The DaR3D System: Detecting Defects for 3D Printed Parts

A Major Qualifying Project submitted to the faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements of the degree of Bachelor of Science

Written By:

Madi Eisenhour (ME, RBE)

Mark Forte (CS)

Ryan Malkowski (ME)

Advisors:

Professor David C. Brown (CS)

Professor Pradeep Radhakrishnan (ME, RBE)

May 2, 2022

Abstract

Low cost and open-type Fused Deposition Modeling (FDM) 3D printers are widely available but can produce various defects. Parts slipping off the print bed, filament runout, warping, etc. are some of the common defects in such 3D printers. This can produce material and time losses. To minimize that, we researched several algorithms and developed DaR3D, a monitoring system to detect defects and alert the user. DaR3D's detection algorithm periodically acquires images through a webcam with controlled lighting, removes the background, and applies a normalized mean square error method to compare successive images. If the images have noticeable differences, the algorithm determines that a defect has occurred. The system was able to correctly identify slippage in prints with 89.6% accuracy, using samples from two different 3D printers and many different printed models. Future work will involve expanding the algorithm to cover more defects during printing.

Acknowledgments

The team would like to give special thanks to:

- Professors Pradeep Radhakrishnan and David C. Brown (Faculty Advisors), for their continued support and guidance throughout the project.
- Participants of the student survey and focus group interviews, for their valuable insight into our final product.
- Daler Kang, for his participation as an actor in the DaR3D video explanation.
- Barbara Furhman, Administrative Assistant, MME Department, for her help with purchasing equipment for the project

Table of Contents

List of Figures

Authorship

Chapter	Author	Editor (s)
I. Abstract	All	All
II. Acknowledgments	All	All
III. Table of Contents	All	All
IV. List of Figures	All	All
1. Introduction	All	All
2. Problem Statement and Goals	All	All
3. Terminology	Mark Forte Ryan Malkowski	All
3.1. 3D Printer Terminology	Ryan Malkowski	All
3.1.1. Print Bed	Ryan Malkowski	All
3.1.2. Nozzle	Ryan Malkowski	All
3.1.3. G-Code	Ryan Malkowski	All
3.1.4. Slicer	Ryan Malkowski	All
3.1.5. Fused Deposition Modeling	Ryan Malkowski	All
3.1.6. PLA	Ryan Malkowski	All
3.1.7. Common Origin of Printer and Slicer	Ryan Malkowski	All
3.1.8. Front View of the Printer	Ryan Malkowski	All
3.2. Other Hardware Terminology	Mark Forte	All
3.2.1. Raspberry Pi	Mark Forte	All
3.2.2. Raspberry Pi Camera	Mark Forte	All
3.3. Software Terminology	Mark Forte	All

3.3.1. OpenCV	Mark Forte	All
3.3.2. OctoPrint	Mark Forte	All
3.3.3. OctoLapse	Mark Forte	All
3.3.4. Photogrammetry	Mark Forte	All
3.3.5. Flask	Mark Forte	All
3.3.6. Canny Edge Detection	Mark Forte	All
4. 3D Printing Defects	Ryan Malkowski	All
4.1. Common Defects	Ryan Malkowski	All
4.2. Defects Relative to this Project	Ryan Malkowski	All
4.2.1. Slippage	Ryan Malkowski	All
4.2.2. Failure to Extrude	Ryan Malkowski	All
4.2.3. Stringing	Ryan Malkowski	All
4.2.4. Warping	Ryan Malkowski	All
4.2.5. Layer Shifting/Misaligned Layers	Ryan Malkowski	All
5. Literature Review	Madison Eisenhour Mark Forte	All
5.1. Analysis of Hardware Options	Madison Eisenhour	All
5.1.1 Sensors	Madison Eisenhour	All
5.1.2 Background and Lighting	Madison Eisenhour	All
5.2 Similar Software Solutions	Mark Forte	All
5.2.1 The Spaghetti Detective	Mark Forte	All
5.2.2. 3DPrintSaviour	Mark Forte	All

5.2.3. Michigan Technological University Project	Mark Forte	All
6. Methodology	All	All
6.1. Hardware Development	Madison Eisenhour	All
6.1.1. Preliminary Decisions	Madison Eisenhour	All
6.1.2. Lighting and Background	Madison Eisenhour	All
6.1.3. Camera Limit Testing	Madison Eisenhour, Ryan Malkowski	All
6.1.4. Camera and Light Fixture	Ryan Malkowski	All
6.2. Software Development	Mark Forte	All
6.2.1. Algorithm Decisions	Mark Forte	All
6.2.2. Machine Learning	Mark Forte	All
6.2.3. 3D Model Analysis	Mark Forte	All
6.2.4. 2D Model Analysis	Mark Forte	All
6.2.5. Environment Setup and First Tests	Mark Forte	All
6.2.6. Outline Extraction Through Edge Detection	Mark Forte	All
6.2.7. Time-Lapse Image Improvements	Mark Forte	All
6.2.8. New Outline Extraction Methods	Mark Forte	All
6.2.9. Combined Method of Outline Extraction	Mark Forte	All
6.2.10. Building the Comparison Model	Mark Forte	All
6.2.11. Outline Extraction with New Edge Detection	Mark Forte	All

6.2.12. Outline Extraction and Initial Comparison Test	Mark Forte	All
6.2.13. Other External Solutions	Mark Forte	All
6.2.14. Further Outline Extraction Struggles	Mark Forte	All
6.2.15. Changing Goals	Mark Forte	All
6.2.16. Initial NRMSE Algorithm Development	Mark Forte	All
6.2.17. Background Removal Attempts	Mark Forte	All
6.2.18. Automatically Blocking out the Extruder	Mark Forte	All
6.2.19. NRMSE Algorithm with Empty Print	Mark Forte	All
6.2.20. Using Rolling Average and Outliers to Detect Slippage.	Mark Forte	All
6.2.21. Connecting the Algorithm to OctoPrint	Mark Forte	All
6.2.22 Runout Testing	Mark Forte	All
7. Final Design	All	All
7.1. Hardware Design	Madison Eisenhour, Ryan Malkowski	All
7.1.1. Hardware Requirements	Madison Eisenhour	All
7.1.1.1. Environment Requirements	Madison Eisenhour	All
7.1.1.2. Camera and Fixturing Requirements	Madison Eisenhour	All
7.1.2. Environment Setup	Ryan Malkowski	All

7.1.2.1. Printer Surroundings	Ryan Malkowski	All
7.1.2.2. Lighting	Madison Eisenhour	All
7.1.2.3. Background and Print Bed Color	Ryan Malkowski	All
7.1.3. Camera and Fixturing	Ryan Malkowski	All
7.1.3.1. Camera Selection	Ryan Malkowski	All
7.1.3.2. Camera Fixture	Ryan Malkowski	All
7.1.3.3. Foot Component	Ryan Malkowski	All
7.1.3.4. Camera Bracket	Ryan Malkowski	All
7.1.3.5. Light Adapter	Ryan Malkowski	All
7.1.3.6. Base of the Fixture	Ryan Malkowski	All
7.1.3.7. Full Fixture Assemblies in Use	Ryan Malkowski	All
7.1.3.8. Monoprice Fixture	Ryan Malkowski	All
7.1.3.9 RepRap Fixture	Ryan Malkowski	All
7.2. Software Design	Mark Forte	All
7.2.1. The DaR3D System	Mark Forte	All
7.2.2. Image Analysis Algorithm	Mark Forte	All
7.2.2.1. Image Analysis	Mark Forte	All
7.2.2.2. Statistical Analysis	Mark Forte	All
8. Algorithms	Mark Forte	All
8.1. Canny Edge Detection	Mark Forte	All
8.2. Background Removal	Mark Forte	All
8.3. Difference Detection	Mark Forte	All
8.4. Color Extraction	Mark Forte	All
8.5. Combined Method	Mark Forte	All

8.6. Canny Edge Detection V2	Mark Forte	All
8.7. NRMSE Comparison Algorithm	Mark Forte	All
8.8. NRMSE with Empty Time Lapse	Mark Forte	All
8.9. Rolling Average Outlier Detection Algorithm	Mark Forte	All
9. Experimentation	Mark Forte	All
9.1. Experiment Definitions	Mark Forte	All
9.1.1. Models	Mark Forte	All
9.1.2. Cameras	Mark Forte	All
9.1.3. Camera Positions	Mark Forte	All
9.1.4. Printer Gantry Position	Mark Forte	All
9.1.7. Lighting	Mark Forte	All
9.2. Experiment 1	Mark Forte	All
9.3. Experiment 2	Mark Forte	All
9.4. Experiment 3	Mark Forte	All
9.5. Experiment 4	Mark Forte	All
9.6. Experiment 5	Mark Forte	All
9.7. Experiment 6	Mark Forte	All
9.8. Experiment 7	Mark Forte	All
9.9. Experiment 8	Mark Forte	All
9.10. Experiment 9	Mark Forte	All
9.11. Experiment 10	Mark Forte	All
9.12. Experiment 11	Mark Forte	All
9.13. Experiment 12	Mark Forte	All
10. Evaluation	Madison Eisenhour	All

10.1. Requirements	Madison Eisenhour	All
10.1.1. System Explanation	Madison Eisenhour	All
10.1.2. Participants	Madison Eisenhour	All
10.1.3. Student Survey	Madison Eisenhour	All
10.1.4. Expert Interviews	Madison Eisenhour	All
10.2 Procedure	Madison Eisenhour	All
10.2.1. Student Survey Procedure	Madison Eisenhour	All
10.2.2. Expert Interviews Procedure	Madison Eisenhour	All
10.3. Results	Madison Eisenhour	All
10.3.1. Student Survey Results	Madison Eisenhour	All
10.3.2. Expert Interview Results	Madison Eisenhour	All
10.3.3. Analysis	Madison Eisenhour	All
11. Conclusion	All	All
11.1. Results	Mark Forte	All
11.3. Conclusions	Madison Eisenhour Mark Forte	All
11.2. Future Work	Ryan Malkowski	All
11.3 Broader Impacts	Ryan Malkowski	All

Chapter 1: Introduction

Today, fused deposition modeling (FDM) is the most common form of 3D printing. It has grown in popularity and has reached a point where consumers or hobbyists can purchase 3D printing machines for under $500. FDM is also very common in the world of manufacturing. Its main role is for rapid prototyping where a potential part can be modeled inexpensively and quickly (Palermo, 2013). This is due to the low cost of filament and short setup time.

While FDM has many benefits for printing models and prototypes, it does not work perfectly every time. It is still a relatively new technology and defects are a major concern when printing. In the worst case, a defect can ruin the structural integrity of the print, making it unusable. The time to 3D print something can be anywhere from under an hour to several days, so if the final product is unusable, all of the time spent printing will be wasted. This is especially true for cases where a print will run its full cycle but no part will be produced. This can be due to filament running out during the print or the extruder becoming clogged. Other more catastrophic defects may cause a mess of filament, which can damage the printer in addition to being a waste of time, money, and material. Currently, there is no widely adopted method to automatically detect a defect as it occurs in order to stop the print before material and time is wasted. The scope of this project was to create a system that could detect and recognize defects and alert a user when the defect occurs.

This report reviews all of the activity that was accomplished by this MQP team for the 2021-2022 academic year. It begins with the problem statement and goals of the project described in Chapter 2. Chapters 3 and 4 introduce the terminology used throughout the report as well as an explanation of the defects that occur during 3D printing. This leads to chapter 5 which is a

review of existing literature pertaining to hardware and software development for existing defect detection research. Following this, the methodology can be found in chapter 6. This chapter tells a summarized story of the iterations and changes that took place over the course of the year. The final design of the system is discussed in more detail in chapter 7. Chapter 8 details algorithms that were considered or developed throughout the duration of the project. Next, the experiments are highlighted in Chapter 9. Here, specific experiments and tests that were completed throughout the duration of the project are recorded. Chapter 10 details evaluation via student surveys and expert interviews about the DaR3D system. Lastly, the report concludes with chapter 11, which includes a discussion of the results, future work, broader impacts, and the project experience.

Chapter 2: Problem Statement and Goals

2.1 Problem Statement

Low-cost and open-type Fused Deposition Modeling (FDM) 3D printers are widely available and often used for various prototyping projects. However, there are many things that can go wrong during the printing process which cause the completed printed parts to be defective. These defects can occur at any point during the print and can range in severity from only affecting the aesthetic of the print to having the entire print turn into a mess of filament. These defects are caused by conditions outside of the users' control or by issues they do not know to check for. Even small defects can waste valuable development time and material, or cause damage to the machine if not caught. Currently, there is no widely adopted system to automatically monitor the 3D printing process for defects. Developing such a system would save time and money for both small-scale prototyping projects and large scale automated FDM production.

2.2 Project Goals

Our goal for this project was to minimize the time and material cost of FDM defects by developing a system of software and hardware. The system would automatically monitor the 3D printing process and notify the user if a defect occurred during printing. The goal for the software was to write an algorithm that could use information about the printed part to detect and recognize defects as they happen, then notify the user about the defect. The goals for the hardware were to control the environment around the printer and gather information about the

print to effectively aid the algorithm in detecting and recognizing defects. An additional goal for this project was to make the final product modular so that the system could be adapted to many types of 3D printers and defects. The proposed system should be able to detect a number of defects. It should also be able to recognize what defect is occurring and notify the user as early as possible in the printing process. In addition, the system should work efficiently, be accurate, have low cost additional equipment, be easy to use, and not interfere with normal use of the 3D printer. Any additional equipment must be easily available or easy to manufacture.

Chapter 3: Terminology

This chapter presents some of the terminology that is used throughout the report. Its purpose is to establish any terms that are new to the reader and serve as a glossary of terminology to refer to while reading the report. This chapter breaks down into subsections that cover 3D printer terminology, other hardware terminology, software terminology, printer specific nomenclature, and camera terminology.

3.1 3D Printer Terminology

3.1.1 Print Bed

The print bed is typically a square or rectangular shaped plate that sits within the frame of a 3D printer. It is where the melted filament goes once it leaves the nozzle of the printer. The print bed must be leveled properly for the base of the print to properly adhere to it. The print bed also has heating elements to increase the temperature of the bed. This allows for the filament to cool at a slower rate which can increase bed adhesion and help reduce defects. Circled in red in Figure 3.1 is the print bed for the Monoprice printer. In this image the print bed has a layer of tan tape on top.

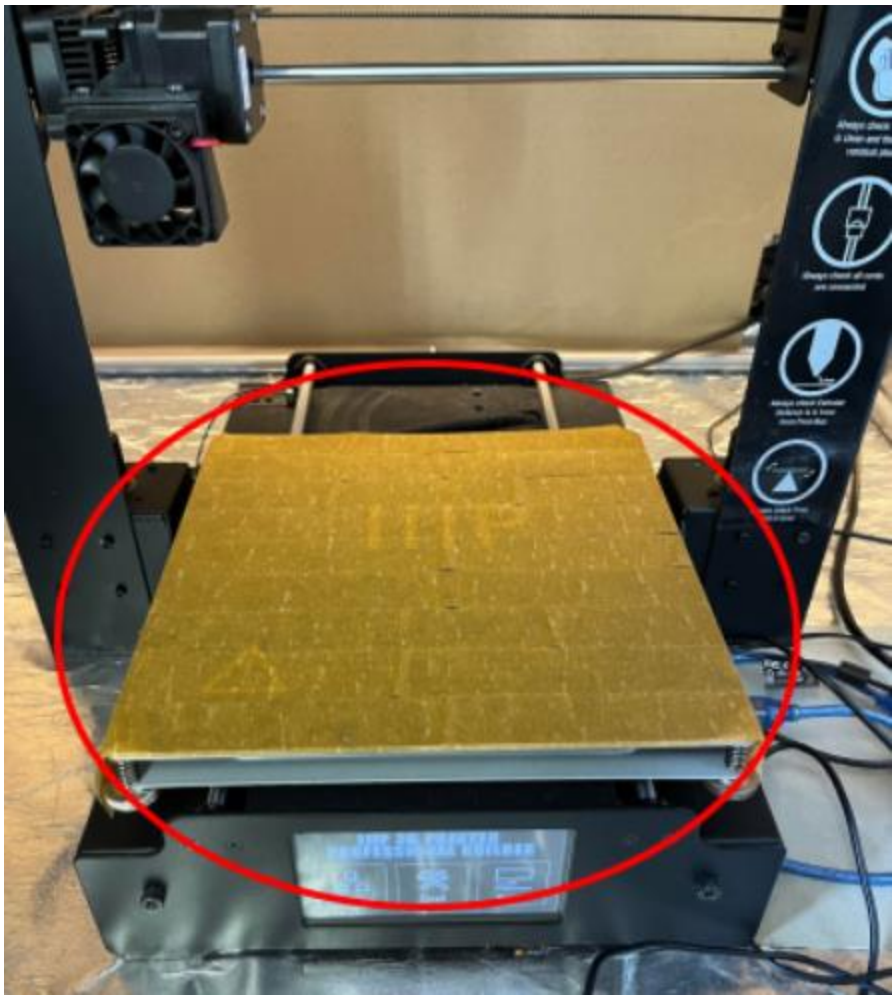


Figure 3.1: Print bed of the Monoprice printer

3.1.2 Nozzle (Extruder)

The nozzle is the part of the printer where the filament comes out. It is also known as the extruder or hot end of the printer. For FDM style printers, this nozzle heats up to a temperature above the melting temperature of the filament. Figure 3.2 shows the nozzle on the Monoprice printer that is used for this project.

Figure 3.2: Nozzle of the Monoprice printer

3.1.3 G-Code

G-code is a programming language that is designed specifically for computer-controlled machines. Its purpose is to deliver directions for machines to follow. Common systems that use G-code to function are computer numerical control (CNC) machines, such as mills and lathes. FDM 3D printers (see section 3.1.6) also utilize G-code to control things such as extruder movement, filament retraction and extrusion as well as heating the nozzle and print bed (Thomasnet, 2022).

3.1.4 Slicer

A slicer can be treated as the connection between a CAD model and the G-code that is required to 3D print a part. The slicing software inputs a CAD model saved in the form of an STL (Standard Tessellation Language) file and outputs the corresponding G-code. Within the slicing software, there are settings that can be changed depending on how the user wants the part to be printed. The software will write the proper G-code for the specified print settings. While G-code can be written manually, this process is very complicated and lengthy. The slicer is needed to generate the code automatically and accurately within seconds. The slicer we use for this project is Cura.

3.1.5 Fused Deposition Modeling

Fused Deposition Modeling (FDM) has 3 main components: the extruder, the print bed, and the filament driving gear. The extruder heats up to a temperature above the melting point of the material being used, then a driving gear pushes the material into the hot extruder. There the filament melts, extrudes out of the nozzle and gets deposited onto the print bed where it cools down and re-hardens. FDM builds a model or part by depositing successive layers of melted material. These layers harden and fuse to one another until the full model is built from the bottom to the top. Figure 3.4 is a diagram of the process of filament going through the hot end of the extruder. (All3DP, 2022).

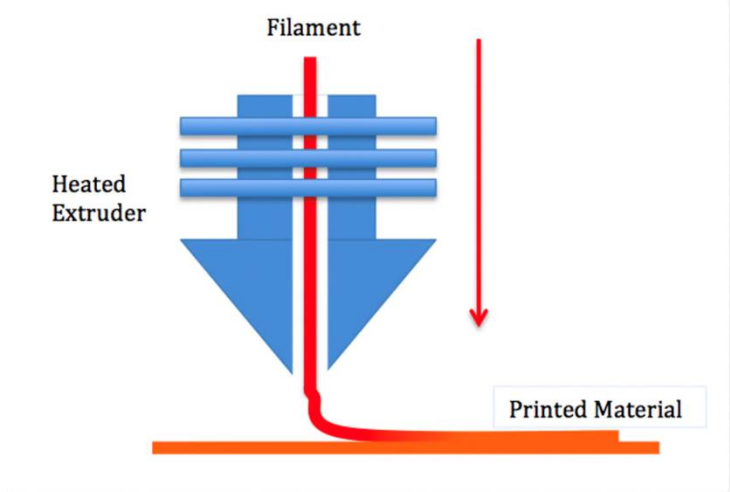


Figure 3.3: An example of the FDM printing process

3.1.6 PLA

PolyLactic Acid (PLA) is one of the most common filament materials for FDM 3D printing. It has a lower melting point than many other filament materials and it generally costs less. The nozzle temperature for PLA filament is around 200◦ C. The material commonly comes in a 1kg spool as seen in Figure 3.5.

Figure 3.4: A 1 kg spool of PLA

3.1.7 Common Origin of Printer and Slicer

There is a common origin that the slicing software refers to when producing the Gcode for the print. The location of this is in the bottom left hand corner of the bed when looking at the bed from a top view. Figure 3.6 shows a print bed with the positive x-axis, positive y-axis, and positive z-axis intersecting at the printer's origin.

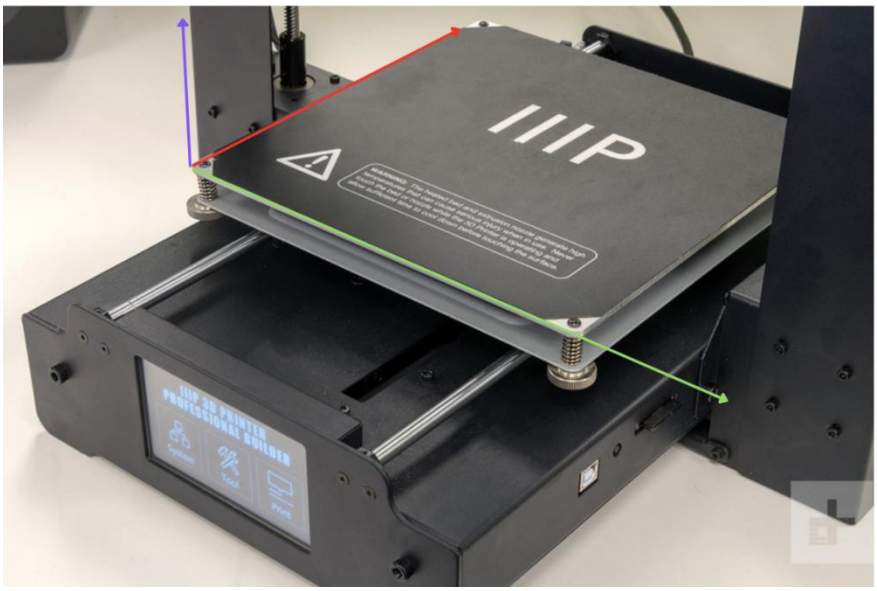


Figure 3.5: Print bed with positive x-axis (green), positive y-axis (red), and positive z-axis (blue) denoted as arrows. The origin is where all three of these arrows meet

3.1.8 Front View of the Printer

When we refer to the front of the printer we are speaking in regards to the view of the image seen in Figure 3.7. Key characteristics of the front of the printer are

- The screen at the bottom of the printer in this image
- The fan on the nozzle

- White lettering written on the frame

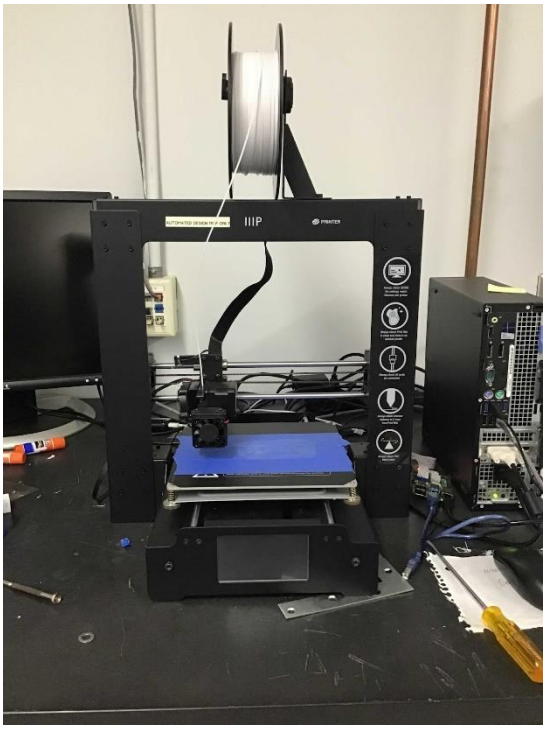


Figure 3.6: The front view of the printer

3.2 Other Hardware Terminology

3.2.1 Raspberry Pi

The Raspberry Pi is a small computer. It is commonly abbreviated as RPI. It can run a full operating system such as Linux, and has built-in ports for USB, Ethernet, HDMI, DSI display, DSI camera and I/O pins. Most models of the RPI are available for less than $60. This project uses a Raspberry Pi 3B+ (Raspberry Pi, n.d.).

Figure 3.7: A Raspberry Pi similar to the one used for this project

3.2.2 Raspberry Pi Camera

The Raspberry Pi Camera is an add-on board for the RPI. Unlike normal webcams, the Raspberry Pi Camera (and other similar products) connects to the RPI through the built-in DSI camera port. Raspberry Pi cameras do not typically come with a case but are just an exposed PCB. This means they are smaller and lighter than webcams but they also require careful handling and protection or they will break (Raspberry Pi, n.d).

Figure 3.8: An example of a Raspberry Pi camera

3.3 Software Terminology

3.3.1 OpenCV

OpenCV (Open Computer Vision) is the standard open source library for image processing, analysis, and other high-level image manipulations. It is written in C++ but there is a Python wrapper for it, which allows it to be used in Python programs. OpenCV contains tested implementations of many computer vision algorithms. This project uses OpenCV for all image manipulation and processing (OpenCV, 2022, April 15).

3.3.2 OctoPrint

OctoPrint is open source software for controlling 3D printers (see Fig. 3.10). It is installed on a computer which is connected to the 3D printer through USB. OctoPrint allows users to upload a GCode file via a web UI. The user can then remotely start and monitor a 3D print. OctoPrint has built-in support for community-created plugins that add a variety of features.

OctoPrint is commonly installed on a Raspberry Pi due to its low cost. The version of OctoPrint specifically for the Raspberry Pi is called OctoPi. This project uses OctoPi to manage the printer and make it easier to use (Octoprint, n.d).

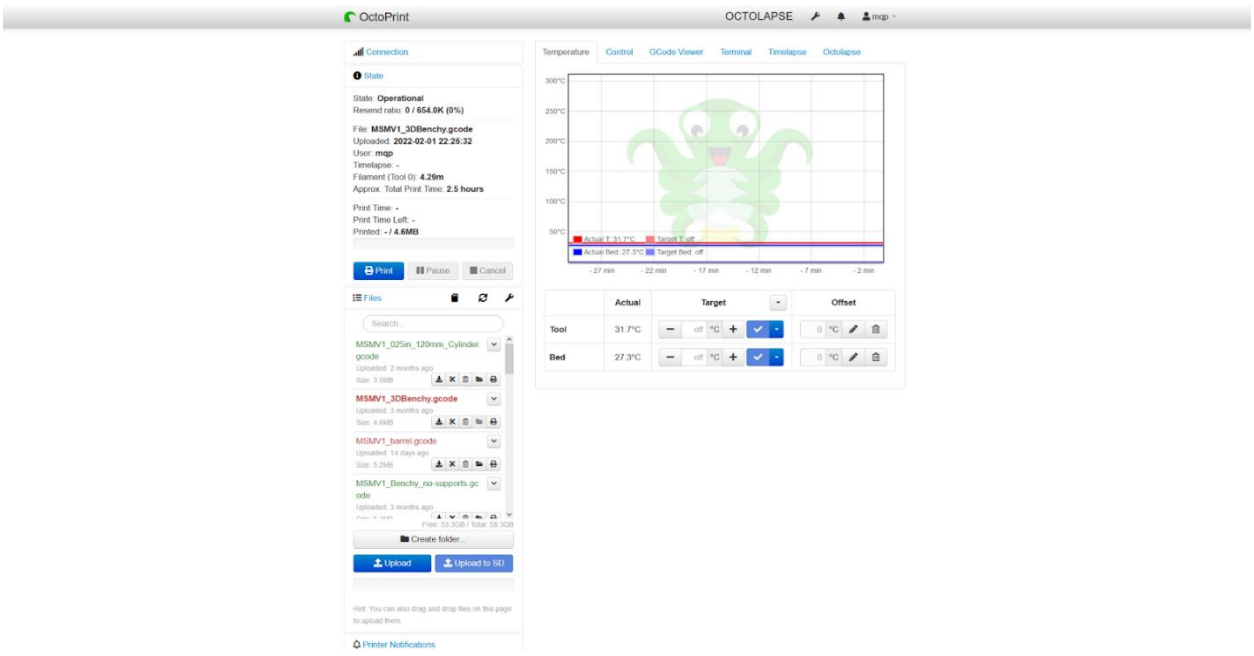


Figure 3.9: The Octoprint home screen used for this system

3.3.3 OctoLapse

OctoLapse is an open source, community-created plugin for OctoPrint. It creates time lapses of a 3D print. It works by taking pictures of a print after a specified amount of layers. It moves the extruder and the print bed into a user-defined position (such as back left) before taking each snapshot so that in the time lapse, the only thing that changes is the progress of the print. In order to use OctoLapse, users need to put custom code at the beginning of the GCode file. This can be done in the slicer before the CAD model gets sliced. The code defines features of the printer such as extruder size and layer height. It does this in order to optimize when pictures are taken. OctoLapse also allows for user-defined programs to be executed after each snapshot is taken. For our project, we can use this to transfer each snapshot of the print to the monitoring program to watch for defects in real time (Hochgesang, n.d.).

3.3.4 Photogrammetry

Photogrammetry is the science of using a photograph to make measurements. There are two main types of photogrammetry, aerial and terrestrial/close-range (Photogrammetry, n.d.).

3.3.5 Flask

Flask is a Python framework made to simplify the creation of web servers. A web server is used to facilitate communication over the internet. Flask allows the user to define endpoints, and functions to handle requests on those endpoints. An endpoint is a location that can receive Internet requests. This project uses a single endpoint to receive incoming images from OctoPrint (Flask, n.d.).

3.3.6 Canny Edge Detection

Canny Edge Detection is an algorithm implemented in OpenCV used to detect edges in an image. It is a multistage algorithm that uses intensity gradients with a threshold to determine what parts of an image are an edge (OpenCV, n.d.).

Chapter 4: 3D Printing Defects

The goal of this chapter is to describe some of the most common printing defects that occur in open type FDM 3D printers. From the list of common defects, there is a subset of defects that fit the scope of this project. These defects were the ones that we attempted to detect and recognize with the DaR3D system. In this chapter, we discuss what these defects are, their root causes and ways to troubleshoot them. By understanding how they occur, the team could replicate the defects for testing purposes.

4.1 Common Defects

There are a wide range of defects that can occur during the print. Some of these defects can be very minor, such as a slight distortion to a final surface finish. Others can be catastrophic to the integrity of the print, causing the final part to be visually unrecognizable and functionally unusable. Table 4.1 lists the common defects that occur during use of open type FDM printers. (All3DP, 2021).

Table 4.1: This is a table of common defects that are found in FDM 3D printing (All3DP, 2021)

Defect Name	*Description*
Failure to Extrude	This is when no material comes out of the nozzle during a print. It can be caused by snapped or stripped filament, a clogged nozzle, the printer's filament spool running out.
Nozzle too Close to the Bed	When the opening of the extruder is too close to the print bed, it can cause filament to get blocked or create a thin first layer.
Printhead Misses the Bed	When the nozzle deposits filament off of the print bed.
Slippage	When the print gets knocked out of position during a print due to a lack of adhesion with the print bed.

Defect Name	*Description*
Broken Supports	When supports fall/break during a print, resulting in overhangs to droop or sag.
Messy First Layer	When the first layer of the print does not print cleanly, this can result in compounding errors as future layers build.
Elephant's Foot	When the first layer of a print splays out due to the weight of the layers on top of it.
Warping (Lifted Edges)	When higher layers cool faster than lower layers causing shrinkage to pull the bottom edges of the print off the print bed.
Gaps in Between Infill and Outer Wall	When there is no material in between the inside of the print and the outer wall, causing lower structural strength of the print.
Infill Visible from Outer Wall	When the outer wall is too thin, the infill pattern can show through.
Misaligned Layers or Layer Shifting	When the print shifts part way through causing layers to not align after the shift occurs.
Missing Layers	When there are gaps along the outer wall of the print signifying a layer is missing.
Messy Fine Details	When small details have strings or missing components.
Ripples in Wall of the Print	When wave-like textures appear in the outer walls of the print.
Drooping Layers	When layers do not adhere well to each other causing sagging in between layers.
Deformed and Melted Print	When the print has melted parts that are not supposed to be there or the structure of the print is deformed.
Defect Name	*Description*
Stringing	When small strings cover parts or all of the print once it is finished.

4.2 Defects Relevant to this Project

The goal for this system was to be able to detect and recognize the five targeted defects listed below. These defects were targeted based on the customer's needs and were chosen from

table 4.1. In order to do this, we planned on creating a working method of detection for one defect at a time until we had a system of five defect detection methods. Furthermore, we needed to understand why and how each defect occurred. We then used this information to design ways to reproduce defects for testing the system.

- Slippage
- Failure to Extrude
- Stringing
- Warping
- Layer Shifting

4.2.1 Slippage

Slippage occurs when the first layer of the print does not properly adhere to the build plate, causing it to shift and sometimes get dragged by the nozzle as the extruder moves. There is not a specific time when slippage occurs: the bottom of the print can detach from the print bed at any time during the print (All3DP, 2021). Figure 4.1 shows before and after pictures of a part that experiences slippage. The first image shows the part correctly printing and the second shows it knocked out of place and falling off the back of the print bed.

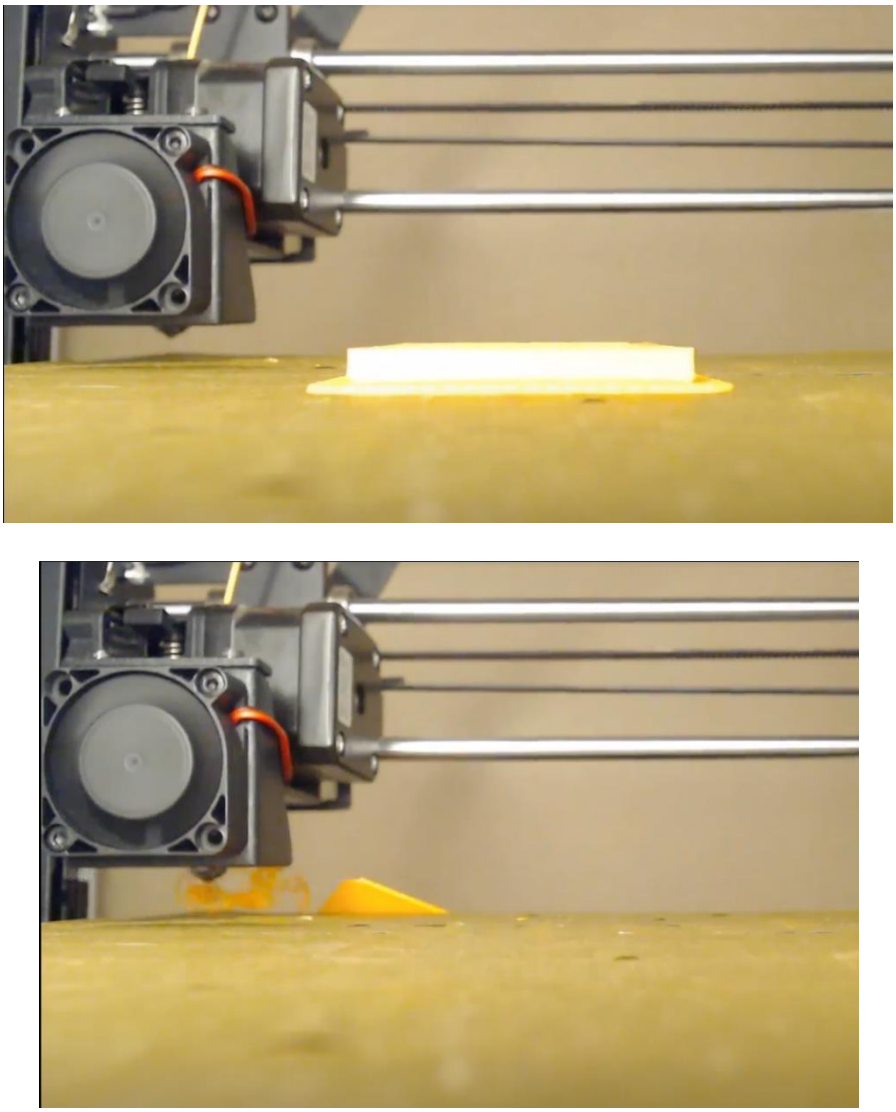


Figure 4.1: A print before and after slippage

The following are ways to minimize slippage:

- Use a heated bed, as heating the bed can reduce the gap in temperature between the newest layers and the previous.
- Use glue or tape on the print bed to help with adhesion, as this can help counter the upward pulling force of the newest layers.

- Control the room temperature and keep the prints away from fans and open windows, which can cool the print more rapidly than intended.

4.2.2 Failure to Extrude

When a printer fails to extrude, no filament comes out through the nozzle. The extruder continues to move as specified by the G-code, but no material is deposited onto the print bed. This defect has several root causes: the printer runs out of filament; the nozzle could be blocked or clogged; the filament could have snapped on the spool; or the filament could be stripped.

- Out of Filament
    - This defect occurs when the printer runs out of filament during a print, causing the nozzle to continue its path without any filament extruding.
- Blocked Nozzle
    - Material can be built up within the nozzle over time and cause the opening of the nozzle to not allow clear passage for material to extrude
- Snapped Filament
    - During the print, too much tension can be placed on the spool of filament as it is being driven through the nozzle. This can cause the filament to snap during a print, cutting off the extruder's access to the rest of the filament spool.
- Stripped Filament
    - The filament is gripped by two gears in the extruder and fed into the hot end. If the gears are placed too close to each other, then the gear teeth will wear down the

filament. That causes the filament to strip and slip on the gear, so that even when the gears are turning, the filament does not feed through them.

4.2.3 Stringing

Stringing, as seen in Figure 4.2, occurs when the material oozes and flows unevenly and often makes extra strings while filament retracts in the nozzle. These extra strings cause the phenomenon of stringing, a quite common defect found in 3D printing. This defect is caused by overheating, which causes the material's viscosity to decrease, allowing it to ooze unpredictably. The stringing happens when the nozzle moves from one location to another while the material is too hot, causing "strings" of material to form in areas that should be empty. This defect can also be caused by incorrect retraction settings. Retraction is when the filament gets pulled back into the nozzle while it is moving between extrusion locations (All3DP, 2021).

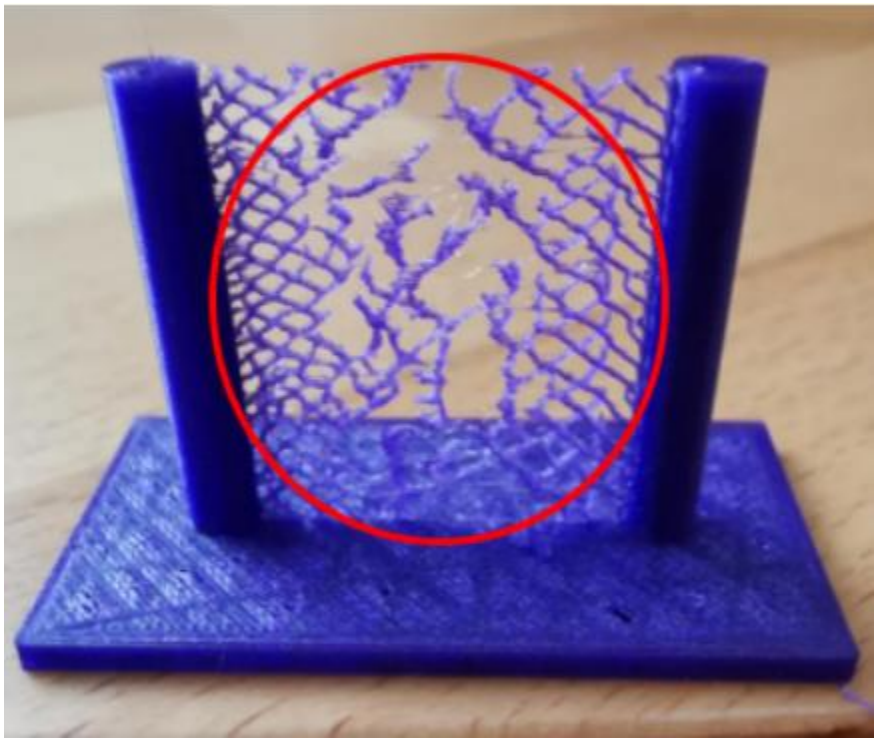


Figure 4.2: A part where stringing has occurred, as seen within the red circle (All3DP)

4.2.4 Warping

Warping is when corners or edges of the print lift off of the print bed during the print. Sometimes it can occur in the beginning of the print and other times it can occur during the print. It is primarily caused by improper timing between the extruding and rehardening of the print material. The forces that occur form internal cooling and heating of the material can pull up on the lower layers of the print and lift sections of the print off of the print bed (All3DP, 2021). An example of a part that has experienced warping can be seen in Figure 4.3.

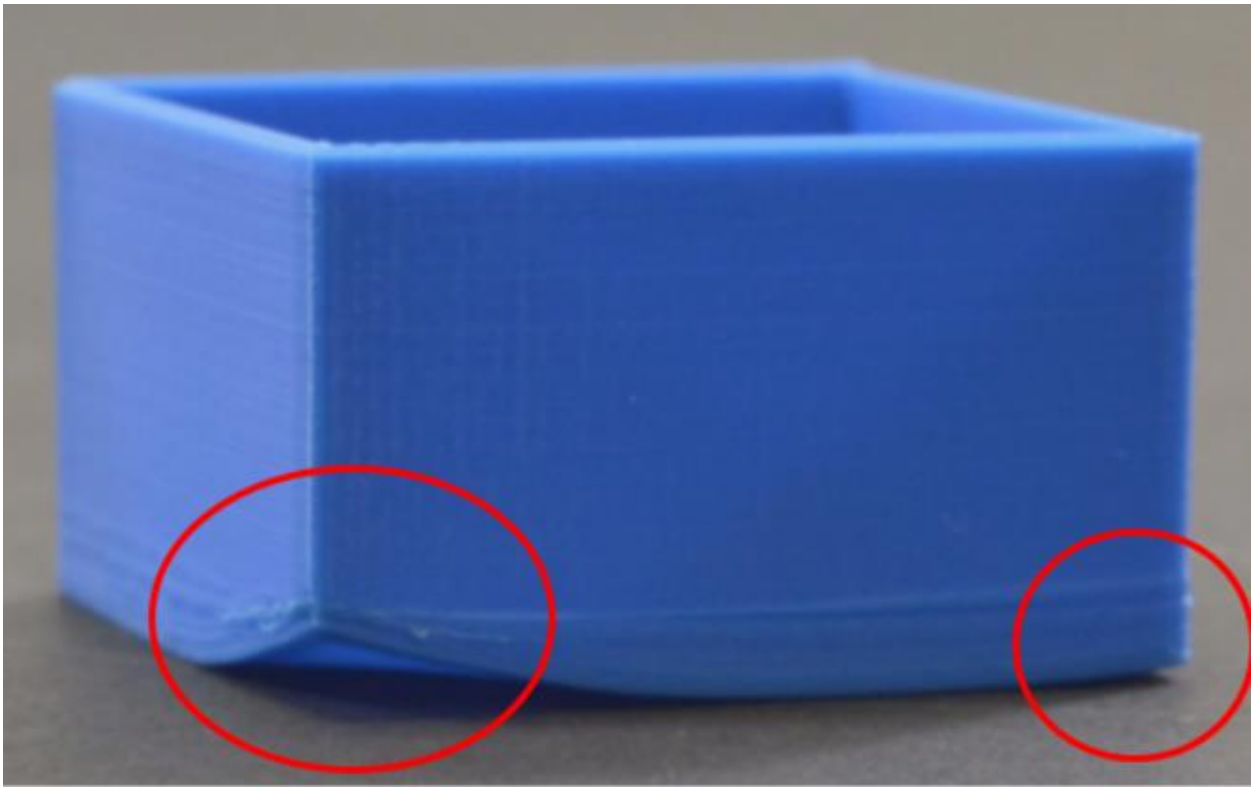


Figure 4.3: A part with warping. The corners have lifted off of the table as seen in the red circles

4.2.5 Layer Shifting/ Misaligned Layers

As seen in Figure 4.4, layer shifting occurs when the individual layers do not line up properly. Where there should be a smooth outer wall there will be ridges and gaps. This may cause misaligned internal supports which can weaken the structural integrity of the part. This typically occurs in only one axis: i.e., front to back or left to right (All3DP, 2021). This problem is primarily caused by hardware within the printer wearing out over time. Things such as belts, pulleys, nuts, and bolts can loosen, introducing the possibility for the extruder to drift and not stop at the exact position specified by the G-Code (All3DP, 2021).

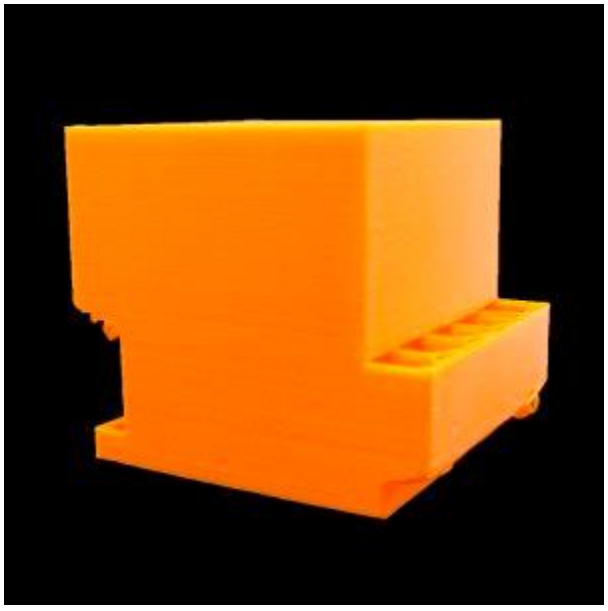


Figure 4.4: A cube-shaped part that has experienced layer shifting

Open type FDM printers have a wide range of defects that can occur throughout the duration of a print. For this proposed system, slippage and failure to extrude are the targeted defects. Gaining an understanding of how these defects occur will allow us to recreate them for testing the system, as well as confirming whether it can both detect and recognize a defect. For this

project, our goal was to be able to detect and recognize five different defects. Those defects are slippage, failure to extrude, warping, stringing and layer shifting. In order to accomplish this goal, we planned to try to successfully detect and recognize one defect at a time.

Chapter 5: Literature Review

The following chapter is a description and review of existing literature pertaining to systems for detecting defects in 3D printing. Many of the sources outlined here discuss two-part systems, with distinct hardware and software designs. Therefore, this chapter discusses hardware options like lighting, sensors, environment, etc. as well as various software options. The literature review serves as a background for preliminary design choices as described in Chapter 6 and Chapter 7.

5.1 Analysis of Hardware Options

A common aspect of existing systems is a hardware setup that can collect quality data from the ongoing print and send that data to be processed. The following is a description and analysis of hardware setups used by relevant existing systems.

5.1.1 Sensors

Depending on the Additive Manufacturing (AM) process, the type of printer, and the environmental conditions, different sensors will be more or less effective. For example, the thermal conductivity of metal makes thermographic sensors very effective for analyzing the progress of a printed part in metal printing processes such as SLM, L-PBF, etc. To detect delamination and splatter in SLM printing, Baumgartl et. al. (2020) used a setup with in-situ thermographic imaging to generate heat maps of likely defects. Another setup used by Ye et. al. (2018) involved a microphone that picked up acoustic information during the melting process, and that information underwent frequency analysis to determine if a defect was occurring. Goh

et. al. (2020) used CT scans post-print to train a machine learning system that used a high-resolution camera in-situ. These systems all got admirable results, but they are not necessarily compatible with an FDM process, and they use equipment that is too expensive or inaccessible for our system. Therefore, we would like to explore the viability of using one or more cameras to gather images of the printing part at various angles.

Some important aspects to consider when looking at different camera setups include the type of camera, number of cameras, location, and angle of the camera(s) in relation to the part or the printer, how the camera is secured to its chosen location, and what kind of data the camera is meant to collect. This section will explore the various approaches used by previous systems and examine the viability of adopting certain aspects into our system.

Kahn et. al. (2020) used a single camera attached to the top of the printer looking almost directly downward. This camera setup is called a ‘static camera setup’ because the camera is stationary with respect to the printer. The goal of this system is to look at the infill pattern of the print to detect inconsistencies that could lead to defects.

Bisheh et. al. (2021) used a Logitech C270 camera with its front panel removed for easier adjustment of the focal length. The camera was attached to the extruder looking downward at an angle and took videos rather than static photos. This system specifically looked for under or over extrusion, so it had the camera focused on a close-up view of the filament directly under the extruder, and had the focal length set so that the filament surface was clear. Since the system was analyzing the pattern of the filament surface, it did not need to capture the edges of the print, so the close-up view was ideal.

Makagonov et. al. (2019) used a single CCD camera attached to the top of the printer looking down at the part. The system took a picture after each layer to analyze porosity in each layer.

Each of these three systems used a single top-down camera to look at a single type of defect, but they all looked for different types of defects. Therefore, this camera position shows promise for getting substantial information about whether these types of defects are forming.

Kirk et. al. (2019) used a few cameras in various positions around the printer to get images of the printing part from several angles. The cameras were mounted on tripods resting on the table around the printer, and they looked at the printing part from a lateral view. The goal of this system was to collect quality images of labeled defective prints in order to train a machine learning algorithm in the future. They concluded that a system that uses higher resolution cameras would be more effective for defect recognition, and that a zoom lens could be effective for capturing minute details on the print surface. However, higher resolution cameras are more expensive and take up for storage and bandwidth in the processor, and a zoom lens may negatively impact the ability to detect defects in larger prints.

Bas et. al. (2020) used an RPi Camera Module v2 (resolution of 2592 x 1944 pixels), and a TTL Serial JPEG Camera with a UCAM-III-116 lens. These two cameras were mounted on the front corners of the print bed to get pictures of a 3D print from different lateral angles. The images from these cameras were combined into a 3D point cloud which was then compared to an ideal model to find defects in the overall form of the printed part. While 3D model recreation was the primary function of this system, it also utilized a backup process of using the image data

to detect the formation of filament blobs from the nozzle. This highlights the potential of using a single camera view for detecting multiple types of defects.

Delli and Chang (2018) developed a setup that collects image data from eight positions from each of five angles using Raspberry Pi cameras each with a Raspberry Pi to process the data.

5.1.2 Background and Lighting

The external environment of the system will affect the quality of the data that the sensors can collect. Different ambient lighting conditions will require different levels of exposure for the cameras, and changes in lighting conditions from the beginning to the end of a print will change how the part looks from the different cameras' perspectives. Additionally, the nature of the background of each image can affect the system's ability to isolate the printed part in any given image. These variations in lighting and background can cause errors in defect detection through image analysis, so it is necessary to explore ways of creating predictable environmental conditions in order to create a system that reliably detects and recognizes defects.

Fastowicz and Okarma (2016) described how the color of the filament changed how light interacted with the part, and therefore how it affected the quality of the photo. They also explained how software solutions could offset this variation. This would include a user interface that prompts the user to input the type and color of filament so that appropriate adjustments can be made in the software before image analysis begins.

Mikolas (2019) explained that controlling the lighting and background was critical for producing high quality photos. Even though this system was not detecting defects in printed parts,

its goal was to produce high quality photos, which will still be an important consideration for our system. Specifically, they explained that good lighting was not as simple as shining a light at the print, and that it was important to consider the camera's white balance and focus when creating a lighting setup. Additionally, they described a background setup made of white foam board which helped to reflect light toward the printing part to create homogeneous lighting conditions.

Kirk et. al. (2019) explained how dim lighting negatively affected the quality of their images and described a setup using an LED panel shining on the surveilled surface of the part to produce higher quality images. They also concluded that the use of a single light source created gradients and shadows in their images which made image analysis difficult. They recommended a lighting setup that would minimize the shadows cast by the part and nozzle while being careful to avoid harsh reflections and high-power consumption.

Gobert et. al. (2018) used a series of five light sources placed in various locations around the print bed. Each of these "modules" could be triggered independently to create eight different lighting conditions.

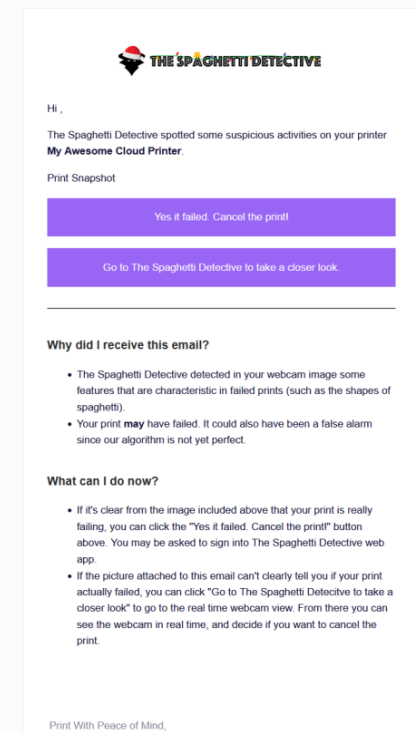
Other systems like the one created by Langeland (2020) used closed-type printers with built-in lighting systems, therefore requiring no additional hardware to create consistent lighting conditions. However, even though the system described by Mwema et. al. (2020) used a closed type printer they, still opted for using two strategically placed light sources in order to create homogeneous illumination around the part. Even with this additional lighting, they cited problems with cast shadows interfering with image analysis and noted that it was sometimes difficult to subtract the background and foreground from the part in the photo because of similarities between the qualities of the part and the unwanted details of the image.

5.2 Similar Software Solutions

There are many projects that attempt to prevent the waste of time and material caused by a 3D printing defect. This section describes some similar projects and the methods they used.

5.2.1 The Spaghetti Detective

Spaghetti Detective (https://www.thespaghettidetective.com/) is an open-source service that detects spaghetti during a 3D print. Spaghetti Detective requires a more powerful computer than a RPI due to the algorithms it uses. Spaghetti Detective can be run on a separate personal computer, or the organization behind it offers a paid subscription service for its use. The separate computer then connects to the computer running OctoPrint through a plugin. The plugin creates a livestream of the 3D print. Then, using an algorithm derived through machine learning, the Detective watches the print for spaghetti. If spaghetti is detected, an email (Figure 5.1) is sent to the user. The user has the ability to confirm if spaghetti is present or not. If spaghetti is present, Spaghetti Detective will cancel the print.



THE SPAGHETTI DETECTIVE

Hi ,

The Spaghetti Detective spotted some suspicious activities on your printer **My Awesome Cloud Printer**.

Print Snapshot

Yes it failed. Cancel the print!

Go to The Spaghetti Detective to take a closer look.

Why did I receive this email?

- The Spaghetti Detective detected in your webcam image some features that are characteristic in failed prints (such as the shapes of spaghetti).
- Your print **may** have failed. It could also have been a false alarm since our algorithm is not yet perfect.

What can I do now?

- If it's clear from the image included above that your print is really failing, you can click the "Yes it failed. Cancel the print!" button above. You may be asked to sign into The Spaghetti Detective web app.
- If the picture attached to this email can't clearly tell you if your print actually failed, you can click "Go to The Spaghetti Detecitve to take a closer look" to go to the real time webcam view. From there you can see the webcam in real time, and decide if you want to cancel the print.

Print With Peace of Mind,

Figure 5.1: An email from Spaghetti Detective of our failed print

5.2.2 3DPrintSaviour

3DPrintSavior (https://github.com/Manicben/3DPrintSaviour) is a final year project developed at Imperial College London. Its goal is to be an automatic print failure detection system. Like Spaghetti Detective, 3DPrintSaviour requires two computers to use. The first computer runs OctoPrint and OctoLapse while the second one has 3DPrintSaviour on it. Each time OctoLapse take a picture, it transfers it to the second computer. Like this project, it also uses OctoPrint, OctoLapse and OpenCV. The current version of the software compares each snapshot from the time lapse to the previous. It then generates the Normalised Root Mean-Squared Error (NRMSE) from OpenCV. The software will decide there is a defect if the NRMSE value is above a given value. If there is a defect for more than 5 layers, 3DPrintSaviour will stop the print. There is little information on how to use the program or if it worked correctly.

5.2.3 Computer Vision Based Layer Wise 3D Printing Analysis

The Computer Vision Based Layer Wise 3D Printing Analysis project (Petsuik and Pearce) is a project done by two researchers, Joshua Pearce and Aliaksei Petsiuk. The goal of their project was not only to detect defects when they occur but to also generate corrective actions that the printer could take to correct the issues. It uses computer vision to analyze every layer of the part as it is printed. It first verifies that the height is correct. Next, it analyzes the shape of the topmost layer and, using the original CAD model, compares the topmost layer to the model to ensure that it is printing the correct shape. The system also has a user interface that shows the current actions the printer is taking. When tested, the system was able to work in real time and took about one minute to analyze each layer.

Chapter 6: Methodology

Throughout this project, we did a lot of experimentation with regard to setting up an ideal system for defect detection and recognition. We tried many different hardware setups, algorithms, and investigated other software choices. This chapter serves as a diary of the various things that we attempted in order to determine the system's final design. The goal of this chapter is to show what we tried, what worked, and what did not work as the project progressed.

6.1 Research and Initial Tests

6.1.1 Research

At the beginning of the project, we realized that each team member had a different level of knowledge about 3D printing, and that all of us would be better equipped to work on this project if we did research about different types of 3D printing and defects. The results of this research can be found in Chapters 3 and 4. Additionally, we decided it was necessary to read about existing systems with similarities to what we wanted to create. This research is laid out in Chapter 5.

Through this research we began to notice patterns of features that many of these existing systems had. At the most basic level, each defect detection system, no matter the method of detection or the type of 3D printing, had some type of sensor paired with an algorithm. The sensor collected information about the print, and the algorithm processed that information to determine if the print was defective. Some examples of sensors included acoustic sensors, thermal sensors, and CT scans, as well as basic cameras such as webcams and RPI cameras. Many of the systems we researched used machine learning algorithms to detect defects, while

others used methods like photogrammetry. All of this initial research led us to a basic assumption that our system should use a sensor to collect information and an algorithm to process information.

6.1.2 Initial Tests

After performing research into relevant systems, we started to think about what we wanted our system to look like. We first set up a MonoPrice 3D printer to use for testing, and explored different options for sensors that were available to us in our lab. We decided to use cameras as sensors because we had access to both a webcam and a RPI camera to test with initially. In addition, other forms of sensors would be far too expensive or elaborate for the scope of this project. We then took measurements of the printer, and analyzed different options for how the camera would record data. To determine the best option, we ran a few initial tests, whose findings are explored in more detail in the following sections. The main purpose of these tests was to help us get a bearing on the software and hardware requirements, so that we would know what direction to take when developing and testing the final system.

6.2 Hardware Development

6.2.1 Preliminary Decisions

Based on the research that we had done, we made some decisions about how we would begin our hardware development. While some similar projects (see Chapter 5) had used multiple cameras to capture many angles of the part while it was printing, a majority of the projects we researched got admirable results with only one camera. So, we decided to start by using a single camera to see if it would be sufficient for our purposes.

Knowing that we were going to use a single camera, our next step was to decide which camera to use, where to place it, and how to mount it. In order to save money and time, we wanted to use a camera that was already in our possession, so we had two choices; a Logitech C920 Pro webcam, or a Raspberry Pi CM v2 with a fisheye lens. The webcam was larger, heavier, and generally would be harder to consistently mount than the Pi camera. However, we did not have another lens for the Pi camera, and we had determined previously that the images taken with the fisheye lens would cause the shape of the printed part to be distorted. This distortion would have added another layer of complexity to our software, which we wanted to avoid. Additionally, the webcam had a larger horizontal field of view, which would help to capture more of the print bed from a closer distance.

Our initial tests had shown that motion blur would be an issue when taking intermittent photos of the part throughout the print. We determined that there were two possible solutions to this problem, with one of them being a hardware solution. If the camera were to be mounted on the print bed, then the relative velocity between the part and the camera would always be zero, which would eliminate any potential motion blur. However, this solution would be impractical, because the weight of the camera might cause problems with the motion of the print bed, and tampering with the print bed could cause additional problems that we might be unable to predict. Ultimately, we decided to solve this problem with software, which will be discussed in more detail in section 6.3. With the software handling the motion blur problem, we had more potential options for where to mount the camera.

The placement of our camera had to meet three requirements. First, the position and angle must be easy to replicate. This is because our chosen method of image analysis to search for defects requires images of a CAD model of the same part, taken from the same position and

angle. Slicing programs and CAD model viewing software often have standardized viewing angles from which the user can select, so we decided to choose one of those angles for the sake of simplicity. Additionally, we decided not to use an isometric or top view because of how difficult it would be to mount our fairly heavy and awkwardly-shaped webcam high off the table.

Second, the images taken from the chosen position and angle must be able to give us a clear outline of the part. At the very least, this meant being able to see the entirety of the part without obstructions while still being close enough to see small details in the silhouette. Effectively, this meant getting the camera as close to the part as possible without the part going out of frame. This left us with front, back, and side views, because the flat edges of the bed were far enough away for the camera to capture the whole part, while views from the corners of the print bed would be farther away from the part, therefore capturing less detail.

The third requirement was for the background to be consistent so our software would not mistakenly detect background elements as something for consideration. Due to the orientation of our printer, the side and back positions could potentially have people walking or working in the background, so we decided to use the front position. As an added bonus, this made it easier to access the camera in case we needed to adjust anything.

Our camera was mounted to a clamp style mount, which was placed on the table in front of the printer. We lined up the camera to be at the exact center of the print bed horizontally, and made sure the angle of the lens (view angle) was perpendicular to the printer's z-axis. Additionally, we had the camera two centimeters away from the print bed when it was extended to its limit in the -y direction. Images of the exact setup are shown in Figures 6.1 and 6.2. The initial lighting and background for the system were the ambient room conditions.

Figure 6.1: The location and the placement of the camera with the print bed extended to its limit in the negative y direction. The square was used to align the camera with the z-axis

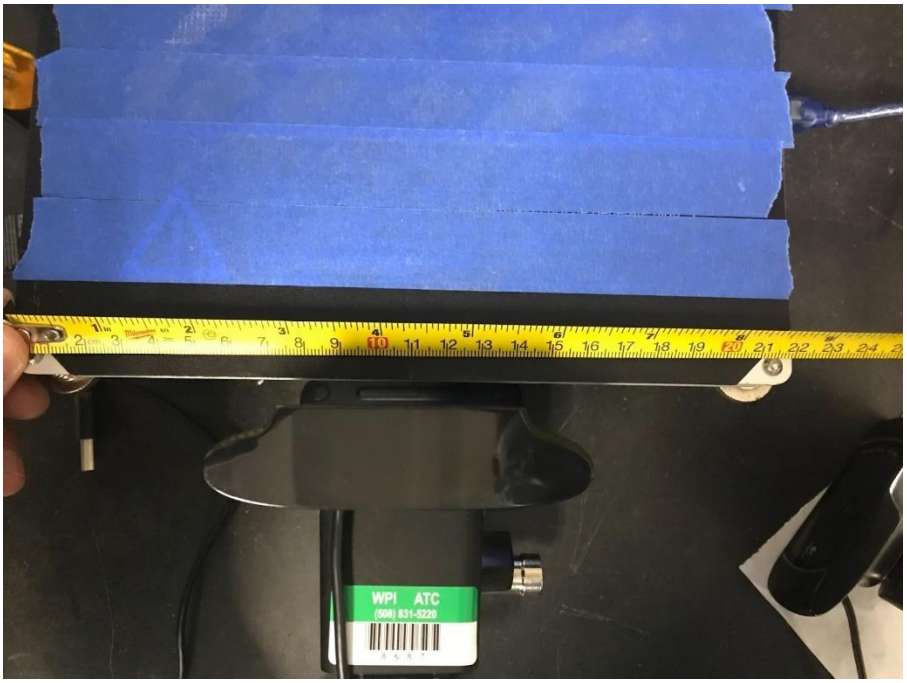


Figure 6.2: How the camera was centered relative to the print bed. The lens of the webcam was placed at 110 mm (half of the width of the print bed)

6.2.2 Lighting and Background

Initially we decided not to make any specific decisions for the lighting or background. We wanted to see how well the system would function with ambient lighting conditions and using the plain white wall of the room as a background. However, in the first experiment we determined that this approach would not work, especially for the white filament we used in our experiments. The results of the first experiment showed us that the white background did not provide enough contrast to extract an outline of the part from the images that were taken. We also found that the camera was autofocusing on the background instead of the part, because there were several wires and other prominent details visible behind the part for the camera to focus on. Additionally, since the ambient lighting was subject to change over the course of the print, the time-lapse images varied between being too dim and too bright.

One solution to the autofocus problem would be to turn off the autofocus and have the camera focus at a distance of about half the bed length, but we wanted to avoid this solution if possible so the system would be effective no matter how close the part was to the camera.

For the second test, our first priority was to place something behind the printer to act as a background. We believed this would help with the contrast issue, and it might make the variable lighting irrelevant as long as the part had good contrast. We also thought that a plain and uniform background would solve the autofocusing problem. Even though a black background would provide the best contrast for a white part, we decided to use a more neutral colored background since we thought it would be more universally suitable for many different filament colors. We chose cardboard (a neutral tan color) as our background because it was easily available to us, and none of the common filament colors are very similar to the color of cardboard. While the cardboard background did solve the contrast issue and negate the variable lighting problem, we

still had problems with autofocus, and decided it would be easier to set the camera's focus at a standard location. This solution proved to be sufficient.

At this point, the most prominent difficulties were caused by the variable ambient lighting, so we decided to address this problem for the next test. Several factors were responsible for changing the lighting conditions, including the natural light changing throughout the day based on the position of the sun, and people walking around casting shadows in various locations. To address all of these factors we decided that a full enclosure for the printer would be the best option. This way we would have full control over how the lighting and background looked. The internal walls of the enclosure were a shiny metallic gray material, and there was a small window on the front of the enclosure which allowed the user to see inside to the printer.

For the next tests we decided to remove the cardboard background because we thought that the metallic wall material would reflect light around the enclosure to create a uniform lighting condition, and we did not want to obstruct any of the walls. We took pictures of two different parts under various lighting conditions to get a sense of what sorts of lighting orientations would create the most useful images of parts on the print bed.

Next, we wanted to see the quality of a time-lapse in the new enclosure, using ambient room light to see if the enclosure alone would solve any of the problems we were having with the ambient room light. Since the lights in the room turned off halfway through this test, we decided to place high priority on standardizing our lighting system.

For the next few tests, we completely covered the window of the enclosure, and set up our own lighting. Our first lighting design involved a simple LED flashlight mounted in the top right corner of the enclosure. We determined from the previous test that a light from above and behind the camera would produce good enough lighting conditions for the part to be clearly

visible. We chose to use the top right corner (farthest point away from the part) and the lowest brightness setting because the white filament that we were using for our tests was highly reflective, and too much direct light would make the part too bright. We also put the cardboard background back because the metallic wall material had many details which tripped up our edge detection algorithm. However, this ended up being irrelevant because, when we put the cover over the enclosure window, it was completely dark inside except for the print bed. Most filament colors that we tested in this lighting setup showed up quite well against the effectively black background, but parts made of darker filament, especially black, almost completely blended in with the background, making it nearly impossible to find the outline of them. For the next test, we made a time-lapse under these conditions, using a 3DBenchy instead of a cube in order to show how shadows affect edge detection for a more complex 3D shape.

At this point there were three major problems that we needed to address. First of all, the darkness inside the enclosure completely negated the original purpose of the cardboard background, especially for the black part. Therefore, we needed to design a way to keep the background well lit so that the cardboard color would be visible in the time-lapses. Second, since the print bed was essentially part of the background, we needed to consider using different tape colors to provide better contrast to all possible filament colors. This was once again shown where the blue part blends in with the blue bed tape in certain areas. Third, our lighting mount was incredibly makeshift, with only tape and paper towel wads holding it in the correct position.

To solve the first problem, we designed a system of LEDs that would shine down on the cardboard background, with an additional cover to prevent the LEDs from lighting up the print bed at all. The cover also served to reflect more light onto the cardboard background.

To solve the second problem, we used manila masking tape in place of the blue painter's tape. However, the manila tape was fairly translucent, which caused the lettering on the print bed to show through. We considered doing two layers of manila tape, but this would have made the print bed too light in color, which would cause problems with the contrast of lighter colored parts. Ultimately, we decided to use colored tape as a bottom layer, and then put the manila tape on top of that. We had many colors of tape to choose from, but to negate the problem of the manila tape being too light, we knew that a darker under-layer would be better. We decided to use red tape, since it gave the closest appearance to the cardboard background when the manila tape was placed over top of it.

To solve the third problem, we acquired a ring fill light, and mounted it on a tripod behind and above the camera. Unlike the setup with the flashlight in the corner, the ring light had minimal harsh shadows, which proved to be friendlier for our edge detection algorithm. Additionally, the ring light had controls for brightness and warmth, giving us further control over how the lighting looked within the enclosure.

To test our new lighting setup, we first determined what brightness setting would work best for each warmth setting by observing the appearance of parts of different colors through the webcam's view, and adjusting the lighting settings to find a good brightness. We determined that the lowest brightness setting created clear images for all three warmth settings, so we kept that setting for all of our tests with the ring light. Next, we tested parts with many different filament colors and shapes in the newly furnished enclosure, taking a picture of each one in cold, neutral, and warm lighting conditions. The purpose of this test was to establish a point of comparison for the three different light settings, and determine which one would be best for the most filament colors. Additionally, we ran one print at each light setting to see how the settings affected the

edge detection in a time-lapse. The details of these tests can be found in Chapter 9. Once we found that the warm light setting worked best, we then printed a 3DBenchy under warm lighting conditions to see if the results held true for a more complex 3D part.

At this point, we started noticing many problems with the manila tape. The biggest problem was that it was made of a material that was not very heat resistant. During every test print, the tape would melt and stick to the bottom of the part. This meant that every time we removed a part from the bed, we would have to replace the manila tape. Additionally, the red tape underneath the manila tape was also not very heat resistant. The heated bed would cause the red tape to lose adhesion in various locations, bubbling up and creating a bumpy surface. This would cause problems for bed adhesion, so we had to fix this after every test. To combat this problem, we ordered heat-resistant, tan tape. This tape was closer to the cardboard background color with only one layer, and the heated bed did not cause it to bubble up as much.

6.2.3 Camera Limit Testing

One of the problems with our system at that moment in time was the maximum size part that the camera can see. For this experiment larger parts that had previously been printed were placed on the print bed and the webcam took photos. All of our testing up to this point had been with simple and smaller parts, so the goal of this was to understand how larger and differently shaped parts would affect the camera's ability to capture the full outline of the part. From this test we learned that there is a size limit to the parts that the camera can see.

Once this information was realized, the team brainstormed about what we should do about this limitation. One idea was to add additional cameras so that the part can be viewed in full from multiple angles. Another idea was to have a moveable camera mount that could change position as the part was being printed. The moveable mount seemed to be too advanced for the

scope of our project. While multiple cameras is the more realistic solution, it still was not our priority. We were trying to learn how to detect defects with one camera: once that was accomplished then we can add more cameras to the system. The team decided that the best solution would be to acknowledge the limitation and make measurements to fully understand it.

The next step that the team took was measuring what the actual size image the current camera setup could capture. In order to accomplish this the print bed was moved into its position that it moves to for time lapse photos. Next a ruler was placed along the width of the print bed. The measurement was taken from the left side of the frame to the right side of the frame. This denoted the width that the camera could see. A similar measurement was taken for the height of the image as well, this is shown in Figure 6.3.

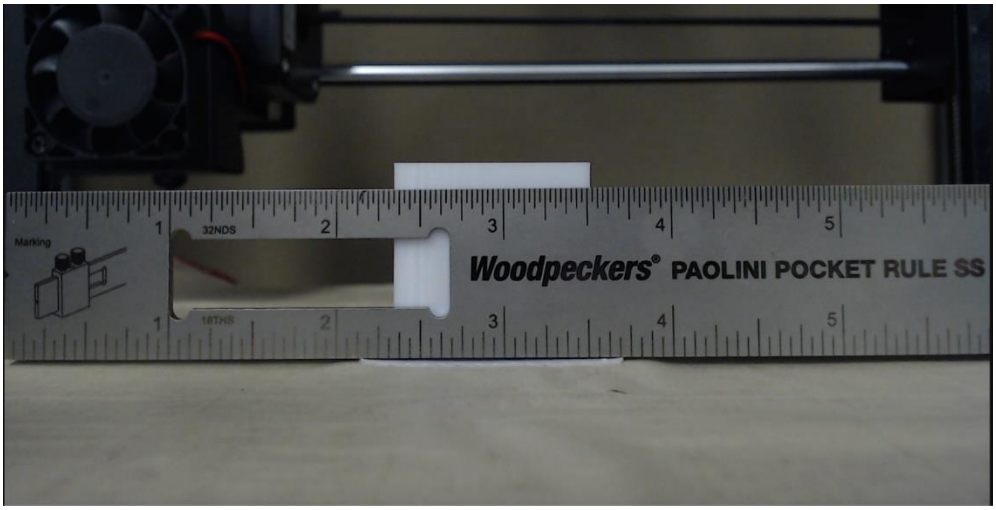


Figure 6.3: The measurement of the width of the frame that the camera can see. From left to right it is roughly 6 inches wide

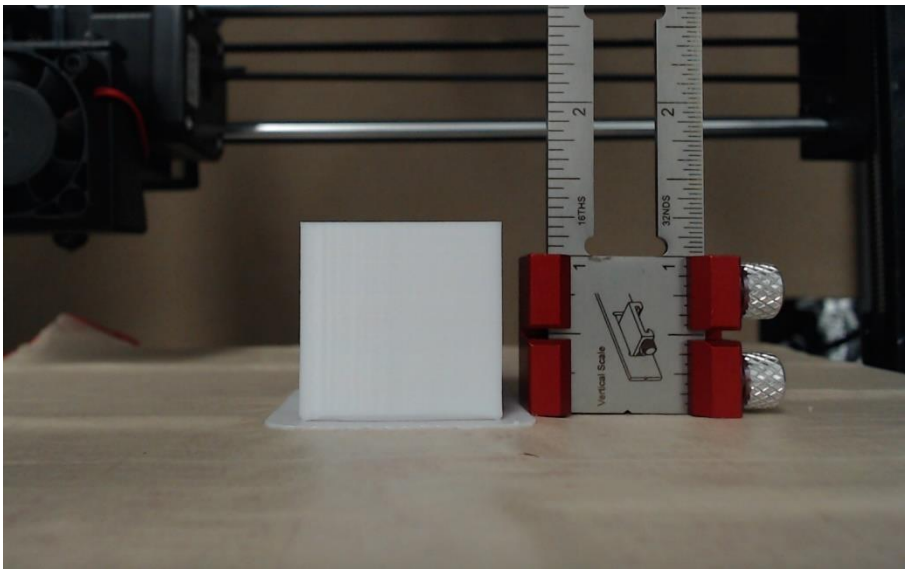


Figure 6.4: The size of the frame in the vertical direction. From the build plate to the top of the frame, the camera in this orientation views roughly 2.5 inches

Another measurement that was taken was the distance from the lens to the part, this measurement can be seen in Figure 6.5. This was done by finding the centerline of the part and measuring from the face of the webcam to this center line.

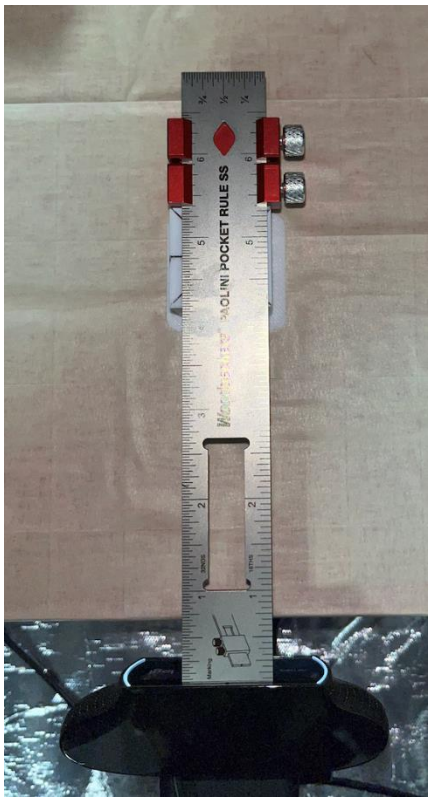


Figure 6.5: The distance of the center of the part to the front of the webcam. The distance is roughly 5.375 inches

Overall from these measurements, we learned that there are some limits to what the camera can see in its current set up. With a part roughly 5.375 inches away from the camera lens, the height and width of a part that can be seen in full by the camera is about 2.5 inches tall by 6 inches wide. Understanding these limits made the team look into other camera options that could improve these limits.

When looking into other camera options, we needed to establish what parameters were valuable for us when it comes to a camera. These parameters would be image size and image resolution. For viewing capabilities, it would not be realistic to have a camera that can view the entire vertical bound of the printer (180 mm). For our project we care more about the beginning of the print, making the vertical limit of the camera less of a priority. This then emphasizes the

need for the camera to view the entire width of the print bed, meaning that no matter the size of the part at its base, the camera will be able to view its entire outline.

The other parameter that we wanted to look into was the resolution of the sensor for the camera, this determines the overall resolution capability of the camera. There are a few more factors that play into resolution such as sensor size, pixel size and number of megapixels that the sensor has. We then created a spreadsheet designed to compare specs of different cameras. Using this we could quantitatively compare different cameras and see if there was a better option other than the Logitech webcam that we had been using.

6.2.4 Second Printer

Once we had mostly finalized our environment design, we decided to set up a second printer in order to perform a large number of test prints in a short period of time. This also had the secondary effect of testing whether our system was adaptable to additional printers. We ordered a duplicate set of equipment which we set up around a RepRap printer. The second setup had a few differences from the first one, some of which created minor difficulties in making sure it met all the relevant requirements.

First, the enclosure around the RepRap was smaller than the enclosure around the MonoPrice. This made it difficult to fit all the necessary components within the enclosure. However, in the end, the whole printer and camera were able to fit inside. Another problem we encountered was that the RepRap printer had exposed red LEDs that would flicker on and off to indicate certain functions of the printer. Even though the effects of these LEDs were minor, it meant that the lighting would not be consistent for each snapshot. This problem was solved by placing tape over the LEDs to block the light.

For the second setup, we did not have access to an additional RPI to run octoprint, so we had to use a PC available in the lab instead. After a bit of troubleshooting, we were able to get OctoPrint and OctoLapse working as intended, with a slight difference from the first setup. Since the snapshots were handled by a 3rd party software, we had to use one that was compatible with a Windows PC rather than the one used for an RPI. The one that we found, Yawcam, worked well, but provided snapshots that were significantly lower in quality. However, throughout our testing, we found that this did not really matter.

6.2.5 Camera and Light Fixture

After learning about the initial set up for the system in regard to hardware as discussed in section 5.1.1, our next steps were to finalize this setup so that there was a consistent background and environment for the printer to run tests. An important aspect of this consistency was to create a fixture that could hold and register the camera and the ring light that is behind it. The term registering refers to the fixture having the ability to touch a part of the printer to allow it to consistently locate to the same place every time. The ability for the fixture to register to a specific part of the printer is important because it would allow us to move the fixture to make any adjustments or fix any problems with the printer and be able to place it back to the same exact location every time it is moved.

The first thing that the team did was research for any existing mounts or fixtures that had already been created for the Logitech C920 webcam. Through this search, we found a small bracket that attaches directly to the webcam with screws when its lens cover is taken off. We printed this for ourselves and made measurements using a pair of digital calipers and created our own model in Solidworks. By doing this, we had the freedom to make design changes to better suit our fixture.

Once we had a model of the camera bracket, we then needed measurements and dimensions from the MonoPrice printer. These values were acquired using a tape measure, calipers and a stainless-steel rule.

Once the measurements of the Monoprice printer were taken, a basic fixture was created. This initial fixture was not intended to be the final design, its purpose was mainly to confirm all of the measurements taken of the printer and make sure that they were correct and make adjustments if necessary. This led to another iteration of the fixture; this again was to confirm that the fixture had the correct dimensions so it fit the system properly.

The last iteration of the Monoprice fixture was to eliminate material to reduce material and time consumption when printing. A majority of the base of the fixture was over designed and much of it was cut down. After the final iteration, the final fixture takes roughly 10 total hours to print, this is roughly 6-8 hours faster than the initial fixture design.

Once the Monoprice fixture was established, measurements of the RepRap printer were taken. With these measurements, the Monoprice fixture's dimensions were altered to match the bed height and distance from the bed. These changes are explained in more detail in Chapter 6 as well as the other design decisions made.

6.3 Software Development

This section describes the development process of the algorithm used for DaR3D.

6.3.1 Algorithm Decisions

The first step in developing the defect detection program was to research methods of image analysis that could possibly be used to analyze a printing part for defects. In order to quantify each potential method, the team listed each option then listed the strengths and weaknesses for each one. In all, the team was able to come up with three major methods of image analysis that could be used for detecting defects. The first method found was to train a program, using machine learning, to detect defects. The second was to build a 3D model of the part as it was printing and then compare it to the original 3D model of the part that was being printed. The third method was to analyze 2D images from the printed part for defects.

6.3.2 Machine Learning

The first method was to use machine learning to train a program to analyze a printing part for defects. There were several strengths of this method. First, there was already existing research on using machine learning to detect specific types of defects so, this method was already proven to have some success. Second, if it was properly trained with a wide variety of parts, this method would be applicable to the widest range of parts without modification. Unfortunately, there were also a few major weaknesses of this method. First, it required many samples to train. Printing a sufficient number of samples for one type of defect would have taken a large portion of the project's time. Second, after the parts were printed, implementing the machine learning algorithm was also going to be difficult. Finally, running machine learning on a

lower powered computer such as the Raspberry Pi would be difficult. The algorithm would take a lot of processing power, so a more powerful computer would have been needed. The team decided that the downsides of machine learning outweighed the upsides so we moved onto other methods.

6.3.3 3D Model Analysis

The second method the team looked at was to build a 3D model of the part as it was printing and then compare that part to the original 3D model of the part that was being printed. This method would use a 3D model construction method known as photogrammetry [See 3.3.4 for full explanation]. On paper, this method has many upsides. Assuming it is done correctly, it would be able to detect any visible defect. Unfortunately, implementing this method correctly is very difficult. Photogrammetry requires many (80+) high-quality images of the part from every angle. Models with the level of detail needed to detect defects require around 50-80 images. This leads to a few issues. First, pausing the print to take that many pictures would affect layer adhesion and could cause defects itself. Second, getting pictures of every angle of the part while it is printing is nearly impossible due to the printer itself being in the way. Even if the model was accurately built, comparing it to the original file would also be very difficult. The built model and the original model would need to be the exact same dimensions. Because the 3D printer prints parts layer by layer and the original 3D model only contains the entire model, the original model would need to be trimmed to the same height as the part on the print bed. These issues made it impractical to implement 3D model analysis, so the team moved onto the final method.

6.3.4 2D Model Analysis

The final method of defect detection was 2D model analysis. This method involves taking 2D pictures of the part as it is printing and analyzing those for defects. This method had many upsides. First, there were many methods of 2D analysis to investigate, so if one method ends up not working out the team can move to a second method. Second, 2D model analysis is the easiest to set up. It only requires one camera facing the print bed. Finally, it is the easiest method to implement. 2D image analysis is a widely researched field which means there are many examples to choose from. The downside of 2D model analysis is that it can only detect defects on the side of the part the camera is facing. This means that any defects not facing the camera would be missed by any 2D method. Because this method was the most feasible to implement otherwise, the team decided this downside was tolerable and decided to use 2D model analysis for the defect detection method. We were then able to move onto setting up the environment and beginning testing.

6.3.5 Environment Setup and First Tests

Before the software testing could begin, the testing environment needed to be set up. The group decided to use OctoPrint (See 3.3.2) to manage controlling the printer. OctoPrint was then installed on a Raspberry Pi. We decided to do this because OctoPrint added many useful features that made our testing easier. The main two we were interested in were the ability to remotely upload a file to print and the ability to add third party plugins. Remotely uploading a file made it so that any one of us could start and manage an experiment without the hassle of going into the lab. This allowed us to spend that time elsewhere.

The ability to use external plugins was the most important reason we used OctoPrint. Without that ability, we would not have been able to use OctoLapse. OctoLapse is able to pause the print every few layers and take a picture of it with an attached camera. The group decided to use OctoLapse (See 3.3.3) because it was a tested and working solution for taking pictures of a print while it was printing. Without OctoLapse, we would have had to spend more time developing a method to take usable pictures of the print. Both of these were installed on a Raspberry Pi Model 4B. This computer was used because it was a cost and space efficient computer that had the power to run both of these programs and it was easily available.

After installing OctoPrint and OctoLapse, the group was able to run our first experiment. The goal of this first experiment was to verify that the software setup was working and that it was able to record a time-lapse of a print. The details of this experiment are recorded in Chapter 9.

6.3.6 Outline Extraction through Edge Detection

After the experiment referenced in Section 9.2 was completed, the group began testing methods to extract the outline of a printed part while it was printing. We needed to extract the outline because we will use it to determine if the part is printing correctly by comparing it to a generated model. The first method that was tried was using an edge detection algorithm (Seen in Figure 6.6). We decided to use an edge detection algorithm because it was an easy way to test the feasibility of extracting the outline of the part while it is printing. OpenCV (See 3.3.1) includes an edge detection method called Canny Edge Detection (See 3.3.6) so implementing the algorithm was simple. A full writeup of the algorithm is included in Section 8.1: Canny Edge Detection. This algorithm showed that extracting the outline of a printed part was possible, but

some improvements needed to be made before it was usable. The results of this algorithm is shown in Figure 6.7.

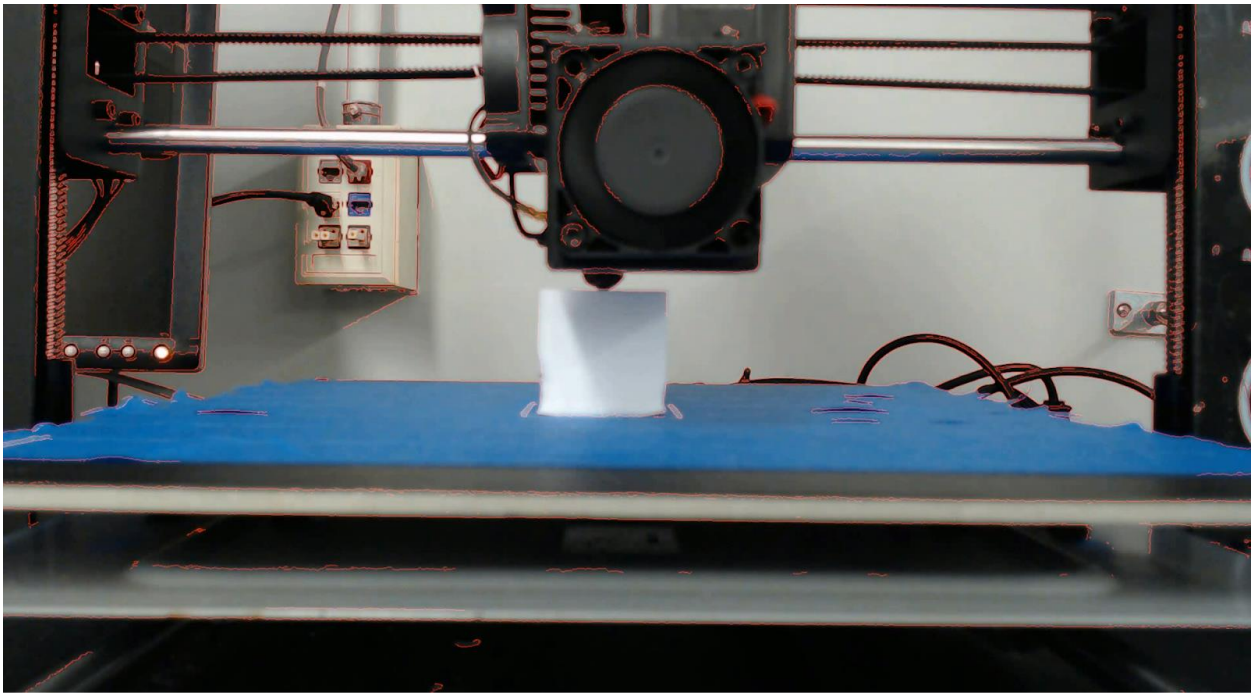


Figure 6.6: Edge Detection Ran on Experiment 9.2

Experiment 9.3 was also carried out. (See Figure 6.7) Because of the background used for that experiment, and the position of the gantry, the algorithm was able to extract more of the printed part. This was because the edge detection algorithm was no longer focusing on the extruder and the background wires. It also showed that image quality improvements still needed to be made.

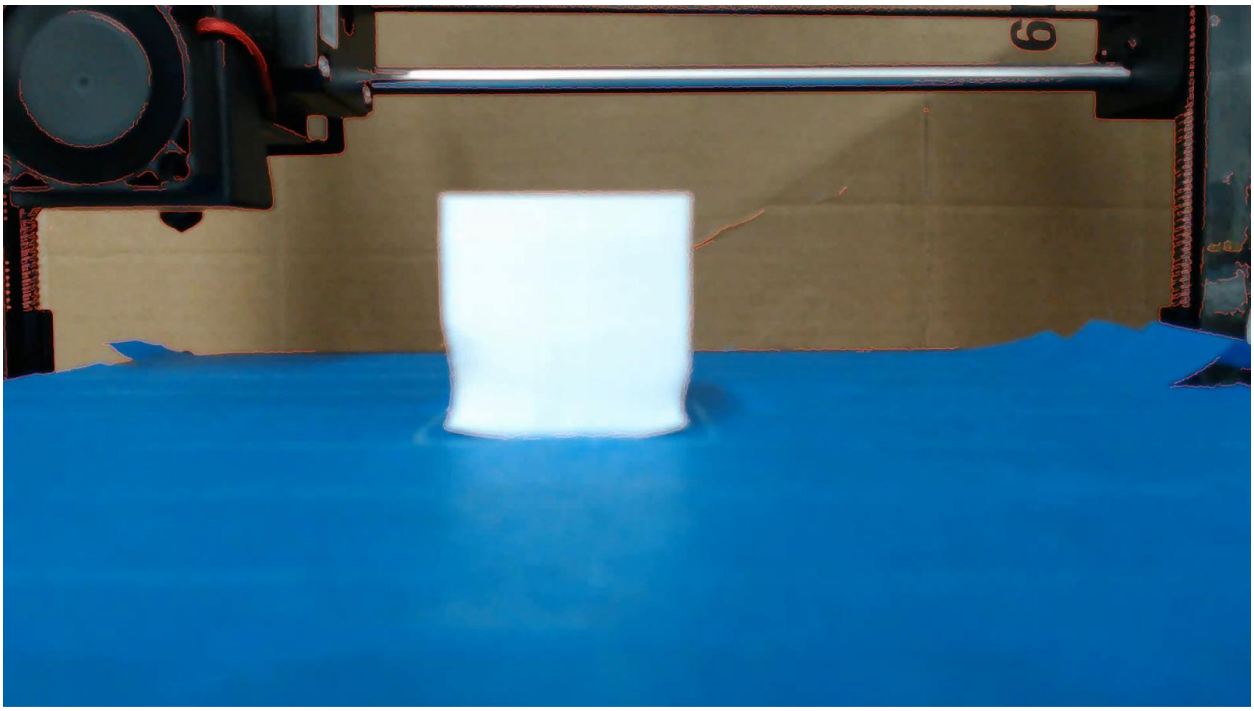


Figure 6.7: Edge Detection Ran on Experiment 9.3

6.3.7 Time-Lapse Image Improvements

Experiment 9.2 showed that the default settings of the camera were not useful for creating a time-lapse (See Terminology) with. The focus was too blurry to make out any details of the print and the exposure was too high, causing the white part to appear too bright. The image quality of the time-lapse was also too low due to automatic video compression in OctoLapse. The group decided that improving the image quality of the time-lapse would make extracting the outline easier. We first researched how to improve the quality of the time-lapse from OctoLapse. We found a setting in OctoLapse that configured the compression of the time-lapse after it was created (See Figure 6.8). This was increased to the minimum, so no compression was being used, allowing for the maximum quality of time-lapse.

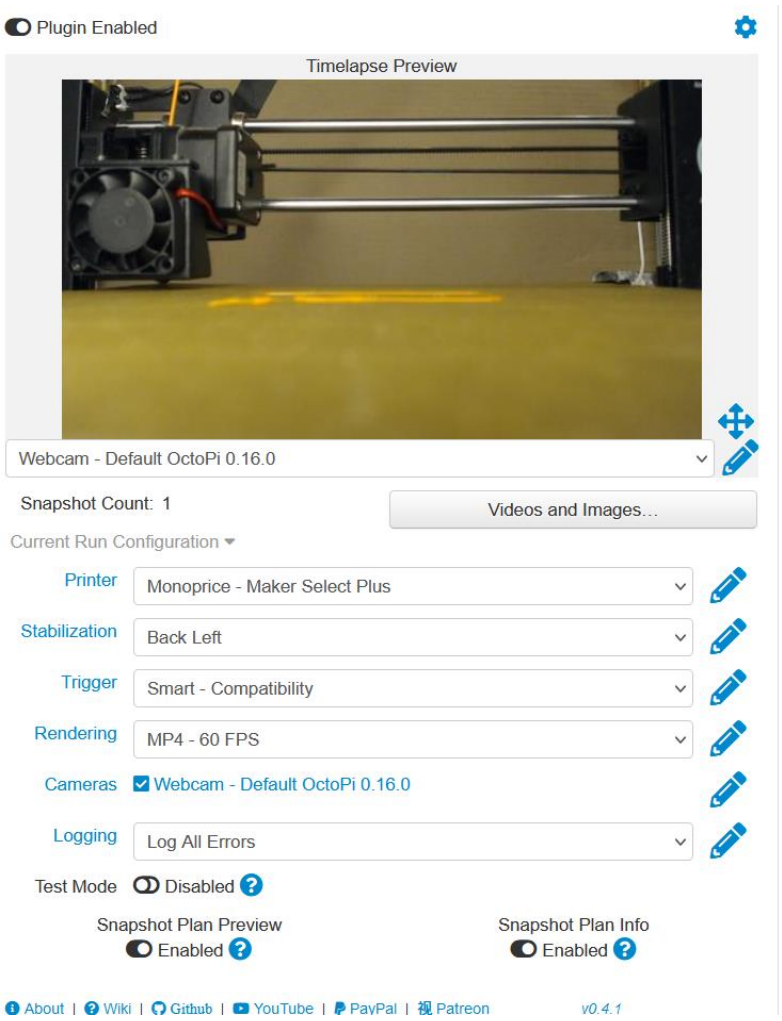


Figure 6.8: Octolapse Settings

Next, the camera needed to be manually focused and the exposure needed to be manually set. There was also a setting for this in OctoLapse. The values shown below in Figure 6.9 were found by looking at the live feed from the camera while adjusting the values in OctoLapse and visually comparing the results.

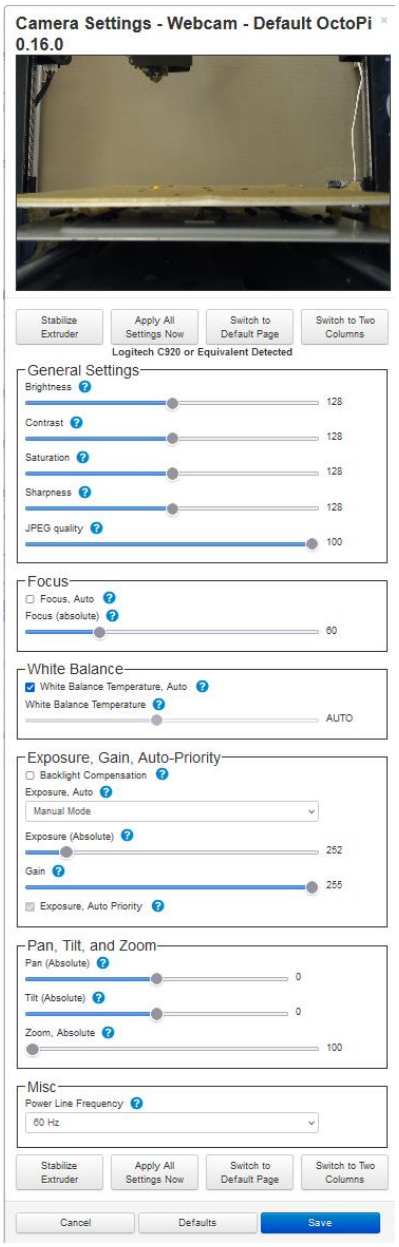


Figure 6.9: Octolapse Camera Settings

After these settings were changed, the quality of the image was greatly improved, and Experiment 9.4 was able to be run. There was no more blur due to video compression and the part remained in focus the entire duration of the print. Show in Figure 6.10 is the results of the calibration.

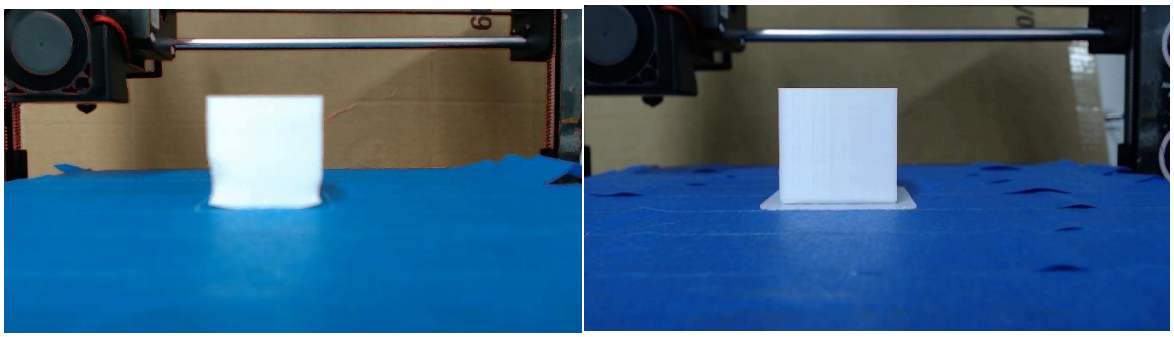


Figure 6.10: Experiment 9.3 (Left) and Experiment 9.4 (Right) Results side by side

6.3.8 New Outline Extraction Methods

After Experiment 9.4 the group met up together to discuss different methods to extract the outline of the printed part. The group created a few new methods to extract the outline of the part.

The first proposed method was background removal. Because of OctoLapse, the print bed and extruder will be in the same position for each picture so, from the perspective of the camera, only the part should change. The empty print bed can be removed through image subtraction. Image subtraction involves removing pixels that are in the same place and color between two images. By saving an image of the empty print bed, we can remove all similar pictures in any other frame. This is described more in Section 8.2: Background Removal. Background removal did not work as well as expected due to slight changes in the image between each frame and because the extruder was not in the same place but actually moving upwards each frame. (See Figure 6.11)

The next proposed method was a slight variation of background removal where we compare each frame with the previous and remove anything similar. Anything that is different between each frame is then added to the outline of the print. This algorithm is described more in Section 8.3: Color Extraction. As for background removal, difference detection also did not work due to

slight changes in the image between each frame and because the extruder was not in the same place.

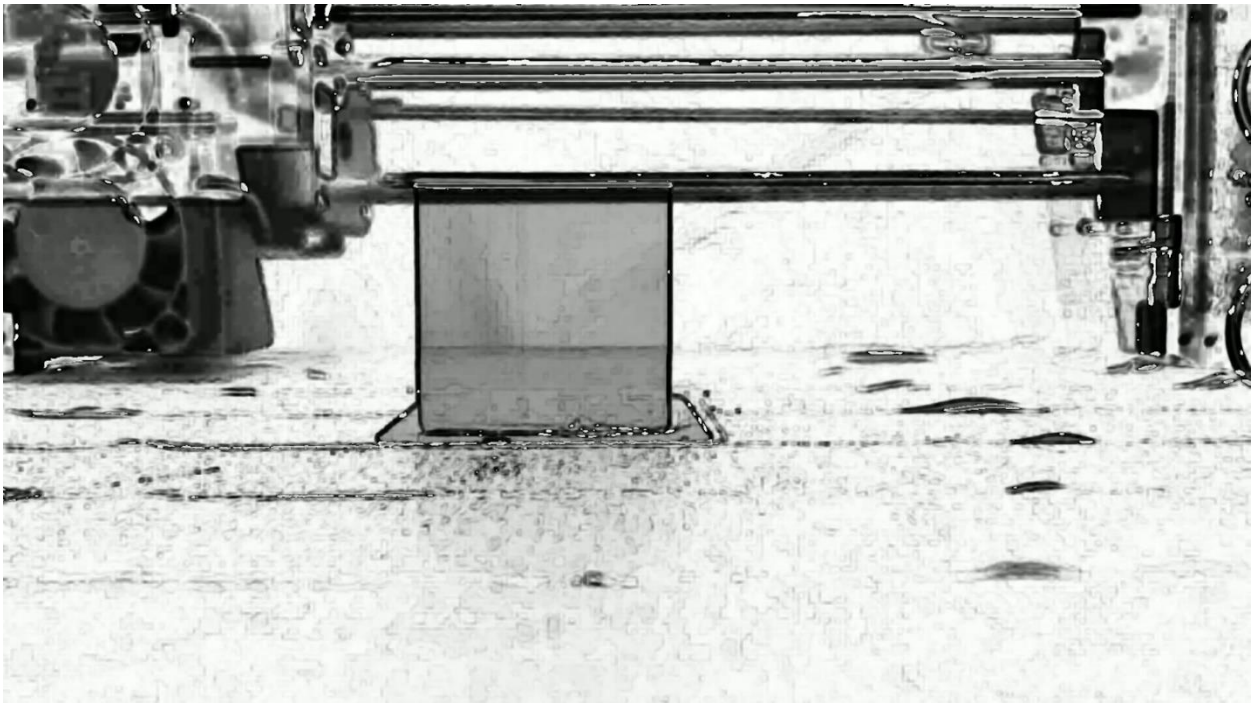


Figure 6.11: Background removal test. The darker regions represent more change

The last proposed method was color masking. The idea behind color masking was that because the printed part is all the same color, that color can be extracted from the whole image. Then the outline of the extracted region can be found. This algorithm is described in Section 8.4: Color Extraction. This algorithm was not usable because there were too many objects in the frame that were the same color as the printed part so, the algorithm picked up too much that was not the part (See Figure 6.12).

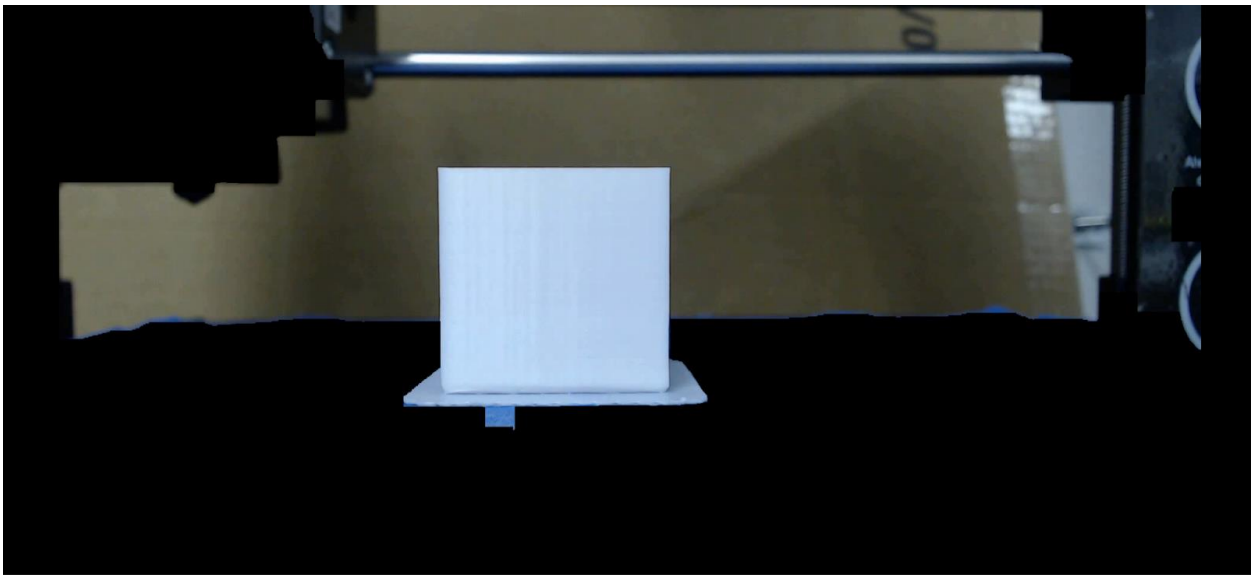


Figure 6.12: White Extraction Test

On their own, these three algorithms were not very useful, but we then had the idea to combine them into one algorithm that could take advantage of color detection, difference detection and background removal.

6.3.9 Combined Method of Outline Extraction

The combined method of outline extraction first used color detection to get everything in the frame that was the color of the part. Next, it took that region and compared it to an image of the empty print bed. Anything in the extracted color region that was different was determined to be part of the print and was outlined. This algorithm is described in Section 8.5: Combined Method. When run against Experiment 9.5 this algorithm produced the most accurate outline so far.

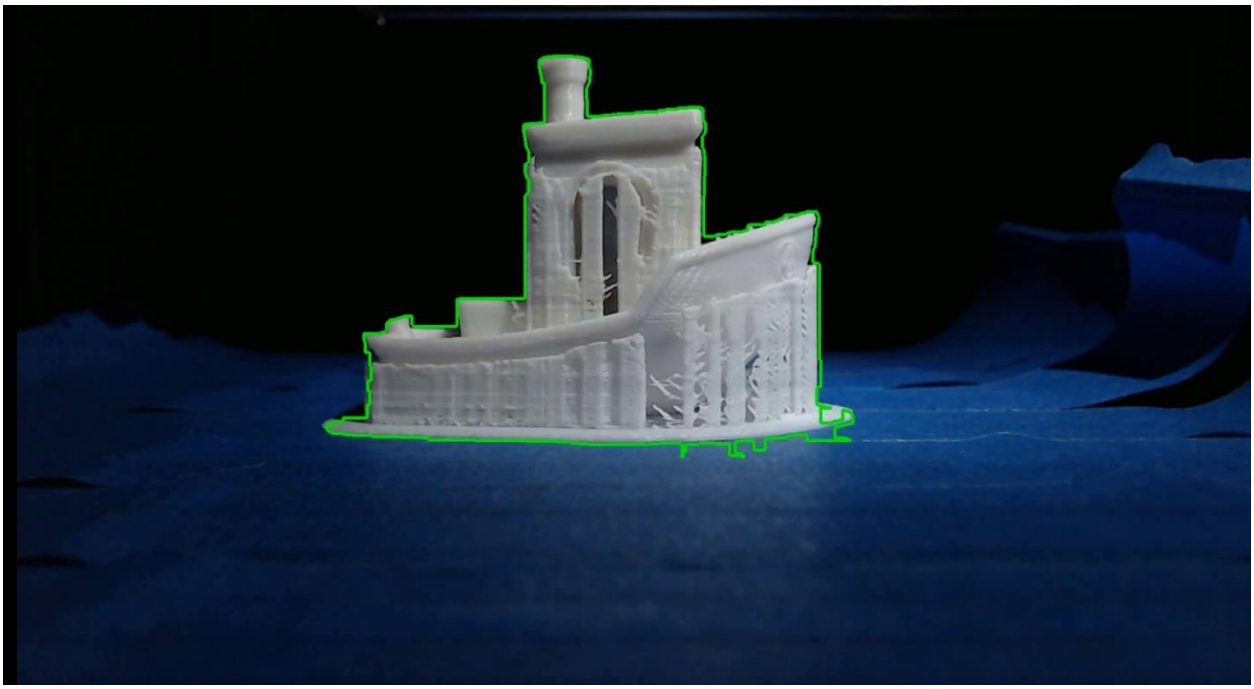


Figure 6.13: Combined Method Outline Extraction

The algorithm was able to extract a rough outline of the part throughout the print. It was not without issues though. It picked up part of the background in the outline still so it was not perfect yet. We continued to work on this algorithm until Experiment 9.6 was run. Experiment 9.6 greatly improved the lighting and print bed color but also caused the combined algorithm to pick up too much of the background.

6.3.10 Building the Comparison Model

At the same time the group was discussing new extraction methods, the group was also discussing methods to extract a generated model that could be used for the comparison. The generated model could be extracted from the CAD model and could be what the outline is compared against. The group came up with two main methods to build the generated model. First was to build the model layer by layer using the G-code generated by the slicing software. The

second was using an image mask to subtract all parts of the print that were above the height of the extracted outline.

The first method discussed, building a model from the G-code, would be the most accurate and the easiest to control but it was the hardest to implement. Ideally, because we knew the layer each snapshot was taken on, we would be able to exactly build the generated model up to that layer. Unfortunately, there were no existing solutions we were able to find so if we did we decided to use this method we would have needed to implement it ourselves. We started to try to do it but, attempting to rebuild a 3D model from generated G-code proved to be too difficult for our group to implement.

After ruling out the first method, we began working on building the generated model by cutting off the unprinted layers. To save time, the program currently relies on the user to take a picture of the model in the slicing software at the correct angle and position but, ideally this will be automated later. The user can then run the program with the given picture and it is able to trim off any percentage of the image.

The program relies on getting the height of the extracted model and then is able to trim the generated model to the same height for comparison.

6.3.11 Outline Extraction with New Edge Detection

After Experiment 9.6 caused the combined method to be no longer accurate, the group decided to revisit edge detection. The combined method was no longer accurate because the change in lighting and background color caused the algorithm to detect too much of the part.
With the more constant lighting, distinct print bed and background color, the new edge detection algorithm was able to get all edges from the printed part. The process of the new edge detection algorithm is described in Section 8.6 Canny Edge Detection v2 and shown in Figure 6.14.

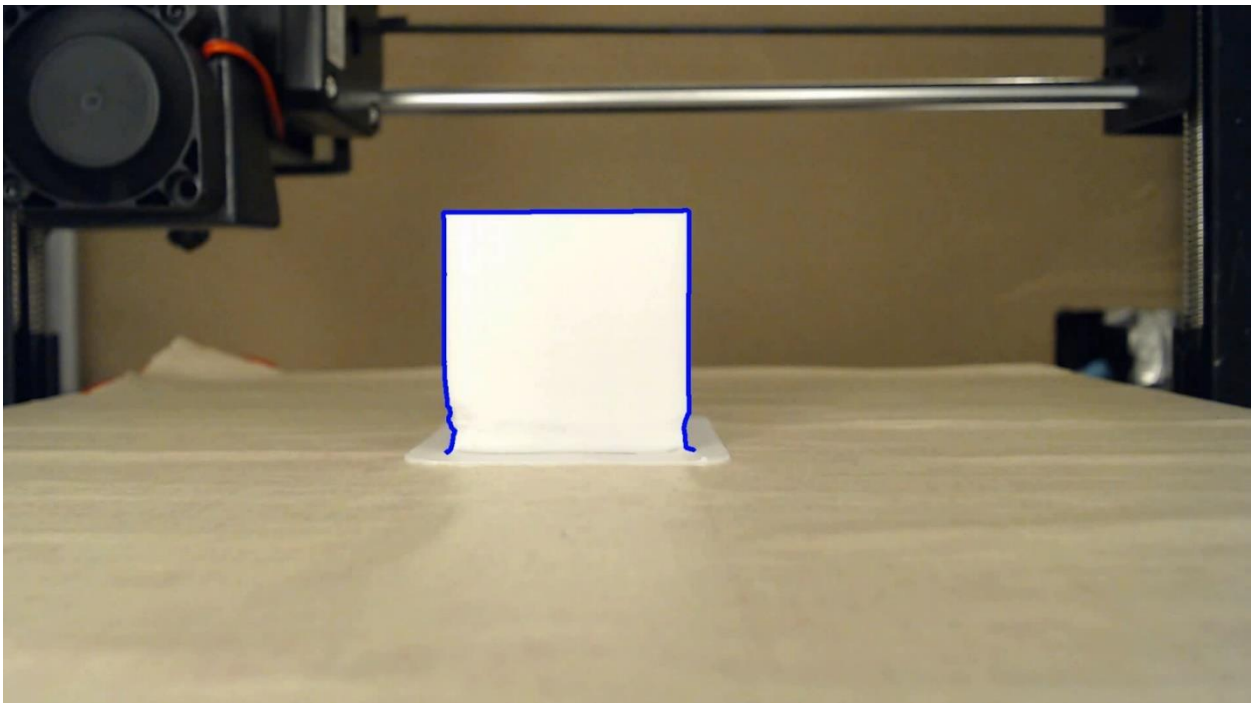


Figure 6.14: New Edge Detection

Like the original edge detection, it also used Canny edge detection. It also included some logic to extract the correct edge from every edge in the frame. First it selects the lowest edge in the image that has a perimeter above a certain threshold then, it ensures that the edge is within a certain percentage of the center X axis. This was needed because there were many other edges picked up that were unneeded such as the edges of the print bed and the edges of the extruder. At this point, the edge detection isolation did not select the correct edge every time, but it was the most usable one so far.

6.3.12 Outline Extraction and Initial Comparison Test

Once the new edge detection algorithm was completed, the group began working on a comparison test to visually see how accurate the generated model generation and outline extraction were. This new program used the new edge detection algorithm defined in Section 8.6: Canny Edge Detection v2 as well as the second model generation method discussed previously.

It then got the height of the extracted outline, cut the generated model to that height percentage and then overlaid the two on a black background. This was an important step in the development of our project because it was the first time that the group had seen accurate outline extraction and model generation together.

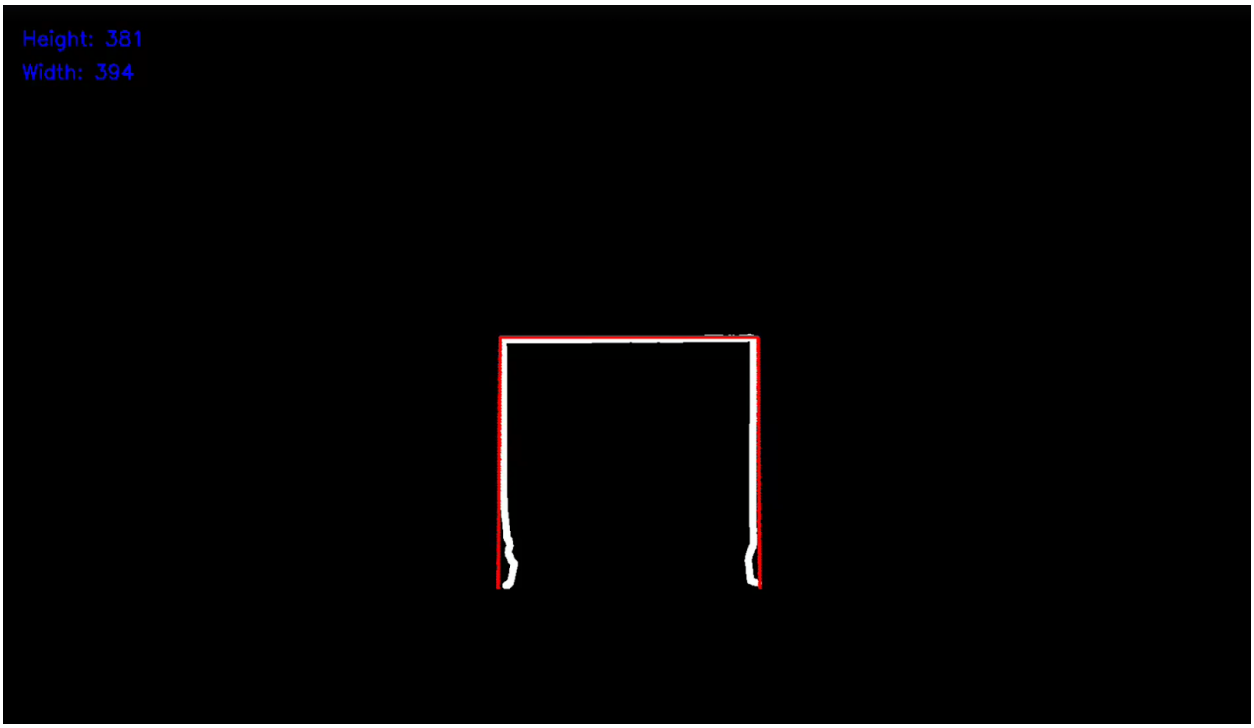


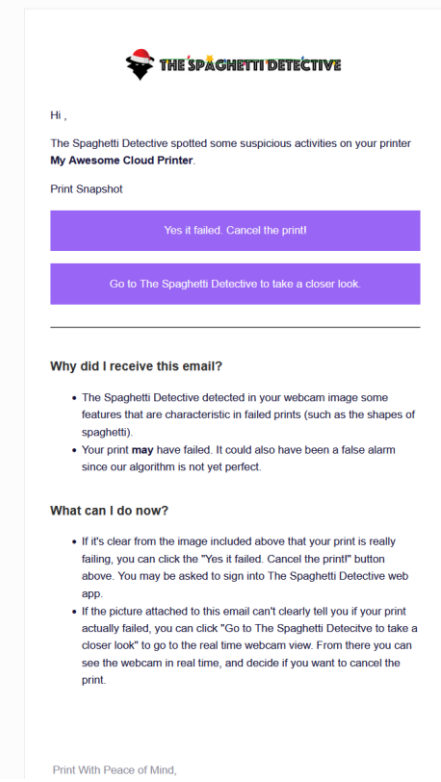
Figure 6.15: Initial Comparison Test. The white shape is the extracted outline and the red shape is the ideal outline

6.3.13 Other External Solutions

Along with developing our own defect detection program, the group looked into other programs. The first one the group looked at was 3DPrintSaviour (See Section 5.2.2 for full description). 3DPrintSaviour did not come with much documentation so, it took a little effort to set up. First, the group needed to set up a second computer to run 3DPrintSaviour. Next, we determined a way to transfer images of the time-lapse from OctoLapse to the second computer in

real time, as they were taken. After that was set up, we ran a print with 3DPrintSaviour running. 3DPrintSaviour did not produce any output so, it was unclear if it was working. When we investigated the code, it appeared to only track if the part is still attached to the print bed and track if breakage occurred. We did not have the capabilities to test those yet so, we moved onto the next solution.

The next existing solution we tested was Spaghetti Detector. To set up Spaghetti Detector we just needed to install another OctoPrint plugin and sign up for the Spaghetti Detector service. Next, we produced a part that would intentionally turn into spaghetti. We then ran Experiment 9.7. Spaghetti Detector successfully found the spaghetti and sent us an email confirming that it was spaghetti (see Figure 6.16).


THE SPAGHETTI DETECTIVE

Hi ,

The Spaghetti Detective spotted some suspicious activities on your printer **My Awesome Cloud Printer**.

Print Snapshot

Yes it failed. Cancel the print!

Go to The Spaghetti Detective to take a closer look.

Why did I receive this email?

- The Spaghetti Detective detected in your webcam image some features that are characteristic in failed prints (such as the shapes of spaghetti).
- Your print **may** have failed. It could also have been a false alarm since our algorithm is not yet perfect.

What can I do now?

- If it's clear from the image included above that your print is really failing, you can click the "Yes it failed. Cancel the print!" button above. You may be asked to sign into The Spaghetti Detective web app.
- If the picture attached to this email can't clearly tell you if your print actually failed, you can click "Go to The Spaghetti Detecitve to take a closer look" to go to the real time webcam view. From there you can see the webcam in real time, and decide if you want to cancel the print.

Print With Peace of Mind,

Figure 6.16: Spaghetti Detective Email for a print with spaghetti

After confirming that Spaghetti Detector found spaghetti through the email, it stopped the print. It could be possible to integrate Spaghetti Detector into our own solution so we can use the working spaghetti detection as well as the working notification system (See Section 11.2).

6.3.14 Further Outline Extraction Activity

After the initial comparison test using the new edge detection algorithm defined in Chapter 8: Method 6, we continued to try to improve the extraction algorithm to better find the outline of a part on the print bed. At this point, the main issue with our edge extraction algorithm was not detecting the edge of the part. The problem was trying to determine a method to isolate the edges that represented the outline of the part from every other edge detected by the algorithm. Figure 6.17, shows all the edges detected by the algorithm in a single frame from Experiment 9.8. Many of the other visible objects in the frame are being detected by the algorithm, such as the extruder gantry and the edge of the print bed. These extra edges need to be filtered out which, although easy to do by eye, proved difficult to automate.

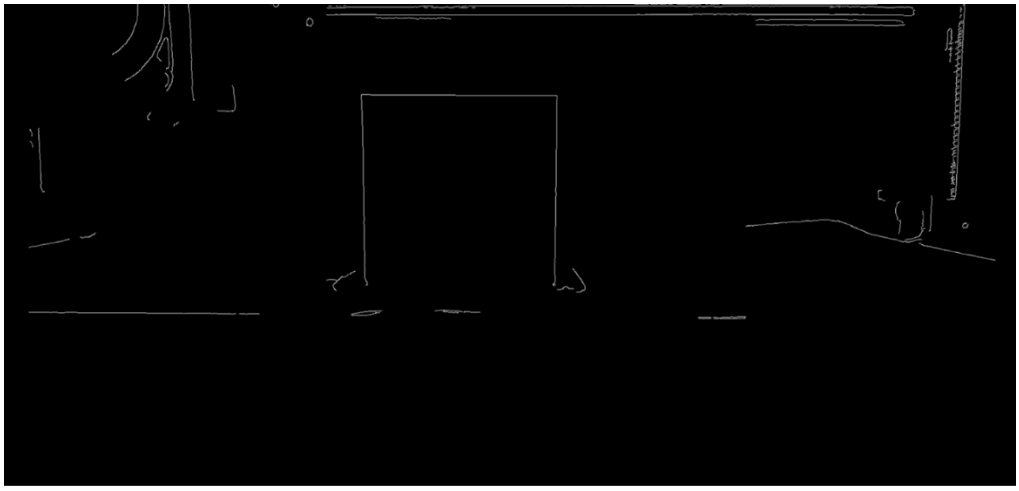


Figure 6.17: A Frame from the Edge Detection Algorithm

After attempting to create an automatic extraction algorithm that was general enough to work on multiple parts, the team had to decide if we should continue working on this. If we decided to continue working on this method, we would have a few steps we would need to complete. First, we would need to find a working edge extraction algorithm. Then we would have to determine a better method to extract the outline of the part from the CAD model. Finally, we would have to create a method to compare the outline from the CAD model to the outline from the part on the print bed. Our deadline for the project was starting to draw near so we were unsure if this path would produce satisfactory results, if any. So, the team made a table of common defects and the methods we could use to detect them. This table is listed below in Figure 6.18.

Defect Name	Able to Detect	Method	Needs CAD Model
Spaghetti	Yes	Spaghetti Detective	No
Warping *(Straight Edge Part)*	Yes	Line Detection	No
Warping *(General)*	No	Model Comparison	Yes
Filament Run Out	Probably *(needs testing)*	NRMSE	No
Bed Detachment	Probably *(needs testing)*	NRMSE	No
Lifting	No	Darkness Comparison	No
Layer Shifting	No	Model Comparison	Yes
Blobbing	No	Unknown	Unknown

Figure 6.18: Table of Defects and Potential Methods to Detect Them

6.3.15 Changing Goals

After listing out several common types of defects and the methods we could use to detect them, the team decided to change the types of defects we were going to try to detect. Originally,

we were planning on detecting warping and layer shifting using outline extraction and model comparison. We decided that this would probably take too long to complete, and it may not produce satisfactory results. So, we decided to try to change our focus to detecting filament run out and bed detachment. These two defects were also targeted in 3dPrintSaviour so, we decided to reference the methods they used. As discussed in Sections 5.2.2 and 6.3.13 3DPrintSaviour uses the Normalized Root Mean-Squared Error (NRMSE) algorithm to compare the differences between two frames generated by OctoLapse. We decided to try and implement our own algorithm using NRMSE to detect filament run out and bed detachment.

6.3.16 Initial NRMSE Algorithm Development

The NRMSE algorithm first converts each frame to black and white then, using OpenCV's absolute difference algorithm, it determines what has changed between the current frame and an initial frame of the empty print bed. Then, using the NRMSE algorithm, it compares this frame to the previous frame. We also compared the current frame to the frame five frames before, to amplify differences. We then plotted the scores on a graph so we could visually assess them for usable patterns. One of these charts is pictured in Figure 6.19. This algorithm is explained in more detail in Section 8.7: NRMSE Comparison Algorithm.

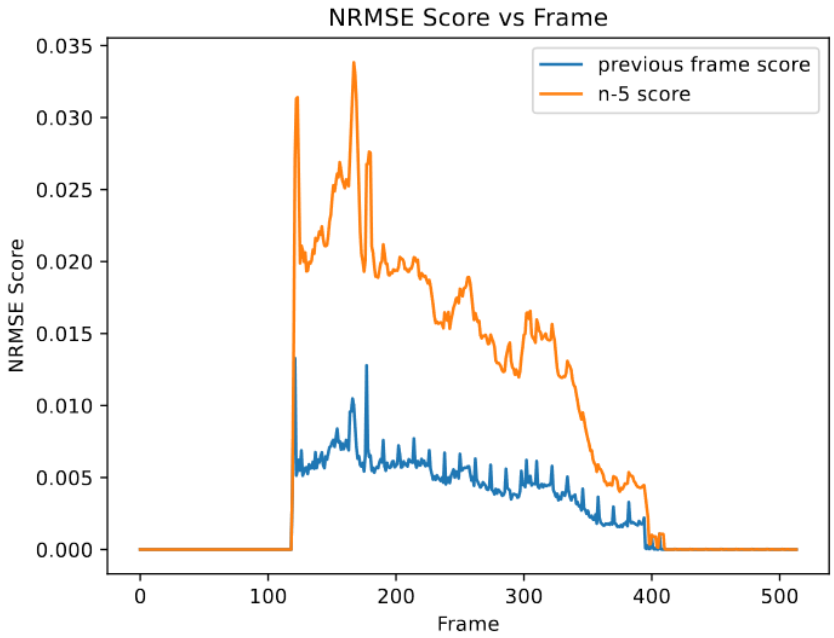


Figure 6.19: An output plot of the NRMSE algorithm run on Experiment 9.8 [Complete – not runout]

This plot seems to have a consistent pattern, so we were hopeful that detecting runout and slippage would be easy. Unfortunately, once we ran a print where the filament ran out midway, we encountered a problem. Figure 6.20 shows the output plot of Experiment 9.10 [Cutoff]. This print had run out of filament around frame 200 (on the x-axis).

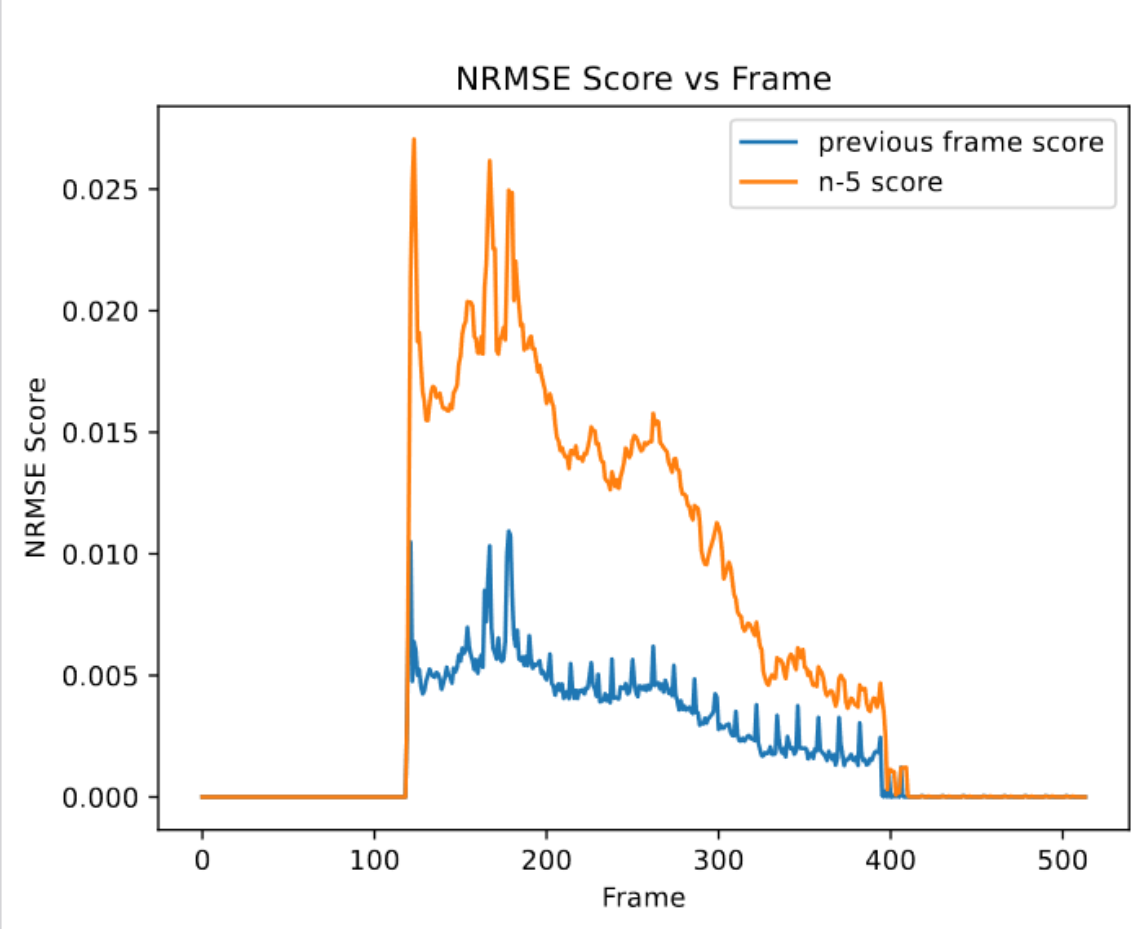


Figure 6.20: An output plot of the NRMSE algorithm run on Experiment 9.10 [cutoff]

The problem was that while the NRMSE score does have a slight downward trend when comparing the current frame to the frame five before when the print runs out of filament, it is not as dramatic as we were hoping. We then realized that the motion of the extruder upward had the effect of constantly adding some difference to the print. We decided that we needed to try and isolate the area around the part or remove the extruder so that we can compare only the differences of the part.

6.3.17 Background Removal Attempts

We decided we needed to determine a method to remove the extruder from the frame automatically. We first decided to try to remove the extruder from the frame by manually cropping it out. We did this just to test our hypothesis that the extruder was causing the extra difference in our NRMSE plots. After manually cropping the time lapse videos, we reran the NRMSE algorithm on Experiment 9.8 and Experiment 9.10. Figure 6.21 and Figure 6.22 show the results.

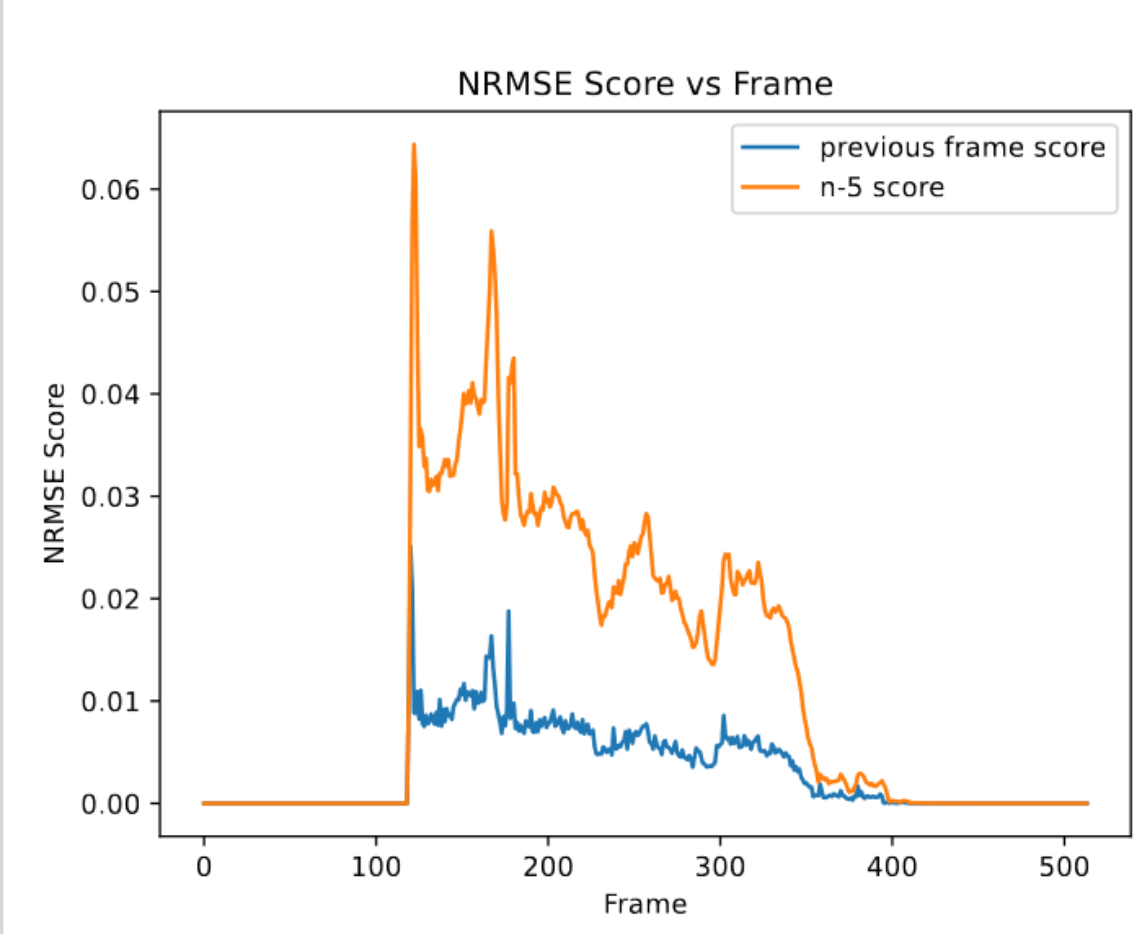


Figure 6.21: An output plot of the NRMSE algorithm run on the cropped version of Experiment 9.8 [Complete – not cutoff]

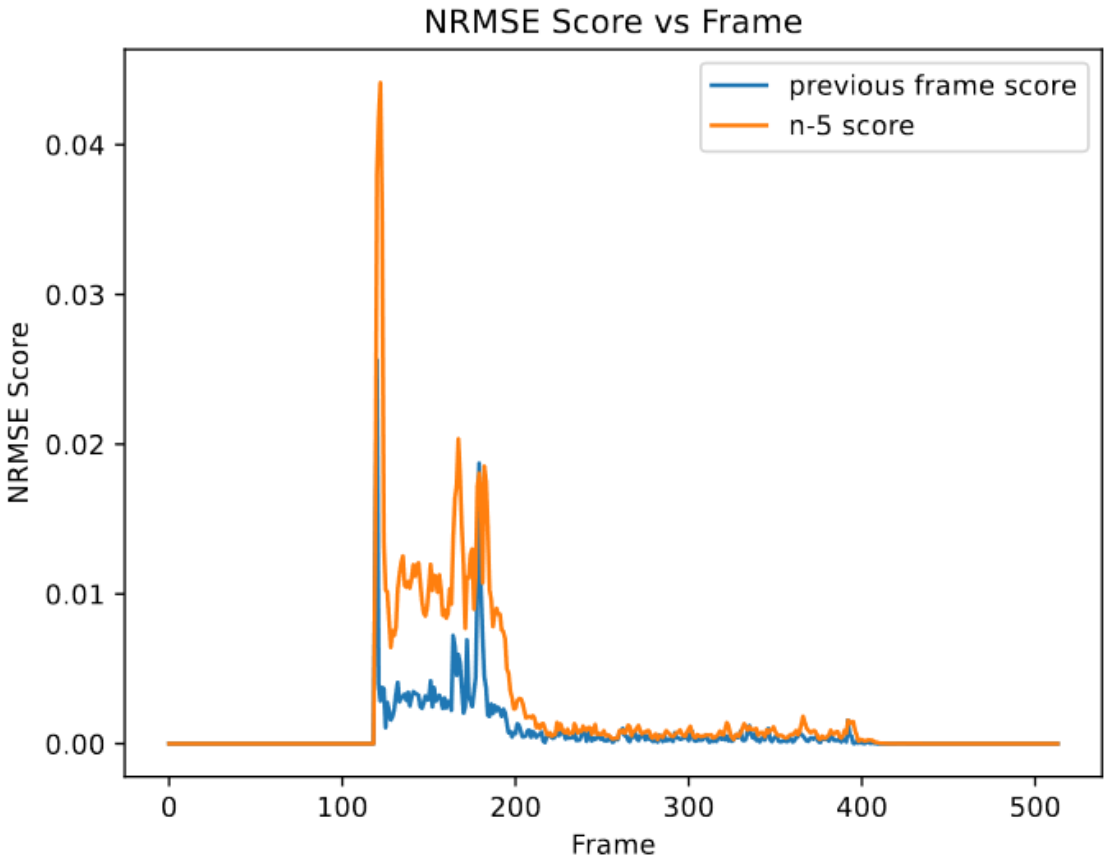


Figure 6.22: An output plot of the NRMSE algorithm run on the manually cropped version of Experiment 9.10 [cutoff]

Cropping made the NRMSE algorithm produce the dramatic change in score that we originally expected. Once Experiment 9.10 and 9.12 stopped printing, the NRMSE score dropped to nearly zero. This test solidified the idea that we needed to determine how to remove or reduce the influence of the extruder. We tried to determine a method to automatically crop the image of the part on the print bed, but this raised the same problem we previously faced when we tried to isolate the outline of the part.

6.3.18 Automatically Blocking Out the Extruder

The next method we tried to use to remove the extruder was to replace the extruder with what was behind it. We could do this using a picture of the empty print bed. Then, by determining where the extruder was, we could replace the extruder with what should be behind it using an OpenCV function. Unfortunately, similar to trying to extract the outline and determine where the part was in the image, automatically determining where the extruder was proved to be difficult.

6.3.19 NRMSE Algorithm with Empty Print

After we completed the camera mount, the camera angle and position were constant between each print. This meant that we could generate a time lapse without any part on the print bed so only the extruder is in the frame. We could then use the OpenCV absolute difference algorithm to remove this part of the frame from each frame in a time lapse with a part on the print bed. Because OctoLapse takes snapshots at a consistent interval, equivalent frames in the empty time lapse and the experiment time lapse would have the extruder at the same height. This algorithm is fully defined in Section 8.8: NRMSE with Empty Timelapse. We then created several more tests to run against the new algorithm. Some with slippage, some without. The team discovered that there was a recognizable pattern in the charts with slippage and the parts without. This is what we were looking for in an algorithm. The prints with slippage were easily differentiable from the prints without. We now just had to create a method to detect when the slippage pattern was occurring.

6.3.20 Using Rolling Average and Outliers to Detect Slippage

The first method we thought of to detect slippage was to use a rolling average of the NRMSE score with outlier detection. An outlier would be any score that is 50% above or below the current average. We decided this because when looking at the figures above, we determined that the successfully printed parts stayed within a consistent range of the average for the duration of the print while the parts with slippage were inconsistent. Because the charts are from different shaped parts, we thought this would be a good general algorithm.

The algorithm we developed is defined fully in Section 8.9. In brief, the algorithm first calculates the NRMSE score using the same method as described in 6.3.19. Then, it adds that score to a list of scores so it can calculate the rolling average. After the list has more points than our defined limit, it removes the oldest score from the list. We decided to use 15 points for our rolling average limit because it was big enough to not be affected by the small spikes of the NRMSE algorithm but big enough to react to consistent trends. We then had to determine when to consider a point an outlier. Through trial and error, we decided that any point 1.5 times greater than the rolling average or lower than 0.5 times the rolling average was an outlier. Figures 6.23, 6.24, 6.25 and 6.26 show output plots of the NRMSE algorithm with the lower and upper bounds of an outlier plotted along with the NRMSE score.

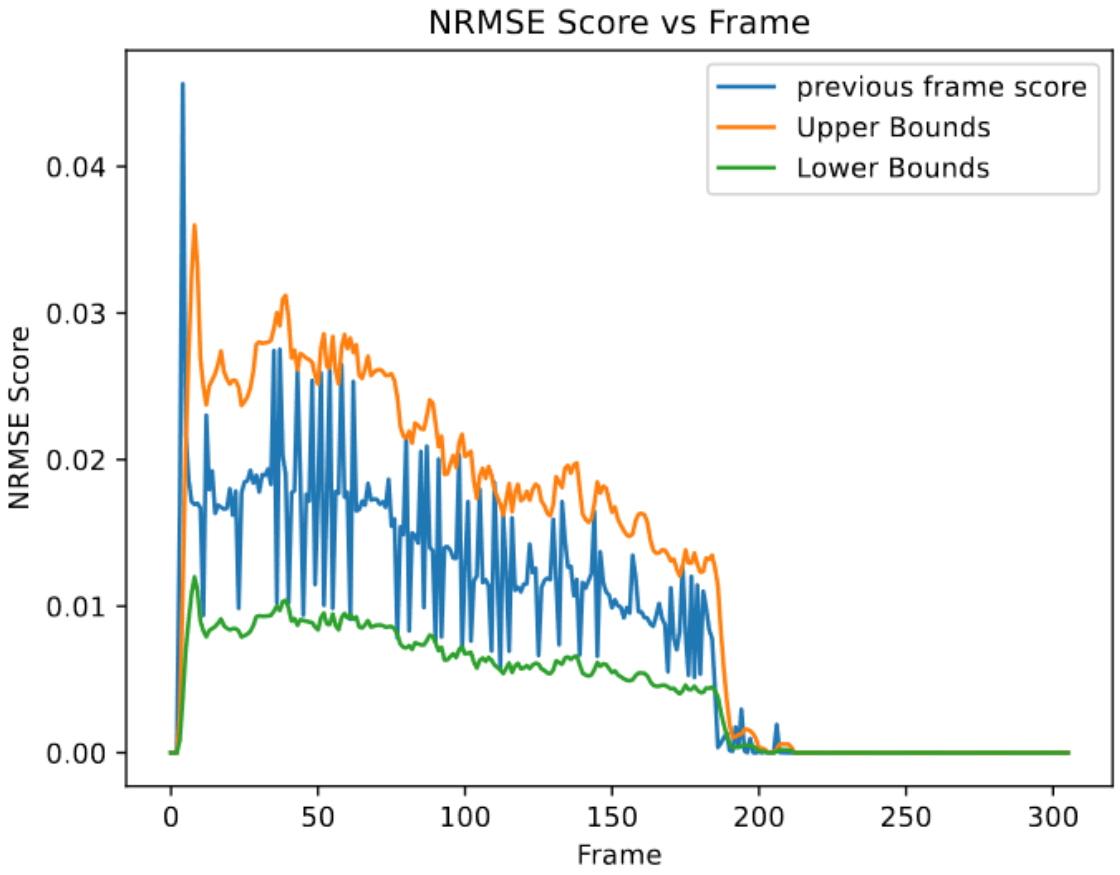


Figure 6.23: NRMSE algorithm plot for a part without Slippage. Note the Y-axis scale

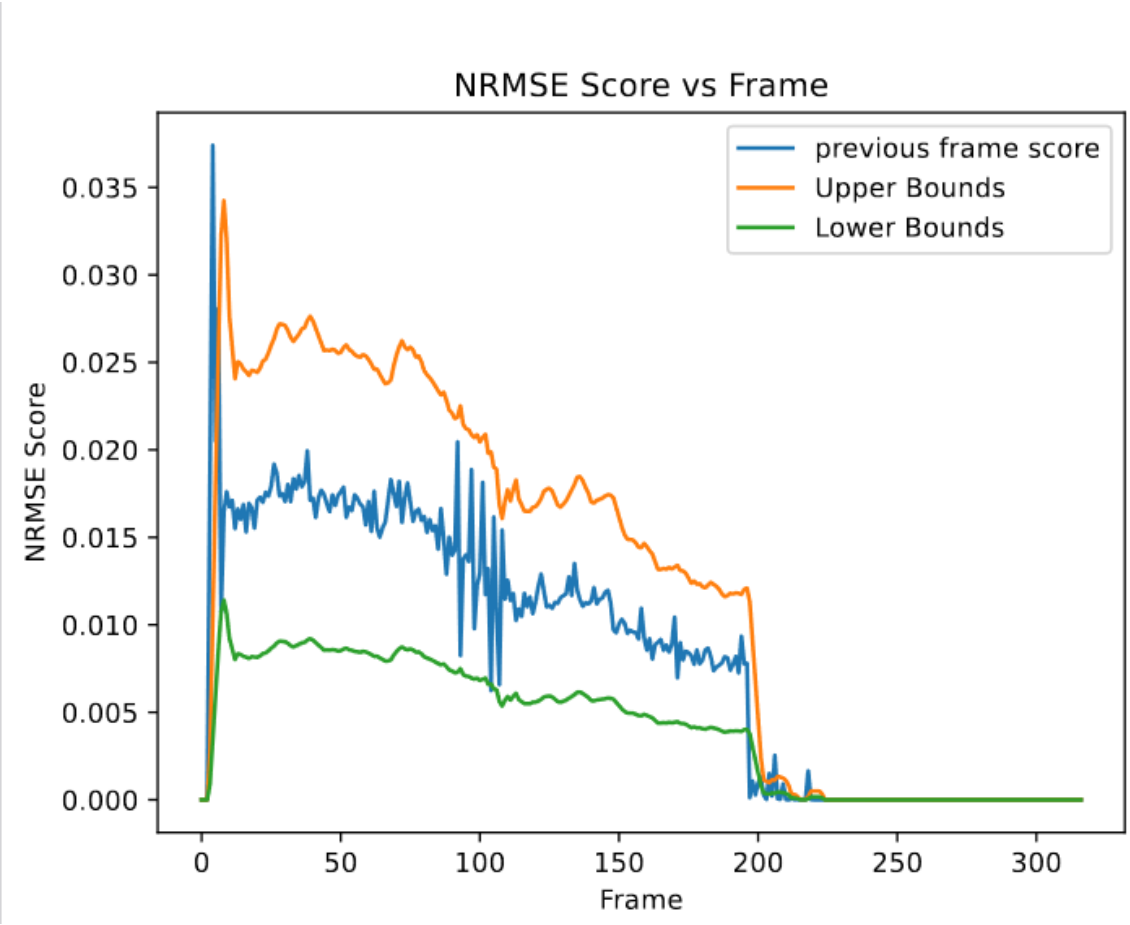


Figure 6.24: NRMSE algorithm plot for a part without Slippage

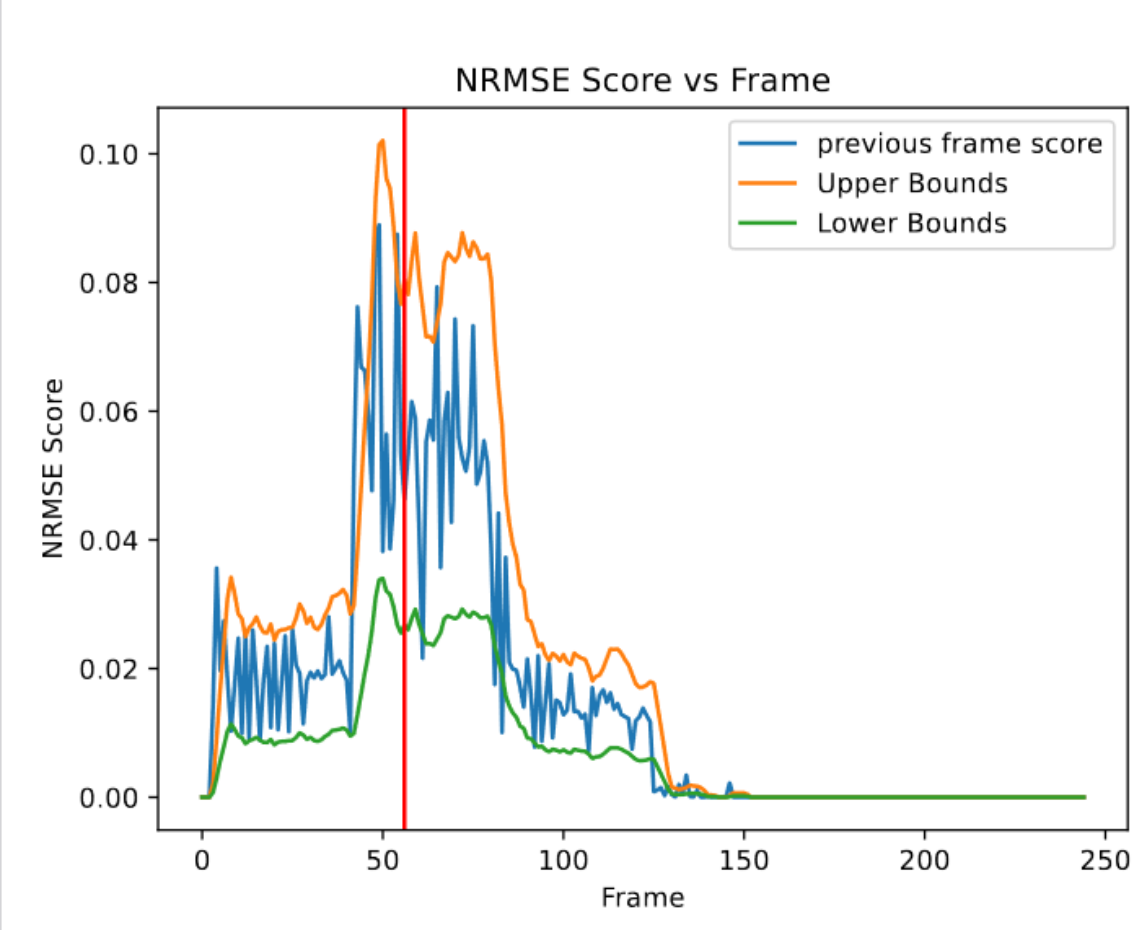


Figure 6.25: NRMSE algorithm plot for a part with Slippage. The red line indicates the frame which the algorithm detected the slippage

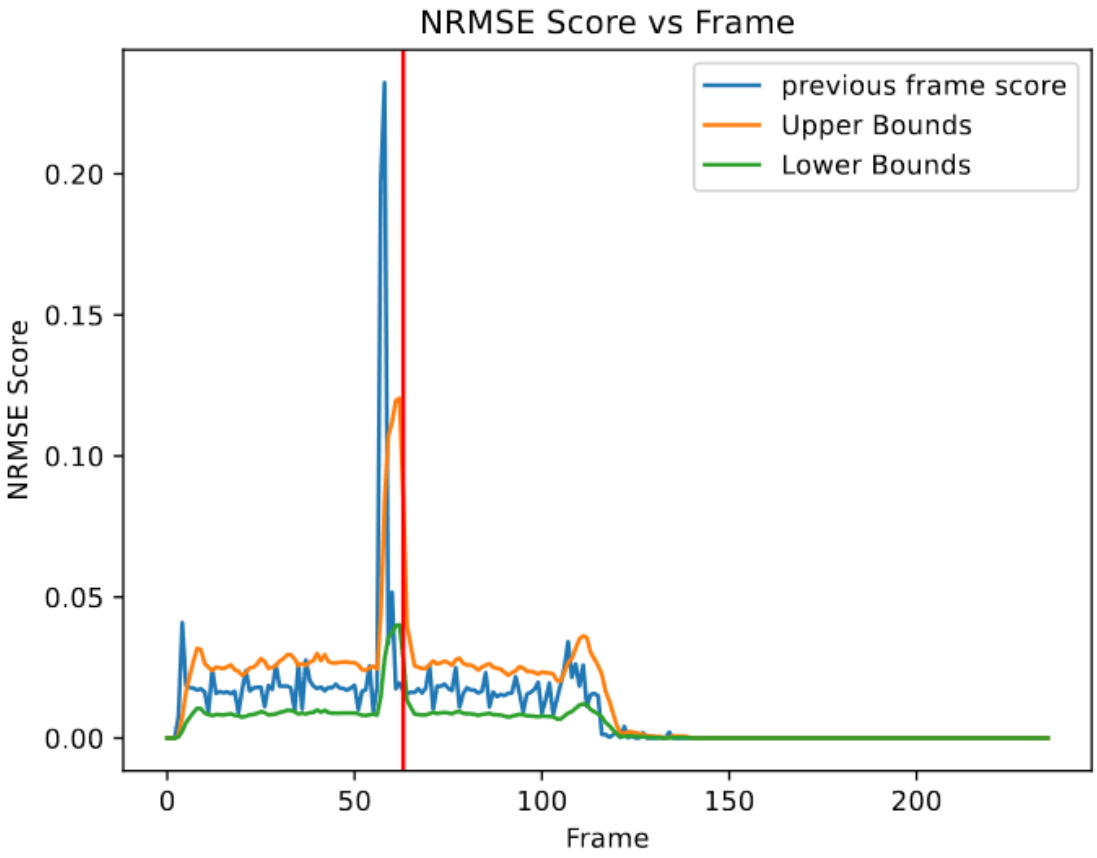


Figure 6.26: NRMSE algorithm plot for a part with Slippage. The red line indicates the frame the algorithm detected the slippage

As can be seen, the successful prints constantly stay within the range (except for the initial difference at the beginning). The parts with slippage, on the other hand, occasionally go either above or below the bounds. We then needed to decide how many outliers were too many in a print. We started with 15 to prevent false detections of slippage but this proved to be slow, not detecting slippage until about 30 minutes after it occured. Decreasing the value to 5 increased the speed and had no effect on the prints without slippage. Anything lower than this caused false positives with the successfully printed parts but improved the detection speeds of the defective

parts. These values could potentially be set by the end user in the completed defect detection system so that they can weigh the pros and cons themselves.

6.3.21 Connecting the Algorithm to OctoPrint

After successfully detecting slippage, the next step was to connect the algorithm to OctoPrint so it could analyze prints in real time. In order to do this, the algorithm would need to be expanded in two ways. First, OctoPrint needed a way to send the timelapse snapshot to the algorithm. Second, the algorithm needed a method to receive the snapshot and process it. We chose to create a web server to receive the images so that the algorithm could be run on a separate computer. This was done by using a framework called Flask (See Chapter 3 for more information). After the web server was set up, the algorithm could be deployed to a server and was almost ready to be used in a live environment. The only thing left to develop was the script to transfer snapshots from OctoLapse to the algorithm. This was quickly created so that every image was sent to the program for analysis. We also added an email that can alert the user when slippage has occurred.

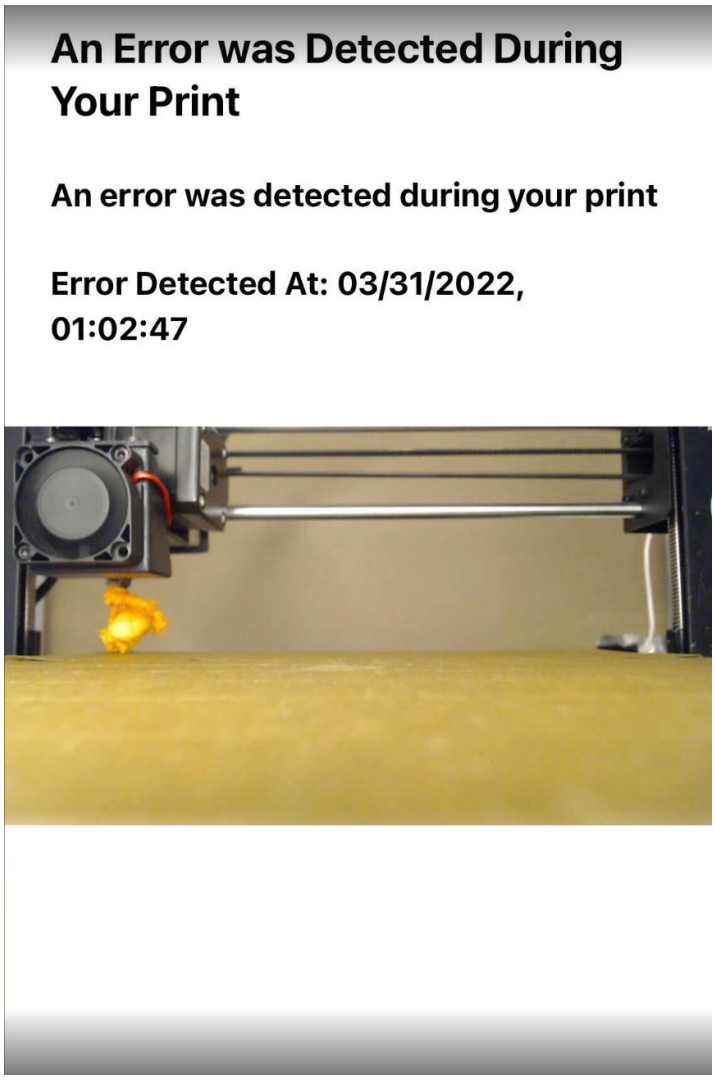


Figure 6.27: An example email sent from the detector once slippage is detected

We were able to test the live detector and successfully received an email alert when slippage occurred.

6.3.22 Runout Testing

Although filament runout was originally one of the defects that was targeted, the team was not able to get the detector working for it. Even with the empty timelapse, the NRMSE score did not substantially change compared to when the printer still had filament. Unfortunately, this meant that the team was unable to detect it. Figures 6.28 and 6.29 show the NRMSE plots from a

print from Experiment 9.9 and 9.12 respectively. Runout was induced around frame 100. Even though the NRMSE score begins to decline, it was not substantial enough to detect.

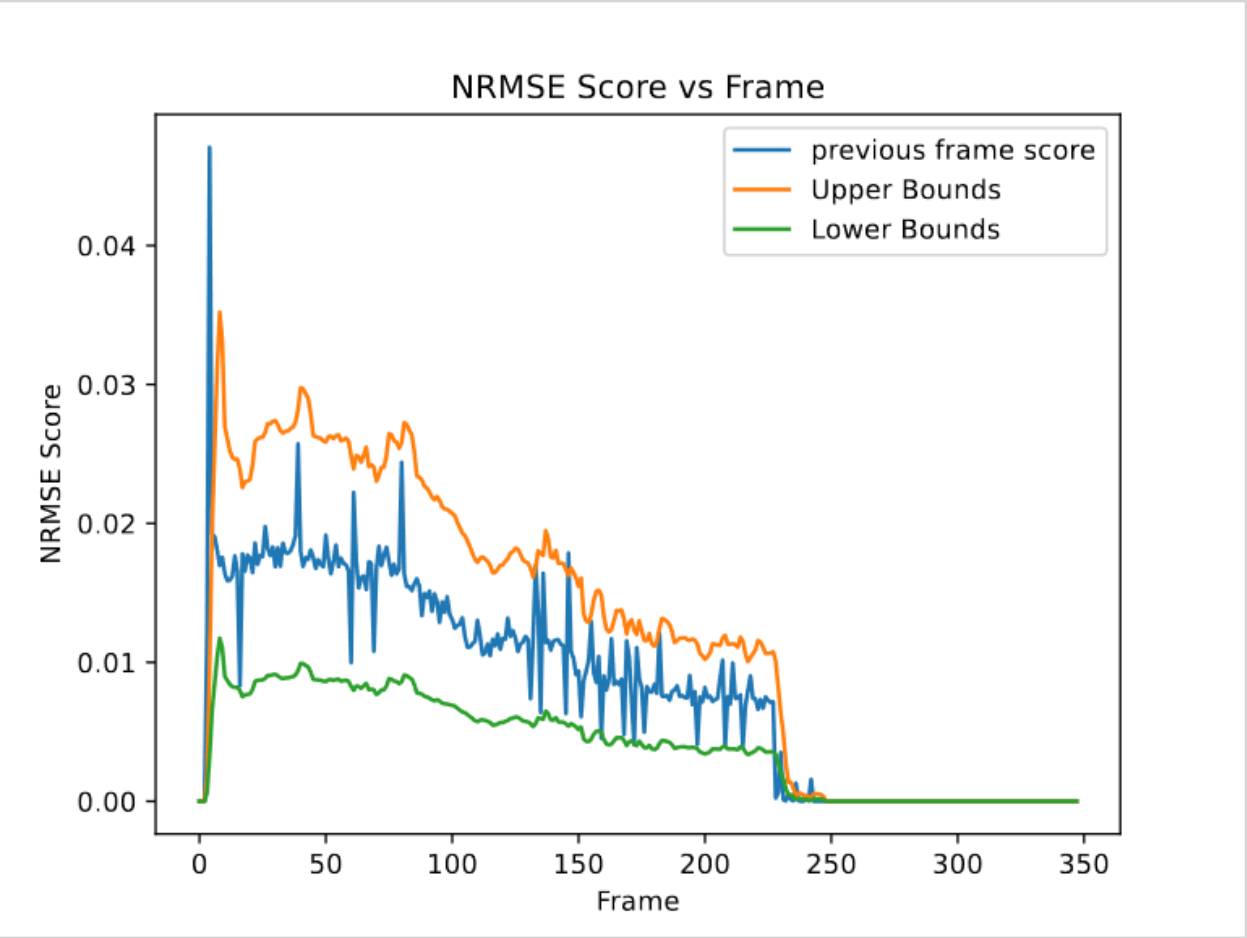


Figure 6.28: NRMSE plot of a part from Experiment 9.9: Runout on the Monoprice Printer

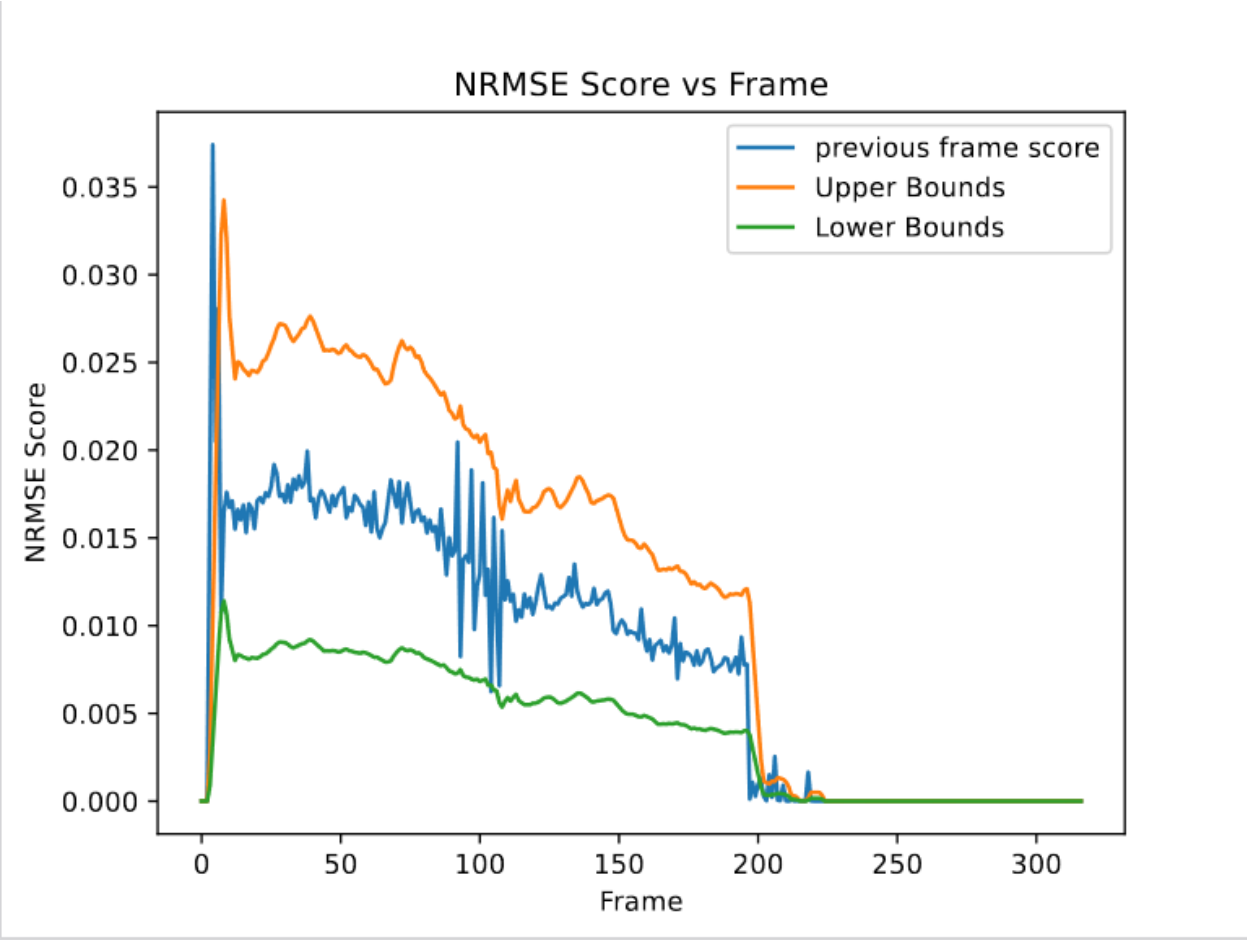


Figure 6.29: NRMSE plot of a part from Experiment 9.12: Runout on the RepRap Printer

Chapter 7: Final Design

There were two main aspects of the project in which design decisions were needed. These two aspects were the hardware setup for the 3D printers and their environments and the software that was used to detect slippage and filament runout. This chapter will discuss the various design decisions that were made for hardware and software.

The hardware subsections will break down further into two sections, printer environment and equipment and its fixturing. Printer environment consists of the decisions related to lighting, background and enclosure. The equipment and fixturing section will discuss decisions related to camera selection and its fixturing.

7.1 Hardware Design

This section will provide details on the finalized hardware setup for the two 3D printers used in this project. These hardware decisions were based on literature and experimentation as illustrated in Chapters 5, 6, and 9. The goal of our hardware implementation was to create a system in which our software and camera could view a consistent background as well as the print bed during a 3D print.

7.1.1 Hardware Requirements

Through several experiments, described in the methodology and experimentation sections, we determined the underlying requirements for different components of the system. This section outlines those requirements for the environment (i.e., printer surroundings, lighting, and background), as well as the requirements for the camera and fixturing device.

7.1.1.1 Environment Requirements

Early on in the project we had made the decision to avoid using machine learning because of the fact that it can be quite complex and we had limited time to complete the project. Due to this constraint, we had to be wary of the various disadvantages of not using machine learning, and develop our hardware to work around those disadvantages. One of the main advantages of machine learning is that it could more easily adapt to inconsistencies and non-standard data sets. In order to compensate for this without using machine learning, we had to make sure that the data we collected was as consistent as possible.

External factors such as the room lighting, time of day, shadows from moving entities, and more, all caused uncertainties in the way that the part looked from the camera's perspective throughout the print. In order to eliminate these factors, it was necessary to find a way of controlling the lighting around the printer. This required blocking all lighting outside of our control, and implementing a controlled lighting design to illuminate the part. We wanted to make sure the lighting design contributed to the goal of making the system modular and adaptable to different setups. To this end, we decided that adjustable lighting was required, since the warmth and brightness of the lighting had different effects on various filament colors, and it would be impossible to account for all of them with a single lighting design. We also determined that harsh shadows would make it harder for the algorithm to distinguish between the part and the background. So, it was necessary to minimize the appearance of harsh shadows from the camera's perspective.

Additionally, the background and print bed color played an important role in the algorithm's data analysis. A machine learning system could be trained to recognize parts and defects against various different backgrounds. However, for our algorithm, it was necessary to control the color

behind the part so that it could be consistently recognized and defects could be more easily detected. Since it was important for the algorithm to be able to tell the difference between the part and the background, we had to choose a background and bed color that would not blend in with common filament colors. We also had to be aware of the functional aspect of the system, and make sure that the method of coloring the bed would not interfere with the printer's ability to perform (e.g., causing problems with bed adhesion).

7.1.1.2 Camera and Fixturing Requirements

Similarly to the environment setup, the camera, with its fixturing device, had to be designed to collect quality data consistently in order to effectively aid the algorithm. The requirements for the camera were related to the quality of the images it collected for the algorithm to use. The requirements for the fixture were related to the consistency of images taken throughout the print.

There were many factors that affected the usefulness of images taken by the camera. The main factor was the resolution of the camera. A higher resolution would mean higher quality images, which meant that smaller defects could potentially be detected by the algorithm. Therefore, we wanted the camera to have the highest resolution possible while also considering things such as cost and availability.

Another factor affecting image usefulness was the dimensions of the image. We needed the images to only contain the necessary information about the part, with no unnecessary details within the frame. This would reduce the need for post-processing of the images. The last factor we considered was the ease of fixturing the camera. Since it was important to take consistent images, we determined that having a consistent camera fixture would be integral in making sure that the camera did not move during the print. Therefore, it was important to have a camera that could easily be attached to a camera fixture.

The camera fixture itself needed to fit three main requirements. First, it needed to be sturdy in order to hold all the necessary components without shaking or sliding during the print. Second, it needed to be modular with interchangeable parts in order to fit different cameras and printers. Lastly, its manufacturing requirement was that it needed to be 3D printed and could be done on any hobbyist FDM printer.

7.1.2 Environment Setup

7.1.2.1 Printer Surroundings

The printers used for testing this system were open style FDM 3D printers without any enclosure built into their structures. Implementing an enclosure for open style FDM printers typically serves to reduce the effects that humidity and temperature can have on filament. However, for our system, we used 3D printer enclosures as shown in Figure 7.1 to meet the requirement of blocking all light outside of our control. The enclosures covered all sides of the printers, with the exception of a clear plastic window on the front flap which can be seen in Figure 7.2.

Figure 7.1: Enclosures with width and height dimensions. RepRap printer with small enclosure shown on the left. MonoPrice printer with the large enclosure shown on the right

Figure 7.2: Clear plastic window on the large enclosure

For this system there are two different enclosure sizes that house the printers. The larger of the two covers the Monoprice printer and the smaller enclosure covers the RepRap printer. Both enclosures are made by Creality and have black exteriors with aluminum lined interiors. The enclosures consist of a series of rods that connect to plastic joints in each corner to create a frame. The black covering then goes over the frame. The larger enclosure has dimensions of 29.5 in x 27.5 in x 35.4 in and costs $87.55. The smaller enclosure has dimensions of 17.5 in x 22.2 in x 26.9 in and costs $64.59. The entire front portion of the enclosures can be opened and closed via a zipper, giving access to the printer inside.

Since the clear plastic windows allowed external light through to the print bed, we added opaque barriers to cover the windows so that no light could enter the enclosure when closed. For each printer, we used a different barrier material and a different manner of fixturing the barrier. It does not matter what method is used if light is blocked from entering the window. Figure 7.3 shows the cardboard that was added to the small RepRap enclosure. It was cut to fit the size and shape of the opening, then attached to the top, front bar of the enclosure via copper wire wrapped around the bar and through holes in the cardboard. The cardboard pivots freely on the top bar, and can be flipped up to the top of the enclosure so the user can access the printer. Figure 7.4 shows the poster board that was placed in the front of the Monoprice enclosure. It is kept in place by wedging it between the enclosure fabric and the frame bars on the bottom and side of the front of the enclosure. To access the printer, the user must remove the poster board entirely and place it to the side.

Figure 7.3: Cardboard window barrier on the small RepRap enclosure

Figure 7.4: Poster board window barrier on the large Monoprice enclosure

With the window barriers in place, little to no outside light has the ability to enter the enclosure when closed. This meets the requirement of blocking all lighting outside of our control. The closed enclosures with window barriers in place can be seen in Figures 7.5 and 7.6.

Figure 7.5: Small RepRap enclosure closed with the cardboard window barrier in place

Figure 7.6: Large Monoprice enclosure closed with the poster board window barrier in place

To measure the effects of the enclosure and the window barrier, we ran the algorithm on blank timelapses with and without the enclosure closed. A blank timelapse is a timelapse of the print bed where the printer is printing nothing. It simulates baseline background changes like the extruder moving upwards with each layer. These time lapses were both taken with the final lighting design, which will be discussed in detail in the next section. Figure 7.7 shows a comparison of the graphical output of NRMSE scores for the two blank timelapses. With the enclosure and window barrier, the printer is able to run prints without any outside light affecting the consistency of the images taken during the timelapse. This is reflected by the more consistent NRMSE scores in the graph on the left side (maximum score of ~0.025), as opposed to the occasional abrupt spikes that occur in the graph on the right (maximum score of ~0.35). Those

abrupt spikes are what the algorithm looks for to detect slippage, so it is crucial that the system's ambient lighting does not cause those spikes.

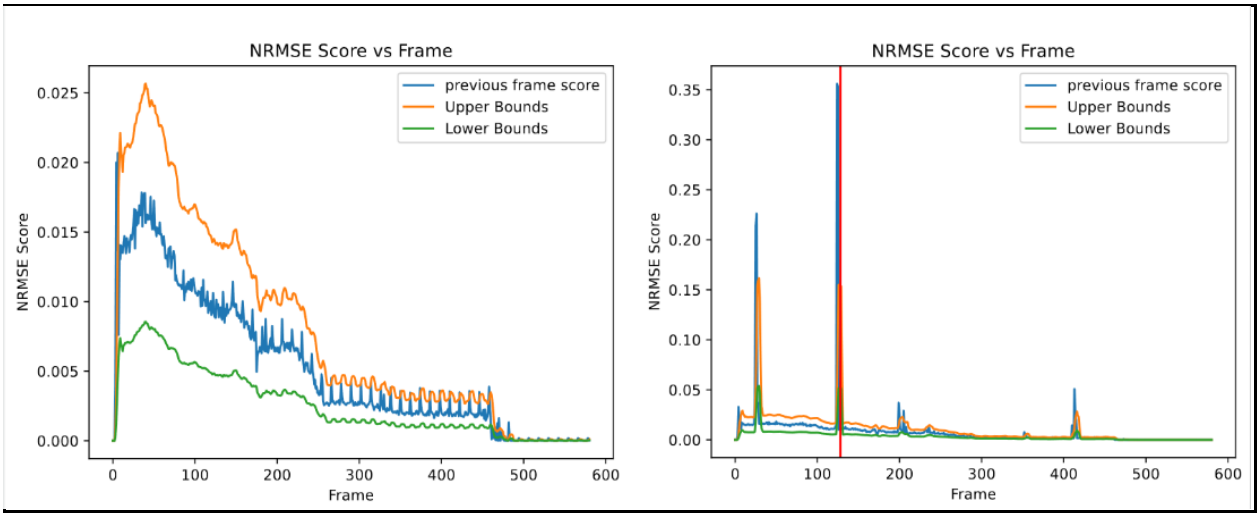


Figure 7.7: Comparison of NRMSE graphs for closed (left) and open (right) enclosure. Note the scale of the y-axis

7.1.2.2 Lighting

Through many iterations, we developed a system of lighting to consistently illuminate the bed while also helping to differentiate the part from the background. The final design consists of two parts; a ring light, and background LEDs, as shown in Figure 7.8.

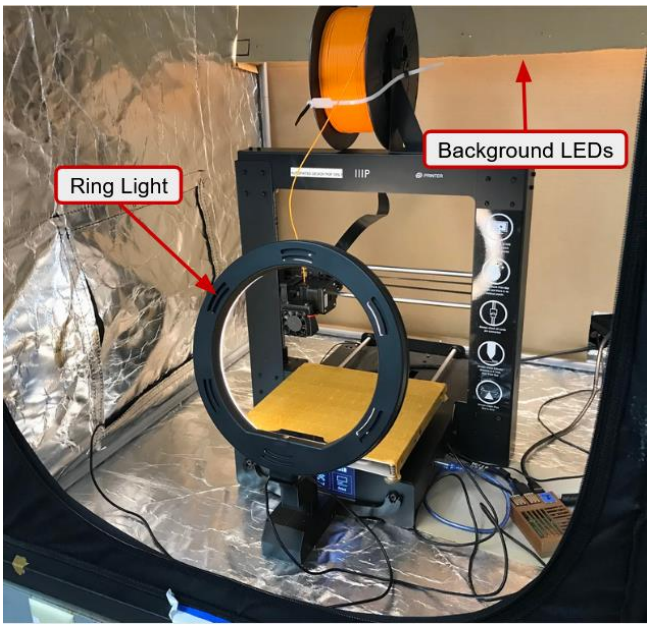


Figure 7.8: Lighting setup in MonoPrice enclosure

The ring light sits behind the camera and illuminates the part and the print bed. It also has adjustable warmth and brightness settings, which helps to meet the requirement of a modular and adaptable system. Figure 7.9 shows a comparison between the three warmth settings on the lowest brightness setting. We found that anything higher than the lowest setting caused the part to appear too bright, obscuring many details of the contours. This was especially true for light filament colors like white and yellow. Additionally, even at the lowest brightness, the neutral light also caused the white part to be too bright, which can be seen in the middle image of Figure 7.9. Therefore, we decided to use the warmest, dimmest light setting, as that proved to be a good baseline that worked for a wide range of filament colors. Details of how we came to this conclusion can be found in Chapters 6 and 9.

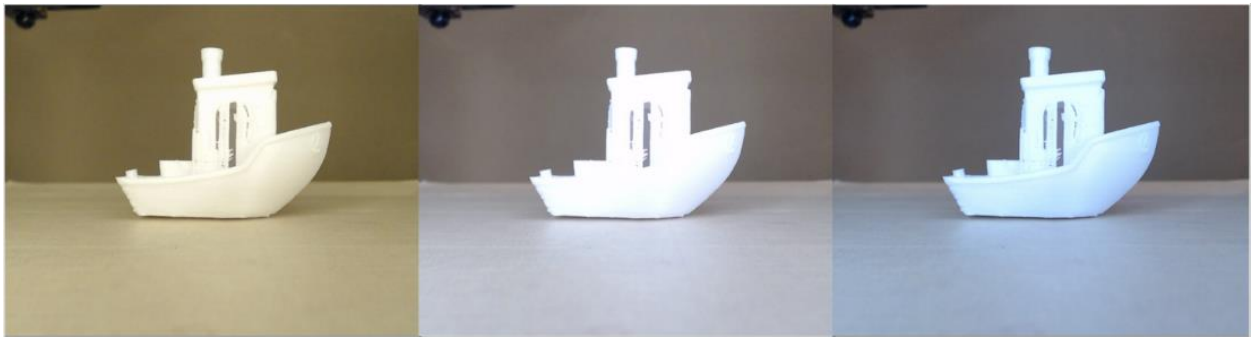


Figure 7.9: Comparison of warmth settings at the lowest brightness, with warm, neutral, and cold from left to right.

The background LEDs are located at the top, back of the enclosure. They illuminate the background to ensure that it can easily be seen from the camera's perspective. The LEDs also have a similar warmth to the warm setting of the ring light, which helps to match the background color to the bed tape color. By matching these colors more closely, the algorithm can more easily differentiate the part from the background in order to subtract the background. The details of this operation will be explained in sections 7.1.2.3, 7.2 and Chapter 8.

7.1.2.3 Background and Print Bed Color

Both enclosures have an aluminum lining that make up the interior of the enclosure, which can be seen in Figure 7.10. This lining has non uniform wrinkles spread throughout, which causes issues with distinguishing the part from the background. To combat this issue, we placed a cardboard background inside the enclosure behind the printer. We determined that cardboard provided the best color for differentiating the background from many different common filament colors while being readily available for us to use. The reasoning and experiments that led to this decision are laid out in section 6.1.2.

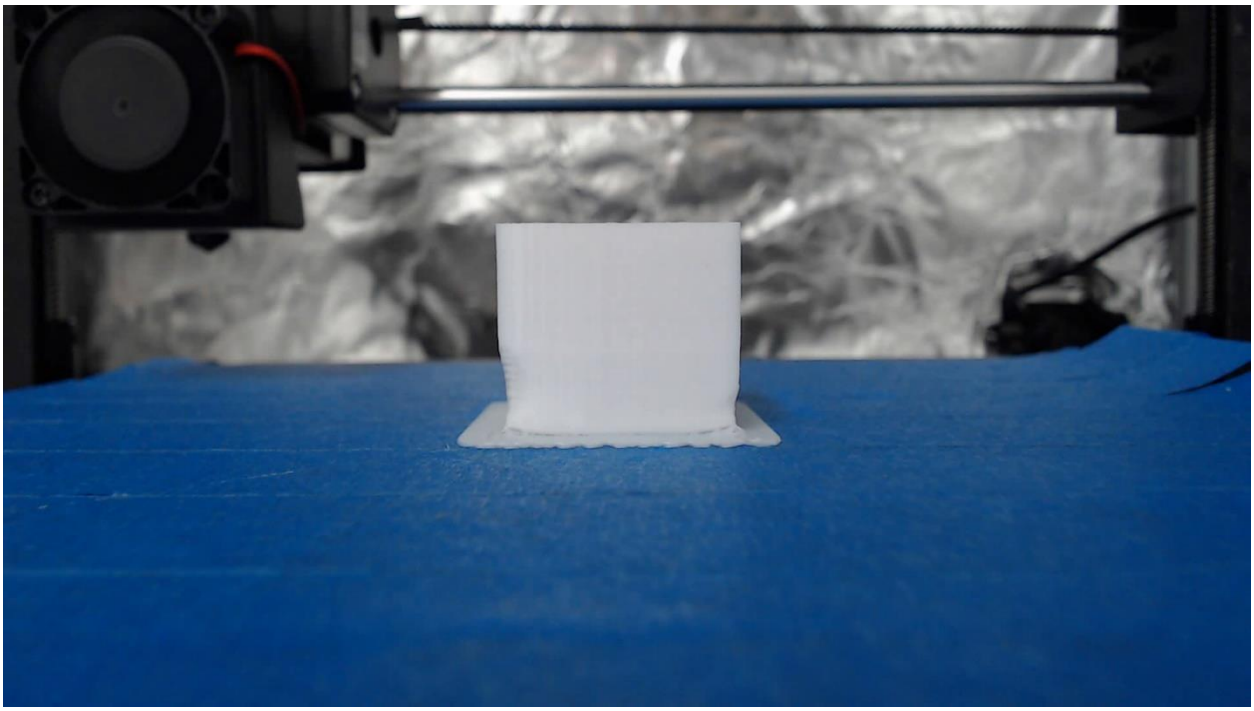


Figure 7.10: Part printed with the aluminum lined wall of the Monoprice enclosure as a background.

Similar to the background, the color of the print bed also had an effect on the usefulness of the images. The final design uses painter's tape that is a similar color to cardboard. Along with being the optimal color, the tape is heat resistant, which is ideal for a heated print bed. It also aids in print bed adhesion, so the part will not slip because of our system. Figure 7.11 shows what the camera sees with the tan bed tape and cardboard background. Since much of the background is the same color, the algorithm had a much easier time differentiating between the part and the background.

Figure 7.11: A part printed with tan tape and cardboard background

7.1.3 Camera and Fixturing

7.1.3.1 Camera Selection

Figure 7.12: Logitech C920 Pro webcam that was used for testing the DaR3D system

The DaR3D system utilizes a Logitech C920 Pro webcam as seen in Figure 7.12. For this system, there is no ideal camera that needs to be used to fulfill certain image quality or field of view requirements. For this project we needed a camera to have a higher resolution, there was no specific target resolution. After comparing models, the team went with a camera that had a higher resolution but was also widely available and that did not cost too much. Another attribute

of the camera we looked at was the field of view of the camera (FOV). FOV is the size of image that the camera can capture in both horizontal directions (HFOV) and vertical direction (VFOV). Appendix G has a table of the different specs compared for multiple webcams. The Logitech webcam has a HOV of about 70 degrees and a VFOV of 43.3 degrees. These fields of views are great for this system because it does not see things wider than the print bed which helps with part focusing. This camera was chosen for two primary reasons; One, the field of view and image quality was upper tier for cameras researched. Two, this webcam can be taken apart to allow for fixtures to be interchanged using small screws. Both attributes allow the system to maintain clear images that can span most of the width of the print bed, as well as create the ability for fixtures to be designed and securely fastened directly to the camera.

The Logitech webcam comes with a mount that can be clipped to a surface (primarily to the top of a monitor). There are a few issues with this stock mount, one is the variability in the orientation of the camera. The stock mount allows for the camera to change angles w.r.t the z axis of the printer. There is no way to lock this tilt angle down which makes the camera very susceptible to shifting during a print. This is a problem for DaR3D because the software relies on consistency between consecutive images of the time lapse. Therefore, if the camera angle changes part way through the print, then the system could detect a defect even when there is not one. Fortunately, this stock mount can be taken off and replaced with a newly designed mount custom made for a specific printer. This section will go into greater detail of these printer specific mounts that attach directly to the camera.

As shown in Figure 7.13, the stock mount that comes with the Logitech webcam is connected using four small screws that connect to the frame of the webcam and metal tabs that extrude from the mount.

Figure 7.13: Front view of the Logitech webcam with cover off

7.1.3.2 Camera Fixture

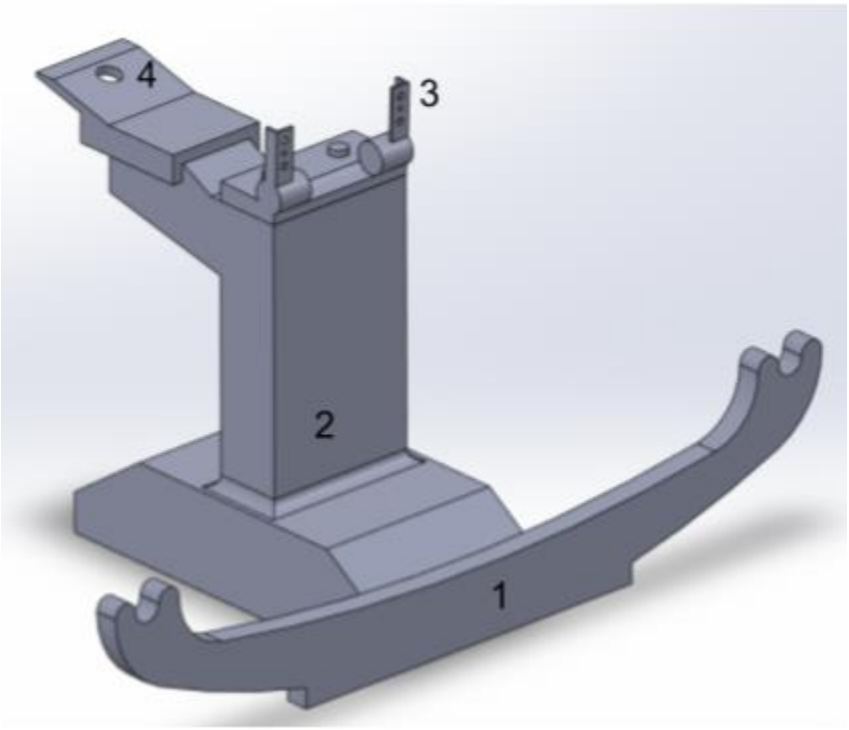


Figure 7.14: The CAD model for the Monoprice camera and ring light fixture assembly. There are four total components. 1 is the foot component of the mount, 3 is the camera bracket, 4 is the light adapter and 2 is the base of the fixtures that hold components 1, 3 and 4

With the Logitech C920 Webcam being our final camera of choice, we needed to create a way to properly secure the camera. One of the main issues we had during our earlier tests throughout this project was locating the camera to a consistent point each time. This meant that there were a lot of eyeball measurements to place the camera in the middle of the print bed along its x axis. Another issue was the camera tended to tilt at different angles throughout the duration of a print. These two problems are substantial for this project because we need a consistent camera placement that views the same background every time and will not move while the print is running. Lastly, the system utilizes a 10" ring light to bring light to the foreground of the print. This light needs to be placed behind the camera during the print, through early testing there was not a consistent way to place either the ring light or camera. Using these problems along with information we had found through our trials in the earlier stages of this project, we created a list of requirements for a fixture for the Logitech webcam and the 10" ring light:

- Compatible with Printers
    - Monoprice Select Plus
    - RepRap
- Must be able to register to a point on the front of each printer for placement consistency
- The fixture must be simple to install to the printer requiring no additional tools
- Must fit within the enclosure for each printer

The concept that was selected for this fixture was called a "base and adapter fixture" concept. This means that there is a single base for the fixture and several adapters that are inputted in the

base to make a custom camera and light fixture for a specified 3D printer. Our final design utilizes base and three separate adapters (components) that attach to the base. The benefit to this system is the ability to make some of the components the same no matter what printer is being used. Having the fixture broken up into multiple components reduces overall print time. If something is to break or needs redesigning, only a portion of the overall fixture needs to change and be recreated. If the fixture was one solid piece of material then the entire thing would need to be designed and manufactured again. The three adapters and their functions are listed in table 7.1:

Table 7.1: A list of the different adapters that connect to the base of the fixture and their function.

Adapter (Component)	*Function*
Foot Component	Attached to the front of the base. Its primary function is to locate the fixture consistently on the front of the printer.
Camera Bracket	This small adapter attaches directly to the Logitech webcam. It is then placed on top of the base to allow for consistent height and distance from the print bed.
Light Adapter	This adapter's function is to hold the ring light using a ¼-20 bolt. It is then attached to the base behind the camera so that light can be brought to the foreground of the print.

The first printer that this fixture was designed for was the Monoprice Maker select plus. With a functioning fixture for the Monoprice printer, aspects of the design were altered so that it properly fit the RepRap printer. In an attempt to decrease the amount of redesign of the fixture for the second printer, the camera bracket and light adapter design were kept the same for each printer. This left only minor redesigning for the base and foot component of the second printer.

7.1.3.3 Foot Component

The purpose of the foot component for the fixture is to locate reference points on the printer without the use of any additional tools or measuring. These reference points were determined before designing and their measurements were identified as critical measurements for the foot components to be designed around. This is so there is a good, tight fit between the foot component and the reference points so that there is consistent placement with little play. The two printers have different designs with regard to their frame which made it necessary to identify separate locating points for each printer because one singular foot component would not be able to reference both printers effectively. Figure 7.15 shows the two reference bolts on the front frame of the Monoprice printer, these two bolts are the locating points for the foot component to register to. These were chosen as the reference points because they are the only features of the front that extrude from the frame, which can be utilized as two robust, consistent reference points for something to register on.

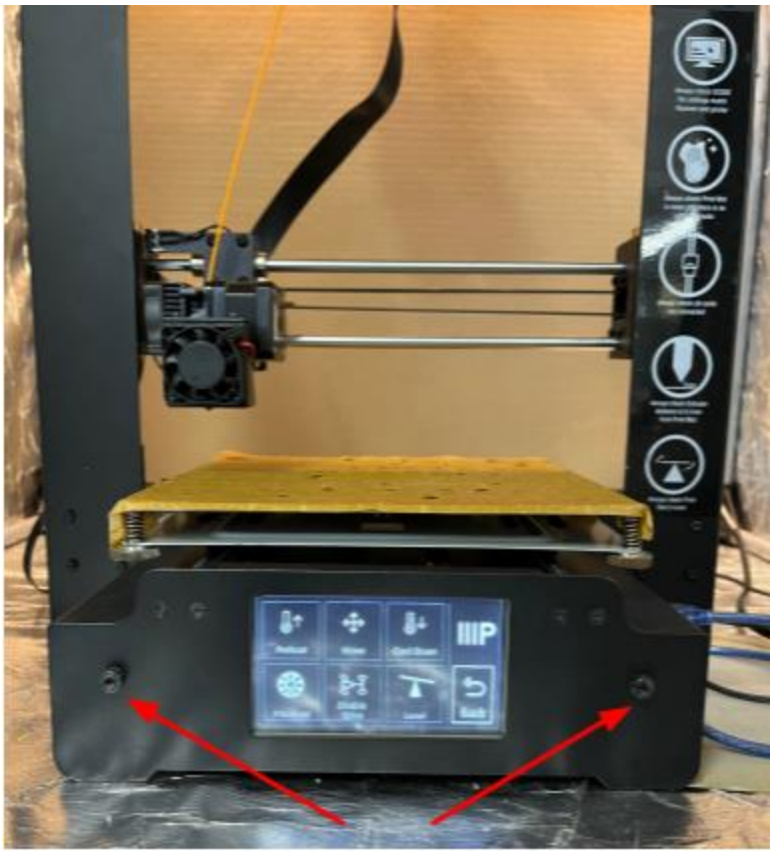


Figure 7.15: The two locating bolts on the front of the Monoprice printer

Similar to the Monoprice printer, the RepRap printer has two locating points on the front of its frame. Figure 7.16 shows the front of the printer. The inside face of these two nuts shown by the teal arrows will be the reference surfaces for the foot component to register to. In the case of the RepRap, there are two cross bars of threaded rod on the front of the frame. Either the top or bottom rod and nuts could be used as reference surfaces, we selected the bottom rod simply because it was the lowest surface from the table, this would result in a part that uses less material because it would not have to reach as high which would require a larger part. In the case for each printer, registering is referring to something touching the two points on the printer to allow for consistent placement.

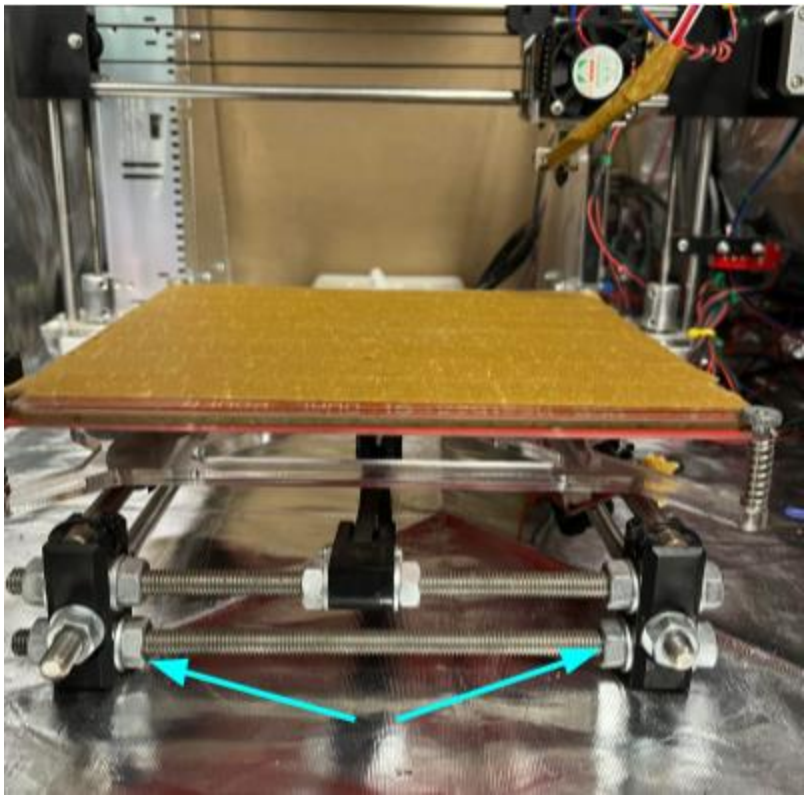


Figure 7.16: Front of the RepRap printer and registering positions

Figure 7.17 shows the front view of the CAD model of the Monoprice foot component with dimensions. The dimensions that are highlighted are the critical dimensions to allow for a proper fit between the foot component and the two bolts on the printer. The Figure shows dimensions circled by a red, blue and green circle. The dimension circled in red represents the diameter of the bolt which is 0.29 in. The dimension in the blue circle is the distance between the center of each bolt, this value was measured to be 7.90 in. Lastly, the dimension circled in green is the height that the bolt sits from the table, this value was measured to be 1.63 in. The thickness of the foot component was held to 0.125 in, this value was chosen arbitrarily as there was not specified thickness that needed to be considered. This foot component attaches to the base via a

square slot with pins to hold it in place. The fit between the base and the foot component is shown in Figure 7.18.

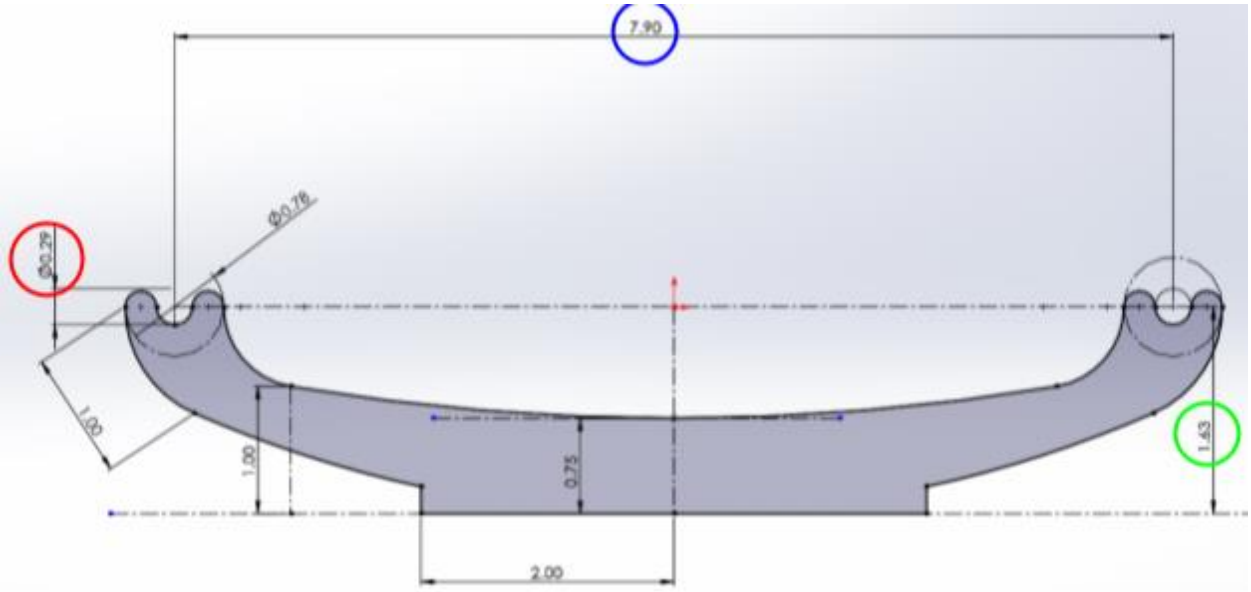


Figure 7.17: CAD Model of the Monoprice foot component

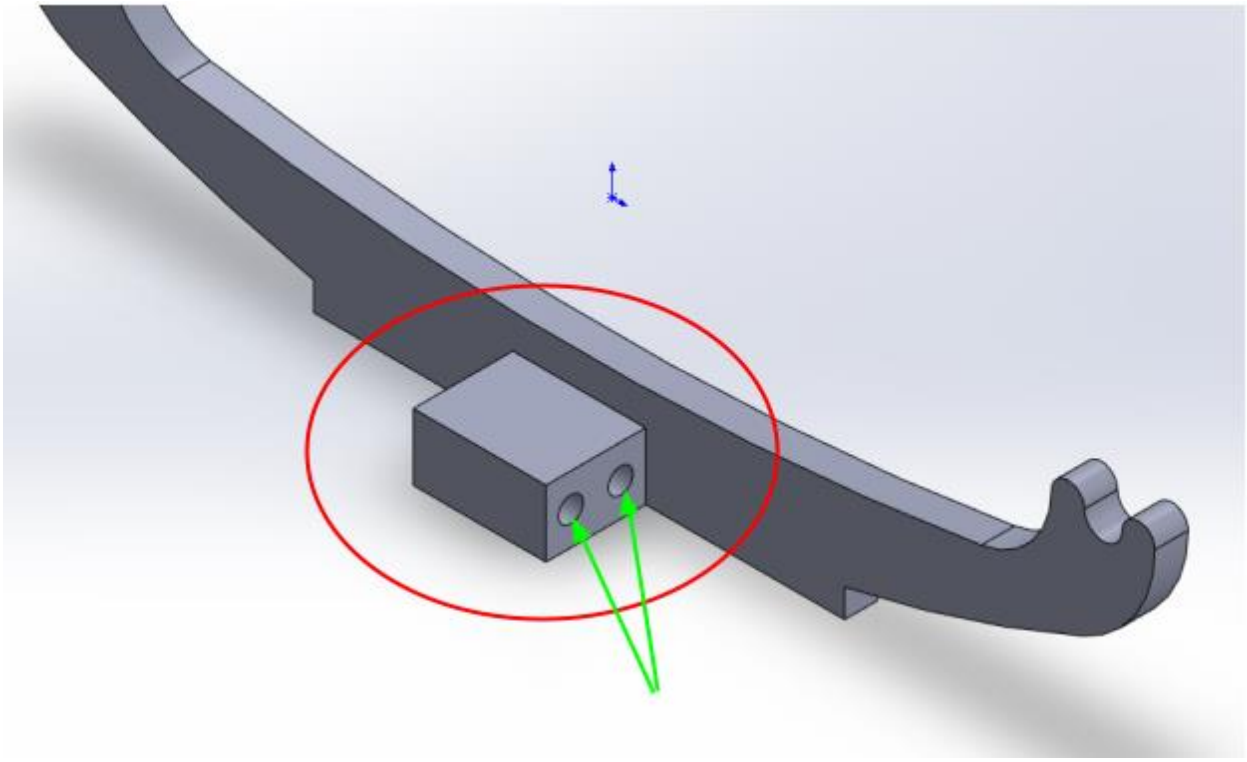


Figure 7.18: Male end of the Monoprice foot component

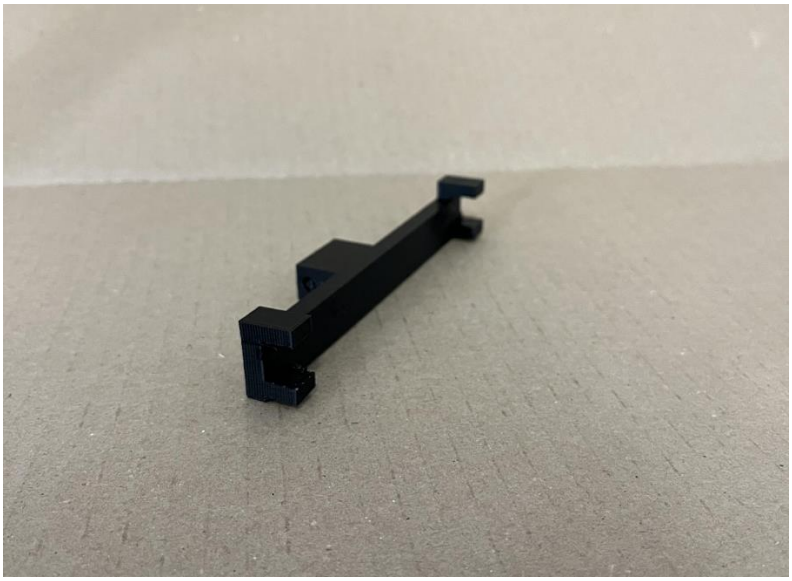


Figure 7.19: 3D Printed model of the RepRap foot component

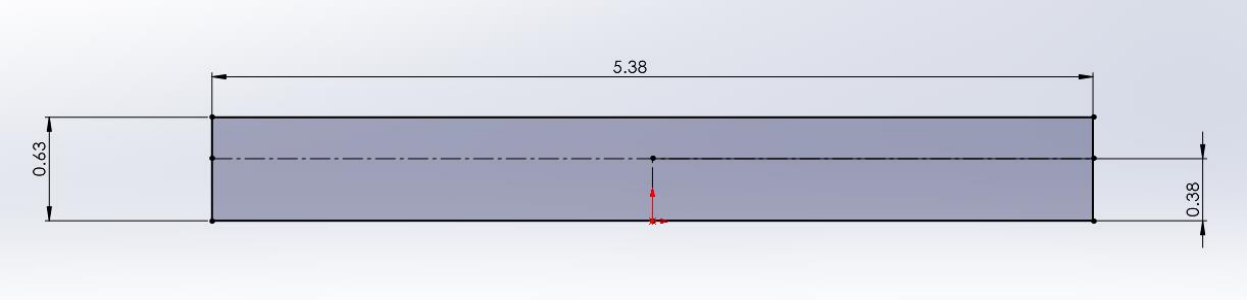


Figure 7.20: Front view of the RepRap foot component CAD model

Figure 7.20 shows one side of the foot component for the RepRap printer. In the image the dimensions shown are highlighting how the tabs of the component need to be oriented to locate on the threaded rod of the frame of the printer. The overall thickness of the rod was measured to be 0.43 inches in diameter. To allow for expansion when printing, the space in between the tabs were set to 0.45 in. The thickness of each tab was set to 0.19 in, this number was arbitrary but needed to be able to fit in between the rods on the frame of the printer and the space between the table and bottom of the threaded rod being located on. Lastly, the bottom of the targeted threaded rod was measured to be roughly 0.25 in from the table, this is where the top of the lower tab begins.

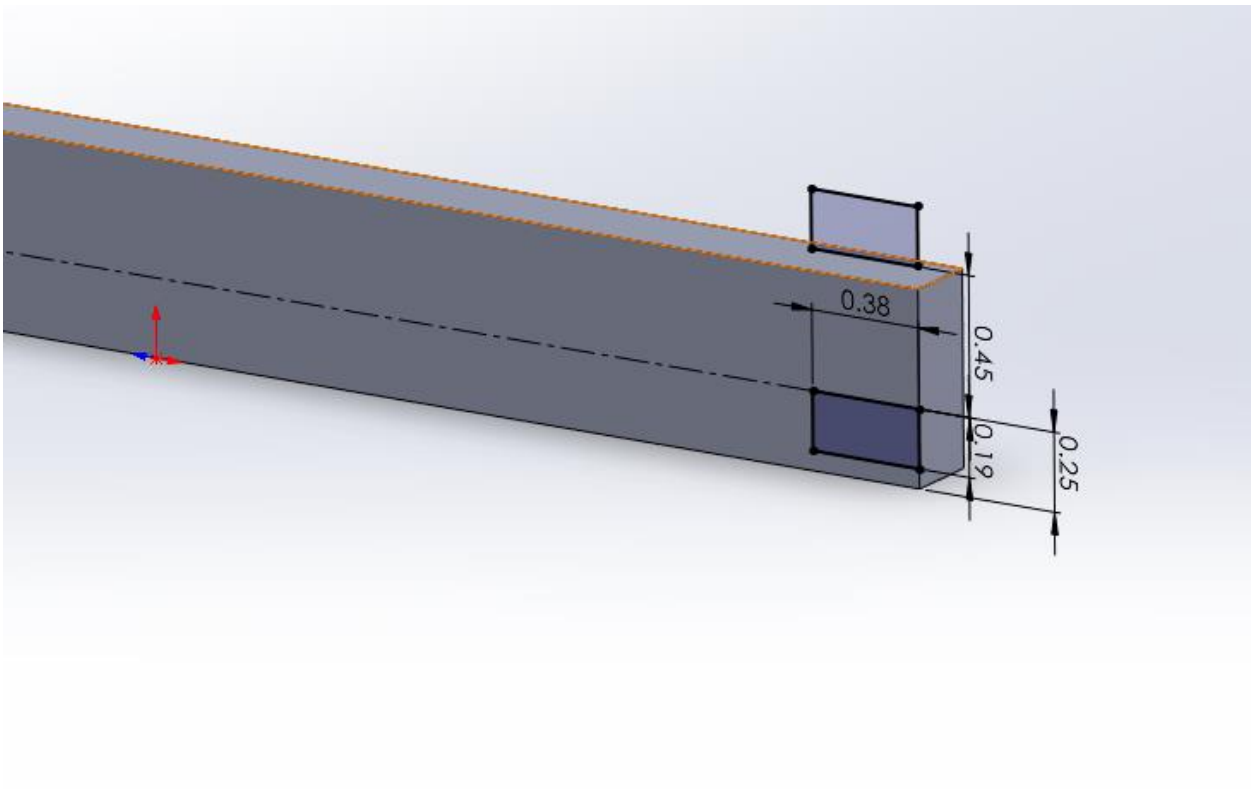


Figure 7.21: CAD model of RepRap foot component showing locating tab dimensions

7.1.3.4 Camera Bracket

The camera bracket shown in Figure 7.23 was heavily inspired from a camera bracket we found on a website called thingiverse.com. Thingiverse is an open forum site where creators can share STL files of designs and anyone can download them and 3D print them for themselves. The model that we found is shown in Figure 7.22, while the exact model would not serve our needs, the tabs that are used to connect the camera to the bracket were of our interest.

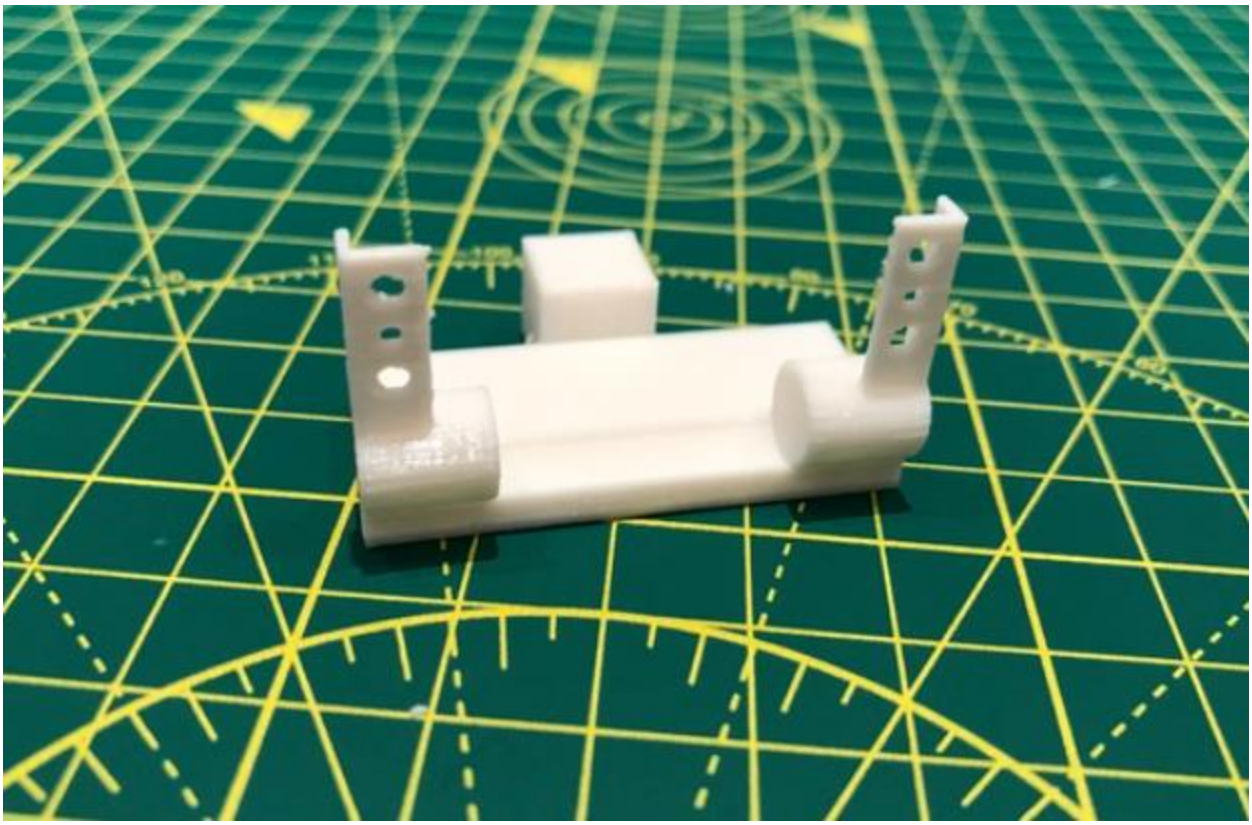


Figure 7.22: Original Logitech C920 mount

The first step in adapting and applying this model was reverse engineering it. To do this we printed the model of the camera bracket from Thingiverse and measured its dimensions and created our own CAD model. After reverse engineering this model, we created adaptations to better suit the camera bracket for our system. Figure 7.22 shows the redesigned model that was created. The requirements of the new camera bracket model were to have a method of attaching to the base of the fixture. Another requirement was to have the camera view parallel to the Z axis of the printer. Some key changes that were made to uphold these requirements was removing the box on the back of the original camera bracket model and substituting that with two holes on the base of the bracket. These two holes are there so that the camera bracket can seat on two locating pins that extrude from the base of the fixture. The only other change made was making the base

have a slight taper of 2 degrees on its bottom. This is to allow for the camera to not tilt upward when placed. This tilt angle is not overly critical as the purpose of the fixture is to create a consistent image, this taper was arbitrarily added to get the camera to seat closer to 90 degrees with the print bed w.r.t the z axis of the printer.

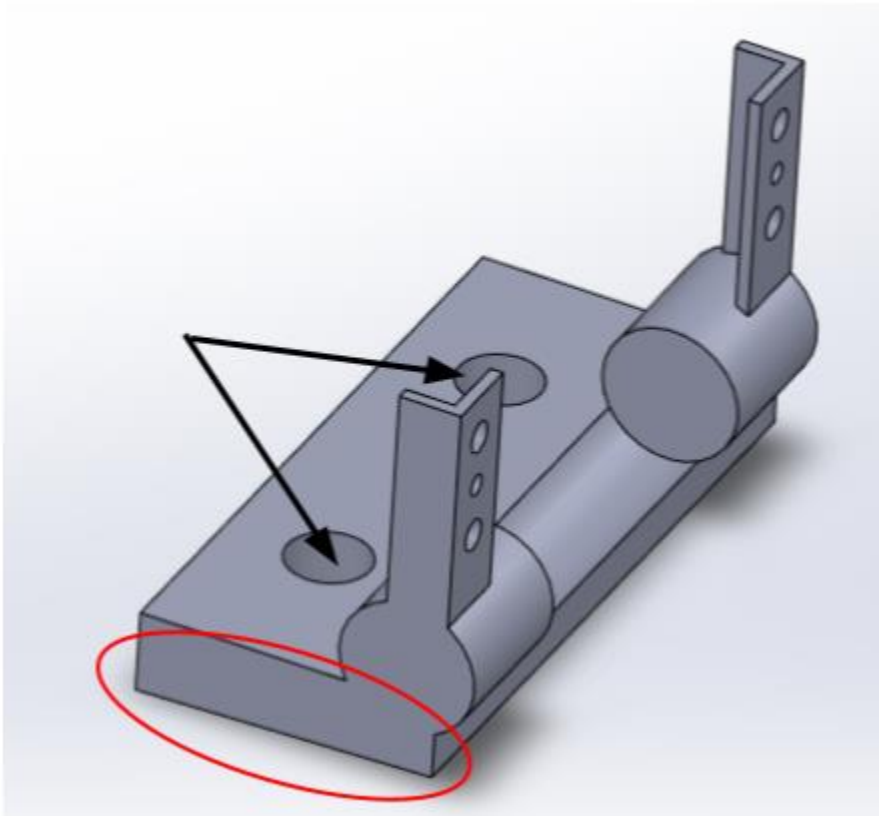


Figure 7.23: CAD model of redesigned camera mount

7.1.3.5 Light Adapter

The third adapter that is used in this fixture system is the light adapter. From previous testing that was discussed in Chapter 5, we decided that the camera needed light shining on the foreground of the component being 3D printed, meaning that a light was needed behind the

camera. We began using a 10” diameter ring light and needed a way to mount it to the same camera fixture. The ring light has an outer diameter of 10” and has a mounting bracket that consists of a ¼”-20 threaded insert. This insert is to allow for tripods to be connected to the bottom of the ring light. The purpose of this adapter was to allow the ring light to be connected to the base of the fixture so that the camera and ring light could be set in one position simultaneously and consistently.

The light adapter can be seen in Figure 7.24. It utilizes a ¼-20 threaded insert in the bottom of the ring light. A ¼-20 bolt is pushed through the hole on the ramped surface of the light adapter and then threaded into the light. This allows the light to be fastened to the adapter securely. This surface is ramped at an angle of 15° to allow for more direct light on the part. This allows the light to be fastened to the adapter. On the bottom of the adapter is a rod that is 0.5 in long that seats into a hole on the base. In order to keep the adapter from swiveling, two tabs were put into the adapter to eliminate any swivel.

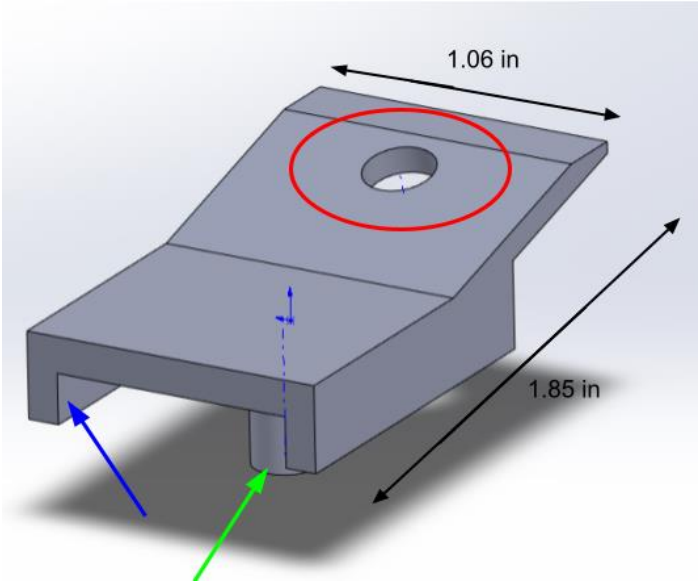


Figure 7.24: This is the CAD model of the light adapter. The red circle is highlighting the hole on the ramped surface of the adapter where the ¼-20 bolt connects to the ring light. The green

arrow is signifying the ½ inch rod that is inserted into the base of the fixture. The blue arrow is showing one of the tabs that eliminate swiveling once installed on the base

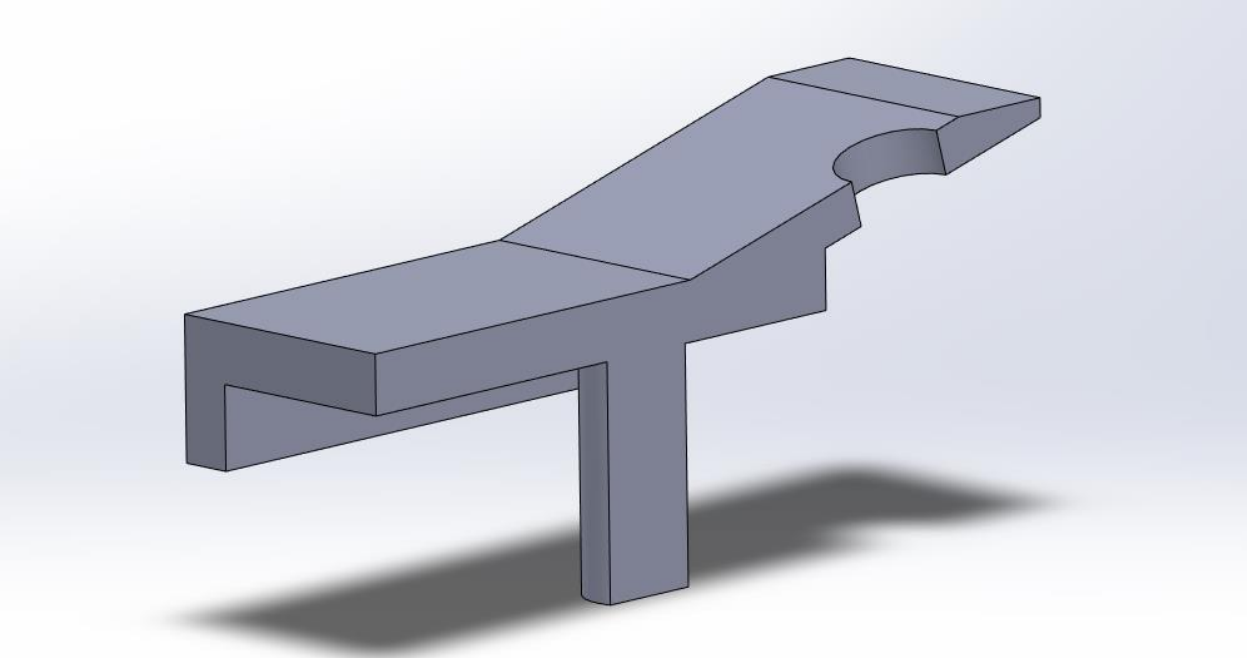


Figure 7.25: Cross sectional view of the light adapter CAD Model

Figure 7.26: Light adapter connected to the ring light

7.1.3.6 Base of the Fixture

All three of these adapters connect to the base or mount portion of the fixture. In this system, the camera bracket and light adapter are the same design for both the Monoprice and RepRap printers. The foot components are different designs in regards to what they reference to each

style of printer but utilize two reference points to ensure consistent location. The mount or base portion are very similar in design, but dimensions are changed in order to properly set the camera less than ½ in from the print bed and less than ½ in above the print bed.

Figure 7.27 shows the base of the fixture for the Monoprice printer. This image has three sections of the base labeled. Section A shows the locating pins where the camera bracket seats. Section B shows where the light adapter seats on the base. Section C is the slot where the foot component is inserted in the base. Out of the two designs, this is the larger of the two because the print bed height for the Monoprice is higher than that of the RepRap printer by 0.625 in. The difference in height is to account for the difference in bed height between the two printers. The purpose of the base is to hold all of the adapters of the fixture. The bases for both printers are identical with regards to dimensions for the adapters, the only changes are the overall height of the mount and the distance from the front of the vertical column of the mount to the front of the mount.

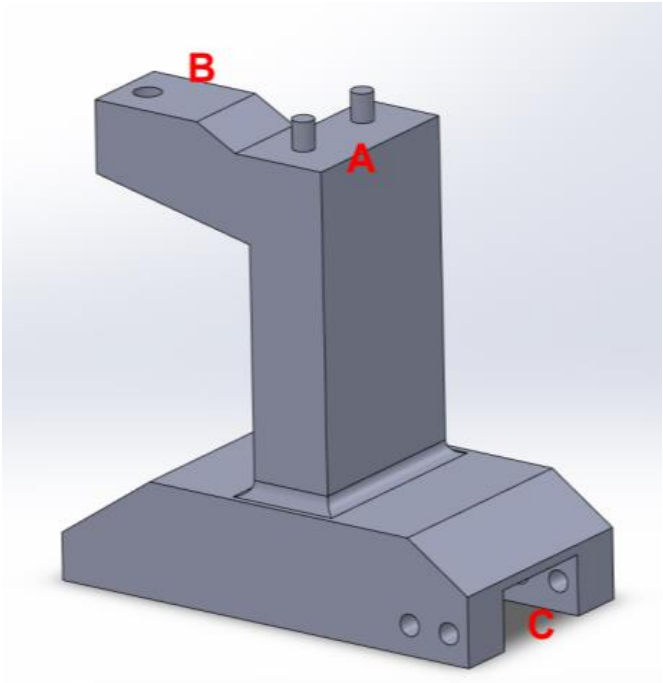


Figure 7.27: CAD model of the base of the camera fixture. “A” is where the camera bracket is inserted on the locating pins. “B” is where the light adapter seats onto the mount by using the hole on the back end. Lastly, “C” is on the front of the mount where the male end of the foot component is inserted

7.1.3.7 Full Fixture Assemblies in Use

This subsection will contain visuals of each of the physical models of the camera and light fixtures for the RepRap and Monoprice printers. The section will show each component of the fixtures and give a step by step display of how the components come together to create the overall fixture for each printer.

7.1.3.8 Monoprice Fixture

Figure 7.28: 3D printed model of the Monoprice foot component

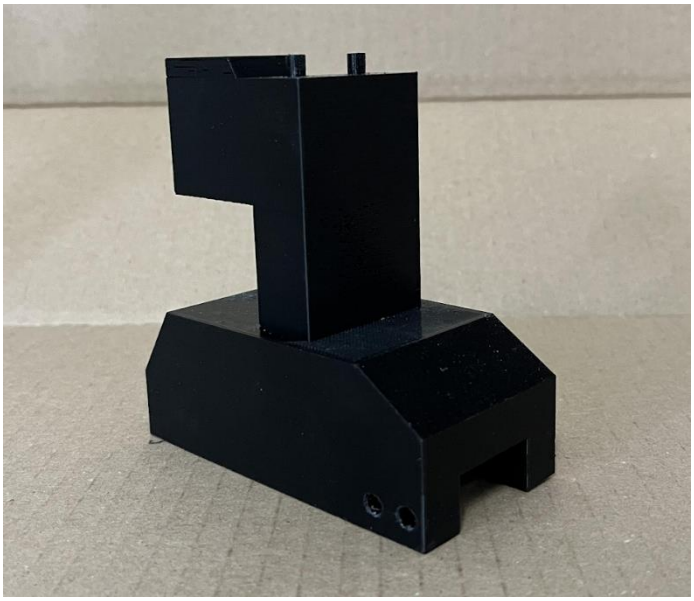


Figure 7.29: 3D printed model of the Monoprice base

Figure 7.30: Foot component connected to the base of the Monoprice Fixture

Figure 7.31: Monoprice fixture, Logitech webcam, foot component and base sub assembly

Figure 7.32: Full Monorpice fixture assembly

Figure 7.33: Monoprice fixture locating to the front of the printer

7.1.3.9 RepRap Fixture

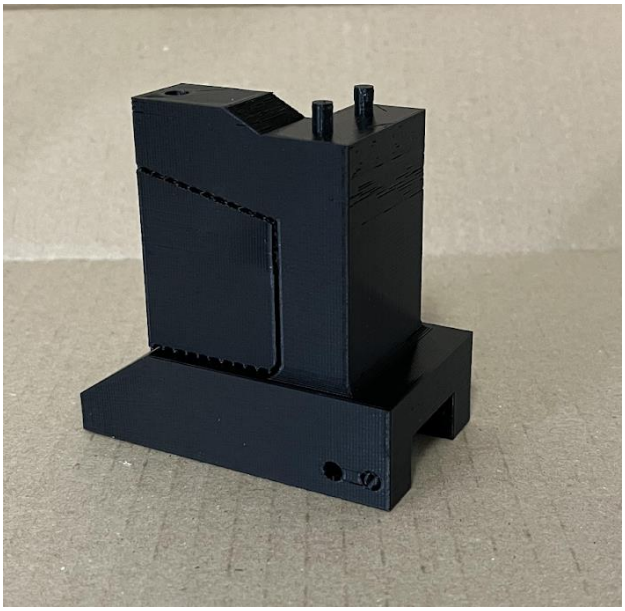


Figure 7.34: RepRap fixture base

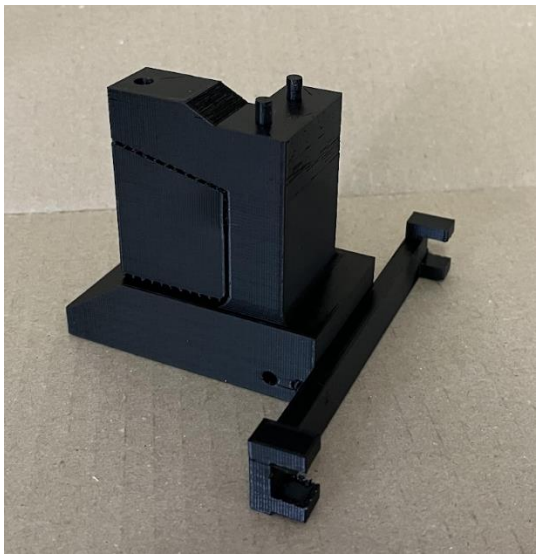


Figure 7.35: RepRap base and foot component sub assembly

Figure 7.36: Full assembly of the RepRap camera fixture

Figure 7.37: RepRap fixture locating on the printer

7.2 Software Design

This section will provide details on the final design of the software side of this project. The development process of this final system and its algorithm is described earlier in Chapter 6.

7.2.1 The DaR3D System

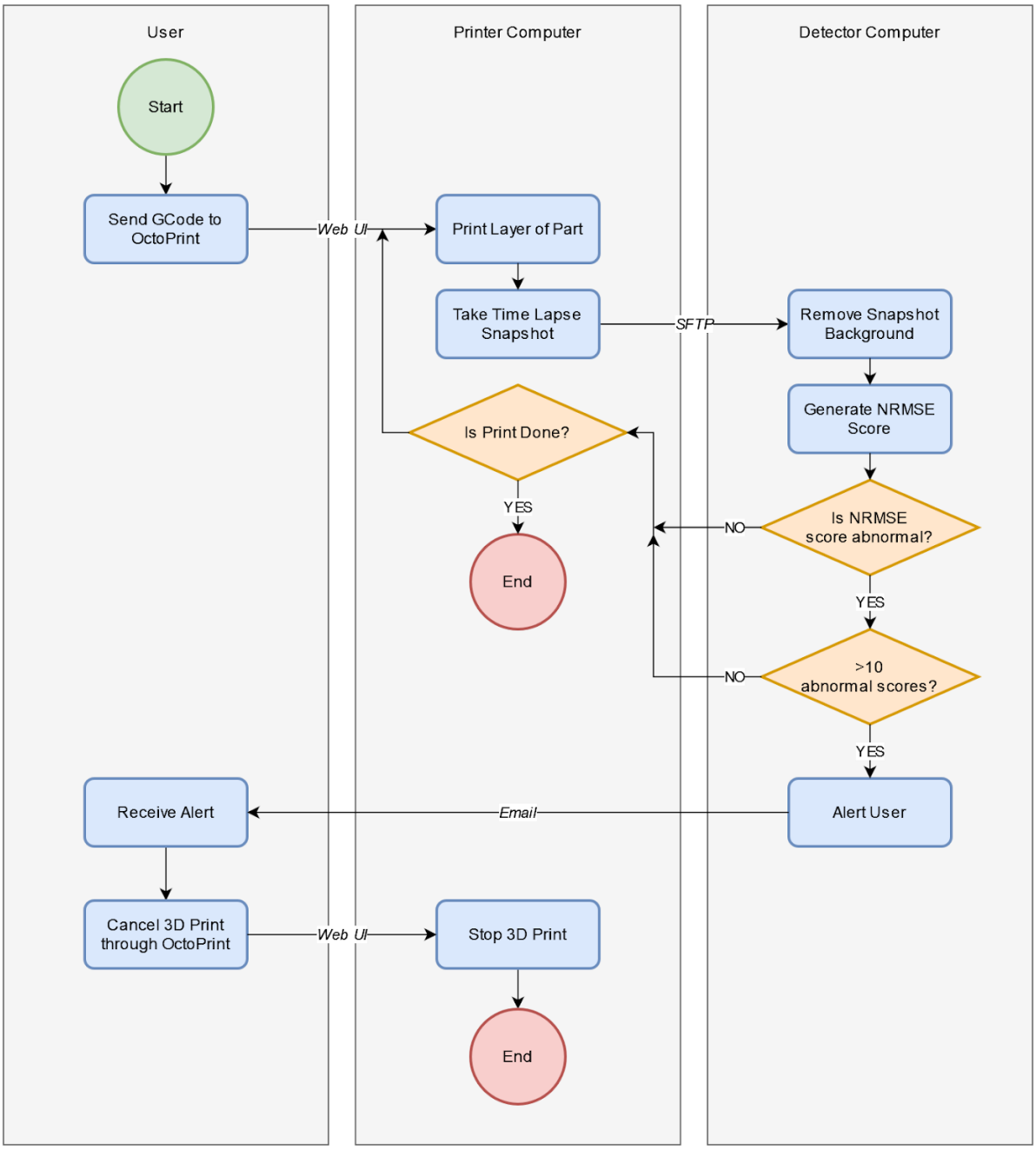


Figure 7.38: High Level Diagram showing the systems in use

Shown in Figure 7.38 is the high-level diagram of the system. The final system consists of the user and two computers: the first runs OctoPrint and OctoLapse (referred to as the Printer

Computer) and the second runs the detector program (referred to as the Detector Computer). To start, the user uploads a GCode file to OctoPrint via its website hosted on the Printer Computer and tells OctoPrint to start printing it. Printer Computer then manages printing the part. Once a layer of the part is printed, OctoLapse takes a snapshot. The Printer Computer then alerts the Detector Computer that a new snapshot is ready for processing. The snapshot is then transferred to the Detector Computer for processing. The Detector Computer then analyzes the snapshot and determines if the part is currently experiencing a defect. It first removes the background from the image then runs the NRMSE algorithm to compare the current snapshot to the previous. The system makes a note if the score generated by the algorithm is abnormal. If more than ten abnormal scores are generated, the Detector Computer sends an alert to the user. The user can then stop the print. If ten abnormal scores are not generated, the part continues to print until it is done.

7.2.2 Image Analysis Algorithm

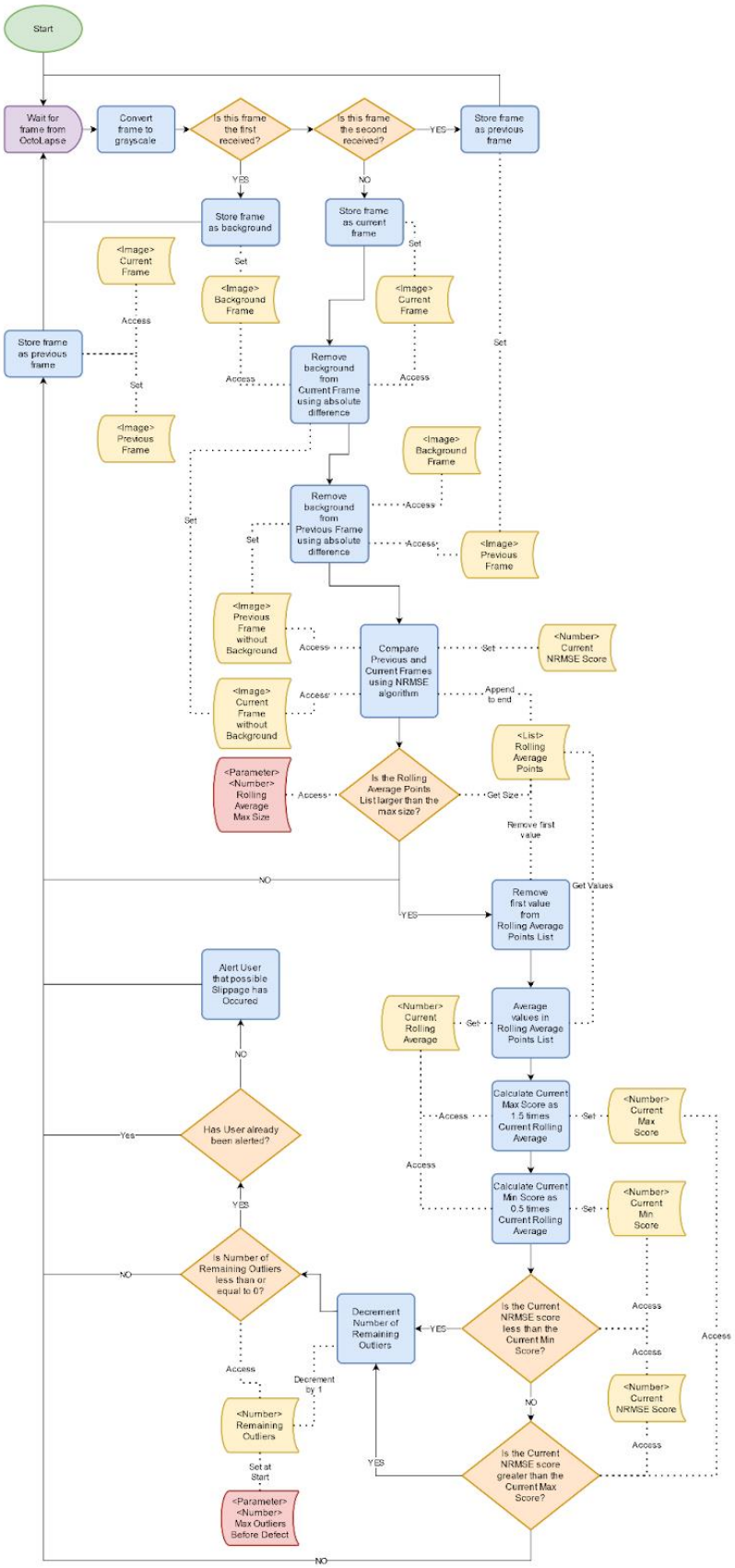

Start
Wait for frame from OctoLapse
Convert frame to grayscale
Is this frame the first received?
Is this frame the second received?
YES
Store frame as previous frame
YES
NO
Store frame as background
Store frame as current frame
Set
<Image> Current Frame
Set
<Image> Background Frame
<Image> Current Frame
Access
Store frame as previous frame
Set
Access
Access
Remove background from Current Frame using absolute difference
<Image> Previous Frame
<Image> Background Frame
Set
Remove background from Previous Frame using absolute difference
Access
Access
<Image> Previous Frame
Set
<Image> Previous Frame without Background
Access
Compare Previous and Current Frames using NRMSE algorithm
Set
<Number> Current NRMSE Score
<Image> Current Frame without Background
Access
Append to end
<List> Rolling Average Points
<Parameter> <Number> Rolling Average Max Size
Access
Is the Rolling Average Points List larger than the max size?
Get Size
Remove first value
NO
YES
Remove first value from Rolling Average Points List
Get Values
Alert User that possible Slippage has Occured
<Number> Current Rolling Average
Set
Average values in Rolling Average Points List
NO
Has User already been alerted?
Yes
Access
Calculate Current Max Score as 1.5 times Current Rolling Average
Set
<Number> Current Max Score
Access
Calculate Current Min Score as 0.5 times Current Rolling Average
Set
<Number> Current Min Score
YES
Is Number of Remaining Outliers less than or equal to 0?
NO
Decrement Number of Remaining Outliers
YES
Is the Current NRMSE score less than the Current Min Score?
Access
Access
Access
Access
Decrement by 1
<Number> Remaining Outliers
NO
<Number> Current NRMSE Score
Set at Start
YES
Is the Current NRMSE score greater than the Current Max Score?
Access
<Parameter> <Number> Max Outliers Before Defect
NO

Figure 7.39: The Image Analysis Algorithm for DAR3D

The main defect detection logic shown in Figure 7.39 for DAR3D takes place on the Detector Computer. It starts when a snapshot (or frame) from OctoLapse is transferred to the Detector Computer. Then, the process is split into two main parts. The first half (Image Analysis) takes the snapshot from OctoLapse and uses previously received snapshots to convert the current one into a score. The second half (Statistical Analysis) takes the generated score and determines if it is abnormal, then determines if there are too many abnormal scores.

7.2.2.1 Image Analysis

Once a frame is received, the system must do image analysis on it to convert it to a score. This is a multiple step process. First, the system converts the image to grayscale. If this frame is the first or second frame received, the system just records them for later use as the background and previous frame respectively. For the third frame and every subsequent one, the system converts it to grayscale then removes the background using the stored background frame using an absolute difference algorithm. The system then removes the background for the previous frame using the same method. Finally, the two images (the current and the previous) are compared using the Normalized Root Mean Squared Error (NRMSE) algorithm. This returns a numeric score which will be used in the next half.

7.2.2.2. Statistical Analysis

After the NRMSE score is generated, the system adds it to a list. If the list has more than the specified maximum number of points (currently set to 15) in it, the first point is removed. If it has fewer than the maximum points, the system cannot yet do any analysis, so it ends until another snapshot is received. After the list of scores is full and the first point is removed, the

system can then begin calculating rolling averages. Every score in the list is added up, then that total is divided by the number of points in the list, giving the current rolling average. Next, the maximum and minimum scores are determined. These are used as ‘limits’ by the algorithm. The maximum is calculated as 1.5 times the current rolling average. The minimum is calculated as half the current rolling average. These values were determined through experimentation. The system then checks if the current NRMSE score is within the maximum and minimum bounds. If it is, the algorithm is done and it then waits to receive a new snapshot. If it is outside of the bounds, the system decrements the current abnormal score counter. The counter starts at a predefined value (for our tests 10) and once it reaches 0, the system determines that slippage has occurred. The abnormal scores do not need to be sequential; other normal scores can come between them. The system (from the Detector Computer) then alerts the user via an email with an image of the latest snapshot. The user can then login to OctoPrint on the Printer Computer and stop the current print.

Chapter 8: Algorithms

This chapter provides a detailed description of the various algorithms that have been introduced and briefly described in Chapter 6: Methodology. It presents the general steps each algorithm takes to accomplish its goal. The goal of the algorithms described here is to either aid in the detection of defects or to fully detect a defect. A working algorithm should be able to either fully detect and recognize a defect or contribute to detecting a defect. The group tested several methods throughout the project in order to get a working algorithm. The final design described in Chapter 7: Design uses a combination of a few of the algorithms described here.

A more detailed description of the goals for each algorithm and how the team developed each algorithm, with pictures, can be found in Chapter 6: Methodology. This chapter will not include such descriptions but will only serve to explain in more detail the algorithms developed.

The algorithms described in this chapter all trigger once they receive an image from OctoPrint. After they receive the image, the algorithms then run a number of processing steps on the image. There are three main groups of processing steps. The first, *For Every Frame*, means that each following step is executed every time the algorithm receives an image. *First Frame* means that each following step is executed only for the first frame the algorithm receives. *Each Subsequent Frame* means that each following step is executed for every frame the algorithm receives except for the first frame.

8.1 Canny Edge Detection

The first attempt to isolate the outline of the printed object used the Canny Edge Detection algorithm (See 3.3.4 for more detail) . This was chosen because it was simple to implement and would show how feasible the group's idea of extracting the outline for a part was.

For Every Frame:

1. The frame is received from OctoLapse.

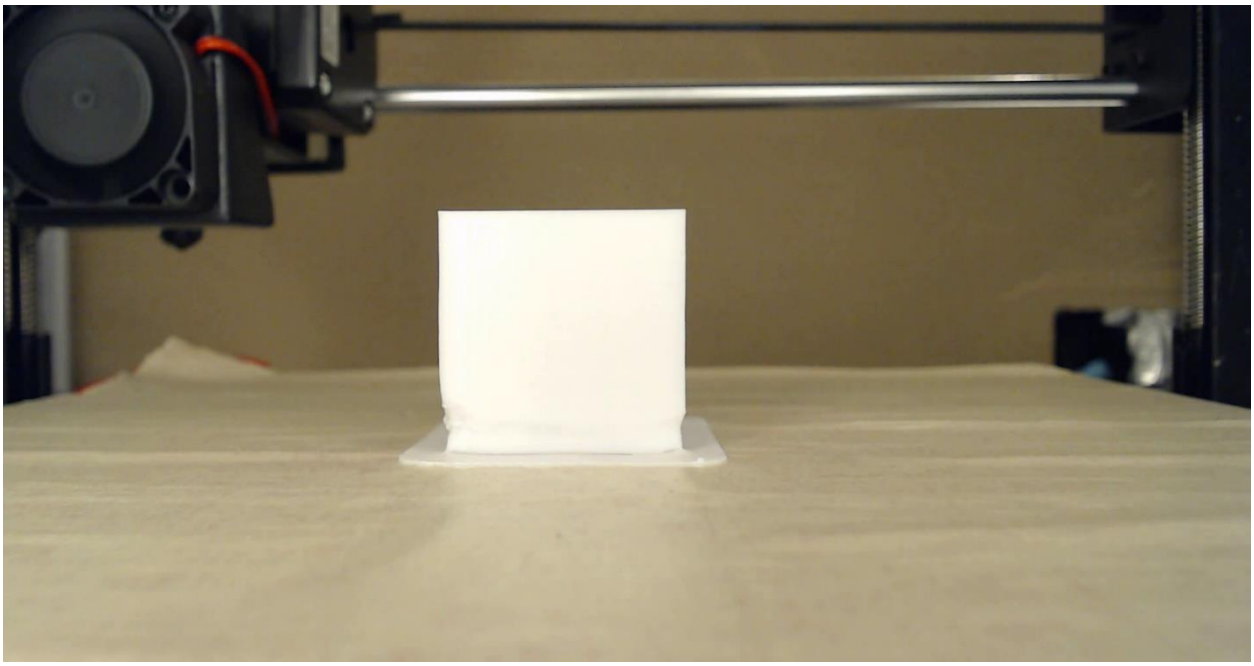


Figure 8.1: Algorithm 1 Step 1 the original frame from OctoLapse

1. Convert the image to grayscale so that each pixel is represented as a number 0 to 255 where 255 is white and 0 is black using a standard OpenCV convert function.

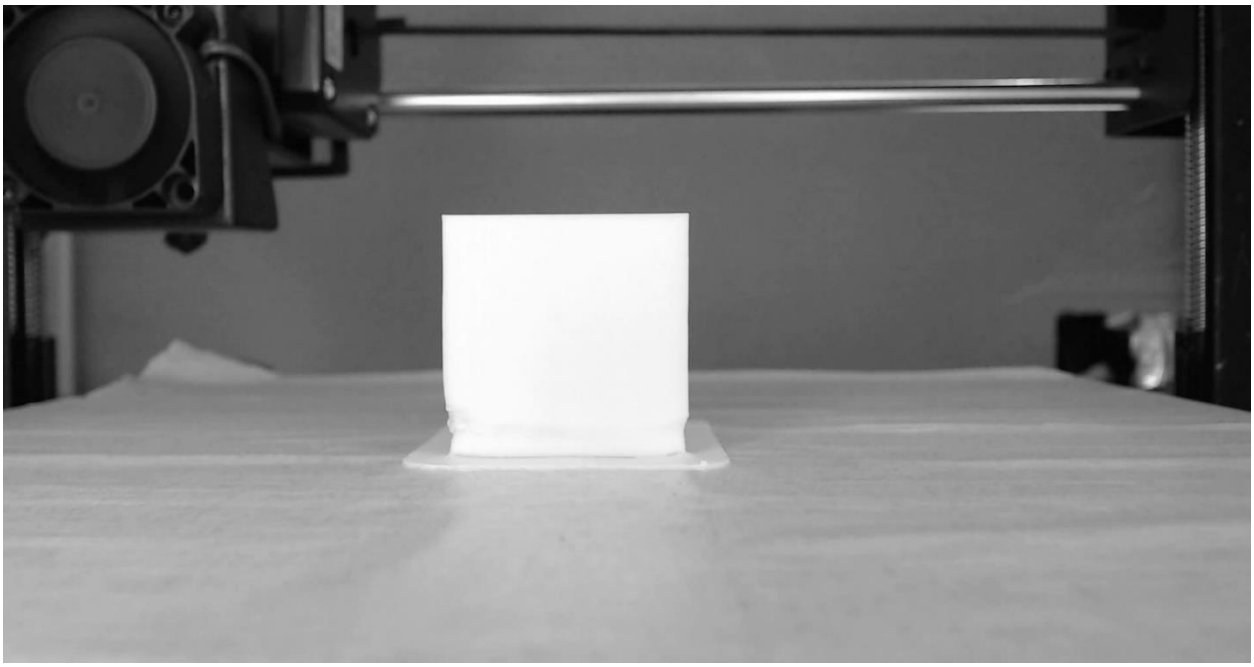


Figure 8.2: Algorithm 1 Step 2: The grayscale version of the original frame

1. Invert the image. In the image for Step 2, 255 is white and 0 is black. This is necessary for Canny Edge Detection

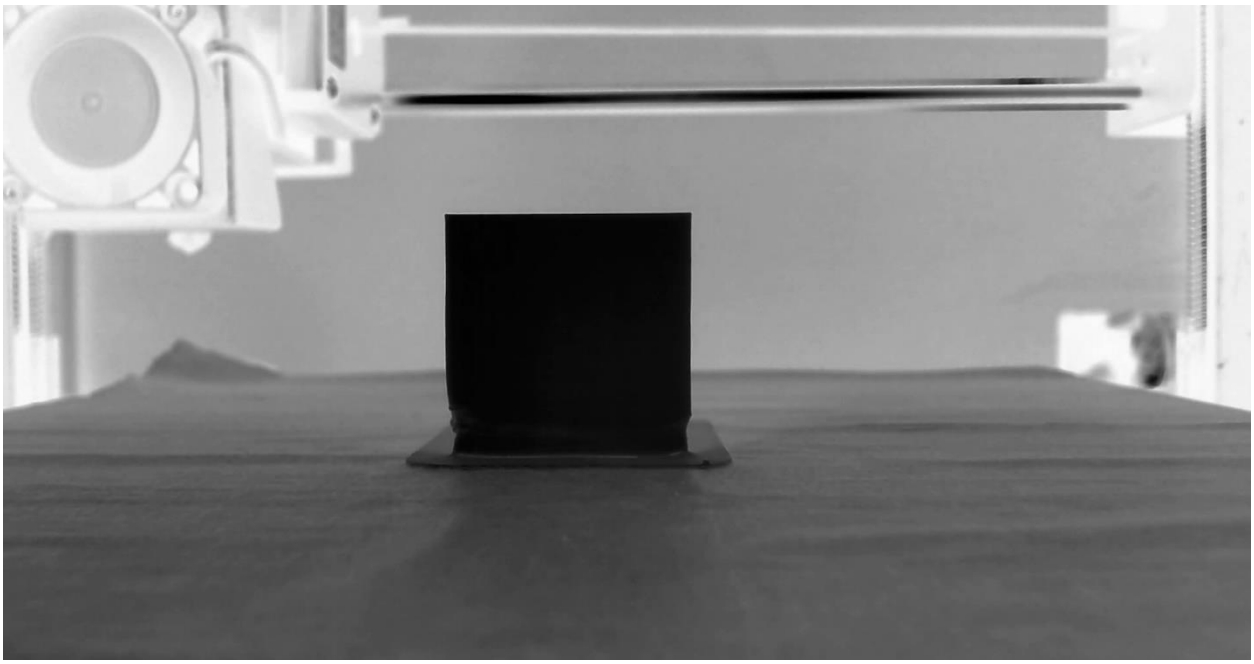


Figure 8.3: Algorithm 1 Step 3: The inverted image

1. Apply a blur to the image. This is necessary for Canny Edge Detection because it helps filter out noise from the background.

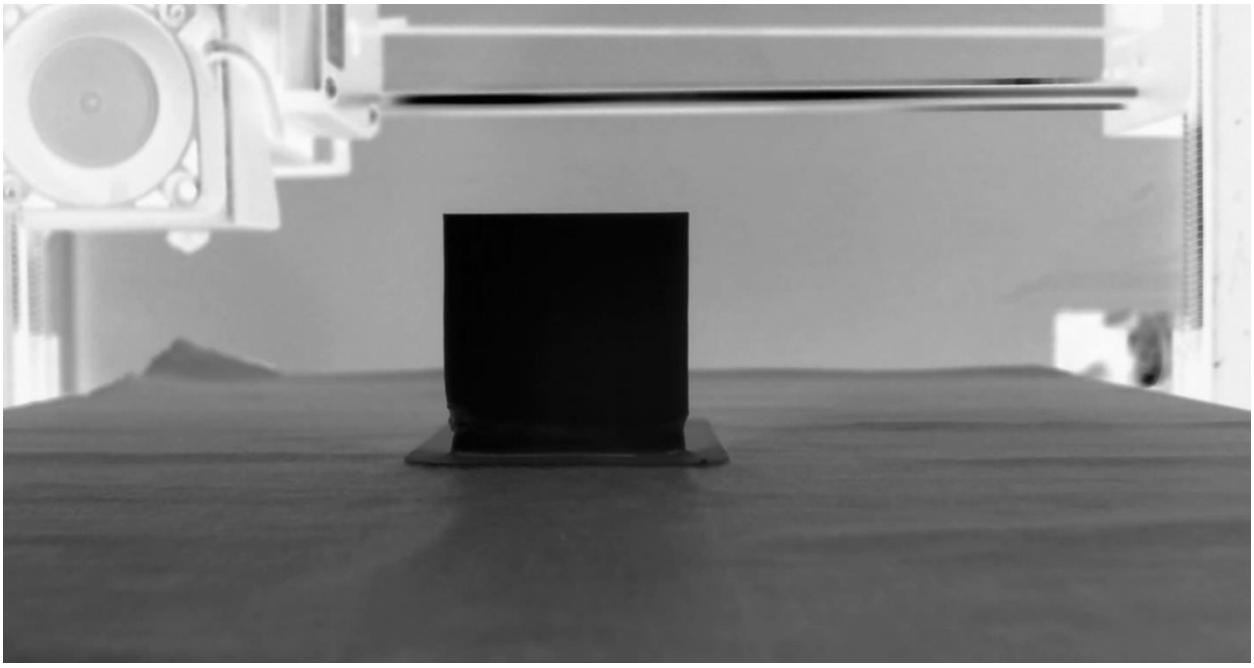


Figure 8.4: Algorithm 1 Step 4: The image with a blurred filter applied

1. Run Canny edge detection on the image.

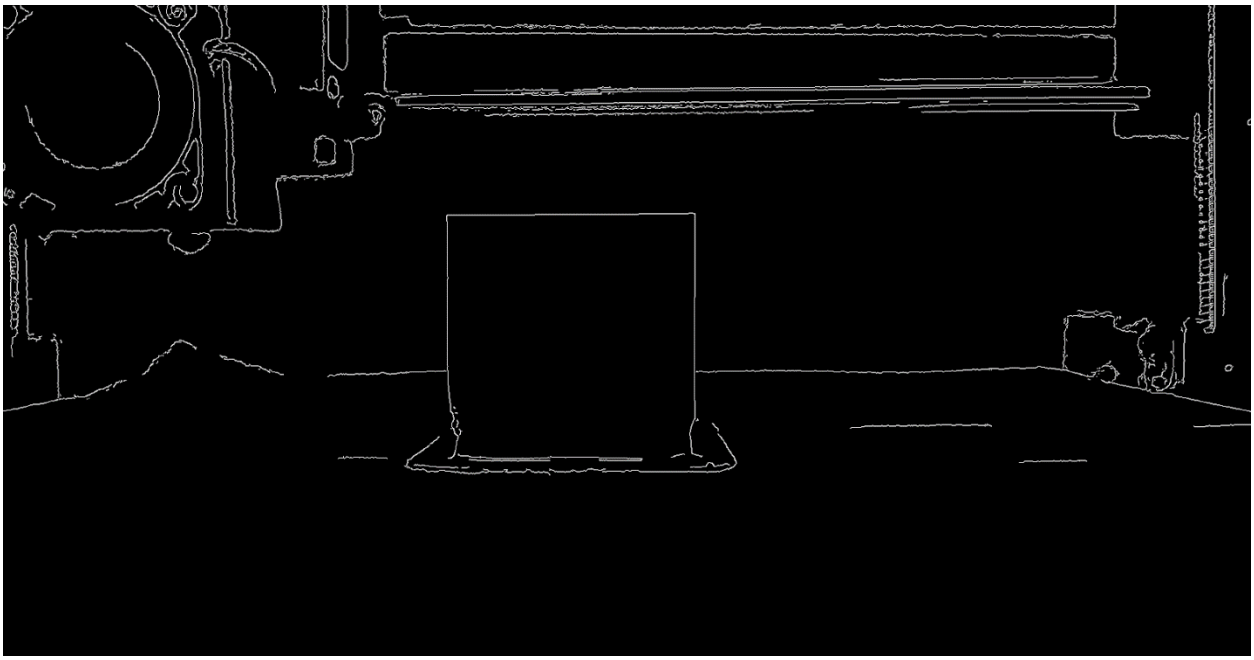


Figure 8.5: Algorithm 1 Step 5: The results of Canny Edge Detection

1. Draw the detected edges to the original frame. The red outline around the white printed part is only faintly visible in this image due to contrast.

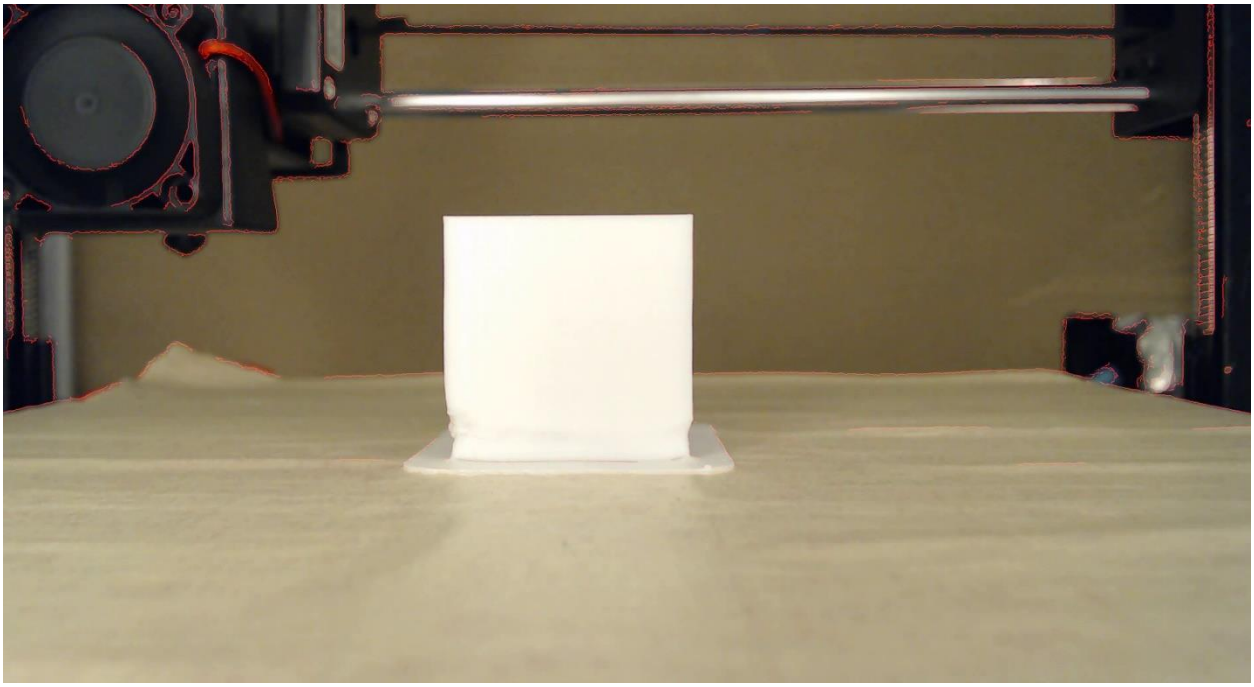


Figure 8.6: Algorithm 1 Step 6: The final image

8.1.1 Results

The Canny Edge Detection algorithm did successfully extract the outline of the printed part but it also produced many other outlines such as the extruder and edges of the print bed. These edges are not useful. The Canny Edge Detection algorithm *on its own* is not useful to isolate the part from the print bed but it is possible with some further optimization.

8.2 Background Removal

The next proposed method to extract the outline of the image was to remove the background from each frame, leaving only the printed part in the image. Because of the way OctoLapse takes

pictures, the background of each frame should be the same. The first frame can then be used as the background and can be removed from every other frame using OpenCV.

First Frame

1. The process begins with saving the first frame. It will not contain any part of the printed part. It will be used later to remove the background of the frame.

Each Subsequent Frame

1. Receive frame from Octolapse
2. Subtract the background from the current frame using an OpenCV defined algorithm.
3. Run Canny edge detection algorithm on the image
4. Draw the detected edges to the frame for visualization.

8.2.1 Results

After testing the background removal algorithm, it was determined that it would not be feasible on its own. Too many parts of the frame changed between each frame of the timelapse. The extruder constantly was moving upwards throughout the print and the background was not exactly the same each time due to the focus of the camera. The lack of focus on the background caused pixels to change shades of color, which is not noticeable to the eye but not similar for the algorithm. Too much of the original frame was extracted as the part.

8.3 Difference Detection

Another method that was attempted was to compare each image to the one immediately previous using a similarity algorithm. Using this method, the only difference between each frame should

be the part getting larger. This part of the frame can then be added to a model which should be the same shape as the part on the print bed.

Each Frame

1. Obtain Original Frame
2. Save a copy of the original frame for use later
3. Run an OpenCV defined similarity algorithm on the frame, comparing it to the previous frame. The similarity algorithm will use a predefined function to highlight differences between two images. The differences will be darker, while the similarities will be lighter
4. Invert the image so that the similarities are now darker while the differences are lighter
5. Turning all values darker than a predefined threshold to black
6. Add remaining difference to current model
7. Run Canny Edge detection on the image
8. Draw edges to original frame

8.3.1 Results

This algorithm had the same issues as Background Removal. There were too many differences between each image to be useful.

8.4 Color Extraction

Because the background and print bed of each frame are a neutral color (blue in the beginning then brown), the idea of this algorithm was to extract all areas with a certain color. The user will select the color of the filament then only that part of the image will be extracted.

For Each Frame

1. Original Frame
2. Use a color mask to select all parts of the image that is a defined color
3. Run Canny edge detection on remaining parts of image
4. Draw edges to the original frame.

8.4.1 Results

Similar to Background Removal and Difference Detection, color extraction *on its own* was not able to extract the outline of a printed model. It picked up too many other areas of the frame that were similar in color to the part.

8.5 Combined Method

The idea is to first find anything in the frame that is the color of the filament, provided by the user. Because there might be things in the frame that are the same color as the print, we need to further narrow the area that the algorithm is searching in. So, we use a different algorithm that compares the first frame, the empty bed without anything on it, to the current frame. Because of how OctoLapse works, only the printed part should be different between frames. Applying this algorithm on the area that was extracted using color extraction should result in the printed part.

First Frame

1. Blur, convert to black and white and save (referred to as *Background*). This frame will not have any piece of the printed part in it

Each Subsequent Frame

1. Using color extraction, create a mask of the space containing the color of the part
2. Mask frame with mask from Step 1, leaving everything else black
3. Convert frame to black and white
4. Mask *Background* using mask from Step 1, leaving everything else black
5. Using an OpenCV defined difference algorithm, compare 2 and 4. Parts of the frame that are more different will be whiter while the parts of the frame that is the same will be more black
6. Extract everything from 5 that is whiter than a set threshold
7. Using an OpenCV defined blob fill algorithm, fill in any inner holes from 6
8. Generate contours from 7 using a defined contour function. This should provide the edges along with info such as the size of the contour.
9. Extract largest contour by area from 8
10. Draw contour on original frame

8.5.1 Results

The combined method showed some promise when working with Chapter 9: Experiment 4 but the change in background and lighting for Chapter 9: Experiment 5 showed that it would not be viable because too many things in the frame were getting picked up.

8.6 Canny Edge Detection v2

The algorithm runs on each frame. It is built off of Method 1: Canny Edge Detection but uses some improvements. After edge detection is run, we detect the contours of that image. Because

some of the edges on the same shape are not connected, we need to combine the contours so one shape has one contour. We do this by drawing the contours to an empty image so that the only things in the image are the edges. Running the contour detector again will then attach the edges together. After that, we are left with a list of contours. The algorithm then iterates through each contour and gets its position and perimeter. It then selects the contour that is lowest in the frame, has a perimeter greater than a given value, and is near to the center of the frame. An exact value for the perimeter was never perfected. This means, for now, the algorithm will only work with parts that are in the center of the print bed. Finally, once the algorithm has only one contour it draws it to the original frame and saves it. This algorithm does have the downside of only being able to detect parts in the center of the print bed. It also has a limit to how large or small of a part could be printed.

For each frame

1. Original Frame

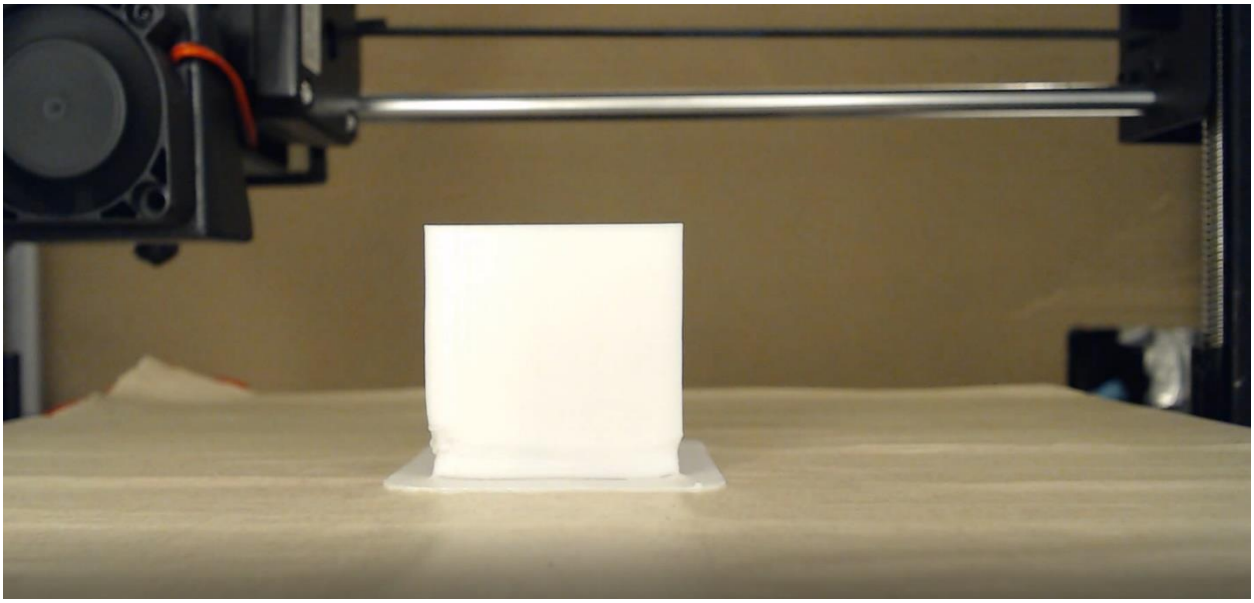


Figure 8.7: Algorithm 6 Step 1: The original frame from OctoLapse

1. Detect edges using Canny Edge Detection

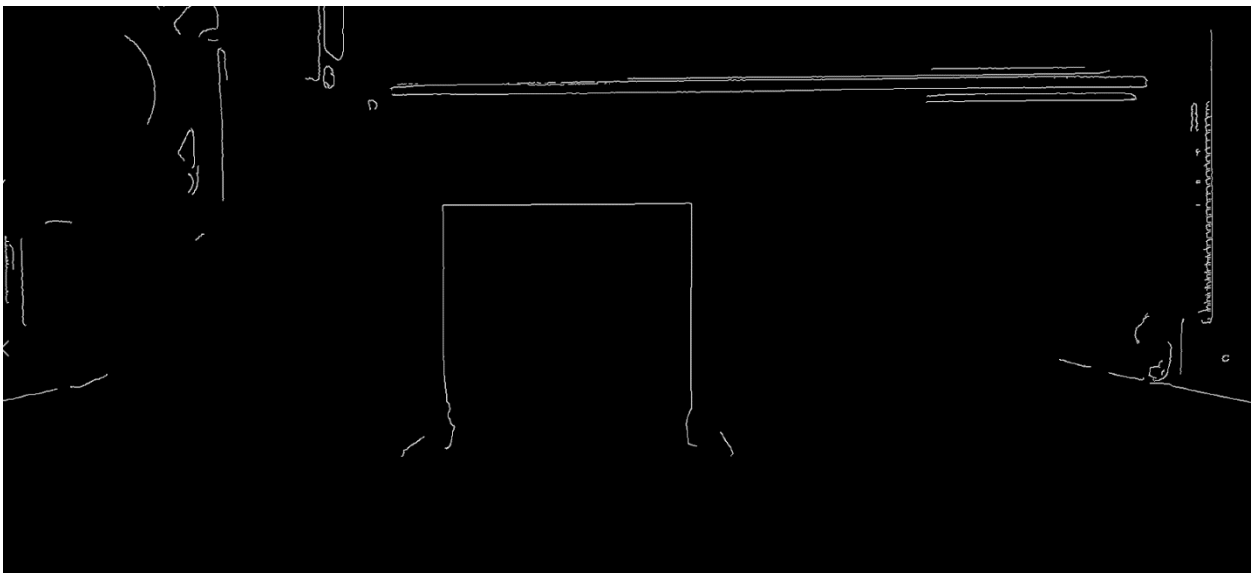


Figure 8.8: Algorithm 6 Step 2: The edges detected from Canny Edge Detection

1. Get contours from edges. Although similar in appearance, contours are closed shapes with an area while edges are just lines.

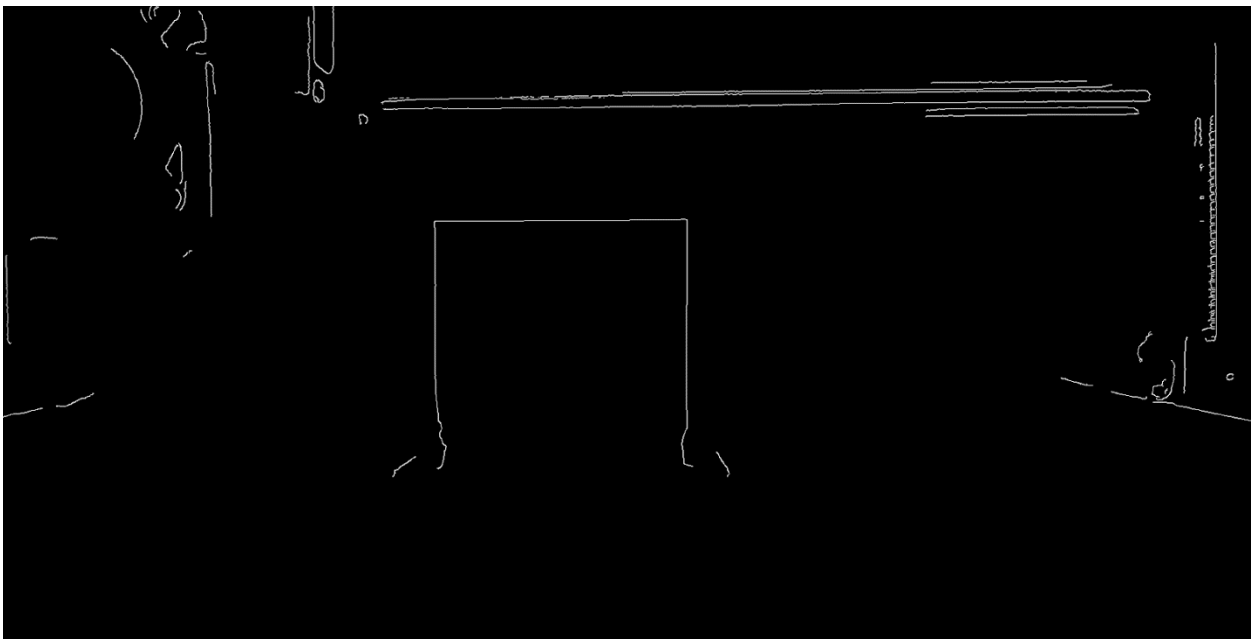


Figure 8.9: Algorithm 6 Step 3: The contours detected

1. Combine overlapping contours by increasing the line width then redrawing to an empty frame. This will make every contour slightly larger so that contours that are touching but not connected are not connected.

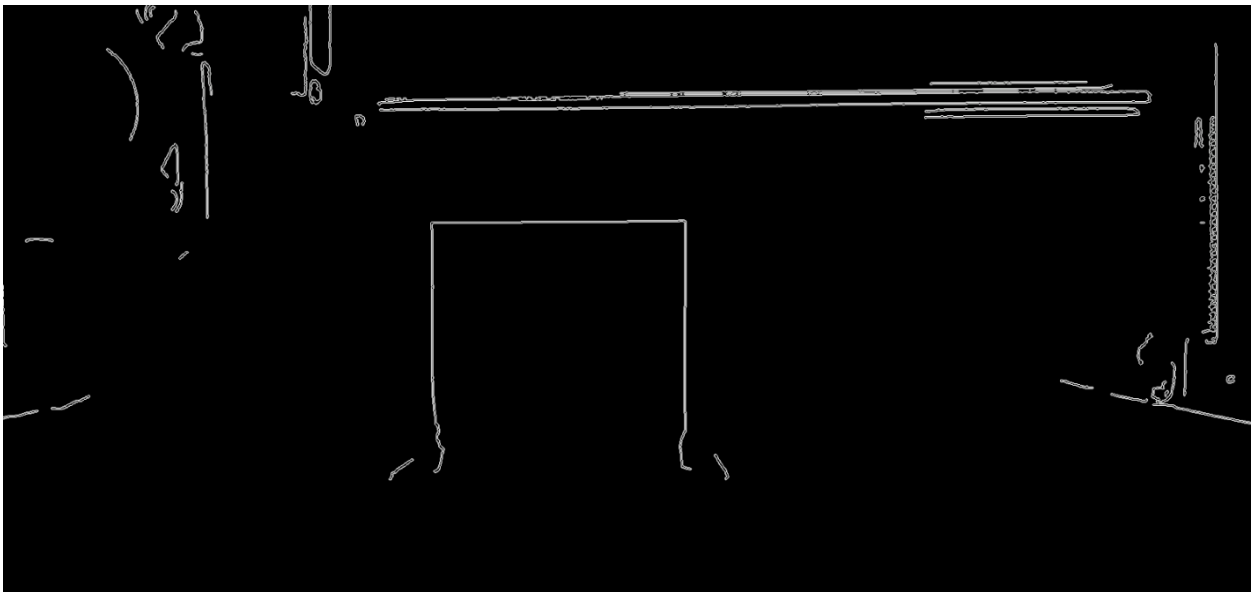


Figure 8.10: Algorithm 6 Step 4: Enlarged contours from Step 4

1. Filter through each contour to select the object. The algorithm first finds the lowest contour in the image that is above a set size then it checks if that contour is within 50% of the center on the X axis of the image. If it is then that contour is considered the contour for the printed part.

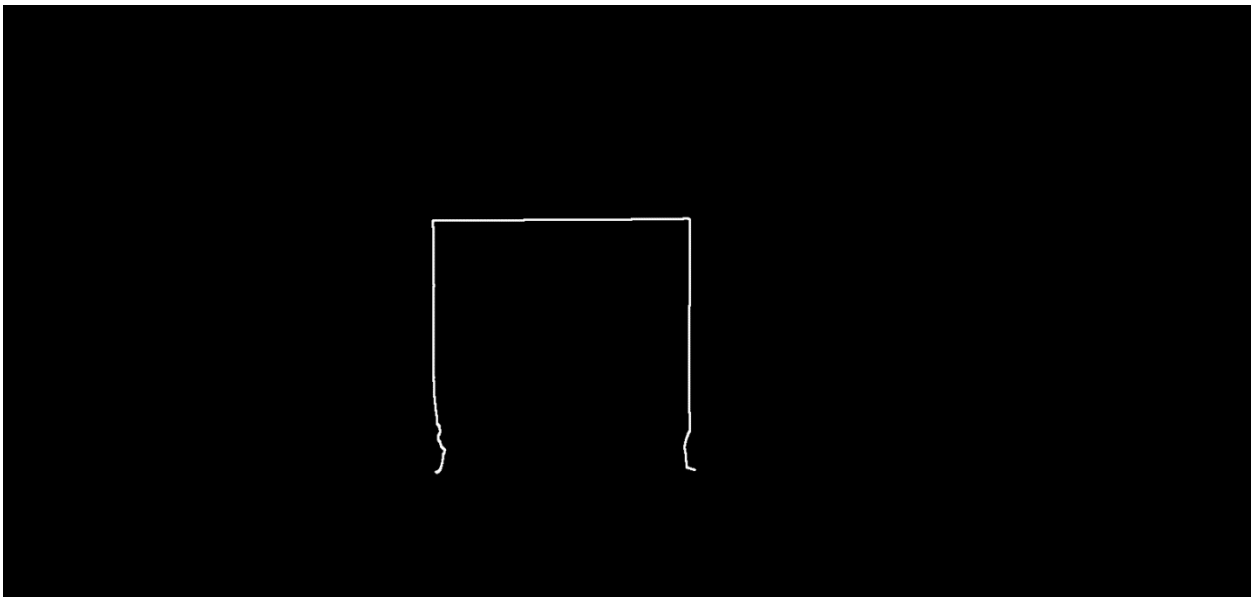


Figure 8.11: Algorithm 6 Step 5: The contour of the printed part as detected by Step 5

1. Draw contour to original frame

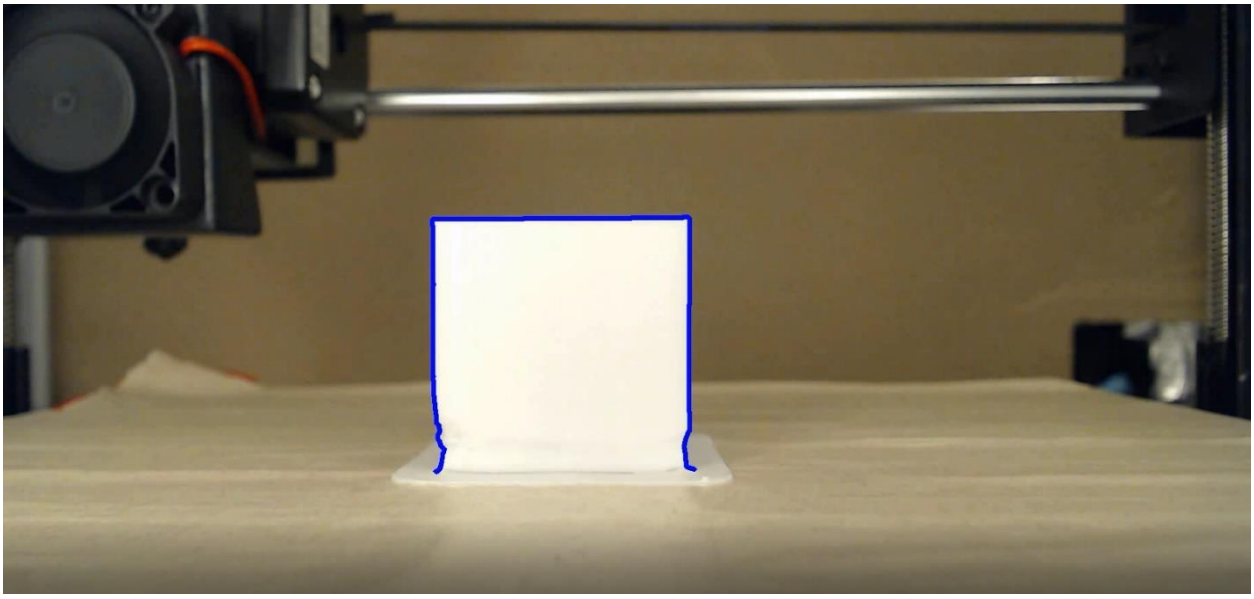


Figure 8.12: Algorithm 6 Step 6: The contour drawn on the original frame

8.6.1 Results

So far, this algorithm has provided the best results. Each algorithm above was run on the same time lapse of Chapter 9: Experiment 5. This algorithm produced edges that fit the shape nearly

exactly when visually compared. This algorithm is not without faults though. While analyzing the time lapse, it occasionally picked up other parts of the print bed and extruder that it thought was the printed part. This was especially common near the beginning of the print when the area of the detection region of the part was closer to the area of other detected regions in the image.

8.7 NRMSE Comparison Algorithm

This algorithm runs on every frame. It is built off the work done by Ben Withers for 3DPrinterSaviour (See Chapter 5.5.2). It records the initial frame from the time lapse in grayscale and uses that as the background. For the next frame, it converts it to grayscale and records it. For every frame after the second one, it converts it to grayscale then, using an absolute difference algorithm, it calculates the differences between the frame and the background frame. It then does the same for the frame previous. Then, using the Normalized RootMean-Square Error algorithm, it calculates the NRMSE score for the differences between the two frames. Finally, it records this score into a plot for analysis.

First and Second Frames

1. Convert to grayscale and record.

Each Subsequent Frame

1. Convert to grayscale
2. Record copy of grayscale for later use
3. Use absolute difference algorithm to find differences between this frame and original frame.

4. Use absolute difference algorithm to find differences between previous frame and original frame.
5. Use NRMSE algorithm to calculate difference score from frame 4 and 5.
6. Record NRMSE score to plot

8.7.1 Results

The NRMSE Comparison algorithm showed some promise on detecting both filament cutoff and slippage without needing any comparison to a CAD model. The issue with it was that the extruder was causing too much of a difference between each frame to be able to accurately detect cutoff. So, there needed to be a method to reduce the impact of the extruder.

8.8 NRMSE with Empty Time Lapse

The NRMSE with Empty Time Lapse algorithm is nearly the exact same as the NRMSE Comparison Algorithm described in 8.7. The only difference is instead of using the first frame of the time lapse as the background, it uses a frame from another time lapse. The second time lapse is run without any filament in the extruder so, only the extruder moves. Because OctoLapse takes snapshots at a constant interval, the height of the extruder will be the same at the same time in two different snapshots. So, because the camera stays in the same position between each print, the background can be eliminated using the empty time lapse. This will take a bit of time depending on the print used.

First Frame

1. Convert to grayscale and record.

Each Subsequent Frame

2. Convert to grayscale
3. Convert frame from empty time lapse to grayscale
4. Run absolute difference algorithm on frame 2 and 3
5. Record copy for use with next frame
6. Run NRMSE algorithm on this frame and previous frame
7. Record score to plot for analysis

8.8.1 Results

The NRMSE algorithm with Empty Time Lapse was an improvement on the original NRMSE algorithm. There was a consistent pattern for prints that ran successfully and prints that had slippage. The NRMSE score spiked when the slippage first occurred and then was erratic afterwards. Some improvements still needed to be made for cutoff to be detected though because there was too much ambient difference caused by the environment.

8.9 Rolling Average Outlier Detection Algorithm

Note: This algorithm uses NRMSE with Empty Time Lapse to generate a NRMSE score. The steps from that algorithm will not be re-explained here. This algorithm definition will start after the NRMSE score is generated. The NRMSE score will typically be below 0.04 for a working print. When slippage is occurring the score can jump up to 0.1. The parameters of this algorithm were detected through trial and error.

After the NRMSE score is generated, it is added to a list of NRMSE scores. If the list has more than 15 values in it, the least recent value is removed from the list. The average score of the

15 scores is then found. This is the rolling average. Next, the bounds are calculated. The upper bound is defined as *1.5 times the average*. The lower bound is defined as *half of the average*. Next time an NRMSE score is received, if it is outside the upper or lower bounds, a counter is decremented. The counter starts at 10 and if it hits 0, the algorithm decides that slippage has occurred, and the user is notified.

Setup

1. Set the counter of remaining outliers to 10

First 15 scores

1. Add to list

Each Subsequent Score

1. Calculate average of score list
2. Calculate upper bound of average as 1.5 times the average
3. Calculate lower bound of average as 0.5 times the average
4. If current score is above upper bound or below lower bound, decrease outlier counter by 1
5. If outlier counter is at 0, alert user that slippage has occurred
6. Otherwise, add score to score list and remove least recent score

8.9.1 Results

This algorithm was able to successfully detect slippage on models that had slippage (See Experiment 9.9 and Experiment 9.11). It also did not detect slippage on several models that did not contain slippage (See Experiment 9.8 and Experiment 9.10). The initial value of the outlier

counter determines how fast and accurate the algorithm is. A lower value, such as 5, makes the algorithm faster but more likely to detect slippage when it is not present. A higher value, such as 15, makes it slower but less likely to detect slippage when it is not present. Too high of a value causes it to not detect slippage while it is present.

Chapter 9: Experimentation

This chapter presents all the experiments and testing that was done with the system throughout the duration of the project. The early stages of the chapter reviews the common definitions that describe experiments and tests. As it progresses, there are more detailed summaries of early experiments that were completed. The primary reasons for these experiments were to understand what hardware would be needed to sufficiently run the system at a consistent level. The later testing that was completed was to actually test the software and the hardware to see if the system in general was successful.

9.1 Experiment Definitions

Many of the experiments in this chapter use similar setups and models with only one or two things changed between each one. In order to save time and space, those common setups will be defined in this section. The setups will then be referenced by title in the experiments below.

9.1.1 Models

This section describes the CAD model each experiment used. Links to all CAD models can be found in a footnote on this page.

Cube: A cube measuring 3cm on each side. Created in Solidworks.

Cube with Raft: Similar to Cube. This model was sliced in Cura to include a raft to help with print bed adhesion.

Benchy: A standard 3D printer testing model. Created by www.3dbenchy.com. This model was sliced with a raft (See Terminology).

Benchy with Support: Same model as Benchy. This model was sliced in Cura (See Terminology) with supports enabled everywhere.

Spaghetti Crescent: A model in the shape of a crescent. It has an overhang that needs support. In order to generate spaghetti (See Terminology) support was disabled in Cura while slicing. This model has a raft.

9.1.2 Cameras

This subsection is to define the types of cameras used in our experiments/testing. The settings used for the cameras are in Appendix D.

Logitech C920: Logitech C920 webcam. It is able to shoot video at 1080p and has autofocus and auto-exposure.

Calibrated Logitech C920: Similar to Logitech C920. Autofocus and auto-exposure are disabled and replaced with custom parameters. (See Appendix D: OctoLapse Camera Settings)

9.1.3 Camera Positions

The purpose of this subsection is to explain the primary position in which the camera was located during testing.

Front-Centered: The camera was placed in the front of the printer with the lens facing the printer. The camera is then centered as pictured in Figure 9.1

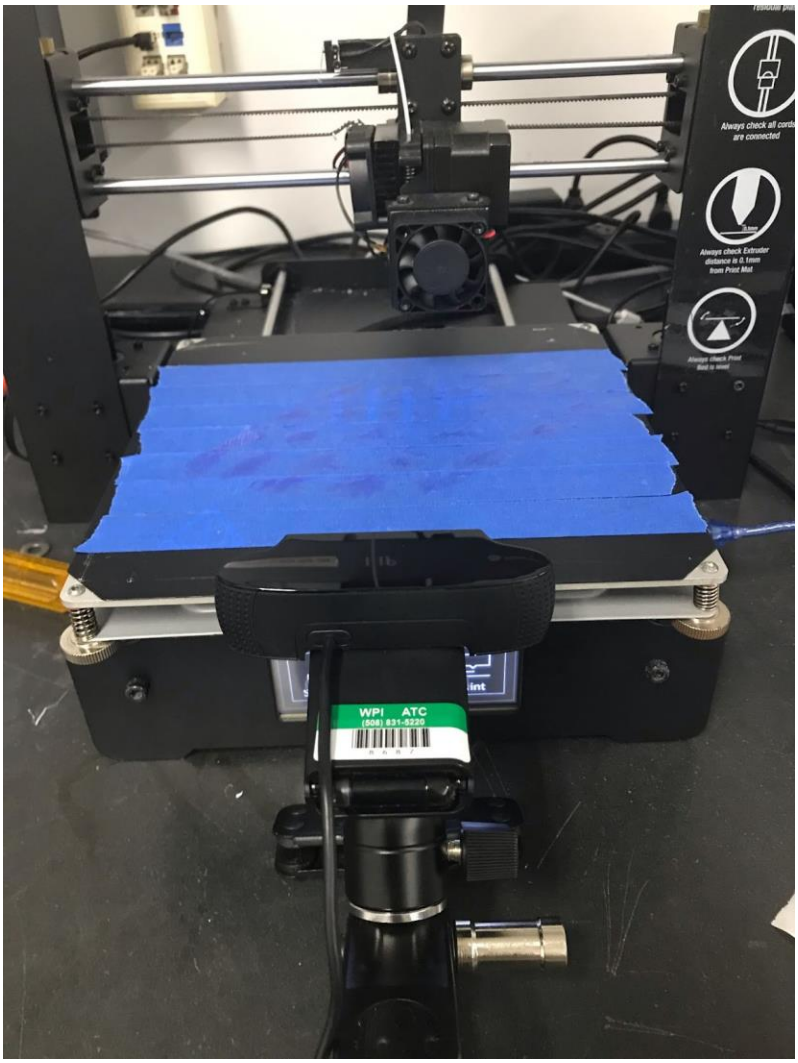


Figure 9.1: Image of the webcam in the front centered position

9.1.4 Printer Gantry Position

The gantry (See Terminology) position refers to the position of the extruder relative to the print bed.

Centered: The print bed is moved to the midpoint of the y-axis while the extruder is moved to the midpoint of the x-axis. This puts the extruder directly over the center of the print bed.

Back-Left: The print bed is put to its limit in the negative y-axis (See Terminology), which puts it as close to the camera as possible. The extruder is then moved to the origin of the X axis which puts it as far left on its rails as possible.

9.1.5 Print Bed Material

This subsection goes through the various types of colored material that was applied to the print bed during testing.

Blue Painters Tape: One layer of blue painters tape applied over the entire surface of the print bed.

Brown Masking Tape: One layer of red tape applied to the print bed with a layer of brown masking tape applied on top.

9.1.6 Background

This section refers to the various backgrounds that were used in testing. The background is placed behind the printer and meant to prevent the wall and wires from being seen by the camera.

None: No background was placed behind the printer. The wall of the MQP lab is visible along with the wires for the 3D Printer.

Cardboard: A piece of browncardboard from a box is placed behind the printer. This covered the wall and the wires behind the printer where the camera was viewing.

Enclosure: An aluminum lined enclosure housed the printer, this covered the wall and the wires from the printer. It creates a textured metallic background when the camera is activated.

Enclosure with Cardboard: The cardboard is placed inside the enclosure behind the printer.

9.1.7 Lighting

This subsection touches upon the different types of lighting arrangements used during the experiments.

None: No additional lighting was used in the print. The only lighting was the ambient lighting of the MQP lab.

Flashlight: A flashlight was attached to the top-right of the enclosure with relation to the printer. It is shining on the print bed as a constant source of light.

Cold Lighting: An LED strip is placed behind the cardboard to illuminate the backdrop to the camera. A ring light is placed behind the camera to illuminate the face of the printed part. The ring light is set to the coldest setting.

Neutral Lighting: Similar to Cold Lighting but the ring light is set to a neutral light.

Warm Lighting: Similar to Cold Lighting but the ring light is set to a warm light.

9.2 Experiment 1

Model	Cube
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Center
Background	None
Print Bed Material	Blue Painters Tape

Lighting Used	None
Goals	• Run first test with OctoPrint and OctoLapse setup • Become more familiar with OctoPrint and OctoLapse

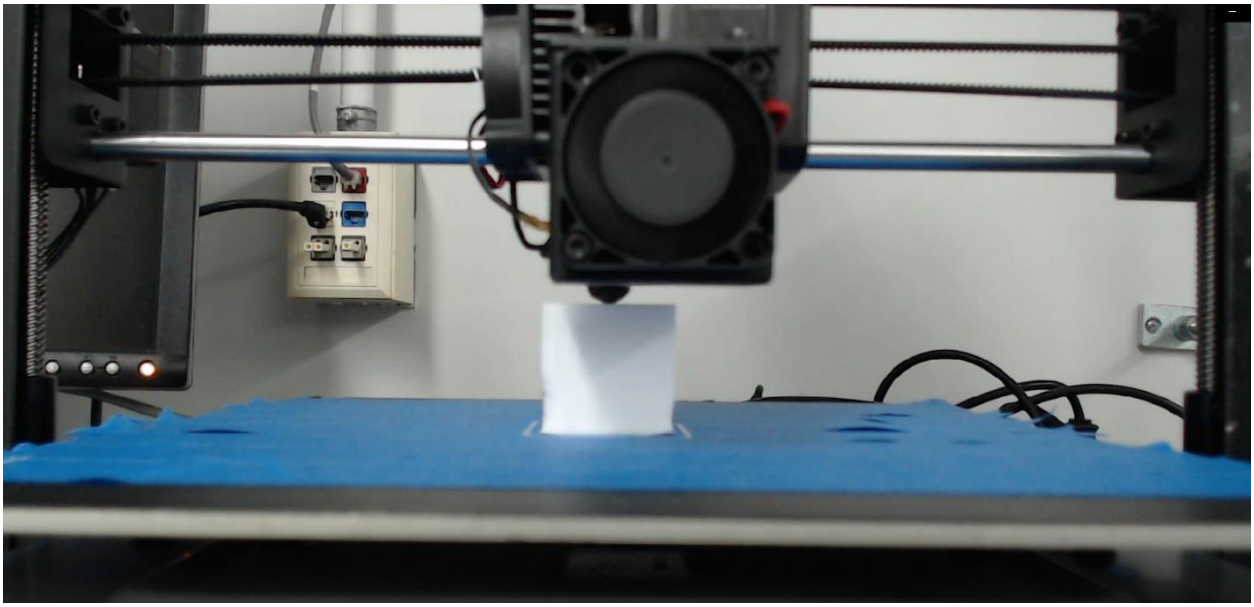


Figure 9.2 Final Snapshot from OctoLapse of Experiment 1

9.3 Experiment 2

Model	Cube
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Cardboard
Print Bed Material	Blue Painters Tape
Lighting Used	None

Goals	• Try to improve camera focus by putting a background behind the printer • Move gantry to back left to get better view of printed part

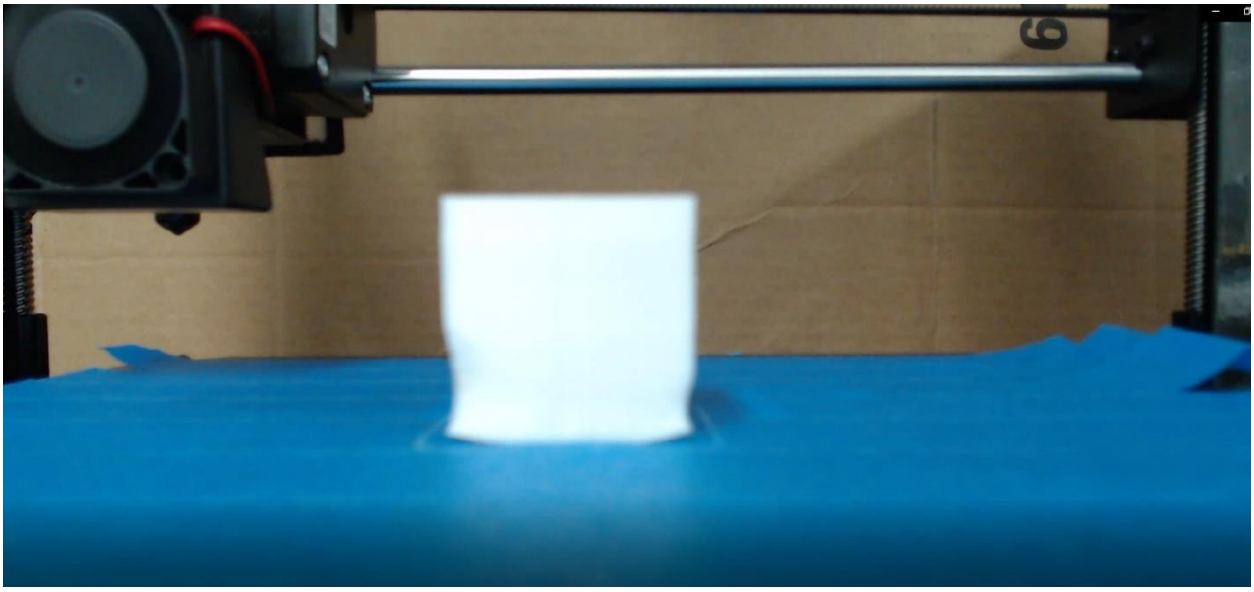


Figure 9.3: Final snapshot from OctoLapse of Experiment 2

9.4: Experiment 3

Model	Cube
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Cardboard
Print Bed Material	Blue Painters Tape
Lighting Used	None

Goals	• Test new manual camera focus and exposure. (See Appendix D: OctoLapse Camera Settings) • Test print bed at lower temperature to prevent bottom layer warping

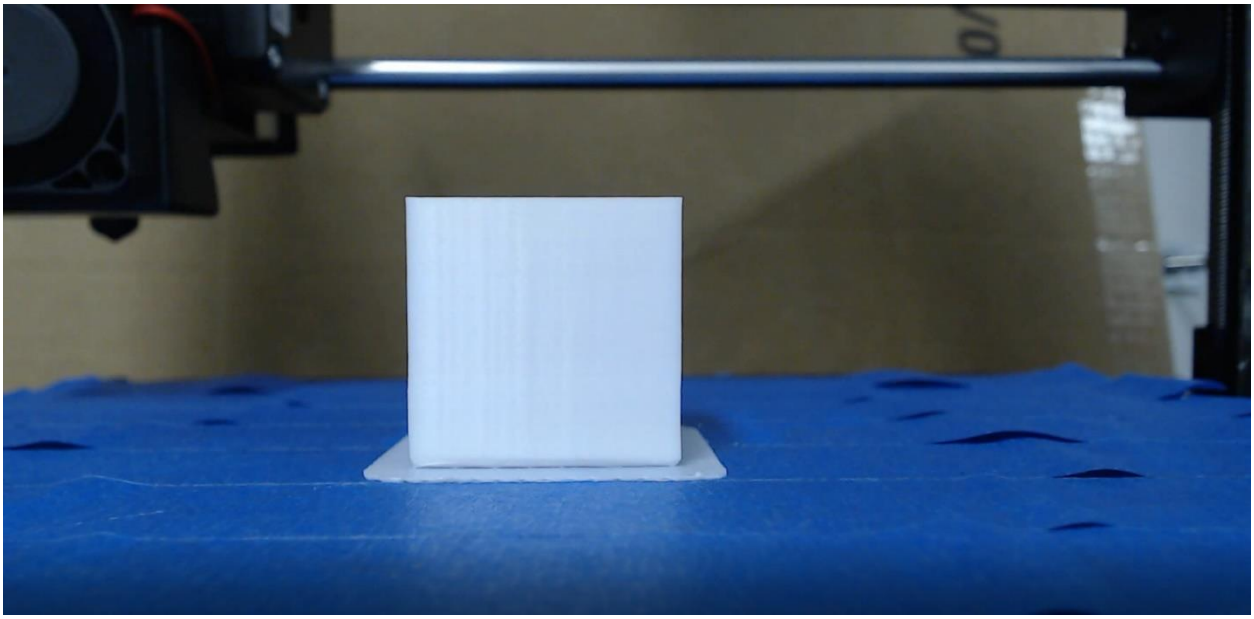


Figure 9.4: The final snapshot from OctoLapse of Experiment 3

9.5 Experiment 4

Model	Benchy
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure
Print Bed Material	Blue Painters Tape

Lighting Used	Flashlight
Goals	• Test new enclosure • Print new part to assess outline extraction algorithms • Test new lighting

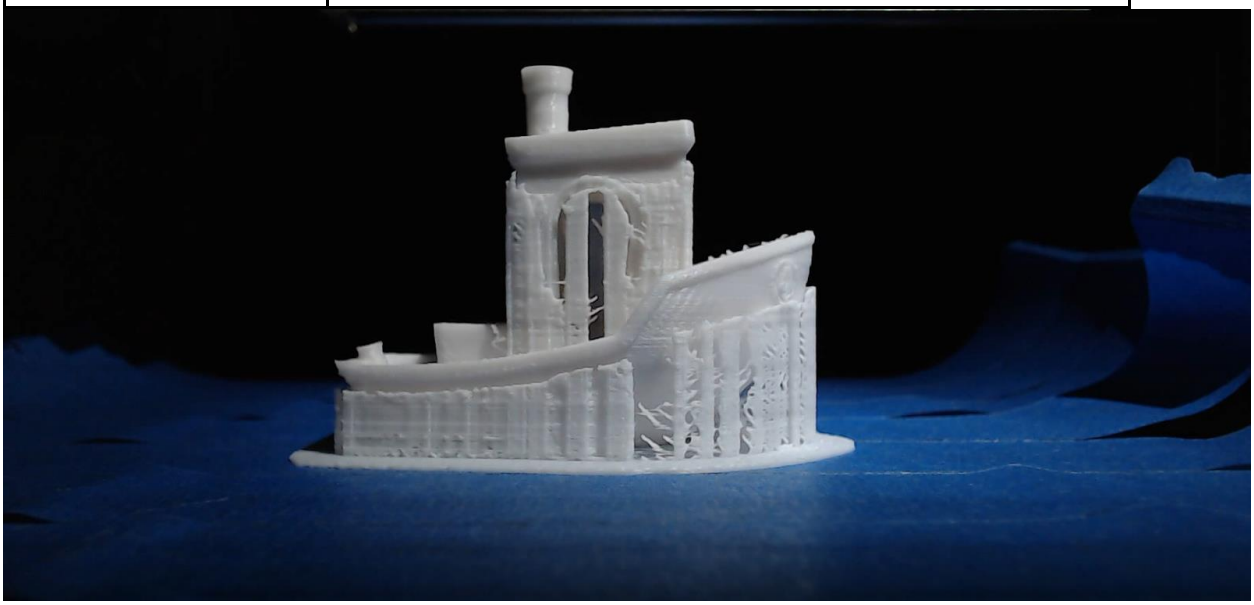


Figure 9.6: Final snapshot from OctoLapse of Experiment 4

9.6 Experiment 5

Model	Benchy
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure with Cardboard
Print Bed Material	Brown Masking Tape
Lighting Used	Warm Lighting

Goals	• Test new tape setup and cardboard background with new lighting setup • Test ring light and back light

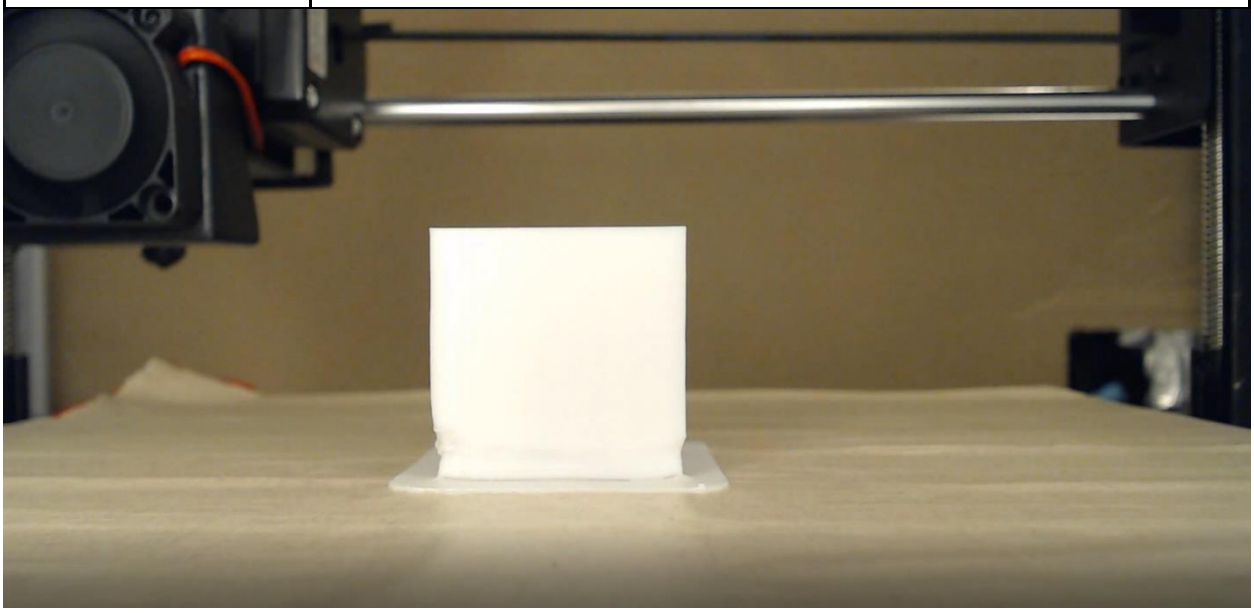


Figure 9.6: Final snapshot from OctoLapse of Experiment 5

9.7 Experiment 6

Model	Spaghetti Crescent
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure with Cardboard
Print Bed Material	Brown Masking Tape
Lighting Used	Warm Lighting

Goals	• To test Spaghetti Detector with a part guaranteed to create spaghetti

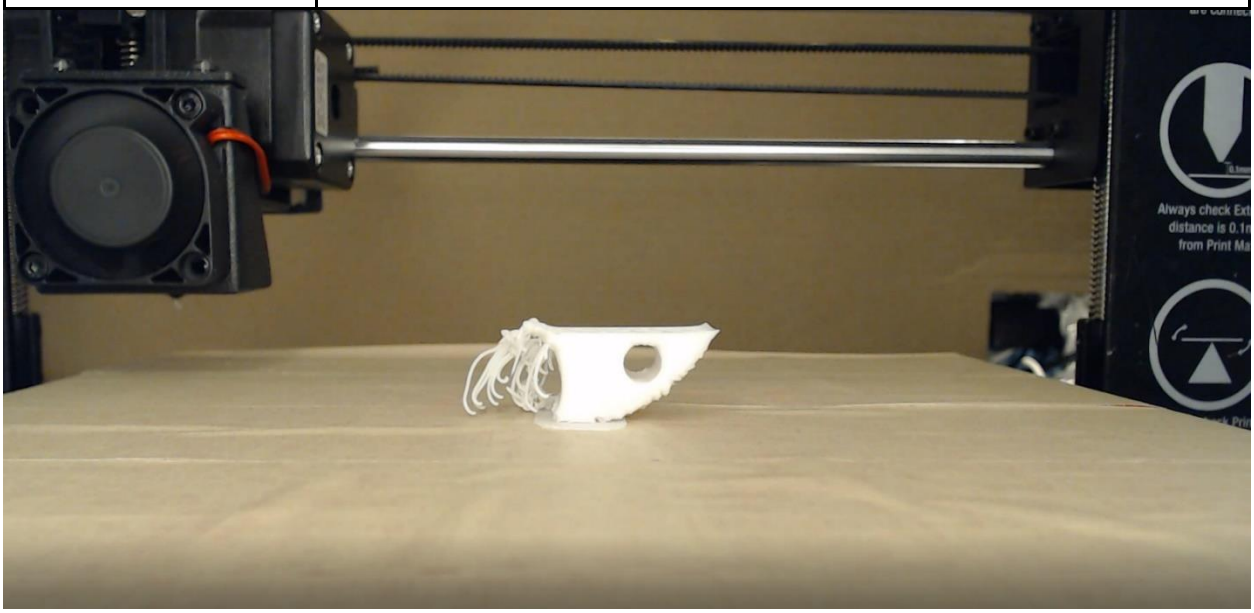


Figure 9.7: The final snapshot from OctoLapse of Experiment 6

9.8 Experiment 7

Model	Various
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure with Cardboard
Print Bed Material	Brown Masking Tape
Lighting Used	Warm Lighting
Goals	• To create several timelapses without any defects occurring

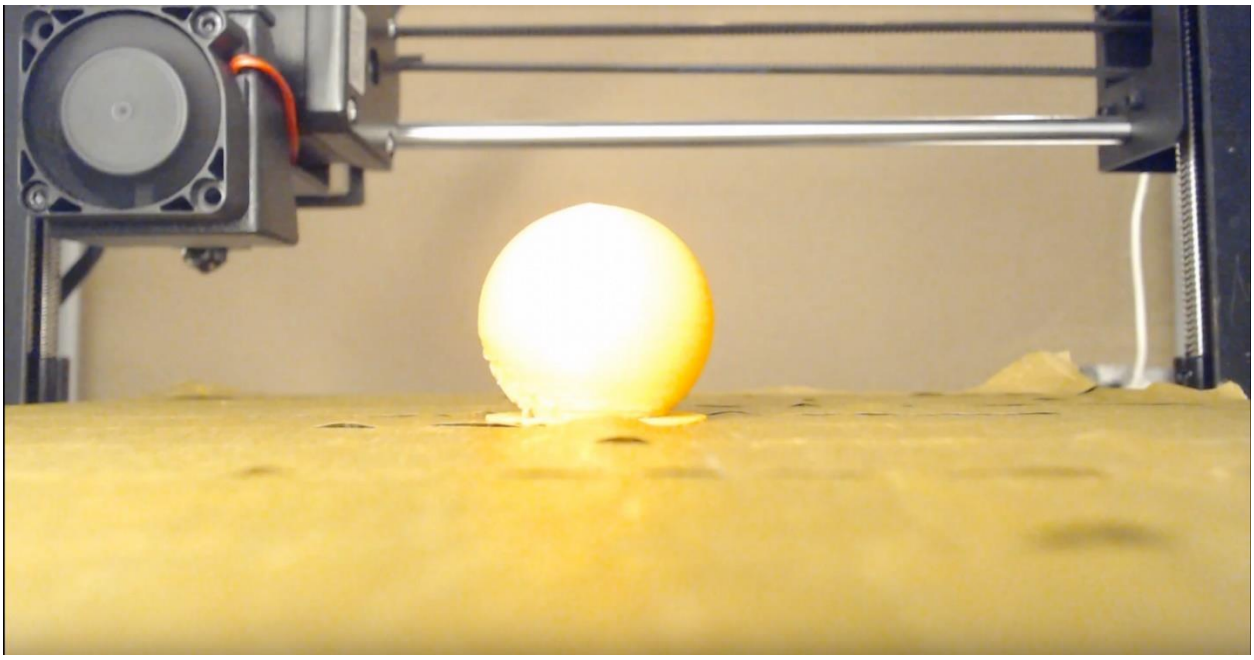


Figure 9.8: The final snapshot from OctoLapse of Experiment 7

9.9 Experiment 8

Model	Various
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure with Cardboard
Print Bed Material	Brown Masking Tape
Lighting Used	Warm Lighting
Goals	• To create several timelapses with slippage occurring

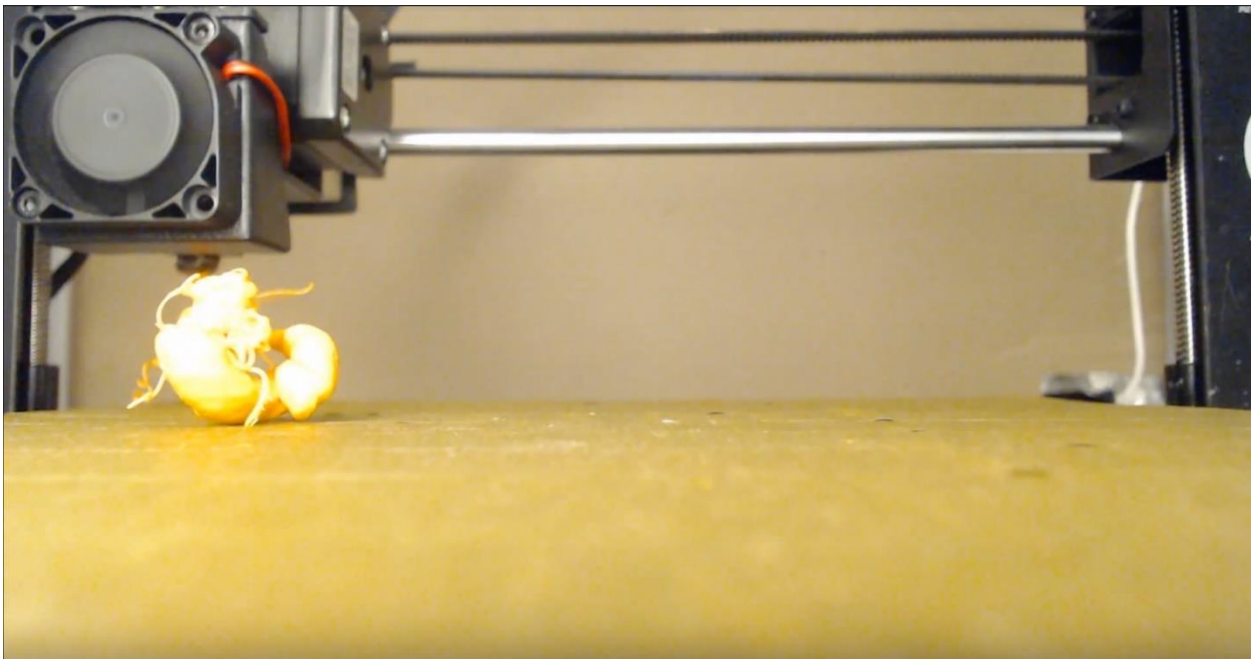


Figure 9.9: The final snapshot from OctoLapse of Experiment 8

9.10 Experiment 9

Model	Various
Printer	Monoprice Maker Select Plus
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure with Cardboard
Print Bed Material	Brown Masking Tape
Lighting Used	Warm Lighting
Goals	• To create several timelapses with run out occurring

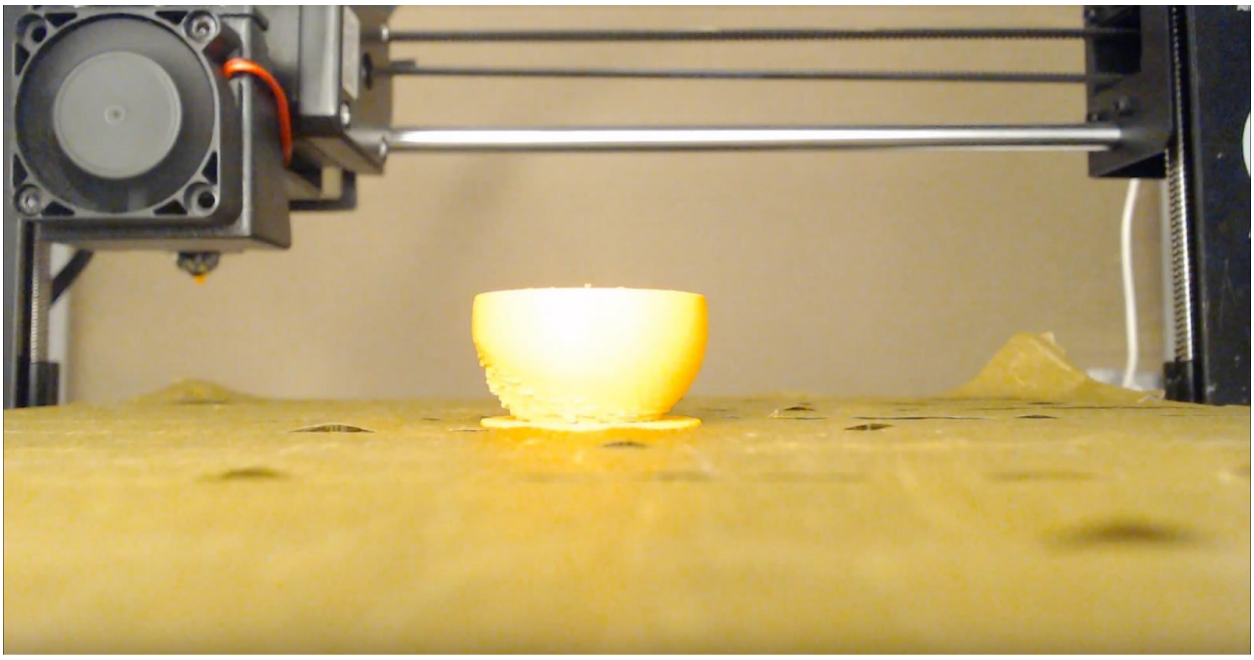


Figure 9.10: The final snapshot from OctoLapse of Experiment 9

9.11 Experiment 10

Model	Various
Printer	RepRap
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure with Cardboard
Print Bed Material	Brown Masking Tape
Lighting Used	Warm Lighting
Goals	• To create several timelapses without any defects occurring

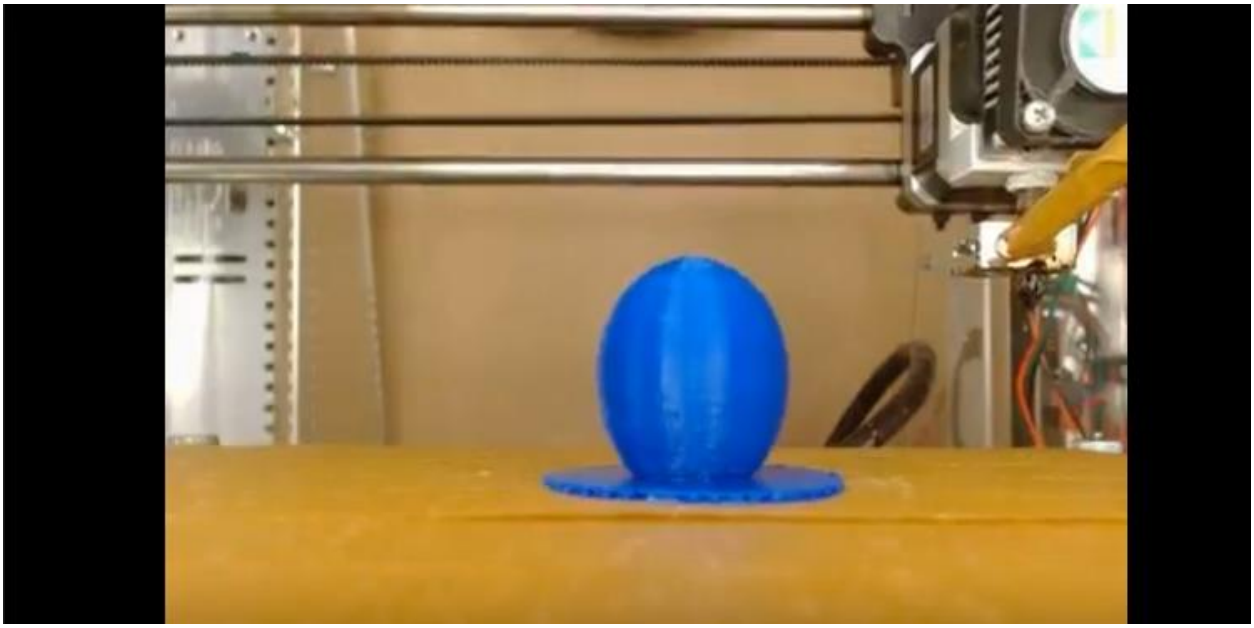


Figure 9.11: The final snapshot from OctoLapse of Experiment 10

9.12 Experiment 11

Model	Various
Printer	RepRap
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure with Cardboard
Print Bed Material	Brown Masking Tape
Lighting Used	Warm Lighting
Goals	• To create several timelapses with slippage occurring

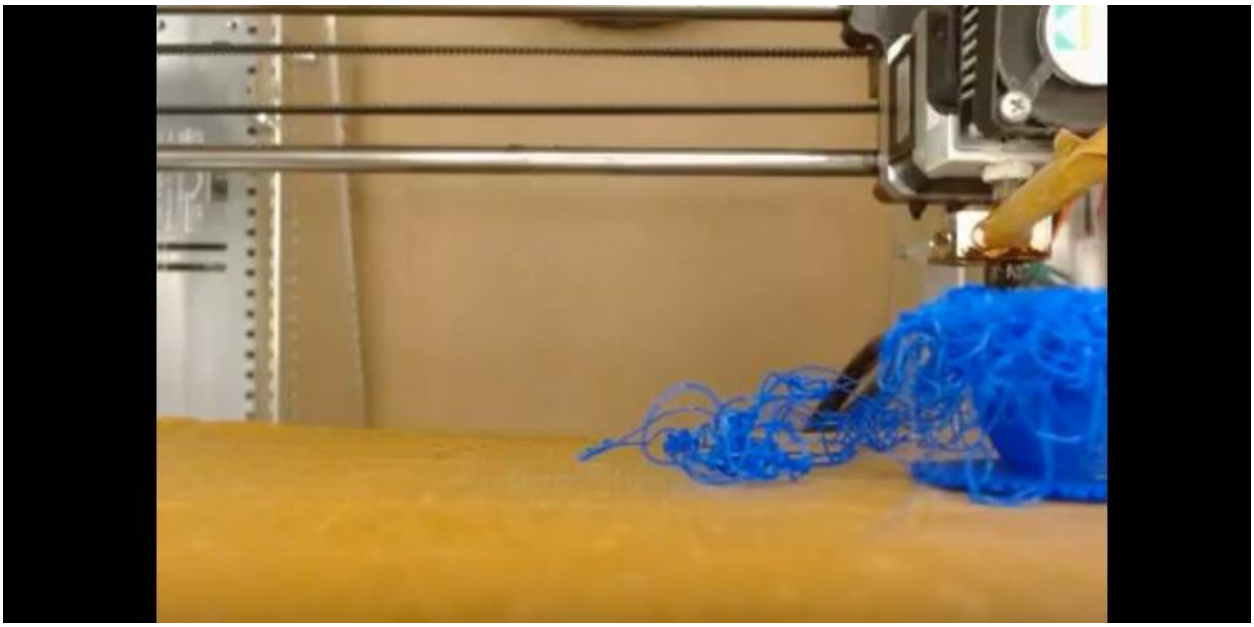


Figure 9.12: The final snapshot from OctoLapse of Experiment 11

9.13 Experiment 12

Model	Various
Printer	RepRap
Camera	Logitech C920
Printer Gantry Position	Back-Left
Background	Enclosure with Cardboard
Print Bed Material	Brown Masking Tape
Lighting Used	Warm Lighting
Goals	• To create several timelapses with run out occurring

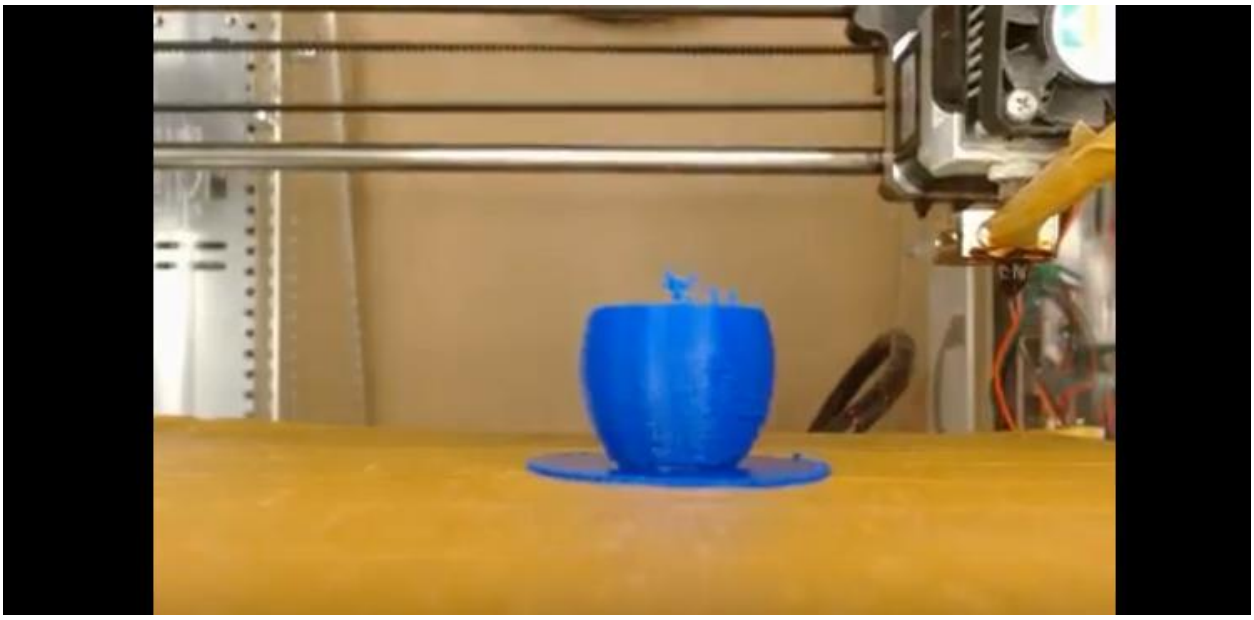


Figure 9.13: The final snapshot from OctoLapse of Experiment 12

Chapter 10: Evaluation

To evaluate the usefulness of DaR3D, it was necessary to get feedback from people who were most likely to use such a system. This chapter first discusses who those people were, and what information was needed from them. Then it describes the procedure for getting this information. Finally, there is an explanation and analysis of the results.

10.1 Requirements

This section is a discussion about the requirements for various facets of data collection. It also serves as a rationale for why things were done the way they were. It explores things such as what information needed to be given to participants, who the participants were, how they were contacted, what information needed to be collected from them, and how data was analyzed, among other things. A bulleted list of requirements can be found in Appendix A.

10.1.1 System Explanation

For people to be able to give their thoughts about our system, they first needed to know what the system was. This meant creating a comprehensive description of the system which could be used as a script. This would ensure that each participant got a sufficient explanation, and that no participant got more or less information than another. Additionally, since many components of the system may not be intuitively understood based only on written or verbal description, it was helpful to have a visual aid to go with the description.

The first thing the script had to do was explain why the system was needed. This included a description of the problem and a statement about how DaR3D would address the problem. Then, participants needed to know how the system was set up and how it worked, so they could

understand what its components were and what was required from a user. In this section, there needed to be a description of the hardware components and how they fit together, as well as a basic explanation of how the software works. Since many of the people who would use DaR3D are more familiar with concepts of mechanical engineering than computer science, we decided to put a higher focus on the explanation of hardware components and keep the software explanation as simple as possible.

Next, we determined it would be helpful to walk the participant through using the system, explaining how it works from the user's perspective and what a user would expect to see. This would help participants decide if the system seemed intuitive and easy to use. Another tool for helping participants decide if the system would be useful to them was statistics and other numerical data. At the very least, we thought a statement about the accuracy of the system was necessary. Other ideas included the approximate cost of the system, or the size of the hardware setup, but ultimately, that information was taken out since it was highly variable. Instead, we included a few statements about the possibility for the user to customize their setup to work for them if needed.

Finally, since there are many possibilities for the future of DaR3D, we wanted to inform participants about the features that DaR3D could have in the future. By gaining feedback about these features, we would be able to provide recommendations on where to focus future efforts, and prompt participants' imaginations so they could provide suggestions of their own. This section explained our ideas for additional defect detection, a wider range of compatibility with different systems, a library of camera mounts, a user interface, and more. The full script can be found in Appendix B.

Since the system explanation might be hard to understand without visual aids, we had to determine what sort of visual aid would be most effective for communicating all the necessary information. The most important visual components of the system were related to the hardware design, such as the printer enclosure, camera fixture, and lighting setup. Hence, we thought that an in-person live demo of the system would be beneficial for participants' understanding. However, if we wanted a large number of responses, then the logistics of organizing a live demo for every participant, or even for small groups of participants, would be too complicated. Additionally, we thought that other aspects of the script, such as the description of how the software works, would benefit from having visual aids as well.

The next idea was to create a video using the script as a voice over and recording a slideshow for the visuals. This would ensure that every participant got a consistent description of the system with sufficient visual aids. Additionally, a video would be useful whether we decided to do interviews or surveys, since participants would get a relatively equal experience whether they saw the video in person or virtually.

10.1.2 Participants

Depending on who we received feedback from, we would get very different responses about DaR3D. Therefore, we had to decide which group of people would be best to ask about the system. The chosen participants would have to be experienced with 3D printing, or at least experienced with 3D printed parts. Ideally, we would have people at varying levels of experience to get a sense of how useful DaR3D would be to different kinds of people. To get a lot of feedback in a short amount of time, we decided to create a survey which would go to students who have at least basic knowledge of 3D printed parts. We also decided that we would do

interviews with people who had more extensive 3D printing experience in order to get more comprehensive feedback about DaR3D.

The students for the survey came from ME4320, a WPI senior level mechanical engineering course taught by our advisor, Pradeep Radhakrishnan. The students in this course were likely to have experience with 3D printed parts, and some would likely have experience with 3D printers from their previous experiences as mechanical engineering students. This got us about 25 responses, which we deemed sufficient for the scope of this project. For people with more 3D printing experience, we reached out to students who we knew to be experienced, such as those who owned personal 3D printers, and those who worked in the prototyping lab on campus which sees dozens of prints each day. We also reached out to faculty members of the university who worked closely with the prototyping lab or 3D printing in general.

10.1.3 Student Survey

The two most important aspects of the student surveys were what to ask and how to ask it. We decided to break the survey into three parts. First, we would ask the students demographic questions. This would include information such as their year and major, as well as several questions about their experience with 3D printing. This type of information would allow us to analyze whether certain features would be seen as more or less useful depending on the user's experience level. It would also give us a sense of if people would be more likely to use the system if they had higher or lower experience.

The next section asks questions about the system with its current functionality. This section is the core of the survey, since one of the main purposes for user evaluations was to find out what people thought of the system. In order to figure out what respondents thought, we implemented two different question styles: Likert scale, and free response. The Likert scale

questions would give us numerical data about how likely people would be to use the system. The free response questions would allow the participants to go more in-depth about the particular features that they liked and disliked. We also decided to include an additional free response question where participants could ask any questions. This information would let us know what aspects of the system were confusing to people so that everything could be made clear for future users.

The state of DaR3D by the end of the project was limited, since the team did not have time to implement many of the features that we had originally thought of. Therefore, in addition to asking participants what they thought about the current system, we decided it would be beneficial to ask them what they thought about some of the additional features that could be added in the future. This would help us gauge which features participants thought would be most useful so future developers would know what to focus on implementing first. The format of this section would be fairly similar to the previous section, with a few additional questions. We would ask them if they had any recommendations for the system in case we overlooked an important idea for future work. We would also ask if they had any further comments on the project in case participants felt that the existing questions did not give them the space to say what they wanted to. Appendix C contains the final list of questions for the student survey.

10.1.4 Expert Interviews

The experts we decided to contact were faculty associated with the 3D printer lab in the makerspace at WPI, faculty with non-makerspace affiliated 3D printing experience, and students with extensive personal 3D printer experience. Since our aim was to contact three different types of interviewees, we determined that the requirements for interviewing those people would be slightly different based on the group they were a part of.

For interviewees associated with the prototyping lab, we wanted to take advantage of the fact that they see a high volume of parts every day. Therefore, it would be the perfect opportunity to ask about the most common defects they see, and how often they see defects. Additionally, since the prototyping lab is a system of many 3D printers, the interviewees associated with it would have a much different experience with 3D printing than someone who owns a single printer for personal use. Because of this, we decided it would be necessary to ask what they think about DaR3D in the context of their system of printers, and whether or not they thought aspects of DaR3D would be helpful within that system.

Faculty members not associated with the prototyping lab would likely have widely varied experiences and reasons for 3D printing, so it would be necessary to have a flexible set of questions to accommodate those experiences.

Experienced students would likely be the easiest to contact for interviews, and their familiarity with 3D printers would be more consistent compared to some of the faculty interviewees. Therefore, we decided it would be best to do a series of focus groups where 3 or 4 students could answer questions together and bounce ideas off of each other. The questions would be fairly standard, but by allowing collaborative answers, we hoped that participants would explore the topic deeper and provide interesting insights.

To keep it as simple as possible while still meeting all the requirements, we decided to develop a set of base questions about participants' general experiences with 3D printers, then add additional relevant questions depending on who we were interviewing. The full set of prepared expert interview questions can be found in Appendix D.

10.2 Procedure

This section outlines the procedure for getting responses from the student survey and the expert interviews.

10.2.1 Student Survey Procedure

The procedure for the student survey was relatively straightforward. Participating students from Professor Radhakrishnan's class were contacted with a basic description of the project and an explanation of what they were expected to do. This included a video for them to watch, and a link to the survey, as well as a deadline for when they should complete the survey. In this initial contact, we also told students that their responses would be confidential, and

The survey was broken up into three sections, with relevant information given at the beginning of each section. The first section stated what the project was and who the contributors were, reiterated the fact that responses were confidential, and told participants when they would be able to access the results. Then, all the demographic questions were asked. The second section pertained to the current functionality of the system. At the beginning of the section participants were asked to read a few paragraphs about how the system worked. These paragraphs were directly copied from the script, which was also used for the video voice over, so participants should already have been aware of the information at this point. However, we decided to include the information in paragraph form within the survey in order to clearly define what the current functionality of the system was, and allow participants to easily refer back to specific details when formulating their responses. The third section of the survey was very similar to the second section, except that it pertained to future features. Participants were given a few paragraphs about future plans for DaR3D, then asked to give their opinion on them.

Once the deadline was reached, we organized the responses into a spreadsheet, then analyzed them. For Likert scale questions, we analyzed responses numerically to find the mean

and mode responses. For free response questions, we did keyword analysis to find general response patterns.

10.2.2 Expert Interviews Procedure

The procedure for each expert interview was slightly different, but there were some common elements. The first step was to email participants with a basic description of our project, and an explanation of what we wanted from them. This email would also ask what times the participants were available to meet. The only exception was for the focus group interviews, where participants were instead sent a link to a when2meet to streamline the process of finding an available time for 3 or more people.

For each interview, we started by showing participants the DaR3D explanation video, so that every participant would get an equal understanding of the system. Before proceeding, we asked if there were any questions about the video in case participants thought anything was unclear. In order to make it easier to go back and analyze data from the interviews, we thought it would be best to get audio recordings. So, we asked the participants if they consented to being recorded. If they said yes, we started recording at that point. Next, we asked questions as laid out in Appendix D according to who was being interviewed. Then we asked additional questions based on what was learned during the previous questions. Each interview was scheduled to take no more than 30 minutes.

Once each interview was completed, the audio recording could be analyzed. This would include an overview of what the responses were, a comparison to responses from other experts to identify patterns, and an analysis of what those patterns indicate about DaR3D.

10.3 Results

This section outlines the results of the student survey and expert interviews. Full results for the student survey can be found in Appendix H. A transcript of the recording for the student focus group can be found in Appendix I.

10.3.1 Student Survey Results

By the time of the deadline for the student survey, we received 25 responses. Figure 10.1 shows how experienced the respondents were with 3D printers and 3D printed parts. The key on the right-hand side is measured in years. About three quarters of respondents had more than one year of experience with 3D printers or 3D printed parts, and almost one quarter of respondents had more than 3 years of experience.

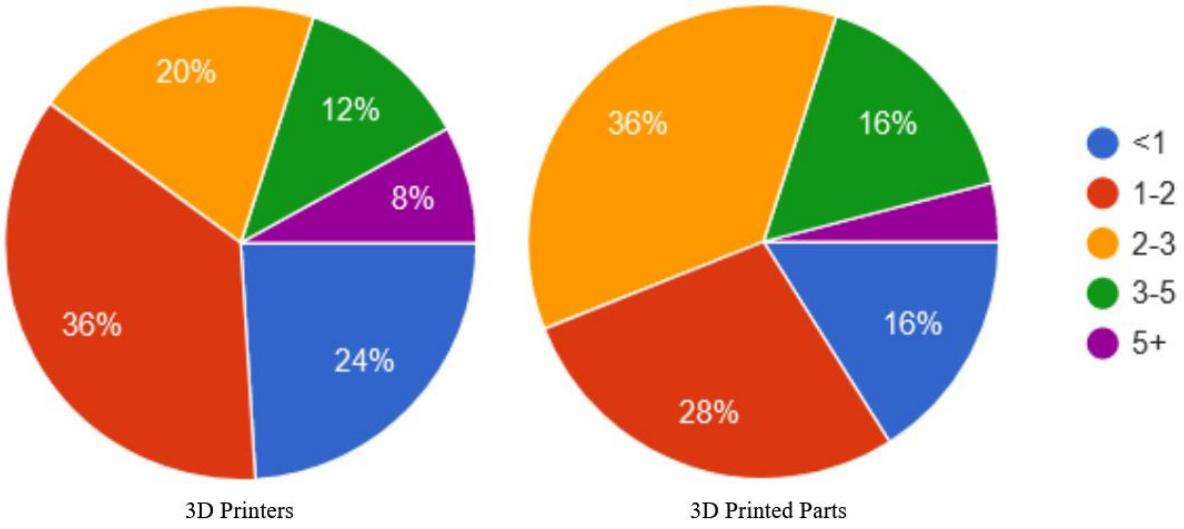


Figure 10.1: Proportion of survey respondents at different experience levels

Out of the 25 respondents, 7 of them personally owned a 3D printer. This, combined with the data from figure 10.1, indicates that participants had a wide range of experiences with 3D printing, and would therefore give a wide range of perspectives.

When asked how likely they would be to use DaR3D in its current state, 19 respondents (76%) said they would be likely or very likely to use it. Only one respondent said they would be unlikely to use it. As seen in figure 10.2, a response of 5, meaning very likely, was most common. The y-axis represents the number of responses. The mean response was 4.12, just slightly over "likely." Among respondents who personally owned a 3D printer, the mean response was a bit higher, at 4.43.

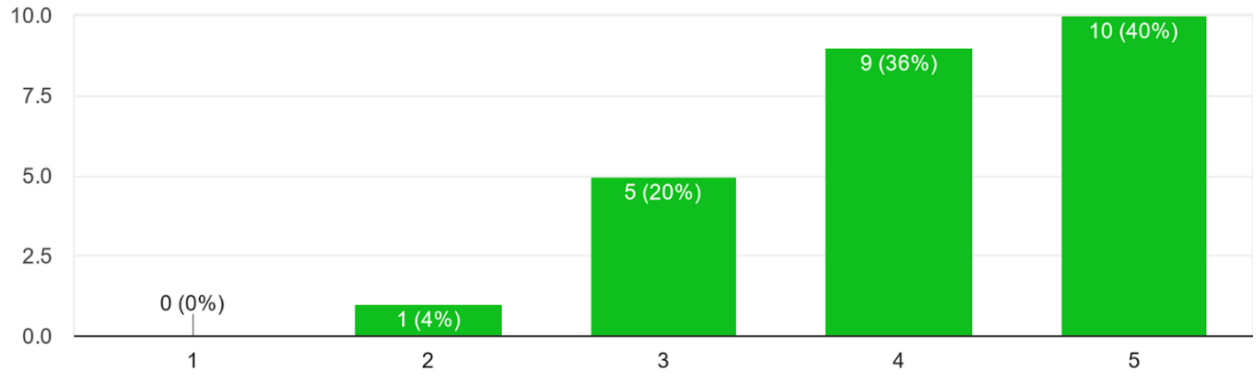


Figure 10.2: Likeliness to use DaR3D in its current state. 1 = very unlikely, 5 = very likely

When asked how likely they would be to use DaR3D with additional functionality, 21 respondents (84%) indicated that they would be likely or very likely to use it. Figure 10.3 shows that a response of 5 was once again the most common, but by a larger margin this time. The mean response was 4.36, indicating an average increase in likelihood of 0.24 across all respondents. Among those who personally owned a 3D printer, the mean response was 4.57; an increase of only 0.14.

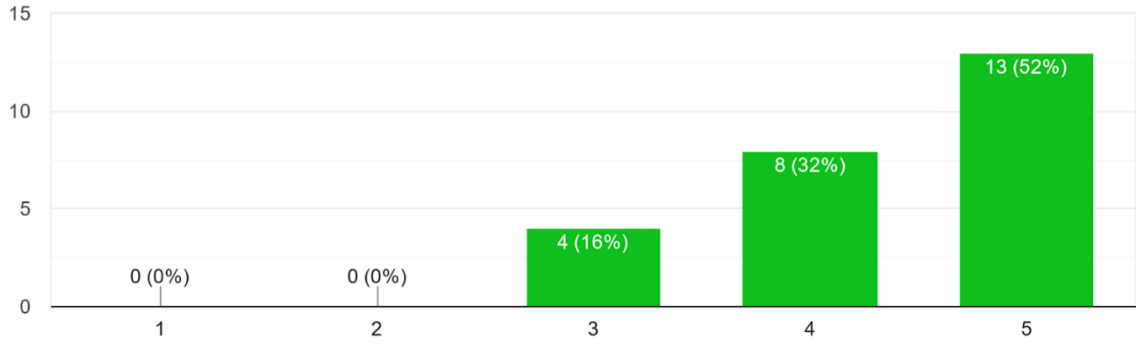


Figure 10.3: Likeliness to use DaR3D with additional features. 1 = very unlikely, 5 = very likely

When asked about what they liked about DaR3D in its current state, respondents commonly mentioned several features through free response answers. The feature participants thought was most useful was the automatic notification system, with ten answers containing something about getting an email, notification, or alert. Two of these respondents additionally claimed that they thought the snapshot and timestamp included in the email would be useful. The next feature participants thought was useful, with eight answers about it, was the ability for users to be away from the printer for long periods of time. With seven answers, the next most useful feature was thought to be the ability to detect defects, specifically slippage. Six respondents cited saving time and/or filament as a useful feature, and five respondents liked the remote stopping feature. Figure 10.4 shows a graph of the features that users thought were most useful. Note that many users mentioned more than one feature in their response, so the total number of responses does not add up to 25.

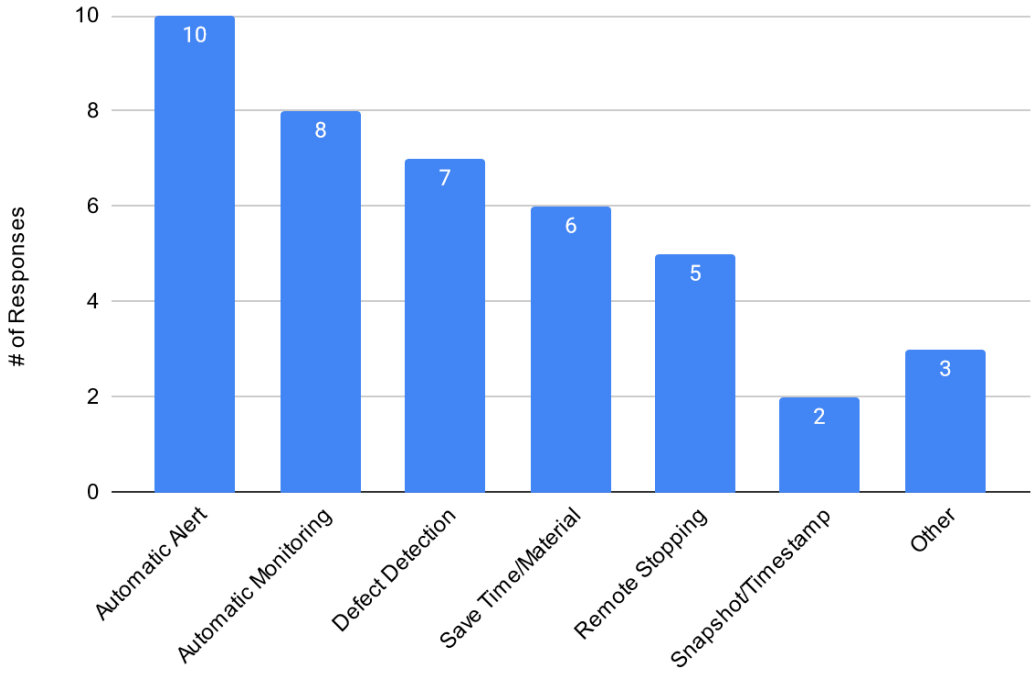


Figure 10.4: Features considered “most useful” by survey respondents

When asked about the features of DaR3D that they thought were least useful, respondents most commonly mentioned the hardware setup. Seven participants wished that certain hardware components such as the enclosure, ring light, raspberry pi, etc. were not necessary. Some explained that they thought it would be a hassle, while others said that they would prefer it if they could look at the part while printing, rather than having their vision be blocked by the enclosure. While figure 10.4 established that participants were fond of the automatic alert feature, four participants specifically mentioned that they did not think email was an ideal method of alerting the user, instead suggesting that something like text message would be more useful. Three responses mentioned the camera fixture as being unnecessary, and another three wished that the system could detect more than just slippage. Figure 10.5 shows a full graph of responses.

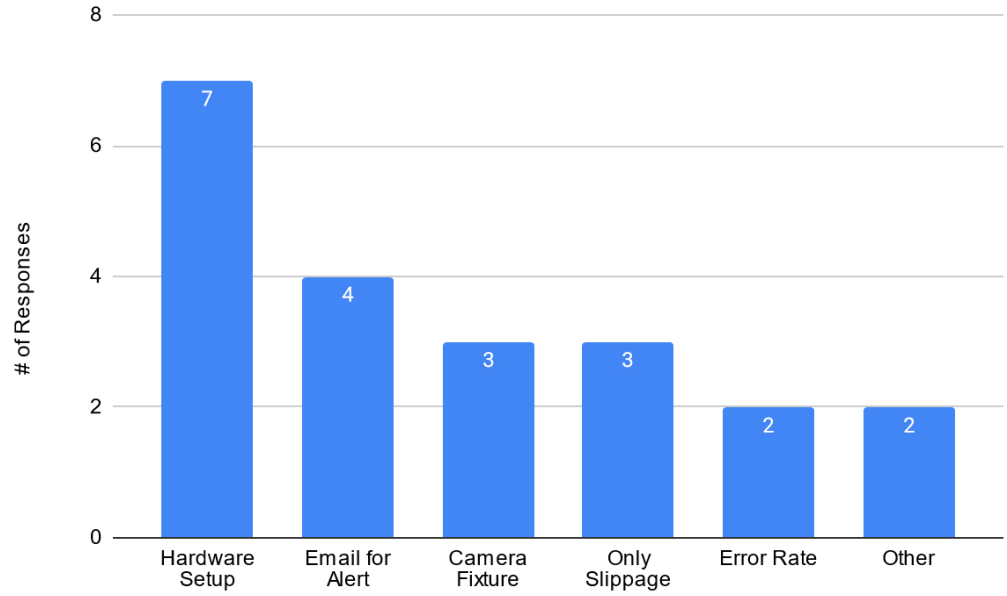


Figure 10.5: Features considered “least useful” by survey respondents

When looking at future features of DaR3D, two features in particular were very well-liked by respondents. The first feature, with twelve responses talking about it, was the ability to detect more defects. Specifically, many of the participants who mentioned more defects wanted DaR3D to be able to detect warping and stringing, with only one response about detecting filament runout in the future. The second feature, with ten responses about it, was the option for a user to choose which notification method they wanted to use. Most of the participants who liked this feature specifically liked the idea of a text message notification, because they perceived it to be more urgent and time sensitive than an email. With only four responses, the next most liked future feature was defect explanations. These participants liked the idea of getting information about what defect occurred and why. A few respondents also liked the idea of automatic stopping, rather than manual stopping of the print. However, other questions in the survey had responses saying they would not want automatic stopping, indicating a split opinion among participants. There were also two responses expressing their appreciation for the idea of

DaR3D being highly customizable to different systems in the future. Figure 10.6 shows a graph of the various future features that participants were fond of.

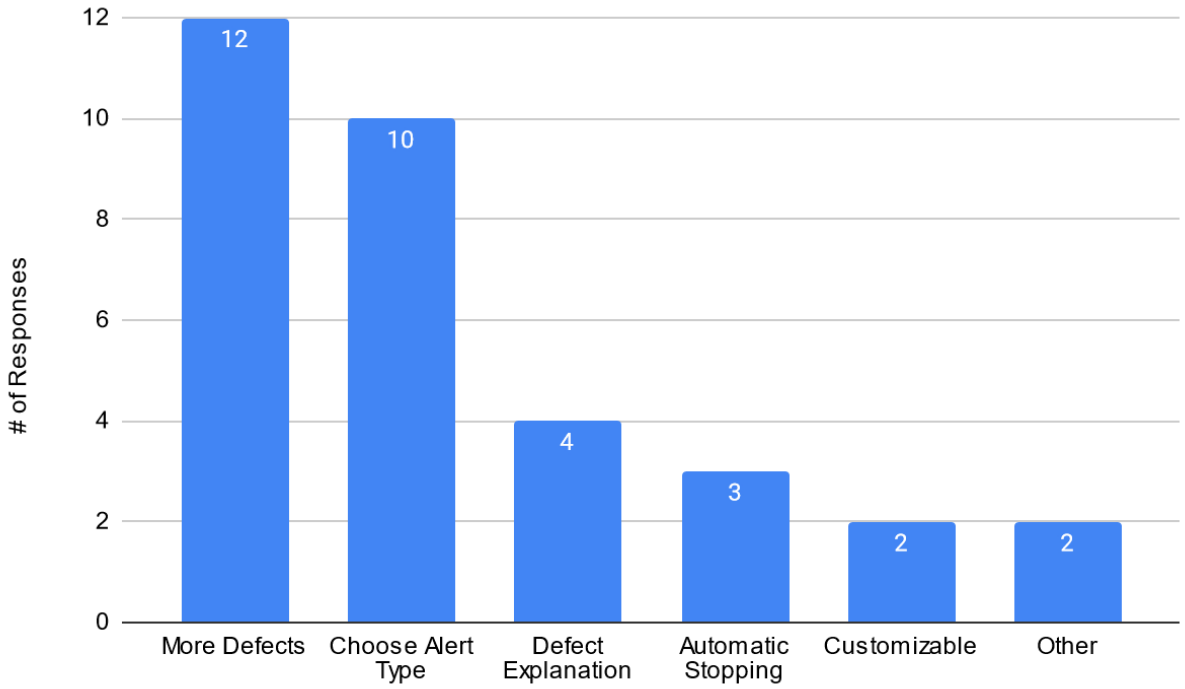


Figure 10.6: Future features that respondents liked the most

It was very difficult to find a concrete pattern of future features that participants did not like. The only one that stood out, with seven responses, was the inclusion of a graphical depiction of the algorithm in the user interface. Participants who mentioned this mostly thought that the average user would not find this feature useful. No other feature got more than two responses. A few participants were not sure how plausible it would be to detect defects based on a static image. Another few participants mentioned specific defects that they thought would not be useful to focus on. Yet another few thought that camera mounts might not be useful and users should develop their own mount for their own system. One respondent showed concern about the

complexity of the system if every future feature was added, explaining that it may be difficult for new users to navigate.

10.3.2 Expert Interviews Results

Due to time constraints, we were not able to interview faculty experts, and we were only able to interview one group of three students. However, the discussion from that focus group still provided valuable insight, and the results will be outlined here.

The three students in the focus group each had very different backgrounds of experience with 3D printers. Participant 1 (P1) mainly had experience with the printers in the makerspace print lab at WPI, since they were a student employee there. This meant P1 saw a lot of prints every day and had a lot of experience with starting and stopping prints due to various reasons. Participant 2 (P2) had experience from helping professors set up 3D printers, and therefore had more knowledge of the inner workings of 3D printers. Participant 3 (P3) had personal experience with creating and using 3D printed parts for robotics projects.

P1 and P2 both agreed that getting a snapshot of the print with the defect alert was one of the most compelling features. P1 explained that the makerspace print lab already has a system for detecting if something has gone wrong, but it does not send snapshots of the potential error. P1 said that the snapshots would be very helpful for knowing where the defect occurred and if it is big enough to need to stop it. P3 built off this response by saying that the ability to stop the print manually seemed very important. P3 cited the possibility for the software to make mistakes as a big reason for this.

When asked about features that thought were not useful, P1 noted that the system seemed fairly streamlined with no extraneous features. All the features that exist were there for a reason,

so none of them seemed unhelpful. P3 agreed by describing the system as “straightforward.” P2 agreed as well.

In terms of future features, P3 suggested the idea for a kit that users could buy to set up DaR3D on their own printer. This would be a way of making DaR3D both adaptable and user friendly. P1 chimed in at this point, saying that their favorite future features were the ones related to the adaptability of the system. P2 agreed, then mentioned how useful the library of camera fixtures would be for someone who is not experienced with CAD. On a different note, P1 liked the idea of receiving notifications on the phone, since P1 does not check emails frequently. P3 added that text notifications seemed great for learning about the defect as soon as possible.

All three participants seemed excited about the idea for a feature that could give descriptions of what the different types of defects are and explanations about why they occur. P3 specifically said that in the past they often encountered a defect then restarted the print without a second thought. P3 stated that if they knew what defect had occurred and why, then they would have been able to avoid a lot of repeated defects. This would have saved time and protected the printer from damage. P2 noted that defect identification was yet another feature that would be excellent for beginners, since people new to 3D printing do not always know what problems to look for when a print goes wrong.

After asking what defects they thought should be targeted by DaR3D next, it became clear that all three participants were lacking in knowledge about the names and causes of various defects. This indicated that a feature to explain defects and causes was quite important, if even experienced 3D printer users were unaware of some of the terminology and root causes.

Chapter 11: Conclusion

The goal of this chapter is to summarize the results and lessons learned from the project. Within this chapter, details of the results from testing the algorithm are shown. Following the results is the future work and goals for DaR3D, as well as broader impacts that this project might have. Lastly, there is a summary of the project experience for each group member detailing things that were learned and skills that were utilized for this project.

11.1 Results

The final DaR3D system was able to repeatedly detect slippage. It was tested with 48 final tests. It correctly identified slippage in 20 of the 24 tests that actually contained slippage and it correctly did not identify slippage in 23 of the 24 tests that did not contain any defect. As shown in Figure 11.1, DaR3D had a final accuracy of 89.6%, a false positive rate of 2.1% and a false negative rate of 8.3%. In terms of this project a successful test was when the system accurately detected when slippage had occurred. It is also considered a successful test when the system does not detect anything when no defect occurs. A false positive test is when the system detects slippage when it has not occurred. Lastly, a false negative test is when slippage occurs, and the system does not detect it. For this project false negatives are what we wanted to avoid, as this would result in a failed print to continue because the system misses it. A false positive would result in a good print needing to be checked, which causes no material waste and little time waste.

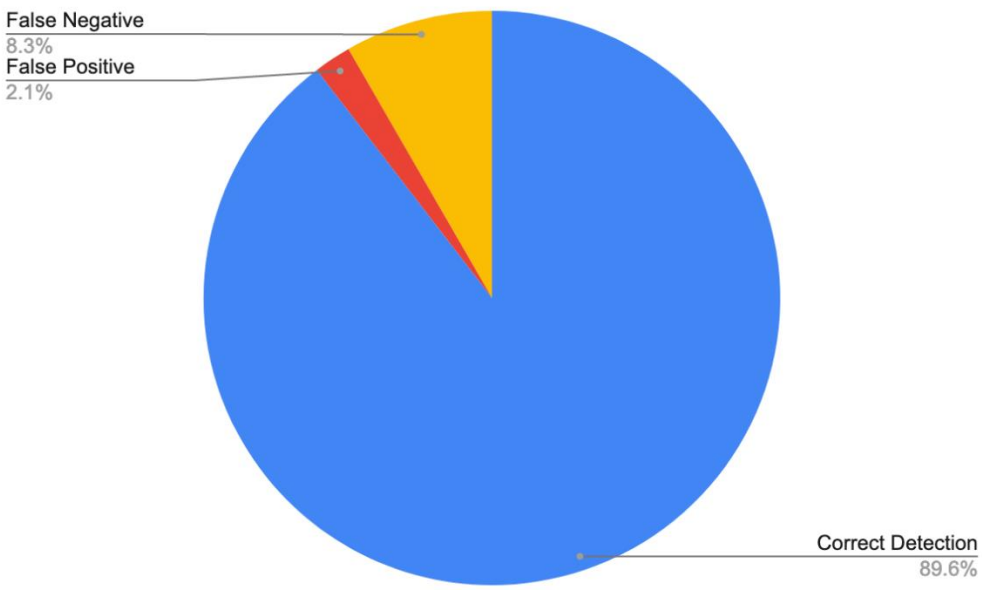


Figure 11.1: The final accuracy of DaR3D

The false positives were caused by a print without slippage being detected as slippage. This was because the part printed was a pyramid and because less filament was printed every layer, the algorithm detected that as slippage. Figure 11.2 shows the NRMSE plot of the false positive part. As seen, the score kept hitting the lower bound.

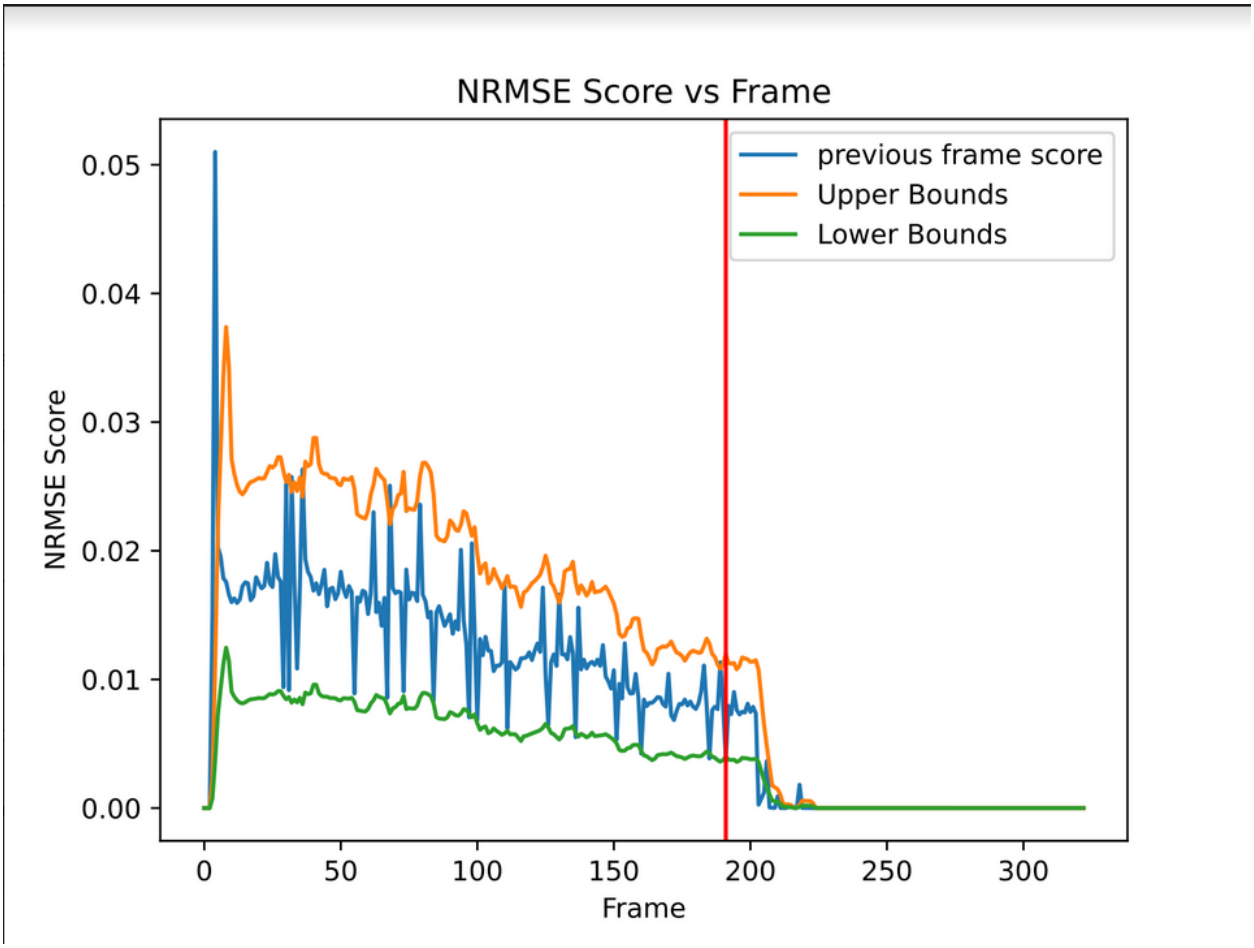


Figure 11.2: The NRMSE plot of the part that generated a false positive.

Based on our evaluation survey, we found that at least 76% of respondents would be likely or very likely to use DaR3D as it currently is. If the system was given additional functionality, that number would increase to 84%. The feature that respondents most commonly cited as their favorite feature was the ability to notify the user and stop the print remotely. A common sentiment among respondents was that they thought the hardware surrounding the printer was too elaborate and bulky. This was especially a concern among less experienced 3D printer users. Many respondents thought that the email system was awkward and were excited about the idea of having a text message option for notifications in the future. It was a common sentiment that the system would be more useful if it could detect more defects, with respondents

mentioning warping and stringing specifically. The more experienced respondents said that they looked forward to having customizable settings.

11.2 Future Work

This section details the groups desired future work of the DaR3D system. This is with the hope that the system can be built upon and improved to be able to:

- Detect and recognize more defects
- Be customizable for a wider range of printers
- Can notify the user via text message where the print can be stopped remotely

11.2.1 Additional Defects to be Detected

As referenced in Section 4.2, there were originally 5 targeted defects in the beginning of this project. For this project we were only able to successfully detect and recognize one defect with the system. We would like to see this system and the algorithms involved be improved and continued upon to be able to detect and recognize the four remaining targeted defects.

11.2.2 Remotely Stop Prints

At the current stage of the DaR3D system, the user can be notified that a defect has occurred. This is accomplished by emailing the user a screenshot of when the defect was detected, leaving the user in charge of going to the printer and manually stopping the print if something is wrong. An improvement that the team envisions is the ability to have an option with the defect notification to either stop the print or let it continue to run. We want this option to be either a text or email to the user.

11.2.3 Library of Camera Fixtures

Currently, there are two camera fixtures that are used for the two printers utilized in the system. Only having fixtures for the two printers limits the variability of the system, one future goal to help fix this would be to construct a library of camera fixtures. This library would allow a user to select the proper camera fixture for their primer and print it for themselves so they can utilize the system. These fixtures would initially be for the Logitech C920 webcam, but we hope it could be expanded to other cameras as well.

11.2.4 Octoprint Plug-in and Customization

There are several parameters used by DaR3D to determine when a defect is occurring. Currently, these parameters are hardcoded into the detection program. In the future, a plugin for Octoprint could be developed that would allow a user to manually set the parameters. This would allow the users to configure how strict or lenient the detection algorithm is, what email to send the alert to, and a few other parameters.

11.3 Broader Impacts

This section is going to reflect upon the broader impacts of our project with respect to engineering ethics as well as societal and global impacts. We will be relating our work to the six fundamental canons of the code of ethics for engineers. These six canons are from the National Society of Professional Engineers (NSPE, 2019). Engineers should:

1. Hold paramount the safety, health, and welfare of the public;
2. Perform services only in areas of their competence;
3. Issue public statements only in an objective and truthful manner;

4. Act for each employer or client as faithful agents or trustees;
5. Avoid deceptive acts;
6. Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.

In this chapter, canons two, three and four will be discussed as they are deemed relevant to this project. In addition, this chapter will present potential environmental impacts related to this project.

11.3.1 Engineering Ethics

The second canon listed above refers to an engineer only working in their areas of competence. We believe that our group satisfied this canon. Our team was composed of three students, two mechanical engineering majors and one computer science major. The scope of this project was to utilize the knowledge of the mechanical engineering students with respect to the manufacturing aspects of 3D printing and hardware design work. This knowledge was combined with computer science and coding to create software to automatically detect defects. Overall, the computer science major worked in his area of expertise by creating the code and algorithms for the software, while the mechanical engineering students worked on the hardware set up to aid the software development and analyze types of defects. In some cases, new things had to be learned outside of areas of knowledge, but this was deemed justified due to the experimental nature of this project.

The third canon requires that any public statements must be made in an objective and truthful manner. At the current stage of this project, there is nothing that would be directly stated to the public. However, any reporting that was done for this project was written and recorded in an objective and truthful manner. This honest recording of findings and results is critical for

future development of the system, without truth future teams and people who read about this work will not be able to improve it.

The fourth canon requires that an engineer is to be faithful to their current employer or client. For this project, our employer and client were our advisors on the project. One of the advisors was considered the client and had his ideas of what he wanted the system to be. We did our best to communicate with what he wanted and applied any input that was given throughout the project. Overall, both advisors were considered our employers. This meant that any direction that they pointed us in we followed, and all work done was to satisfy their requirements. We had to trust their judgment and work closely with them for the duration of the project.

11.3.2 Environmental Impact

The motivations for this system are to limit material and time waste during 3D printing operations. 3D printing materials that are utilized for this system are thermoplastics, these materials are not good for the environment and there is no widely used method to recycle the material after use. This system can allow consumers to use less material during failed prints. Additionally, this system can save time when a print fails, meaning that the run time for the printer can be cut down if a defect is to occur which can reduce the consumption of power that it takes to run a printer. These two effects from the proposed system and future improvements offer a positive environmental impact, especially if implemented in larger scale 3D printing operations.

11.3.3 Economic Factors

This project is aimed to reduce the amount of material waste and time that a failed print can cause. If this were to be implemented on a larger manufacturing scale then it has the potential to save money for the user. Money could be saved on material due to lower waste

amounts. Money could also be saved in power consumption that the 3D printer uses, if the print is stopped when the defect occurs the machine will no longer run and waste energy.

11.4 The Project Experience

This section describes what knowledge and skill each group member already had that was used during this project. It also describes what knowledge and skill they needed to learn. Note that this included technical, writing, organizational, communication and social skills.

The main role in this project for Ryan was to create the fixturing for the cameras and ring light that were used for testing using Solidworks. He brought this skill from previous classes and experiences where he learned about CAD and mechanical design skills. He also came into this project with FDM type 3D printing experience. Having this experience helped him hit the ground running and not need to learn a lot of the basics about 3D printers and how they worked. This experience also included knowledge of various printing defects and how they occur. Some of the main skills that Ryan learned through this experience were time management, communication and writing skills.

Madi worked primarily on developing the lighting and background environment for the system. She worked a lot with coordinating test prints, including compiling a library of test parts. She also created and administered the student survey and interviews for getting feedback on DaR3D. She had a small amount of experience with 3D printing before this project, and by the end she became adept at 3D printer troubleshooting. She also improved her communication, presentation, and information organization skills.

Mark’s main role in this project was to develop the software algorithm used for detecting defects. He also helped set up the software for OctoPrint and OctoLapse. He started this project with experience in 3D printing, software development, and the skills needed to set up the 3D

printers and computers used in the project. He improved on his skills in image processing and analysis throughout this project. He also learned more about the causes behind common types of 3D printer defects.

References

3dnatives.com. (2020, October 26). *How to avoid 3D printing problems*. 3Dnatives. Retrieved April 19, 2022, from https://www.3dnatives.com/en/how-to-avoid-3d-printing-problems-26102020/

All3DP. (2022, March 8). *3D printing troubleshooting all common problems*. All3DP. Retrieved April 19, 2022, from https://all3dp.com/1/common-3d-printing-problems-troubleshooting-3d-printer-issues/

All3DP. (2021, August 27). *The spaghetti detective: What is it & how to use it?* All3DP. Retrieved April 19, 2022, from https://all3dp.com/2/spaghetti-detective-octoprint-guide/

All3DP. (2022, March 7). *What is FDM 3D printing? – simply explained*. All3DP. Retrieved April 14, 2022, from https://all3dp.com/2/fused-deposition-modeling-fdm-3d-printing-simply-explained/

Anonymous. (2021, December 28). *Go to https://wsnv.nspe.org/resources/ethics/code-ethics and download NSPE code of Ethics.write down contexts, conditions, actions for each of codeexample for understanding: Code: 11.1.A (code no...* Unifolks. Retrieved April 12, 2022, from https://www.unifolks.com/questions/go-to-https-wsnvnspeorg-resources-ethics-code-ethics-and-download-nspe-code-of-361001.html

Bas, J., Preston, H., & Moiseyev, L. (2020). *Course.ece.cmu.edu*. Retrieved April 19, 2022, from https://course.ece.cmu.edu/~ece500/projects/s20-teame1/wp-content/uploads/sites/93/2020/03/Design_Report.pdf

Baumgartl, H., Tomas, J., Buettner, R., & Merkel, M. (2020, February 18). *A deep learning-based model for defect detection in laser-powder bed fusion using in-situ thermographic monitoring - progress in additive manufacturing*. SpringerLink. Retrieved April 19, 2022, from https://link.springer.com/article/10.1007/s40964-019-00108-3

Bisheh, M. N., Chang, S. I., & Lei, S. (2021, April 20). *A layer-by-layer quality monitoring framework for 3D printing*. Computers & Industrial Engineering. Retrieved April 19, 2022, from https://www.sciencedirect.com/science/article/pii/S0360835221002187?casa_token=ix4-hiVjTYsAAAAA%3AXZR5k01YXepAlBX7MFFGARaFHTQXBwd6S4xDWaNvTQm-w1sPHABIpXtNOUqT6PlyjyZgG3PSeA

Carmel. (2022, March 14). *Best photogrammetry software in 2022: The Ultimate Guide*. Sculpteo. Retrieved April 19, 2022, from https://www.sculpteo.com/en/3d-learning-hub/3d-printing-software/photogrammetry-software/

CasualPythoner. (1968, June 1). *Measure a length of a line segment in pixels from an image*. Stack Overflow. Retrieved April 19, 2022, from

https://stackoverflow.com/questions/63690237/measure-a-length-of-a-line-segment-in-pixels-from-an-image

Chris. (2015, September 1). *Maker musings*. Maker Musings. Retrieved April 19, 2022, from http://www.makermusings.com/2015/09/01/making-time-lapse-video-of-3d-prints/

Datacarpentry.org (n.d.). Image processing with python. Datacarpentry.org Retrieved April 19, 2022, from https://datacarpentry.org/image-processing/aio/index.html

Delli, U., & Chang, S. (2018, August 3). *Automated process monitoring in 3D printing using supervised machine learning*. Procedia Manufacturing. Retrieved April 19, 2022, from https://www.sciencedirect.com/science/article/pii/S2351978918307820

Fastowicz, J., & Okarma, K. (1970, January 1). *Texture based quality assessment of 3D prints for different lighting conditions*. SpringerLink. Retrieved April 19, 2022, from https://link.springer.com/chapter/10.1007/978-3-319-46418-3_2

Flask. (n.d.). *Welcome to flask*. Welcome to Flask - Flask Documentation (2.1). Retrieved April 27, 2022, from https://flask.palletsprojects.com/en/2.1.x/

GeeksforGeeks. (2021, December 7). *Realtime distance estimation using OpenCV - Python*. GeeksforGeeks. Retrieved April 19, 2022, from https://www.geeksforgeeks.org/realtime-distance-estimation-using-opencv-python/

Gobert, C., Reutzel, E. W., Petrich, J., Nassar, A. R., & Phoha, S. (2018, April 7). *Application of supervised machine learning for defect detection during metallic powder bed fusion additive manufacturing using high resolution imaging.* Additive Manufacturing. Retrieved April 19, 2022, from https://www.sciencedirect.com/science/article/pii/S2214860417302051?via%3Dihub

Goh, G. D., Sing, S. L., & Yeong, W. Y. (2020, July 16). *A review on machine learning in 3D printing: Applications, potential, and Challenges - Artificial Intelligence Review*. SpringerLink. Retrieved April 19, 2022, from https://link.springer.com/article/10.1007/s10462-020-09876-9?error=cookies_not_supported&code=66f50098-8c6f-4df9-8f33-9c0e7990f899

Hochgesang, B. (n.d.). *Octolapse*. OctoPrint Plugin Repository. Retrieved April 27, 2022, from https://plugins.octoprint.org/plugins/octolapse/

IEEE Xplore. (2017). *Development of Visual Inspection Systems for 3D printing*. IEEE Xplore. Retrieved April 19, 2022, from https://ieeexplore.ieee.org/abstract/document/7910849

Image.sc. (n.d.). *Great overlaid rigid rod segmentation ... - image.sc forum*. Image.sc. Retrieved April 20, 2022, from https://forum.image.sc/t/great-overlaid-rigid-rod-segmentation-challenge/3829

Khan, M. F., Alam, A., Siddiqui, M. A., Alam, M. S., Rafat, Y., Salik, N., & Al-Saidan, I. (2020, November 28). *Real-time defect detection in 3D printing using machine learning*.

Materials Today: Proceedings. Retrieved April 19, 2022, from https://www.sciencedirect.com/science/article/pii/S2214785320381037

Khandpur, M. S., Galati, M., Minetola, P., Marchiandi, G., Fontana, L., & Stiuso, V. (2021, June 1). *IOPscience*. IOP Conference Series: Materials Science and Engineering. Retrieved April 19, 2022, from https://iopscience.iop.org/article/10.1088/1757-899X/1136/1/012044/meta

Khanzadeh, M., Chowdhury, S., Marufuzzaman, M., Tschopp, M. A., & Bian, L. (2018, April 15). *Porosity prediction: Supervised-learning of thermal history for direct laser deposition*. Journal of Manufacturing Systems. Retrieved April 19, 2022, from https://www.sciencedirect.com/science/article/pii/S0278612518300402?via%3Dihub

Kirk, K., Lee, M., & Edwards, S. (2019). *Toward smart monitoring of additive manufacturing - DTIC*. Retrieved April 20, 2022, from https://apps.dtic.mil/sti/pdfs/AD1080523.pdf

Langeland, S. A. K. (2020, January 31). *Automatic error detection in 3D printing using Computer Vision*. Bergen Open Research Archive. Retrieved April 19, 2022, from https://bora.uib.no/bora-xmlui/handle/1956/21450

Mhaski, R., Chopade, P. B., & Dale, M. P. (2015). *Determination of ripeness and grading of tomato using image analysis on Raspberry Pi*. IEEE Xplore. Retrieved April 19, 2022, from https://ieeexplore.ieee.org/abstract/document/7437911?casa_token=0y7TyHpN5WcAAAAA%3ASKHd6-XyLM7IDLzow0rQTZuKAQkMwU_V44G0WD_J-lnARqpA4c-j_EyKkr3Lps6jSm24HBIWDQ

Mikolas Zuza (December 3. 2018), & Zuza, M. (2019, April 26). *Photogrammetry 2 - 3D scanning simpler, better than ever!* Original Prusa 3D Printers. Retrieved April 19, 2022, from https://blog.prusa3d.com/photogrammetry-2-3d-scanning-simpler-better-than-ever_29393/

Mwema, F. M., Akinlabi, E. T., & Fatoba, O. S. (2020, March 3). *Visual assessment of 3D printed elements: A practical quality assessment for home-made FDM products*. Materials Today: Proceedings. Retrieved April 19, 2022, from https://www.sciencedirect.com/science/article/pii/S2214785320310683?casa_token=GDFfay3jOj4AAAAA%3AMYmCgR_6kydm_kjI7VZc4L616HcuaSrbp_pdDStY8_yKsYM5S_mzWaup4ZocxKpYZR_yuo9cHg

N. G. Makagonov, E. M. Blinova and I. I. Bezukladnikov. (2017) *Development of visual inspection systems for 3D printing.* IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering. EIConRus

NSPE. (2019). *Code of ethics*. Code of Ethics | National Society of Professional Engineers. Retrieved April 26, 2022, from https://www.nspe.org/resources/ethics/code-ethics?gclid=CjwKCAjwsJ6TBhAIEiwAfl4TWPWR1tIu9YrcMm8Nne7dDq-DveudRwkE-VSjA54QxybXSr_hCyXeiRoC2zAQAvD_BwE

Nuchitprasitchai, S., Roggemann, M., & Pearce, J. M. (2017, June 17). *Factors effecting real-time optical monitoring of fused filament 3D printing - progress in additive manufacturing*. SpringerLink. Retrieved April 19, 2022, from https://link.springer.com/article/10.1007/s40964-017-0027-x

Octoprint. (n.d.). *Octoprint.org*. OctoPrint.org. Retrieved April 27, 2022, from

Okarma , K., & Fastowicz, J. (2016). *No-reference quality assessment of 3D prints based on the GLCM analysis*. IEEE Xplore. Retrieved April 19, 2022, from httg/abstract/document/7575237?casa_token=WnSXVWoRCjMAAAAA%3Au6slejmw9vZ8mRw-Oyn-icJJkEC4HcEyZmycYwn-svvQed7hJG-_mdGbXasG7a1a1jqMgThDBQ

Okaro, I. A., Jayasinghe, S., Sutcliffe, C., Black, K., Paoletti, P., & Green, P. L. (2019, February 11). *Automatic fault detection for laser powder-bed fusion using semi-supervised Machine Learning*. Additive Manufacturing. Retrieved April 19, 2022, from https://www.sciencedirect.com/science/article/pii/S2214860418306092?via%3Dihub

OpenCV. (2022, April 15). *Home*. OpenCV. Retrieved April 27, 2022, from https://opencv.org/

OpenCV. (n.d.). *Canny edge detection*. OpenCV. Retrieved April 27, 2022, from https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

Paraskevoudis, K., Karayannis, P., & Koumoulos, E. P. (2020, November 16). *Real-time 3D printing remote defect detection (stringing) with Computer Vision and artificial intelligence*. MDPI. Retrieved April 19, 2022, from https://www.mdpi.com/2227-9717/8/11/1464

Photogrammetry. (n.d.). *Photogrammetry*. Retrieved April 27, 2022, from https://www.photogrammetry.com/

Raihan, F., & Ce, W. (2017). *PCB defect detection using OPENCV with image subtraction method*. IEEE Xplore. Retrieved April 19, 2022, from https://ieeexplore.ieee.org/document/8273538/authors#authors

Raspberry Pi. (n.d.). *Buy A raspberry pi camera module 2*. Raspberry Pi. Retrieved April 27, 2022, from https://www.raspberrypi.com/products/camera-module-v2/

Raspberry Pi. (n.d.). *Buy A raspberry pi 4 model B*. Raspberry Pi. Retrieved April 27, 2022, from https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Simplify3D Software. (2019, June 21). *Print Quality Guide*. All-In-One 3D Printing Software. Retrieved April 19, 2022, from https://www.simplify3d.com/support/print-quality-troubleshooting/

Stack Overflow. (1956, July 1). *How can I quantify difference between two images?* Stack Overflow. Retrieved April 19, 2022, from https://stackoverflow.com/questions/189943/how-can-i-quantify-difference-between-two-images

Steve, G., Rosebrock, A., Dhinakaran, V., Ivan, Liebenberg, F., V, A., Zhang, T., Arungandhi, T., Yashwanth, A., shahi, A., Harry, Oneill, A., Sai, A., Tedi, Ankit, Iqbal, M., Clement, Hainsworth, K. Truyen, P. N. (2021, July 9). *Measuring size of objects in an image with opencv*. PyImageSearch. Retrieved April 19, 2022, from https://pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/

Straub, J. (2015, April 2). *Initial work on the characterization of Additive Manufacturing (3D printing) using software image analysis*. MDPI. Retrieved April 19, 2022, from https://www.mdpi.com/2075-1702/3/2/55/htm

Straub, J., & Kerlin, S. (2014, May 20). *Development of a large, low-cost, instant 3D scanner*. MDPI. Retrieved April 19, 2022, from https://www.mdpi.com/2227-7080/2/2/76/htm

Team, T. S. D. (n.d.). *Access anywhere - the spaghetti detective*. OctoPrint Plugin Repository. Retrieved April 19, 2022, from https://plugins.octoprint.org/plugins/thespaghettidetective/

Thomasnet. (2022). G-Codes Explained: An Introduction to Common G-Code Codes. Thomasnet Retrieved April 14, 2022, from https://www.thomasnet.com/articles/custom-manufacturing-fabricating/introduction-gcode/

Tutorialspoint.com (n.d.). Line detection in python with OpenCV? Tutorialspoint.com Retrieved April 19, 2022, from https://www.tutorialspoint.com/line-detection-in-python-with-opencv

Vora, H. D., & Sanyal, S. (2020, July 9). *A comprehensive review: Metrology in additive manufacturing and 3D printing technology - progress in additive manufacturing*. SpringerLink. Retrieved April 19, 2022, from https://link.springer.com/article/10.1007/s40964-020-00142-6

Ye, D., Hong, G. S., Zhang, Y., Zhu, K., & Fuh, J. Y. H. (2018, February 24). *Defect detection in selective laser melting technology by acoustic signals with deep belief networks - the International Journal of Advanced Manufacturing Technology*. SpringerLink. Retrieved April 19, 2022, from https://link.springer.com/article/10.1007/s00170-018-1728-0

Appendices

Appendix A: Evaluation Requirements

Script goals:

- *Why the system is needed*
    - Describe the problem
    - Explain goals for the solution
- *How the system works*
    - Hardware
        - Components
        - Basic setup
    - Software
        - Where to find it
    - Walkthrough
        - How the system works from the user's perspective
        - What the user should expect to see
- *Statistics/Measurements*
    - System accuracy
    - Cost
    - Dimensions
- *Ideas for future improvement*
    - Library of camera mounts
    - Additional defect detection
    - Wider range of compatibility for different systems
    - Octoprint plugin with user interface

Student Survey goals:

- *What information needs to be given to the students (potential users)?*
    - Basic information about the purpose of the survey and what the responses will be used for
    - System description (So they understand how it works)
        - Basic premise of how it works
        - Components of the system
        - Procedure for the user
        - Images, or a video of the system in use
    - Future work (So they get a sense of the possibilities)
        - More defects
        - More printers/cameras
        - Ease of use modifications
- *What information is needed from students (potential users)?*
    - Demographic information

        - Year
        - Major
        - 3D printer experience
            - Years of experience
            - Reasons for 3D printing (work, school, hobby, etc.)
            - Do you have your own 3D printer?
            - Types of filament used
            - Types of printers used
            - Types of parts printed (decorative, functional, prototype, etc.)
    - How likely they are to use the system
        - In its current state
        - With additional features
    - What they like about the system
        - Most compelling functionality
    - What they dislike about the system
        - Parts they think are counter-intuitive/confusing
        - Parts they think are not useful
    - What they think could be improved
        - Changes to current features
        - Additional features
        - (After reading the future work ideas) What they think would be the most compelling improvement/next step
    - Additional questions about the system
- *When should the information be supplied in the survey?*
    - After Basic instructions, reason for the survey
        - Demographic questions
    - After System description
        - How likely are you to use the system described above?
        - What do you like about the system?
        - What do you dislike about the system?
    - After Future work description
        - How likely are you to use the system with the features described above?
        - Favorite future features?
        - Least favorite future feature?
        - Recommendations?
        - Additional remarks/questions?
- *Question styles*
    - How likely?
        - Likert scale
    - Likes, dislikes, recommendations, additional remarks
        - Free response
    - Demographic
        - Multiple choice
            - School year
                - (freshman, sophomore, junior, senior, grad) [choose one]
            - Years of experience

            - (<1, 1-2, 2-3, 3-5, 5+) [choose one]
        - Reason for printing
            - (N/A, hobby, work, class, other:___) [choose one or more]
        - Own a printer?
            - (yes, no) [choose one]
        - Types of filament used
            - (N/A, PLA, ABS, TPU, PETg, other:___) [choose one or more]
    - Short response
        - Major (and second major if applicable)
        - Types of printers used (Make and model)
- *Getting students to take the survey*
    - Get student participants from Pradeep
        - 25 students from his course
        - 10-15 additional names
    - Procedure:
        - Contact all participating students with a quick explanation of what the survey is for.
            - Basic info, survey deadline
        - Include a video with the message that walks through how the system works and what it looks like.
        - Include a link to the survey in the message
    - Set a deadline for the survey to be completed by
        - April 20th (Allows time for data analysis before the report is due)

Expert Interview goals:

- *What information needs to be gathered*
    - Makerspace affiliated:
        - Approximately how many prints they see every day
        - Approximately how many defects they see every day
        - What defects occur most often
        - What defects waste the most time, filament
        - If they see value in a real-time defect detection and notification system
            - Based on what it currently does
            - Based on what it could do
        - What they view as the most valuable features of the system
        - What they see as not being very useful
        - Recommendations
            - Based on personal experience
            - Based on what would work best for the prototyping lab
    - Not makerspace affiliated:
        - Experience with 3D printing
            - What applications?
        - What they view as the most valuable features of the system
        - What they see as not being very useful

        - Recommendations based on personal experience
- *Who to interview*
    - Makerspace:
        - Mitra Anand: Makerspace Advanced Technology & Prototyping Specialist, Innovation & Entrepreneurship
            - Experienced with 3D printers
            - Experienced with makerspace's 3DprinterOS system
            - Would know if it is plausible to integrate XXXX into the prototyping lab
        - Erika Stults: Interim Director, Makerspace
            - Experienced with rapid prototyping, 3D printers
            - Interested in optimization
        - Makerspace employees: Students
            - They work with the makerspace printers every day.
            - They have personal experience with individual printers, and a system of printers
            - They would know about common defects
            - They would know what might make their jobs easier
    - Other:
        - John Sullivan: Mechanical Engineering Professor
            - Filament recycling MQP
            - Knows about 3D printing
        - Curtis Abel: Executive Director of Innovation and Entrepreneurship, ad interim
            - Professor in IE who teaches 3D printing
        - Students who own 3D printers
            - Personal experience with 3D printers
- *Procedure*
    - Schedule interviews
        - About 30-60 mins each
        - 1 on 1 with experts
        - focus groups of 3 - 4 students
        - For in person, book rooms and/or tech suites ahead of time
    - Ask permission to record audio
    - Show the demo video
        - Over Zoom: share screen or send a file or link to the interviewee
        - In-person: Show the video on a laptop or projector screen
    - Ask questions as outlined above

Data analysis:

- *For student surveys*
    - Likert scale:
        - Mode
        - Median
    - Free response

        - Keyword analysis
            - Choose several important keywords for each question, search through responses to find the keywords, record the data as the percentage of responses that had each keyword (or associated group of keywords)
    - Demographic
        - Cross examine data with keyword and likert scale analysis
        - Sort responses based on year, major, years of experience, etc. and perform analysis as described above on only the responses from certain groups
        - Compare results across demographics, determine if there are any worthwhile differences (might not have enough time or responses to do this)
- *For expert interviews*
    - Data is recorded in audio or video format
    - Take note of response patterns, e.g. commonly liked/disliked features, common recommendations or questions, etc.
        - Perform qualitative analysis, i.e. a few sentences explaining general trends, common thought processes, or important outliers (not enough responses for quantitative analysis)

Appendix B: Video Script

Intro: (Problem Statement, Project Goal)

Low cost and open-type Fused Deposition Modeling 3D printers are commonly used for prototyping projects, but they can also produce various defects. Parts slipping off the print bed, filament failing to extrude, warping, and more are common defects that can result in losses of time and material. To minimize these losses, we developed DaR3D (pronounced “dread”), a 3D printer monitoring system that can detect and recognize defects in 3D printed parts and notify the user as soon as a defect is detected so they can cancel the print.

Demo: (System Walkthrough)

DaR3D uses a simple hardware setup around the printer to aid an algorithm in monitoring parts as they are printing. The hardware includes a 3D printed camera fixture which holds a webcam and a ring light. It is customizable to different cameras and 3D printers. Currently, there are versions of the fixture for fitting a Logitech HD webcam to either a MonoPrice or RepRap printer, but the user is free to create a camera setup that works for their printer as long as the camera stays in a consistent position throughout the print. Around the printer is a 3D printer enclosure, which blocks out all external light and has a strictly controlled internal lighting setup to consistently illuminate the part and the print bed. Alternatively, the user can bypass the need for an enclosure if their printer is in a location where external lighting is not an issue. DaR3D functions through a third party software which requires either a Raspberry Pi or other form of computer connected to the printer to run. DaR3D’s algorithm works by receiving pictures of the print at each layer, and comparing consecutive images. If there is too much of a difference between two images, then the algorithm reports an error. The system can correctly identify

slippage in prints with 89.6% accuracy, with a false positive rate of 2.1% and a false negative rate of 8.3%.

Once the user sets up DaR3D, they can start a print and simply walk away. The algorithm will automatically run and notify the user if something has gone wrong. Currently, DaR3D can detect when the part has slipped from its original position on the print bed. If this were to occur, the user would receive an email with information, such as a picture of the part in its current state, and the time that the defect occurred. The user can then decide if they want to cancel the print through the web-based printer application, Octoprint.

Future Plans: (Possibilities, work to be done)

We envision that in the future DaR3D will be able to detect and recognize more defects, like filament runout, stringing, and warping, among others. It will also be highly customizable to a wide variety of 3D printers. This would include a library of camera fixtures to fit various cameras to different printers; an option to use integrated cameras for printers that have them; and compatibility with more printer monitoring applications such as 3DPrinterOS.

Additionally, in order to make the system more user friendly, there would be an interface where you can choose how you want to be notified about potential defects, like email or text message, and whether or not you want the print to stop automatically when a defect is detected. The interface would also display the most recent snapshot of the part, and a graph depicting the algorithm's interpretation of the snapshot. For people who regularly use 3D printed parts, but don't necessarily interact with 3D printers first hand, there could also be a feature that tells the user what defects a part may have based on a static image of the completed part.

Appendix C: Student Survey

This survey is for the evaluation of DaR3D, a 3D print monitoring system that can detect defects and notify the user in real time. This system was developed by Madison Eisenhour (RBE,ME), Mark Forte (CS), and Ryan Malkowski (ME), and advised by Professors David C. Brown (CS) and Pradeep Radhakrishnan (ME)

All responses to this survey will be confidential and shared only with the DaR3D development team. If you have any questions or concerns, please contact gr-dar3d@wpi.edu. A report about the system and the survey results will be available once the development is complete. You will be able to request the results after May 2nd, 2022.

1. What is your full name?
    0. (Short Response)
2. What is your academic year?
    0. Freshman
    1. Sophomore
    2. Junior
    3. Senior
    4. Graduate
3. What is your major? (Include second major if applicable)
    0. (Short Response)
4. How many years of experience do you have with 3D printers?
    0. <1 year
    1. 1-2 years
    2. 2-3 years
    3. 3-5 years
    4. 5+ years
5. How many years of experience do you have with using 3D printed objects?
    0. <1 year
    1. 1-2 years
    2. 2-3 years
    3. 3-5 years
    4. 5+ years
6. What kinds of 3D printers have you used before? (Make and Model)
    0. (Short response)
7. Do you personally own a 3D printer?
    0. Yes
    1. No
8. What types of filaments are you familiar with? (Choose all that apply)
    0. None
    1. PLA
    2. ABS
    3. TPU

4. PETg
5. Other: ____

9. For what purposes do you 3D print? (Choose all that apply)
    0. Hobby
    1. Class
    2. Work
    3. Other: ____
10. What kinds of things do you typically 3D print? (Choose all that apply)
    0. Prototypes
    1. Functional parts
    2. Decorative objects
    3. Other: ____

Please read the following paragraphs about the system's design and answer the questions.

DaR3D uses a simple hardware setup around the printer to aid an algorithm in monitoring parts as they are printing. The hardware includes a 3D printed camera fixture which holds a webcam and a ring light. It is customizable to different cameras and 3D printers. Currently, there are versions of the fixture for fitting a Logitech HD webcam to either a MonoPrice or RepRap printer, but the user is free to create a camera setup that works for their printer as long as the camera stays in a consistent position throughout the print. Around the printer is a 3D printer enclosure, which blocks out all external light and has a strictly controlled internal lighting setup to consistently illuminate the part and the print bed. Alternatively, the user can bypass the need for an enclosure if their printer is in a location where external lighting is not an issue. DaR3D functions through a third party software which requires either a Raspberry Pi or other form of computer connected to the printer to run. DaR3D's algorithm works by receiving pictures of the print at each layer, and comparing consecutive images. If there is too much of a difference between two images, then the algorithm reports an error. The system can correctly identify slippage in prints with 89.6% accuracy, with a false positive rate of 2.1% and a false negative rate of 8.3%.

Once the user sets up DaR3D, they can start a print and simply walk away. The algorithm will automatically run and notify the user if something has gone wrong. Currently, DaR3D can detect when the part has slipped from its original position on the print bed. If this were to occur, the user would receive an email with information, such as a picture of the part in its current state, and the time that the defect occurred. The user can then decide if they want to cancel the print through the web-based printer application, Octoprint.

1. How likely would you be to use the DaR3D system as described above?
    0. Very Likely
    1. Likely
    2. Neutral
    3. Unlikely

    4. Very Unlikely
2. What features of DaR3D seem most useful to you?
    0. (Free response)
3. What features of DaR3D seem least useful to you?
    0. (Free response)
4. Are there any features that you do not understand?
    0. (Free response)

Please read the following paragraphs about potential future features and answer the questions:

We envision that in the future DaR3D will be able to detect and recognize more defects, like filament runout, stringing, and warping, among others. It will also be highly customizable to a wide variety of 3D printers. This would include a library of camera fixtures to fit various cameras to different printers; an option to use integrated cameras for printers that have them; and compatibility with more printer monitoring applications such as 3DPrinterOS.

Additionally, in order to make the system more user friendly, there would be an interface where you can choose how you want to be notified about potential defects (email, text, etc.), and whether or not you want the print to stop automatically when a defect is detected. The interface would also display the most recent snapshot of the part, and a graph depicting the algorithm's interpretation of the snapshot. For people who regularly use 3D printed parts, but don't necessarily interact with 3D printers first hand, there could also be a feature that tells the user what defects a part may have based on a static image of the completed part.

1. How likely would you be to use the DaR3D system with one or more of the additional features described above?
    0. Very Likely
    1. Likely
    2. Neutral
    3. Unlikely
    4. Very Unlikely
2. Which future feature(s) do you like the most?
    0. (Free response)
3. Which future feature(s) do you like the least?
    0. (Free response)
4. Do you have any recommendations for changing the system?
    0. (Free response)
5. Additional comments/questions?
    0. (Free response)

Appendix D: Expert Interviews

Procedure:

- Ask if they consent to being recorded (expert interviews will be conducted over Zoom)
- Show prepared video
- Allow interviewees to ask questions directly after the video to make sure they understand it sufficiently.
- Ask them questions as laid out below
- Allow for any additional comments or closing remarks that people may have if there is time. Interviews will last for 30 minutes each.

Makerspace Affiliated:

1. How often do defective prints occur in the prototyping lab (number or percentage)?
2. What are the most common types of defects you see in the prototyping lab?
3. Do you think a system like this would be suitable for use on the prototyping lab printers?
4. Do you think DaR3D would save time and/or money for the makerspace?
5. Would such a system narrow the tolerance for acceptable parts?
6. What are some features that you think would make DaR3D worthwhile for the prototyping lab? Are there any changes that you think are necessary in order to integrate DaR3D with the prototyping lab printers?
7. Are there any features that you see as not being useful to the prototyping lab?
8. Do you have any other thoughts or questions about DaR3D? (Either based on personal experience or experience in the prototyping lab)

Other Faculty Experts:

For these interviews, participants will have fairly different experiences and potential applications of DaR3D. Additional questions will be asked based on what they say during the interview.

1. What is your experience with 3D printing, especially relating to defective parts?
2. Are there any features of DaR3D that stand out to you as being useful?
3. Are there any features that you see as not being very useful to you?
4. Do you have any recommendations about the system based on your personal experiences with 3D printing?

Student Focus Groups (these will be in-person):

Focus groups will be instructed to be interactive with each others' responses; adding ideas and agreeing or disagreeing with their peers.

1. What is your experience with 3D printing, especially relating to defective parts?
2. Are there any features of DaR3D that stand out to you as being useful?
3. Are there any features that you see as not being very useful to you?
4. Do you have any recommendations about the system based on your personal experiences with 3D printing?

Appendix E: Octolapse Camera Settings

General Settings

Brightness	128
Contrast	128
Saturation	128
Sharpness	128
JPEG quality	100

Focus

Focus, Auto	Unselected
Focus absolute	70

White Balance

White Balance Temperature, Auto	Checked
White Balance Temperature	AUTO

Exposure, Gain, Auto-Priority

Backlight Compensation	Unchecked
Exposure, Auto	Manual Mode
Exposure (Absolute)	238
Gain	68
Exposure, Auto Priority	Checked

Pan, Tilt, and Zoom

Pan (Absolute)	0
Tilt (Absolute)	0
Zoom (Absolute)	100

Misc

Power Line Frequency	60 Hz

Appendix F: Library of Test Parts

The full collection of test parts can be found at
https://drive.google.com/drive/folders/1TY4cRVuQBvEpocuv1pqdUw88k--FCdye?usp=sharing

Also included in the directory is the NRMSE plot generated by running DaR3D on the part.

Appendix G: Camera Selection Criteria

	A	B	C	D	E	F	G	H	I
1	Camera	Link to Specs	Focal Length [mm]	hFOV [deg]	vFOV [deg]	dFOV [deg]	vResolution [pixels]	hResolution [pixels]	Pixel Size [micrometers]
2	C920s HD Pro Webcam	https://support.logi.	3.67	70.42	43.3	78	1080	1920	2.697770427
3	TECKNET webcam	https://www.amazo	?	90			1080	1920	
4	BAVEEL webcam	https://www.amazo	1 - 40	110			1080	1920	
5	RasPi CM v1	https://www.raspbe	3.6	55.14912792	41.66920089	65.73884584	1957.142857	2685.714286	1.4
6	RasPi CM v2	https://www.raspbe	3.04	62.36994932	48.8310967	74.22076438	2464.285714	3285.714286	1.12
7	RasPi HQ Camera	https://www.raspbe	depends on lens	#VALUE!	#VALUE!	#VALUE!	3040	4056.129032	1.55
8	RunCam Nano 2	https://www.rcplane	1.8	170	160.3864136	171.0731594	494	976	42.16002899
9	Arducam	https://www.arduca	4.04	62	51.70910543	77.81160259	3496	4656	1.12

*RunCam Nano 2 is highlighted red because its field of view (FOV) was too wide for our applications.

Appendix H: Student Survey Results

Demographic/Experience Questions

Participant #	What is your academic year?	What is your major? (Include second major if applicable)	How many years of experience do you have with 3D printers?	How many years of experience do you have with using 3D printed objects?	What kinds of 3D printers have you used before? (Make and Model)	Do you personally own a 3D printer?	What types of filaments are you familiar with? (Choose all that apply)	For what purposes do you 3D print? (Choose all that apply)	What kinds of things do you typically 3D print? (Choose all that apply)
1	Senior	ME	5+	5+	Ultimaker, Original Prusa i3	No	PLA, ABS, TPU	Work, Class, robotics team in high school	Prototypes, Functional parts
2	Senior	ME	5+	3-5	I've used one industrial type 3D printer that I cannot recall, however, I've also used the Prusa printers, makerbots, and whatever is available in the innovation studio as well.	No	PLA, ABS	Hobby, Class	Prototypes, Functional parts, Decorative objects
3	Junior	ME	3-5	3-5	Prusa i3 mk3, Markforged Mark Two, Markforged Onyx Pro	No	PLA, ABS, Chopped Nylon/NylonX/Onyx	Work, Class	Prototypes, Functional parts, Decorative objects
4	Senior	ME	3-5	3-5	QiDi X-Plus, Formlabs 3+, Creality Ender 3 V2	No	PLA, ABS	Hobby, Work	Prototypes, Decorative objects
5	Junior	ME	3-5	3-5	Creality ender 5, ender 5 pro	Yes	PLA	Hobby, Class	Prototypes, Functional parts, Decorative objects
6	Senior	ME	2-3	2-3	Ender 5 Pro	Yes	PLA	Hobby, Class	Prototypes, Functional parts, Decorative objects
7	Junior	ME	2-3	2-3	Creality ender 3 pro, prusa mini	Yes	PLA, ABS, TPU, PETg	Hobby, Class	Prototypes, Functional parts,

									Decorative objects
8	Senior	ME	2-3	2-3	Creality cr-10s V2 pro, ultimaker, TAZ	Yes	PLA, ABS, TPU	Hobby, Class	Prototypes
9	Senior	ME, BME	2-3	2-3	Ender 3	Yes	PLA, ABS, TPU, PETg	Hobby, Class	Prototypes, Functional parts, Decorative objects
10	Senior	ME	2-3	2-3	Ultimaker & luzbot taz	No	PLA, ABS	Hobby, Class	Prototypes, Functional parts, Decorative objects
11	Senior	ME, BME	1-2	2-3	Creality 5 Pro	No	PLA, ABS	Class	Prototypes, Functional parts
12	Senior	ME	1-2	2-3	Unknown, Ive used the ones in the innovation studio and a friend's	No	PLA, ABS	Hobby, Class	Prototypes, Functional parts, Decorative objects
13	Senior	ME, BME	1-2	1-2	Creality Ender 3 Pro	Yes	PLA	Hobby, Class	Prototypes, Functional parts, Decorative objects
14	Junior	ME	1-2	1-2	taz and ultimaker 3	No	PLA	Class	Prototypes, Functional parts
15	Junior	ME	1-2	1-2	Ultimaker	No	PLA, ABS	Hobby, Class	Prototypes, Decorative objects
16	Junior	ME	1-2	1-2	Ender pro	No	PLA	Hobby, Class	Prototypes, Functional parts
17	Senior	RBE, AE	1-2	1-2	Creality Ender 3 Pro, Lulzbot Taz	No	PLA, ABS, TPU	Hobby, Class, MQP	Prototypes, Functional parts
18	Junior	ME	1-2	1-2	Lulzbot Taz	No	PLA	Hobby, Class	Prototypes, Functional parts, Decorative objects
19	Junior	ME	1-2	1-2	Ultmaker	No	PLA	Hobby, Class	Prototypes, Decorative objects
20	Junior	ME, RBE	<1	2-3	Prusa MK3S+	Yes	PLA, ABS	Hobby, Class	Prototypes, Functional parts

21	Senior	ME	<1	2-3	Lulzbot TAZ , Ultimaker, Ender(This was over 4 years ago), MakerBot (This was over 4 years ago)	No	ABS	Work, Class	Prototypes, Functional parts, Decorative objects
22	Senior	ME	<1	<1	Taz Lulzbot	No	PLA	Class	Functional parts
23	Junior	ME, BME	<1	<1	unsure	No	PLA	Class	Prototypes
24	Junior	ME	<1	<1	none	No	None	I dont	none
25	Senior	BME	<1	<1	I'm not sure	No	PLA	Class	Prototypes

Questions about Current DaR3D

Participant #	How likely would you be to use the DaR3D system as described above?	What features of DaR3D seem most useful to you?	What features of DaR3D seem least useful to you?	Are there any features that you do not understand?
1	4	The user being notified automatically if something has gone wrong, so they can stop the print before wasting additional time, money, and filament.	It does not seem like an easy set up for the user and could be confusing to set up for someone new to 3D printing.	No
2	4	Getting a notification when my print fails would save me so much time when it came to class and work applications of 3D printing. There were many times I would leave long prints going while I would be away for hours, unaware if anything was going wrong.	The 3D printed mount doesn't seem too useful. Like you said, it would be difficult to design a mount that would work well for many printers, so I believe it would be best to just let the end user create their own camera mount and simply use the software you create. I think it would be more effective for your group to focus on detecting more than just slippage.	No
3	3	The fact that the algorithm can notify the user if a part has slipped does seem useful to me. I actually think that by extension of this, the most useful thing is the timestamp/picture of the part's current state. While many parts "slipping" on printer beds I've had were due to poor application of adhesive/not using a heated bed on a printer, being	I'm not sure an email is the ideal method for delivering this information, especially if we would like to use this in a work setting.	I'm not sure I understand what "too much of a difference" means in this context. For example, imagine I wanted to use this with a dual-extruder headed printer and print alternating colors for layers

		able to identify a problematic section could be useful for part redesigns (maybe a lower infill is necessary or making a part that doesn't use a raft).		(maybe for just aesthetic reasons). Would the algorithm be able to compensate for this?
4	3	The ability for the program to notify the user is very useful, especially for large parts. If I were to start a print that is 4+ hours I would want to know when it fails ASAP so I can stop the print to avoid wasted filament and time. I think an adaptive function to stop, remove, and start a new print could be an amazing feature. That being said very difficult to accomplish in a non industrial setting.	The sheer size of the fixture seems quite large. If I were operating a small scale 3D print buisness in my own house/apartment it would be hard to have multiple of these setups in such a tight space.	It is very straight forward. I like that it compares images between layers, but could it also compare images of the render to the real part?
5	5	Being able to notify the user of defects while they're away	The fact that it's via email. Personally, my email application is not the greatest at notifying me of new emails	no
6	5	I like that it would be able to automatically determine if the print has slipped and would not be able to recover, as this would allow anyone to leave the printer unattended for longer periods of time. I also like how it can allow the user to remotely stop the print.	The design seems to be effective, however it would be nice if the enclosure was not necessary, so the print could be physically observed without disrupting the program.	N/A
7	3	The ability to be compatible to many different types of printers and cameras. Being able to sense if there is a defect or problem in a print.	If constantly printing, the amount of storage of images after a 300 layered print	Isn't this almost basically like what octoprint already does? Except for the notifying part. Just an online monitoring device that can be looked at and controlled anywhere?
8	5	Stopping the print when it goes wrong	N/A	N/A
9	5	The alert system because I hate when an object is printing and I check on it 8 hours later to find out it failed on hour 2 etc.	The automatic cancellation. If the user gets notified of the problem, then there isn't a need to automatically stop the print, especially with the current false negative rates. If I have a 48 hour print that is automatically stopped at hour 46 because of a false positive then I would be	Nah I think I got it all

			very upset because that's a lot of time and filament wasted. But if I had the option to cancel it then I might say eh its good enough and continue printing especially if it is due to a warp etc.	
10	3	Being able to remotely stop the print	Only detects slippage right now	Why is the false neg. So high? Why do I need an enclosure
11	5	Detect slipping, so time isn't lost when printing large parts	Not sure	No
12	5	The slippage detection and email when an error occurs	The fact that it runs on a raspberry pi and have to be connected to the computer	No
13	4	The ability to remotely notify the user when they are not in the same physical space as the printer.	Notification via email as some emails get lost in my inbox.	No
14	4	stopping when there is a fail	the high false negative and having to get the ring light and other stuff for it	what about 3d printers where the floor moves up and down
15	4	I could save filament using DaR3D which is nice.	I wish if someone could improve the actual performance, actually reduce the accidents.	Not really.
16	3	Print stops and alerts you when there is an error	The webcam holder	N/A
17	5	Stringing detection - this issue had happened many times SMS Notification with option to stop the print - This will save time and reduce the waste produced	None. This is a very simple but effective system.	None.
18	5	Being able to recognize differences between layers	Having to put the whole system inside an enclosure	It all makes sense and is a very great idea.
19	4	Deflect detection	Email user when error occur, instead of shutting down print	No
20	4	The monitoring when you are away	I think the emailing is pretty useless if I don't look at my email fast enough. It sounds like if I don't do anything, the print will just keep going with the errors.	This is not really something I don't understand but if the print was to make multiple objects, I 'm just unsure if it would be able to see every object from its angle.
21	4	The email notification with picture of the print.	Having to enclose the printer in a darking box(I understand that keeps light level constant) but I enjoy glancing over every once in	Does the system require you to use Octoprint? Is the raspberry pi

			a while to check on the progress, also the box seems like a hassle to take the parts out of.	an additional microcontroller or can you use the ones built into the printer?
22	4	Knowing if the part is printing incorrectly without having to be there watching it print.	Requiring a lighting enclosure around the printer.	No.
23	5	being able to walk away/ work on something else while printing	camera point in only one direction. I understand that this is good for slipping but would this miss defects out of the cameras view/ on the side of the print not facing the camera?	can dread stop the print when there is an issue or can you only manually stop it?
24	5	As someone who does not know how to 3D print or what to look for if something is going wrong it seems nice that the DaR3D can tell if something is going wrong.	I feel that it would be more useful if the program fully stopped the 3D print in progress and the email was sent so that you had to start the program to continue. This way maybe the user could correct the print and continue.	I don't understand how slippage occurs but that is probably because I am not familiar with 3D printers
25	2	I think the 3D printed camera fixture is very interesting.	I think all features seem useful	I'm not sure about how it prints by using pictures of each layer and comparing.

Questions about Future Features of DaR3D

Participant #	How likely would you be to use the DaR3D system with one or more of the additional features described above?	Which future feature(s) do you like the most?	Which future feature(s) do you like the least?	Do you have any recommendations for changing the system?	Additional comments/questions?
1	4	Text notification with photo of potential defects, being able to text/email back to stop the print would be very convenient	N/A	Have a clear instruction/set up guide as people may not be comfortable setting up the wiring, camera, and raspberry pi on their own	
2	5	Personally, and for other 3D printer users as well in the consumer space, the	A different UI, different communication lines, etc. Basically	I know the shroud is required for lighting, however, I think that is	

		ability of detecting many different types of errors accurately and notifying them in any way would be perfect.	all these creature comforts don't really seem necessary for consumers, however, in the professional space, they will probably matter more.	probably the largest downside of your project. Out of the dozens of people I know with 3D printers, only one of them has a shroud and they barely ever leave it closed. A lot of people don't have space for something that large, so if there is some sort of work around that will make the shroud optional, I believe that would be very beneficial.	
3	3	I think that the customizable features within cameras and different monitoring software is the most promising feature. I think it's fair to assume that your current scope includes the most common 3d printers but having a wider variety of options makes your system more usable by someone with a variety of different printers at their disposal so they can integrate everything together.	The idea to try and tell the user what defects a part may have based on the completed part does sound interesting but a little bit of a stretch to me. I think it has the most limited use case of any of the future features proposed while also sounding the most difficult to implement (there are such a huge variety of edge cases for 3d-printing failures that are virtually indistinguishable) but I'd be happy to be proven wrong.	If we have a large production house and a high volume of prints running, I think using an alternative method for delivering this "warning" such as a webbapp, might be ideal and can be more scalable.	This is a really neat concept for a project and I'll be curious to hear how it ends up!
4	4	the custom fixtures adn the ability to specify mode of contact to the user. I would be more receptive to an emergency text over an email.	I think the feature that classifies the defect could be overkill. I feel that the person that this product is marketed for would be able to know the defect on site and knowing the defect instantly	maybe smaller housing.	for open-faced printers could an pneumatic system be used to dislodge or remove the part to automatically start a new print. This could be used to run through the night to create as many of the parts

			wouldn't make that big of a deal of difference when I am going to stop the print and start over anyway. It could be useful, but overkill.		as possible without human intervention.
5	5	Being to receive texts and descriptions of specific defects	The graph seems a little unnecessary, but more data cant hurt	I'm not sure, right now the setup seems very simple and the team is working towards helpful, substantial improvements	I actually saw a kickstarter with this kind of concept, with a fully integrated camera to warn of defects, only this one was comparing the print to the actual base 3d model. It was also very expensive, so this seems like a very nice, semi-diy version that you can use to incorporate your older printers (and not spend like 1k on a fancy new printer)
6	5	I like the idea of warping detection the most, as that can ruin the usefulness of a print and generally results in more problems than stringing or other issues.	I personally do not use other monitoring applications, so that would not be a priority for me.	If it is possible to monitor filament before it runs out and provide warning, that would be very helpful for some of the lower-tech printers that do not have that feature built in.	
7	4	Having it be texted instead of email. Detection of stringing and warping.	Although probably cool to look at, I wouldnt see myself using or paying attentio to the graph all that much	At least for the warping idea, those type of printing errors happen overtime and some times cant be detectable until 5 layers passed. So that type of detection might be different cause it'll take time to detect.	Since youre using octoprint, will this data be incorperated into the dashboard or will there be a seperate interface that you have to have open along side octoprint?

8	5	Notification method selection	Qualification of defect	I think defect classification is only useful if the software were to offer suggestions on how to fix the defect it characterizes.	N/A
9	5	Being notified by text because I check those most often.	Potentially the graph of the interpretation because I may not need that if I see the image of the part.	Having real time monitoring where I can go see what the latest image is despite if there is a problem or not just to see how far along it is etc.	This is awesome and it would have saved me hours of failed prints and material waste.
10	5	Detect More types of errors, text instead of email	Automatically stopping a print w/out asking	An app on your phone	
11	5	Detect warping	Graph depicting the algorithm's interpretation of the snapshot	No	No
12	5	Filament running out	Warping	no	none
13	5	The ability to detect which defect occurred and tell the user	The automatic stopping of the print.	Customize the system to have the ability to stop the print based on the defect that occurred, as warping can affect non-vital functions of the part as opposed to filament failing to extrude which will require a reprint	
14	4	sends a photo of the problem so the user gets to ultimately decide	im worried that the software will be very complicated and not very user friendly	no	
15	4	I really like the choice that how to be notified.	Not really, I feel this is going to work.	Please notify all 3D printer company about this feature. I really wish if able to, please make it compatible with other software.	Thank you very much for doing that.

16	3	Stopping automatically on defected print to prevent material Easter	I don’t see the use of camera mounts		
17	5	Stringing and Warping detection	filament runout	In addition to comparing the consecutive frames, comparison between the CAD model and the 3d printing piece would help with increasing accuracy of the system. Have you thought of using two cameras? Sometimes defect might be on the blind spot the camera being used. Two cameras should give 360 view of the print.	Having a need of using an enclosure with the 3D printer would keep some people from using DaR3D. Is there a possibility of removing the need of an enclosure? I know that you can customize your vision pipeline to deal with the different level of brightness and still get the same pictures. For the survey, I would have loved to have an optional section asking me the reasons for liking/disliking a certain feature of the project. This project is simple but effective, and has a lot of potential. Also, kudos to getting approx. 90% accuracy on the slippage detection.
18	5	Remote notification suchas text	I guess sending the graph when a defect occurs is not as necessary		
19	4	Detect part slip or other defect	The email system	Add a option to shut own print when error is detected	
20	3	I like the monitoring/reporting of different types of errors.	There isn't really a feature I don't like. I just think there are things you should try and do like if there is a certain deviation		

			(like the print went super wrong) the system will stop the print automatically to prevent wasting more filament and possibly making the situation worse if not stopped.		
21	4	Text message, as those tend to be more urgent to me than just an email. Sensing other errors would also be helpful, especially if it can be done as they happen or right after, to save both filament and time.	I'm not sure how much help a graph will be if a picture is already included, because it should be clear to the user that the system has moved, maybe included the closest before image instead of a graph.	I think if you provided a picture as described in the last sentence, a quick note on easy ways to fix that problem, like your extruded is too hot or the table that the printer is on is shaking too much	The problems I have run into using the 3d printers on campus are not the ones addressed by your software, because most of them have been design problems with my parts. I don't think this is in the scope of your project but maybe a preload program for your 3-D models where it highlights errors such as too large an overhang and it will likely fail. I now know what is going on with the 3d printer in the back of the MQP Lab :) This is not related to this project in particular, but I think it would be cool for each storage space in the MQP Lab to be labeled with both the year of the project and a short abstract.
22	4	Having the algorithm stope the print automatically and having the algorithm tell you what kind of defect it thinks the part has.	Probably the graph showing the algorithms interpretation of the snapshot. For an average user this seems like unnecessary information.		

23	5	automatic stopping as an option	none	360 camera view of part/ being able to see all sides of a part	cool project!!
24	5	I like the feature that the device would be able to recognize more areas of error.	I don't know they all seem like useful features.		
25	3	Detecting warping because I had that issue when 3D printing.	I don't find it necessary to have various cameras.	No	No

Appendix I: Focus Group Transcript

Interviewer:

Okay. So what is your experience with 3D printing, especially related to defective parts?

Participant 1:

So I work in the 3D printing lab, so through 3D printer OS, it does also also have a notification system when it detects that a print has gone wrong. So that's similar to what you guys are doing, but through your material, it doesn't send us a picture of the defective print, which is something I really admire from our project.

Participant 2:

My main experience has been setting up a couple of 3D printers for professor and so I've had to calibrate. And through that process, there's been a couple of times that there's been some off prints. Definitely either not sticking or it's been a couple of times that the extrusion process is a little off.

Participant 3:

Yes, I've done a lot of three printing for robotics work, and there's plenty of times where I've set a print to go and left the room and came back 30 minutes later to check on it's just a big ball of filament. So, I mean, it seems like a really useful project to me, so you don't have to constantly be looking at what you're printing.

Interviewer:

Are there any features of Dread that stand out to you as being very useful?

Participant 1:

I guess again, like sending the picture is really cool because then I know exactly where it failed. And if it's a big enough fail where I would want to stop it or not or let it continue?

Participant 2:

Yes, definitely agree with that. Sending the pictures is the biggest thing that's good.

Participant 3:

I think having the option to turn it off yourself too is good because maybe the software messes up and gets something wrong so it doesn't automatically shut down.

Interviewer:

Yeah, that's how it is currently. And I think that is very good for because there is that chance that the software will mess up. Are there any features that you see as not being very useful?

Participant 1:

No, there aren't that many different features, and all of them are useful, so I think there aren't any extraneous features.

Participant 3:

Yeah, I feel the same way. It's pretty straightforward, I guess.

Participant 2:

Yeah.

Interviewer:

So the last half of the video was talking about features that we could add in the future. Are there any of those that seemed most compelling to you?

Participant 3:

I think I mean, obviously it would be really hard to do, but more ability to put different cameras in or have maybe a specific product that's like this is our camera with our lighting and everything, so you don't have to use your own stuff.

Interviewer:

Pretty good idea.

Participant 1:

I agree. I like the adaptability. Like you can use different cameras for different printers.

Participant 2:

I don't want to just keep agreeing, but yeah, I think the idea of making it so that because there's such a wide range of 3D printers on the market, being able to at least have models where people can print out the camera and ring light holder would be really nice for those people, because some people don't really know how to use CAD or any design software. For the most part, they just kind of take it off the web, print it. So I think having that there is very good for that group of people.

Participant 1:

I also like the idea of having it sent to your phone as well. That's pretty cool. Because I don’t check my email

Participant 3:

Yeah. Because you would want to see as soon as possible. Right.

Interviewer:

I like that idea as well. Do you have any recommendations about the system based on your personal experience with 3D printing?

Participant 3:

You said in the presentation option to have it stopped the print itself. Right. I think maybe if you do, like an overnight print or something, you could turn a feature on.

Interviewer:

So another idea that we had that kind of was at the end of the video was the idea that you could show it a static image of a part and it could spit out like it could be this defect or this defect with a percent chance that it's that defect. And then also with information about what may have caused it and how you can fix that. Do you think that would be a useful feature?

Participant 1:

Yeah, definitely.

Participant 3:

I think knowing anything about how it happened would be good because then make sure it doesn't happen again.

Participant 1:

There are different fixes for different errors. It'd be nice to know what the error is.

Participant 2:

Yeah. I think having the ability to more easily figure it out, especially for people who aren't as knowledgeable as people who would ignore that feature, they would just look at a part and be like, oh, that's wrong with it. But definitely for some of the more inexperienced, that would be very helpful.

Participant 3:

I think maybe if I see a defect, I might just be like, all right, run it again, see what happens. So if there is a way to tell, like, there's something wrong with the printer, that'd be good.

Interviewer:

Based on your experience with 3D printing, what do you think would be the best defect to focus on detecting next?

Participant 3:

I don't know names for these things, because if it doesn't slip off, like the build plate. Right. But sometimes I've seen where something messes up and I come back and it's just a giant ball of filament. I don't really know how it happens, but then that's really annoying because you have to clean up the entire printer.

Interviewer:

And so generally that's called blobbing. I've heard blobbing. Yeah. That usually happens when the filament sticks to the nozzle.

Participant 3:

Okay, that makes sense. So usually you need to clean your nozzle. Yeah.

Participant 1:

When the nozzle gets clogged and so it can't extrude it. So it just kind of tries to do a couple of layers, but there's just nothing extruding.

Participant 2:

I don't know how much of a problem this one is, because I had this problem printing at the center here, but it was also because there's a huge print and I had to basically under fill everything. Warping?

Interviewer:

Warping, because of the underfilling. Like the low percentage of infill it sags in different places.

Participant 3:

Yeah, I've had a little bit of that. I can't remember one but I know it has happened. It's annoying.

Interviewer:

Yeah, that's definitely an important defect. I think especially well, if you're doing parts for aesthetics that's definitely an issue. But then also mechanically if you want like a straight part and it sort of warps off the printed then you can't use that part anymore. So it's definitely a good one.

Participant 3:

Exactly.

Participant 1:

There's also some parts where we can visibly see the different layers. That's because sometimes when the bed isn't properly aligned or if the nozzle is like a bit off.

Interviewer:

Yes, I believe that's called layer shifting. Sometimes that will happen if the belt slips. Do you guys have any additional comments or questions?

Participant 3:

Yes, well, there was like the Lightbox thing.

Interviewer:

Light box?

Participant 3:

Well, just like the enclosure from a printer is that something that obviously better technology could go away?

Interviewer:

Yeah, with something like machine learning that could definitely go away. The other thing is if your 3D printer exists in a closet somewhere then you don't really need to have the enclosure. I know that the enclosure is kind of the gaudiest part. Is that the word? Sure. It is a little bit bulky so I can see how it would not be very appealing but there are definitely ways around it.

Participant 3:

Yeah, I don't think it's that big of a deal any. Like you don't need to be staring at your print especially if it's detecting it for you.

Interviewer:

Plus when you set this up you actually get a live stream of the print that you can look at.

Participant 3:

That's good. What exactly is your project? Are you guys making a custom picture or the software?

Interviewer:

We're making the software so we've actually made the software. We can already do this so yeah, we're just sort of getting feedback on the project right now. Okay. Well if there isn't anything else and I think we're all set. Okay, cool. Thank you guys very much for participating.

Participant 3:

Thank you. Exciting project.