is constrained by the requirements to use MATLAB to refactor the code.

3.5 Design Analysis

The team is in the beginning stages of implementing the proposed design in 3.3 and it does not appear to need any modifications or iterations at the moment. We are prepared to make any alterations as additional information arises.

3.6 Development Process

The group is moving forward with an Agile approach for development of the system. We are utilizing the system provided by GitLab to track issues and label them according to their respective

milestones (sprints). While the team will not be performing Agile in the exactly proper way insofar as daily meetings go, the team is using Agile as a starting point and adapting it to our needs.

The team decided on Agile because it is a system some members were quite familiar with from previous experience and those members felt that it would be an effective way of organizing the work for a project of this scale. The project has many concurrent pieces of work going at the same time, and so Agile provides an effective way of keeping track of these sets of tasks and making sure the team is completing each set of tasks in a desirable amount of time in accordance with the other sets of tasks.

3.7 Design Plan

Although we are inheriting this project from 2 other senior design groups, Dr. Bigelow has authorized a rewrite of the software in MATLAB. The main non-functional requirement asked of us by Dr. Bigelow is “clarity”. We will use this opportunity to redefine design choices and write code under an imperative programming paradigm instead of the object oriented one used by previous senior design groups.

The figure below shows the architecture of the project, as well as how the data flows from one module to another:

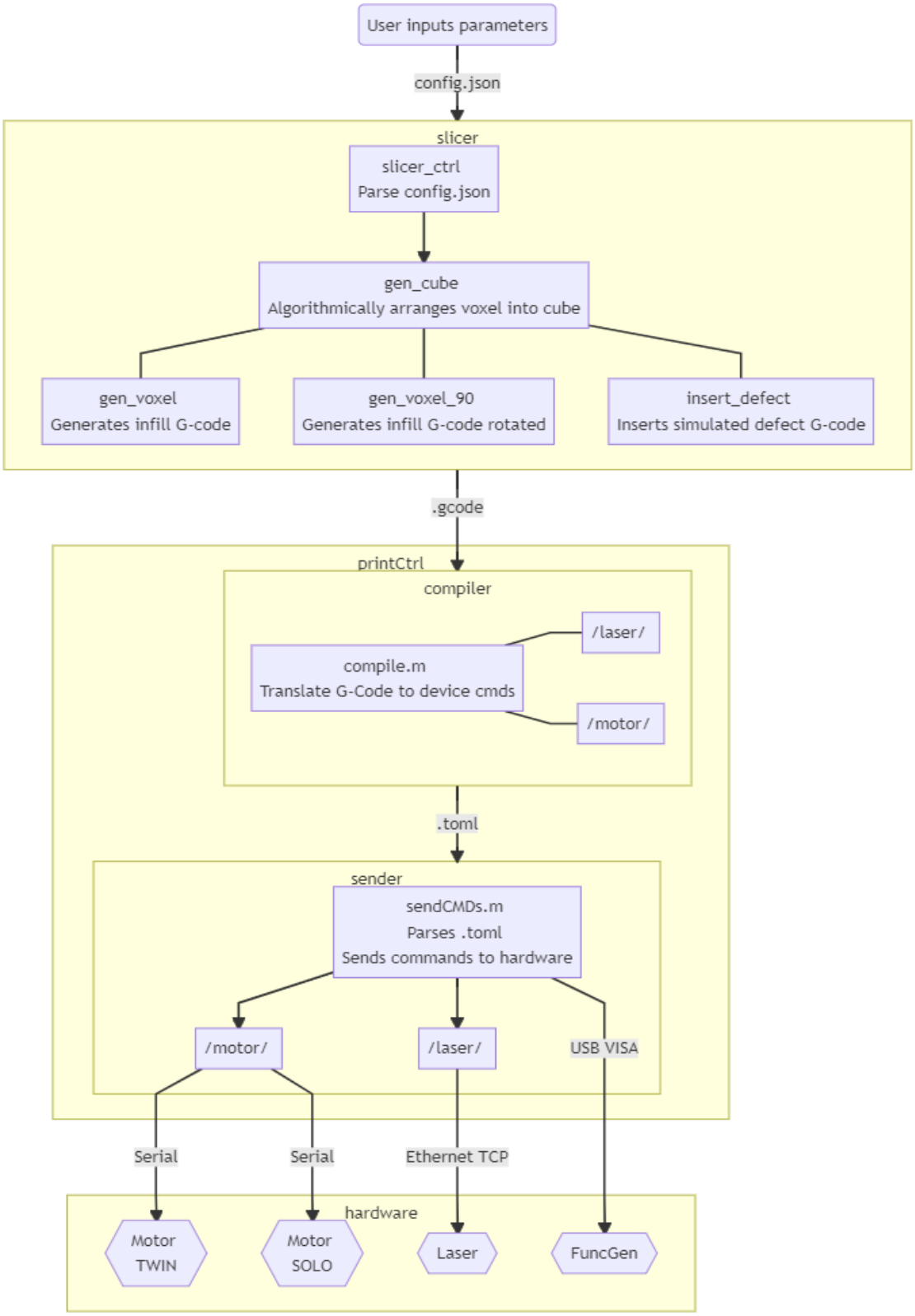

Figure 4: Software Flowchart

This project architecture is largely based on the previous group's i.e. there are three modules used to break up the software into parts. Each module has an input and output:

Module	**Input(s)**	**Output**
Slicer	Cube specifications	Gcode File
Sensor System	Oxygen Sensor Data Temperature Sensor Data Pressure Sensor Data	Sensor Status
Printer Control	GCode File Sensor Status	Velmex Motor Control

Table 3: Project Modules

Importantly the Slicer is independent from the other two modules; it is run separately before the print whereas the Sensor System and Printer Controller are run together during execution of the print. As a result, both the Slicer and Printer Controller will have two separate interfaces. Further, the Sensor System will be run asynchronously and will send information about the status of the sensors if an error occurs when the sensor safety conditions are not met.

4 Testing

4.1 Unit Testing

Unit testing was not done for this project because the deliverable to use the laser to print cubes was removed. However, custom TOML and G-code files in the /testFiles/ directory were written to test the printCtrl functions and Slicer functions respectively.

4.2 Acceptance Testing

The team will demonstrate that the design requirements are being met for both function and non-functional requirements by demoing the project to our client (Professor Bigelow) and showing that each component of our project is meeting the design constraints and requirements. Such functional testing, nonfunctional testing, and modeling and simulations that we will demo to the client are the following.

Functional Testing:

- Test slicer program
- Test gcode to vxm motors program
- Test infill algorithm
- Test that vacuum chamber seals correctly
- Test camera
- Test temperature and pressure sensors
- Test stepper motors
- Test Main Menu GUI
- Test Slicer GUI

Nonfunctional Testing:

- Test the speed at which printing is taking place
- Test structural integrity of part being printed

- Test that the GUI is easy to use

Modeling and Simulation Testing:
- Simulate how the gcode will trace the printing process
- Simulate how the actual 3D printer will trace the object

4.3 Results

Initial testing of the overall system and printer in this project is completed. We have done testing on functions that are used in the slicer file with success. The printer can communicate with the COM ports for the vxm motors. We were able to successfully communicate and send commands to the laser arm through an Ethernet cable. We learned that it was best to approach this project in iterative steps, starting with developing simple implementations of the project and working our way up to more complicated implementations. This project required a lot of doing our own research so it is best to start with simple implementations and increase with complexity as we learn more. After debugging the software and overcoming setbacks, our results are mostly positive. The only initial goal that we did not complete was actually printing a metal cube. This was not accomplished due to insufficient funding for this semester.

5 Closing Material

5.1 Conclusion

We have accomplished all of our goals set in place at the beginning of the first semester. We have the motor systems fully functional, as well as have the algorithm in place for the actual printing process. We also have an internal oxygen, temperature, and pressure sensor set up. We finished the defect algorithm for the Slicer. Initial testing on the laser was finished and we were able to demonstrate that the laser can trace the cube pattern.

5.2 References

"Makers of MATLAB and Simulink," *MathWorks*. [Online]. Available: https://www.mathworks.com/. [Accessed: 10-Mar-2021].

S. Gaisford, R. J. Crawford, and J. L. Throne, "Powder Bed," *Powder Bed - an overview | ScienceDirect Topics*, 2017. [Online]. Available: https://www.sciencedirect.com/topics/engineering/powder-bed. [Accessed: 10-Mar-2021].

6 Appendices

6.1 Operation Manual

First, ensure that Matlab r2020b is installed along with the Instrument Control toolbox. Other versions are incompatible with the function generator because of deprecated functions.

Load Matlab and run startup.m, this will automatically add project files to path and install the matlab-toml-forked add-on.

Switch on all three VXM motor controllers. The jog light should turn off when Matlab establishes serial communications when running a program. If you encounter issues with motors, verify that the USB-Serial cables are connected to the motors properly.

If you only want to test the laser with the pilot or targeting laser, ensure that the laser interconnect is unplugged before turning on the laser's mains power. Connect the ethernet cable to the Laser. If you encounter any problems, Wireshark and R4GUI2.0.exe can be used to debug Laser communication. See the R4GUI manual for details.

To connect the function generator to Matlab, start the function generator and obtain the USB address. Add the USB address to the appropriate Matlab file to connect to the function generator.

Printer Control

The compiler and the sender fall under the printer control unit and are located in the \printCtrl\compiler\ and \printCtrl\sender\ directories respectively. Bundled with the project is also printCtrl.pdf, which contains more in-depth information about Printer Control.

The user should edit CONFIG.m to configure printCtrl's configuration file with ports and addresses. The instructions on how to load serial port and tcp client information can be found in Matlab documentation.

printCtrl.m is a simple script that can be used to run the slicer, compiler, sender, and extraneous utility functions. Entering "printCtrl" into the matlab console will bring up a CLI where one of the following commands can be entered:

```
>> printCtrl
At any time, press Ctrl+C to stop the program.
Enter one of the following commands:
   setup    to check connected devices.
   slice    to run slicer and generate .gcode
   compile  to compile .gcode into .toml
   test     to run testing functions
   send     to run the command sender
>> test
Enter one of the following commands:
   arbWaveForm     to test function generator.
   freeLaserTCP    to send a laser command
   freeMove        to test an individual motor
   freeVXMCMD      to run a VXM cmd (useful for -0 cmds)
   printerHome     to move the motors to home.
>> exit
Unknown command. Ending program.
>>
```

Commands will then walk through the various input parameters, prompting the user for either text, or variable input.

printCtrl.m will exit after each walkthrough to allow the user to run individual parts of the program in isolation. It is also suggested that users manually verify G-Code and TOML files during testing.

Slicer

All files regarding the Slicer are located under the directory \stl_to_gcode\Slicer. Under this directory, you first want to edit the config.json file. This file contains the cube parameters. Change the parameters to your liking.

- "**filename**" is the path to and name of the gcode file that will be generated by the slicer.
- "**length**" is the total length of the cube you want to print in cm.
- "**width**" is the total width of the cube you want to print in cm.
- "**height**" is the total height of the cube you want to print in cm.
- "**layer_height**" is the height between layers in the cube in cm.
- "**voxels_per_length**" is the amount of voxels you want per layer length-wise.
- "**voxels_per_width**" is the amount of voxels you want in the cube per layer width-wise.
- "**voxel_margin**" is the margin between voxels in cm.
- "**voxel_padding**" is the padding between voxels in cm.
- "**defects**" is a 2D array of defects, each defect with its own parameters. If any defect has a parameter with a value of -1, that defect will be ignored and not inserted into the cube. You may add as many defects as you want to this 2D array.
    - "**defect_x_origin**" is the x origin coordinate point of the defect.
    - "**defect_y_origin**" is the y origin coordinate point of the defect.
    - "**defect_z_origin**" is the z origin coordinate point of the defect.
    - "**defect_width**" is the total width of the defect in cm.
    - "**defect_length**" is the total length of the defect in cm.
    - "**defect_height**" is the total height of the defect in cm.

After you have edited the config.json file to your liking, run the function slicer_ctrl("config.json") in Matlab's IDE terminal. This will run the necessary functions to generate the gcode file describing the cube with your own custom parameters.

Test Files

Bundled with the project are test files for testing out printer functionality. The following section will walk through printing an example cube from start to finish using these test files without printCtrl.m:

The Slicer's default config.json has the parameters for a 3cm cube with no defects. In /stl_to_gcode/Slicer/config.json, the filename should be set to "./testFiles/test.gcode". Run the following command in the Matlab console to generate G-Code:

```
>> slicer_ctrl("./stl_to_gcode/Slicer/config.json")
```

You should see test.gcode within the testFiles directory. Review it before continuing.
Next, we run the compiler to translate the G-Code into printer commands:

```
>> compile("./testFiles/test.gcode", "./testFiles/test.toml")
```

The testFiles should now have a test.toml file, which contains a list of operations, with each operation denoted by [[printerAction]]. After verifying that the TOML looks similar to other examples in /testFiles/, run the following command to start the print.

>> sendCMDs(“./testFiles/test.toml”)