3D printing a transposed design in biopolymer materials using an articulated robot and pellet-based extrusion

A thesis presented in partial fulfilment of the requirements for a degree of

Master of Engineering

in

Mechatronics

At Massey University, Albany,

New Zealand

Byron Brooks

2016

Abstract

The aim of this project was to develop a new method of 3D printing. This method is a mix between Fused Deposition Modelling and freeform printing, using a 6 degree-of-freedom articulated robot and a pellet-based extruder to mix and distribute the biopolymer, to create commercial quality thin-shelled parts with aesthetic aspects unique to the process and with a reduced amount of material wastage. There is the potential for many industries to benefit from this new technology.

Initially this project is focused on applications for artists as thin-shelled designs rarely provide the physical properties required for functional parts. An artist has provided a design to test the printer.

The hopper is designed to work with a range of different polymer pellets. It is based off a previous student's design and mimics the operation of an injection moulder by pushing the pellets through a heating chamber with an auger.

The robot controlling the movement of the platform is an ABB IRB120. This robot has six degrees-of-freedom that allows it to reach several positions that would otherwise be impossible with a Cartesian system. The IRB120 has a very high spatial accuracy and repeatability.

The design's original format is converted to a flattened 2D format and the lines are interpolated to produce a 2D set of points. The overlaps in the shapes are removed to reduce the number of times the nozzle traces over previous paths, which helps to keep the layer thicknesses the same. These shapes are filled in with points so the contours are not empty. The points are then projected onto a mathematical model of the platform to produce a 3D point cloud. Finally, these points are converted into data for the robot to read. The design data points stream to the robot, which interprets them on the fly.

Many iterative changes and improvements were done to the hardware and software as the result of continuous testing of the process and analysis of the print.

The pellet-based extruder is an elementary design with numerous variables that affect the resulting extrusion. After many design iterations and improvements to the extruder, the extruder can produce a continuous strand of material, with relatively constant flow.

The software accurately converts a design from the given format into a path for the contours, and another path to fill the contours. These paths are projected onto a model of the moulded platform. Each point along the path is put through multiple affine transforms to generate a

location and orientation for the end effector of the robot. The robot is moved by streaming each point to the robot one at a time. The extruder was controlled simultaneously to create a printed design.

The printed design is geometrically correct. However, the width of the extrusion path needs to be improved to increase the accuracy of the design to the reference one. The current prints achieve the correct visual properties in the extrusion. However, they require a secondary process to improve the surface finish.

This project has produced a new 3D printing process, mixing Fused Deposition Modelling and freeform printing. This process can be adapted to be used in a wide range of applications. It has also produced a low-cost, effective pellet-based extruder that can be used to test a range of different materials, and their effectiveness in being used for 3D printing.

Acknowledgements

I would like to thank everyone who has supported me through this project. Firstly, I would like to thank my family and friends for providing me with their love, support, and patience, all the while encouraging me to do my very best.

Thank you to my partner, Nastassja, for all her patience, support and love that supported me through this degree.

I am deeply grateful to my supervisor, Dr. Johan Potgieter, B.Sc., M.Sc., Ph.D., College of Sciences, School of Engineering & Advanced Technology, Massey University, for his continual guidance, insight and wise words, without which this project would not have succeeded.

I would like to acknowledge the support, advice, and ideas provided to me from the other staff in the School of Engineering & Advanced Technology at Massey University, namely Dr. Khalid Arif, Dr. Frazer Noble, and Dr. Steven Dirven.

I am thankful for the financial support of the Scion research institute, without which, this project would not have been possible. I am grateful to Dr. Marie Joo Le Guen for her constant aid throughout this project and her belief in my abilities.

Lastly, I would like to acknowledge David Trubridge for providing me with his design to test this project on, along with his thoughts and creative vision.

Table of Contents

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: LITERATURE REVIEW 4

2.1 ADDITIVE MANUFACTURING 4
2.2 INDUSTRIAL ROBOTS 10
2.3 EXTRUSION 17
2.4 CHAPTER SUMMARY 22

CHAPTER 3: MECHANICAL DESIGN OF THE ARTICULATED PLATFORM AND EXTRUDER 23

3.1 PLATFORM 23
3.2 ROBOT ARM 26
3.3 HOPPER AND EXTRUDER DESIGN 34
3.4 CHAPTER SUMMARY 48

CHAPTER 4: DEVELOPMENT AND IMPLEMENTATION OF ALGORITHMS IN A SOFTWARE APPLICATION 49

4.1 COORDINATE SYSTEMS 50
4.2 CLASSES 52
4.3 GRAPHICAL USER INTERFACE 54
4.4 OVERALL PRINTING PROCESS 57
4.5 DXF FILE IS CREATED FROM A CAD PACKAGE 58
4.6 FILE LOADING 60
4.7 CONTOUR POINT GENERATION 60
4.8 GEOMETRIC OPERATIONS 63
4.9 OVERLAPS ARE REMOVED 63
4.10 FILL POINT GENERATION 65
4.11 2D TO 3D PROJECTION 69
4.12 SAVING/LOADING FILES 70
4.13 PRINTING 70
4.14 COMMUNICATION TO HOPPER AND ROBOT 71

4.15 CONVERTING THE POINT3D TO ROBTARGET FORMAT 72

4.16 LIMITATIONS 76

4.17 FUTURE IMPROVEMENTS 77

4.18 CHAPTER SUMMARY 81

CHAPTER 5: SYSTEM INTEGRATION AND DESIGN IMPLEMENTATION 82

5.1 DEVELOPMENT 82

5.2 TESTING 84

5.3 CHAPTER SUMMARY 90

CHAPTER 6: RESULTS AND DISCUSSION 91

6.1 DEVELOP A PELLET-BASED EXTRUDER SYSTEM FOR 3D PRINTING BIOPOLYMERS 91

6.2 DEVELOP SOFTWARE AND PARAMETERS TO CONVERT THE DESIGN INTO COMMANDS FOR THE ROBOT. 91

6.3 DEVELOP SOFTWARE TO CONTROL THE ROBOT 92

6.4 BUILD A FUNCTIONAL DESIGN USING THE DEVELOPED 3D PRINTER 92

6.5 MR TRUBRIDGE'S PERSPECTIVE 96

CHAPTER 7: CONCLUSION 97

CHAPTER 8: REFERENCES 98

APPENDIX A – DESIGN DRAWINGS 101

APPENDIX B – PCB DRAWINGS 107

APPENDIX C – JOURNAL ARTICLE 108

List of Figures

Figure 2-1: Filament extrusion process (Carneiro, Silva, & Gomes, 2015) 6
Figure 2-2: Methods of printing curved surface (Diegel, Singamneni, Huang, & Gibson, 2011b) 7
Figure 2-3: Pellet based extruders. (a) shows a design to be attached to a delta 3D printer. (b) is designed for a large scale 3D printer 7
Figure 2-4: Freeform process. (a) The wire fed process (Donghong Ding, 2015), and, (b) The droplet based process (Tseng, Lee, & Zhao, 2001). 9
Figure 2-5: Applications of freeform technology (a) in art (MX3D, 2015a), (b) in manufacturing (MX3D, 2015b) 9
Figure 2-6: The six lower-pair joints (Craig, 2005). 10
Figure 2-7: Hudson Robotics' PlateCrane EX robot (Hudson Robotics, n.d.). 12
Figure 2-8: Cartesian assembly line robot (Schunk, 2015) 12
Figure 2-9: Spherical robot arm by Kawasaki (Robotics Bible, n.d.) 12
Figure 2-10: Espon's G3 SCARA robot (Epson, n.d.) 13
Figure 2-11: ABB's delta FlexPicker robot (ABB, n.d.-a). 13
Figure 2-12: ABB's IRB 1520id articulated robot (ABB, n.d.-b). 13
Figure 2-13: Single screw extruder (Douroumis, 2012) 17
Figure 2-14: Schematics for screws: (a) Flow restriction/mixing type; (b) Barrier-to-melt type; (c) Barrier between solid bed and melt pool type; (d) Solid/melt mixing type (Chung, 2011) 19
Figure 2-15: Illustration of dispersive and distribution mixers 19
Figure 2-16: Diagram showing how a gear pump moves polymer through it (Skibba & Thoma, 2001) 21
Figure 3-1: Process to create the mould. (a) Laminating MDF sheets, (b) CNC milling of mould, (c) sanding and filling required areas, and (d) sealed and waxed mould 24
Figure 3-2: (a) Fibreglass platform, (b) ribs inside platform, (c) platform base with locking mechanism, and (d) bracket to attach platform to robot 25
Figure 3-3: Robot with platform attached 26
Figure 3-4: Robot controller 26
Figure 3-5: ABB's Flexipendant 27
Figure 3-6: Workspace of IRB 120 27
Figure 3-7: Link lengths of IRB 120 27
Figure 3-8: ABB's RobotStudio 28
Figure 3-9: Different robot configurations for the same point 30
Figure 3-10: Pellets used for printing 34
Figure 3-11: Previous student's extruder 36
Figure 3-12: CAD model of the extruder 37
Figure 3-13: Cross section of hopper section with coupling ring 37
Figure 3-14: Arduino Uno (Aduino Inc, n.d.) 38

Figure 3-15: Motor shield (RobotShop, n.d.) 38
Figure 3-16: Extruder motor controller 38
Figure 3-17: Prototype heat sink 39
Figure 3-18: Close up of hot end 40
Figure 3-19: Screw after being used 40
Figure 3-20: cross section of CAD model of extruder, and extruder attached to test platform 41
Figure 3-21: Fan bracket 42
Figure 3-22: Worm gear driven 4RPM motor 42
Figure 3-23: 2 mm laser cut steel plates 42
Figure 3-24: Heating components. (a) PID heating controller, and (b) Heating band 43
Figure 4-1: Cartesian coordinate system 50
Figure 4-2: (a) Platform coordinate system, and (b) Robot coordinate system 51
Figure 4-3: User Interface to control printer 54
Figure 4-4: Process to convert DXF 55
Figure 4-5: Flow diagram of the printing process 57
Figure 4-6: (a) Isometric view of the CAD design, and (b) top view of the CAD design 58
Figure 4-7: Interpolated design 62
Figure 4-8: Close-up of interpolated design 62
Figure 4-9: (a) Entities before overlap is removed, (b) Entities after overlap is removed 63
Figure 4-10: Design after overlaps are removed 65
Figure 4-11: (a) Points of intersection along a scanline, (b) Fill line within polygon between start and stop points, (c) Points of intersection along scanline and open contour, and (d) Fill line with scanline intersecting polygon on a tangent. 67
Figure 4-12: 2D design with fill points 68
Figure 4-13: Side views of the (a) single point projection method, and the (b) vertical projection method 69
Figure 4-14: View of point normal to the XY plane 69
Figure 4-15: Planes normal to surface rotated around a central axis 73
Figure 4-16: Starting point of diagram used for mathematics calculation 73
Figure 4-17: Endeffector vector in relation to origin 74
Figure 4-18: Black diagram is transposed to (0,0,150) 74
Figure 4-19: Black diagram has been rotated around (0,0,150) 75
Figure 4-20: Clustering of points around centre 79
Figure 5-1: Bonding test for overlapping material 82
Figure 5-2: Flow inconsistency of extrusion 82
Figure 5-3: Test designs. (a) Rectangular design, (b) circular design, (c) hexagonal design 83
Figure 5-4: Methods of simulation. (a) Through RobotStudio, and (b) with a pencil 84
Figure 5-5: Guides resting on platform to keep the nozzle height constant 85
Figure 5-6: (a) Tags and (b) artefacts left by the printing process 87

List of Figures

Figure 5-7: Horizontal fill path trial 88
Figure 5-8: Extrusion quality of 4 RPM motor 90
Figure 6-1: 3D printed design 92
Figure 6-2: Close-up of surface finish 93
Figure 6-3: Comparison of reference design and print 94
Figure 6-4: Comparison of print and reference design 95
Figure 6-5: Printing along a contour. (a) Over the path. (b) Within the path 95
Figure 6-6: Result from testing the new motor 95
Figure 6-7: Underside of 3D printed design 96

List of Tables

Table 2-1: Table of processes and technologies used in Additive Manufacturing....4
Table 2-2: Type and description of robot joint types (Saha, 2014). 11
Table 2-3: Pros and Cons of twin screw extruders (Chung, 2011). 17
Table 3-1: Link lengths for the IRB 120....27
Table 3-2: Joint ranges and Max speed for IRB 120 28
Table 3-3: Table of RAPID movement commands....31
Table 3-4: Pros and cons of moving hopper 35
Table 3-5: Pros and cons of stationary hopper 35
Table 3-6: Commands to control the extruder 44
Table 4-1: Different reference frames provided for programming ABB robots 51

List of Code Snippets

Code Snippet 3-1: Main function of robot controller 32
Code Snippet 3-2: Interrupt service routine for setting the robot's movement speed....33
Code Snippet 3-3: Interrupt service routine to move the robot 33
Code Snippet 3-4: Algorithm to calculate speed from the encoder....45
Code Snippet 4-1: Method of saving to DXF 55
Code Snippet 4-2: Overlap removing algorithm 64
Code Snippet 4-3: Scanline algorithm to create the fill path....66
Code Snippet 4-4: Gets the points of intersection from the scanline and the polygon 68
Code Snippet 4-5: Projection function....70
Code Snippet 4-6: Method to create the command for the robot 72

List of Equations

Equation 3-1: Equation of the platform 24
Equation 4-1: Equation of a line 61
Equation 4-2: Rotation Matrix for 2D points....66
Equation 4-3: X value of intersection....67
Equation 4-4: Y value of intersection....67
Equation 4-5: Matrix for initial translation 73
Equation 4-6: X value for rotation about a point....74
Equation 4-7: Y value for rotation about a point....74

List of Terms

Abbreviation	Expansion
CAD	Computer Aided Design
CNC	Computer Numerical Control
DOF	Degrees of Freedom
FDM	Fused Deposition Modelling
IPC	Inter-process communication
ISR	Interrupt Service Routine
MDF	Medium Density Fibreboard
PCB	Printed Circuit Board
PID	Proportional, Integral, Derivative control
PLA	Polylactic acid
PWM	Pulse-width modification
STL	Stereolithography

Chapter 1: Introduction

Additive manufacturing has been around since the 1980's and since then has grown and developed as both an industry and an area of research. Fused Deposition Modelling (FDM) is one of the most common forms of 3D printing. Currently there is no method of creating certain parts without using support material. Freeform printing overcomes this challenge, but it can only be used for a few select applications. Because of this, it was proposed to create a new method of printing that would combine the benefits of freeform printing with the FDM process.

The current method of testing new materials for FDM printing involves turning the material into filament before running it through a FDM printer. This is normally a lengthy and costly process. To improve this and make it easier for testing new materials it was suggested to create a method of printing directly from pellets, cutting out the need for filament creation.

It was determined this printer would focus on printing a thin-shelled design as the process could later be adapted to creating solid bodied parts. To test this printer a design was provided for a wall-mounted lampshade. A wall-mounted lampshade could be perceived, as a thin-shelled design so is an excellent choice as a test design. The material was also provided by the biomaterial research institute, Scion, which currently creates the material used for the artist to create his designs. The provided material is similar to the one the artist generally uses for his designs. A student previously attempted to create an extruder for similar purposes, which was used as a basis to develop further.

As the provided design is a paraboloid style design, it was decided it would be easier to create a platform of that shape than to develop support material algorithms. These algorithms could be added in a later date, but as the provided design is a single layer, printing on a shaped platform would give the best surface finish results.

In the previous project, it was attempted to have the hopper attached to the robot. This led to multiple problems that are outlined in Section 3.3.2. The main problem was the inconsistency in the flow of the material. As the platform is not flat like most 3D printers use, the standard 2.5D method using a gantry style system to print a model cannot be used. Moreover, because the platform is paraboloidal with the surface being equation driven, rather than a hemisphere, it is not possible to rotate it around a single point to keep the extrusion normal to the surface. Therefore, the platform would require multiple degrees-of-freedom. It was decided to attach the platform to the robot, as it would be faster and easier than creating a custom system.

The objectives of this Masters project is to:

1. Develop a pellet-based extruder system for 3D printing biopolymers

   The extruder system is to print using pellets rather than filament. This requires a method to store and feed the pellets, a method to mix and melt them together, and a method to extrude a set width of material.

   The pellets are provided by Scion, a Crown Research Institute that has a focus on biopolymers. They consist of a mix between Polylactic acid (PLA) polymer and fibres from the native New Zealand flax plant (Phormium genus). The pellets are between 1 mm and 3mm long with a diameter of 3 mm. The extruder system must be able to extrude these pellets.

2. Develop software and parameters to convert the design into commands for the robot

   Software must be developed to convert the provided design to instructions that are readable by the robot. Because this project to test the viability of 3D printing as a method of manufacturing commercial versions of artistic thin-shelled designs, this software must be able to convert a wide range of designs. The software must be easy to use and tuneable through a user interface so it can be configured without needing to modify the source code.

   Mr Trubridge provided CAD files that contain the designs. The software must be able to convert these specific designs into a path for the robot to follow. The path must accurately represent the design.

3. Develop software to control the robot

   Software to move the robot to the correct position and orientation must be developed. This software must be able to move the platform without collisions or interference or danger to the user.

   Because the printer is to be used by end users to create commercial products, the software must be robust to prevent errors from occurring, which could affect the commercial viability of the project.

   The software must also be easy to read and modify. As the development of this software will continue after the end of this project, it must be possible for further development to be done easily.

4. Build a functional design using the developed 3D printer

   With the developed 3D printer, the user must be able to print a full design that accurately represents the model loaded into it. Currently the model will only be a single layer, but this may change in the future. The printer must require the least amount of user interaction once it has been sent the command to print, as the user should be able to leave and come back knowing there will not be any major problems with the printer. The resulting print should require the least amount of post-processing possible to make it into a commercially viable design.

Chapter 2: Literature Review

2.1 Additive Manufacturing

2.1.1 Additive Manufacturing technologies

Additive manufacturing has been around since the 1980s and different materials and methods have been developed since then to improve the process. Generally, the additive manufacturing technologies are categorised by the process they use. The processes and technologies can be seen in Table 2-1.

Table 2-1: Table of processes and technologies used in Additive Manufacturing

Process Type	**Technologies**
Extrusion	Fused Deposition Modelling (FDM)
	Direct Ink Writing (DIW)
Light-polymerized	Stereolithography (SLA)
	Digital Light Processing (DLP)
Powder-based	Inkjet 3D printing (3DP)
	Electron-beam melting (EBM)
	Selective laser melting (SLM)
	Selective laser sintering (SLS)
	Selective heat sintering (SHS)
	Direct metal laser sintering (DMLS)
Laminated	Laminated object manufacturing (LOM)
Wire-fed	Electron beam freeform fabrication (EBF^3)

Each process has its benefits and drawbacks, as does each technology. All of them work by printing a layer of material and then building upon the previous one.

Extrusion works by heating the material and passing it through a nozzle onto a bed. The nozzle and/or bed moves around to create the part's layer. Extrusion systems require support structure for certain parts which have undercuts that are too large. Some of the constraints of this method are; it doesn't have very good resolution due to a typical layer thickness of 0.2 mm; the parts require support structure for any overhangs or unsupported parts; and some printers have proprietary filament cartridges and are unable to run with others.

Light polymerization uses light curable resins, which are hardened and set by flashing light onto the resin, using either visible or ultra-violet light depending on the resin. These parts are built in a bath of resin and the light cures each layer of resin. This method provides very accurate parts with layers as thin as 0.025 mm thick when using an ultraviolet laser. Because the parts are made from resin, they are more brittle than parts made using other methods so are aesthetically superior but can fail when subjected to load.

Most powder-based systems use heat to bond the powder together by melting them. The exception to this is the 3DP technology where powder bonds by applying a resin or glue that hardens and joins the powder together. The methods that use heat generally use lasers in the infrared spectrum to fuse the powder together. Powder-based systems do not generally require support material to help create parts as the layers of powder provide the support while building the model. When the part is removed, this powder can be shaken out. However, not all the powder is reusable. Guidelines for most machines state a maximum of 50% used and a minimum of 50% fresh powder should be used in the print. This process therefore generates a large amount of waste.

The laminated process works by moving sheets of material over the platform, gluing them together and cutting them to form each layer. It is very easy to make large parts this way. However, their resolution and dimensional accuracy is not as good.

Wire-fed systems feed a metal wire onto the substrate. This wire is melted using various methods, such as electron beams or lasers just before it reaches the substrate so a melt pool is created on the part to build up material as it cools and hardens.

2.1.2 Current FDM process

The Fuel Deposition Modelling (FDM) process is well documented. Firstly, a three-dimensional model is developed in CAD software. This model is exported as a Stereolithography (STL) file, a format that defines a solid model as a set of triangles. This STL file is sliced horizontally in very small increments to define the layers that will be printed. The software generates the contour and fill path for each layer. These paths are transferred to the printer, which follows the instructions written.

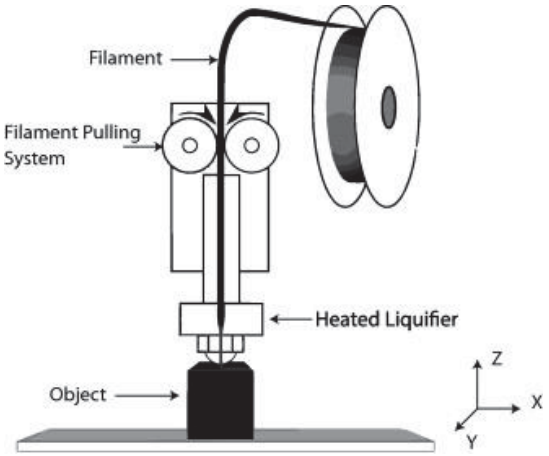


Figure 2-1: Filament extrusion process (Carneiro, Silva, & Gomes, 2015)

In the physical process of printing, shown in Figure 2-1, the filament, which is generally a thermoplastic polymer, is fed through the drive wheels, and past a heating element, that turns it into a semi-molten state. This material is extruded through a nozzle onto a platform. The nozzle moves along the XY plane to print each layer and is then moved down along the Z axis to begin the same process again for the next layer. Once the part is finished, the base and any support material is removed (Ahn, Montero, Odell, Roundy, & Wright, 2002).

With FDM printing, there are limited variables that be modified within the process. Sood, Ohdar, and Mahapatra (2010) found the most common of these are; layer thickness, part build orientation, raster angle, bead width and distance between nozzle and part (air gap). The characteristic that affected the surface finish quality of an ABS part the most significantly was found to be layer thickness but bead width and deposition speed also had minor effects (Anitha, Arunachalam, & Radhakrishnan, 2001). The characteristics founds to be responsible for increasing the mechanical strength of the part are the distance between the nozzle and the part (air gap) and raster orientation (Montero, Roundy, Odell, Ahn, & Wright, 2001).

Lots of research is currently going into new applications of FDM printers along with new materials to use with them. The FDM process has been applied to areas such as tissue engineering, which is focused on creating functional replacement tissues or organs. New materials such as carbon nano-fibre reinforced filament and biopolymers are also active areas of research in the FDM field.

There has been little development on printing curved parts. However, the research that has been done shows a technique to achieve quality parts by creating a scaffold beneath the part, which is removed afterwards, leaving a curved print rather than staggered layers (Diegel, Singamneni, Huang, & Gibson, 2011a; Diegel et al., 2011b). An illustration of this can be seen in Figure 2-2. This method has already been applied to create curved conductive layers within parts with a continuous extrusion (Diegel et al., 2011b). As this is a relatively new area of research, many improvements can be made to the process to improve part quality.

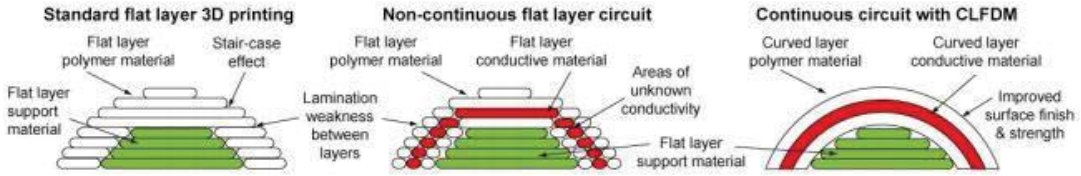


Figure 2-2: Methods of printing curved surface (Diegel, Singamneni, Huang, & Gibson, 2011b)

2.1.3 Pellet-based 3D printers

Little research has been done on applying direct extrusion from pellets to 3D printing. Reddy, Reddy, and Ghosh (2007); Valkenaers, Vogeler, Ferraris, Voet, and Kruth (2013) both use custom designed pellet extruders similar to each other to replace the filament extruder on a 3D printer to produce parts. The extruder prints directly onto the print bed, as the filament one does. A number of hobbyists have attempted creating pellet-based extruders that can be seen in Figure 2-3. However, no pellet-based 3D printers are commercially available yet.

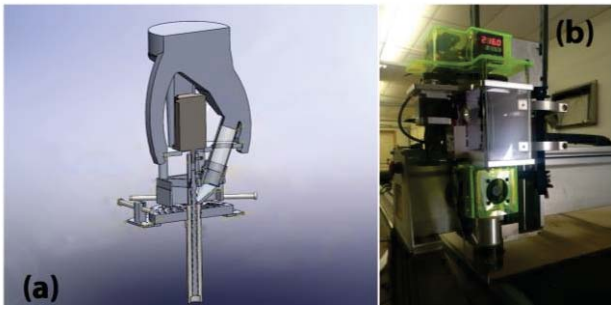


Figure 2-3: Pellet based extruders. (a) shows a design to be attached to a delta 3D printer. (b) is designed for a large scale 3D printer

While creating filament, there are a number of processes applied to the material, which could potentially alter the structure and properties of the material or introduce impurities. The process of creating filament also increases the cost of production. Any additives, such as fibres, that are to be mixed into the material must be included during the filament creation stage. This means when creating and testing new materials for FDM 3D printing, new spools of filament must be created for each one.

Current pellet extruders offer several advantages, such as:

- New materials can be tested quickly and easily by purging the extruder and filling it with the new pellets;
- There is a reduced chance of having degradation in the material as it has been through less thermal processing;
- Wider range of materials are available including thermoset plastics;
- Additives can easily be added to the polymer to change its properties.

However, current pellet-based extruders do have some disadvantages, such as:

- Impurities can be introduced easily;
- Air bubbles and voids must be avoided and can be hard to remove;
- Hopper must be kept full;
- Extruder pressure must remain constant.
- Not designed or optimized for printing with fibres or biopolymers

2.1.4 Freeform 3D printing

Freeform printing is a fluid version of 3D printing. Material is still deposited upon itself or another object. The material, which is usually a metal in the form of a stick, is heated up in a small, localised area to create molten droplets that fall onto the desired location. Early versions use an arc welding tool attached to a movement mechanism such as a robotic arm to form models (Harris & Director, 2011).

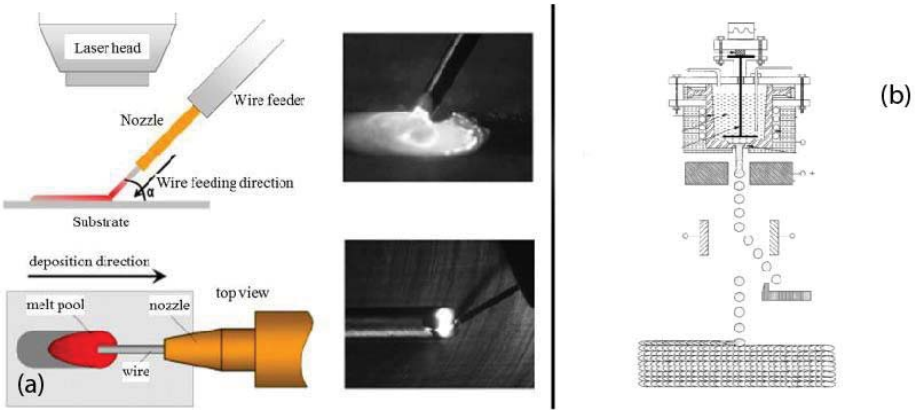


Figure 2-4: Freeform process. (a) The wire fed process (Donghong Ding, 2015), and, (b) The droplet based process (Tseng, Lee, & Zhao, 2001).

Figure 2-4 (a) shows the wire fed process, where a melt pool is created by heating the wire. The head and the wire are moved simultaneously, and as the melt pool cools, it solidifies to form the part. The print resolution of this method is determined by the thickness of the wire. Figure 2-4 (b) shows a set-up to create droplets from molten metal. These droplets fall past a pair of charged plates, which, depending on the location of the charges on the plates, can move the droplets around to position them on the substrate.

MX3D, a company in Netherland, is using this technology for both art and functional designs such as bridges. By attaching it to an articulated robot, it is possible for the printer to run in any direction. The material sets in position as the robot moves. Because of this, support material is not required in freeform 3D printing. This allows for much greater flexibility in design and significantly enhanced capabilities of the printer with reduced waste material. Figure 2-5 depicts some applications of MX3D's freeform printer.

Figure 2-5: Applications of freeform technology (a) in art (MX3D, 2015a), (b) in manufacturing (MX3D, 2015b)

2.2 Industrial robots

Robots are used for many different applications in lots of industries. Due to the high demand to automate many different industries, a large number of robots have been developed for a wide range of applications. These applications include pick and place, welding, painting, CNC milling, assembly, and palletizing.

Biological dexterity is very complicated and difficult to mimic using robotics because of the size and strength of actuators and the mechanical limitations of materials. Degrees of freedom (DOF) defines the number of ways the robot's end effector can move in three dimensional space, encompassing both translational and rotational motion (Schilling, 2013). A robot requires six degrees of freedom, translation along the X, Y and Z axes, and rotation along the X, Y and Z axes, to achieve maximum flexibility and reach any point in three-dimensional space.

2.2.1 Robot motion

Robots can be classified into serial and parallel robots. Serial robots are made up of a series of links, each one joined to the previous. One link becomes the base, and the other end is known as the end effector. Parallel robots have two or more individual links coming off their base that attach to the end effector through a series of links. Some of these links can be passive, meaning they do not have actuators on the joint, but most joints are normally active.

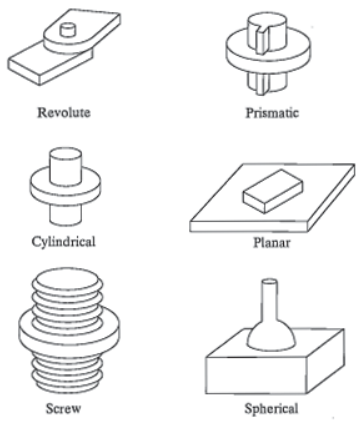


Figure 2-6: The six lower-pair joints (Craig, 2005).

Each link is connected by a joint that provides either linear or rotational motion to the link. There are six types of joints: Revolute, prismatic, cylindrical, planar, screw, and spherical. A description of each of these joints can be found in Table 2-2, and a schematic can be found in Figure 2-6.

Table 2-2: Type and description of robot joint types (Saha, 2014).

Joint type	**Description**
Prismatic	Prismatic joints are linear joints and slide along a vector. They only have a single degree of freedom.
Revolute	Revolute joints pivot around an axis. This type of joint has one degree of freedom.
Cylindrical	Cylindrical joints have two degrees of freedom as they permit rotation around the shaft, and translation up and down it.
Planar	Planar joints allow translational motion along a plane and rotational motion about the axis normal to the plane, giving it three degrees of freedom.
Screw	Screw or Helical joints rotate and translate at the same time. The translational motion is dependent on the rotation of the screw. Because of this dependency, screw joints have only one degree of freedom.
Spherical	Spherical joints allow rotational motion in any direction but no translational motion. Because of this, there are three degrees of freedom in this joint.

The workspace of a robot is the area of space that the end effector can reach. This shape is unique to each robot. This workspace is determined by the length of each link, type of joint and mechanical limitations of the robots' movement. These mechanical limitations are generally designed to protect parts of the robot during operation.

2.2.2 Types of industrial robots

The robot arms used in industry today are variations of six basic types. They are:

- Cylindrical
- Spherical
- Cartesian
- SCARA
- Delta
- Articulated

Cylindrical robots are named as such after the geometry of their workspace. Figure 2-7 shows the construction of a cylindrical robot. The slide attached to the vertical column can move up and down, but can also move the end effector, which is attached to the end of the slide, closer or further from the column. Rotating the column gives the robot a cylindrical workspace (Groover, 2015). Cylindrical robots are generally used in the electronics industry in pick and place tasks (Wilson, 2014).

Figure 2-7: Hudson Robotics' PlateCrane EX robot (Hudson Robotics, n.d.).

Spherical robots, also known as polar robots, consist of a sliding arm that rotates about the vertical and one of the horizontal axes (Aized, 2010). This provides a spherical workspace and is good for applications that require little vertical movement (Schilling, 2013). Spherical robots, like the one in Figure 2-9, were the first type of industrial robot developed.

Figure 2-8: Cartesian assembly line robot (Schunk, 2015)

Cartesian robots, like the one in Figure 2-8, are also known as linear or gantry robots. Altintas (2015); Low, Chong, Salleh, and Johnny (2010) define Cartesian robots as having their principal axes of control as linear and perpendicular to each other. All three axis are perpendicular to each other and by moving along these three slides, the robot can achieve a cuboid workspace. They are used in many applications such as CNC mills and 3D printers.

Figure 2-9: Spherical robot arm by Kawasaki (Robotics Bible, n.d.)

SCARA is an acronym for Selective Compliance Assembly Robotic Arm. SCARA robots are normally serial robots, but some do exist as parallel too. This type of robot is similar to the cylindrical robot but limits the movement along the vertical axis, while being compliant in the horizontal plane. Generally, this compliance is achieved with two horizontal parallel rotational joints and a prismatic joint at the end as seen in Figure 2-8. This makes it great for assembly type work where vertical insertion or side-to-side alignment is needed (Groover, 2015; Sahari et al., 2012).

Figure 2-10: Espon's G3 SCARA robot (Epson, n.d.)

Delta robots, like the one shown in Figure 2-11, are a form of parallel robot. Three arms extend from the base to the end effector. A Delta robot's end effector stays parallel to the base and does not change its orientation. This is achieved by having each arm made of two beams attached to universal joints (Groover, 2015). Delta robots have three degrees of freedom, giving just translational motion, but modifications can be made to increase the number of degrees. Isaksson, Brogårdh, Watson, Nahavandi, and Crothers (2012) propose one way to achieve six degrees of freedom is by individually moving each beam for each arm, although some motions are limited.

Figure 2-11: ABB's delta FlexPicker robot (ABB. n.d.-a).

Articulated robots, like the one shown in Figure 2-12, are the most commonly found configuration of robotic manipulator. They consist of a series of joints, generally revolute and prismatic. These robots are used in lots of different applications and are generally the most versatile type of robot due to having many degrees of freedom and big workspace. Their movement closely resembles that of a human arm. Most of these arms have six axes but variations do exist. These variations are generally designed for specific applications (Wilson, 2014).

Figure 2-12: ABB's IRB 1520id articulated robot (ABB, n.d.-b).

2.2.3 Kinematics

Kinematics is a branch of science that studies geometry in motion. This motion is any type of displacement, including changes in position and orientation, and all derivatives with respect to time (Jazar, 2011). The end effector position and orientation of articulated robots and joint angles between links can be calculated using kinematics. Forward kinematics calculates the position and orientation of each of the robots' links using the joint angles and link lengths. These can be cascaded to get the position and orientation of any one link from another. Using this, the coordinates of the end effector relative to the base can be calculated. Inverse kinematics calculates the reverse of this: by using a known coordinate and orientation of a link end and the link lengths, the joint angles to achieve that position can be determined. These equations can also be cascaded to find the joint angles relative to one link from another.

2.2.4 Singularities

When the kinematic mapping is processed by the robot to determine the end effector position in terms of the robot's joint angles, there are certain points where the equations are unsolvable and give erroneous results. These points are known as singularities and can lead to large joint velocities, forces and torques and can also reduce mobility as the robot moves through them (Donelan, 2007).

Singularities can occur in all types of robot, but are commonly an issue in articulated robots. ISO 13482:2014 defines a singularity as "occurrence whenever the rank of the Jacobian matrix becomes less than full rank." The Jacobian matrix is used to calculate the kinematics of the robot and is typically defined as a matrix of the first order partial derivatives of the robots' degrees of freedom.

In a six-axis articulated robot there are three types of singularity, wrist, elbow, and shoulder. These are formed when there are two or more joints that become collinear. When these joints align, the number of degrees of freedom of the robot is reduced, which changes the rank of the Jacobian matrix.

2.2.5 Robot Programming Systems

Industrial robots require some form of control to move them. Normally each controller will come with a teach pendant to control the robot directly. Through the teach pendant it is generally possible to create simple programs. Most robot manufacturers provide software that is used to program them. This software is generally proprietary. However, there are third party software packages, such as Robot Operating System (ROS), Microsoft Robotics Developer Studio, or LabVIEW that can program many different robot controllers.

2.2.5.1 Manual

Manual systems give the user control over the code of the robot. The user can create and manipulate the code that controls the robot. There are two main formats of manual program control: Text, and Graphical.

Text-based systems use a programming language and are one of the most common methods for programming robots. There are numerous programming languages but many industrial robot manufacturers have their own. These are generally very simple languages with basic commands for controlling program flow. These controller-specific languages are unable to be used on different systems and are generally programmed through proprietary software rather than third party software.

Graphical languages require the user to move icons around to form a program. Generally, graphical languages have the same program flow operations as text-based systems, but are easier to use. Graphical languages generally sacrifice flexibility for ease of use and simplicity (Biggs & MacDonald, 2003).

2.2.5.2 Automatic

Automatic systems do not require the user to have direct contact with the program code of the robot. There are three main methods of automatic robot programming: Learning, Demonstration, and Instructive.

Learning systems give the robot some set parameters and the robot determines the best way to achieve the outcome. Many high-level algorithms, generally using heuristics and large amounts of data are used.

Programming through demonstration is the most common automatic method and generally has the user moving the robot with a teach pendant. The robot records this motion, which it plays back when run. This method is very effective for teaching the robot the path to follow, which can be optimized through manual methods once the path has been defined. This method has a very shallow learning curve and does not require the user to know how to write code to teach the robot, making it very easy to learn to program.

Instructive systems respond to the environment, whether it is through gesture, voice, vision control or other methods. These controls will often be linked to predefined actions or motions. For example, each voice command is linked to a different action that runs when the command is detected. This is high-level software and generally requires sensors to interact with the user or environment (Biggs & MacDonald, 2003).

2.3 Extrusion

2.3.1 Extrusion Systems

Polymer extruders work by changing materials from solid pellets to a viscous liquid form that is pushed out through a nozzle. This is achieved by adding heat to melt the pellets and pressure to force the pellets through the orifice. Extruders can be classified into two different types based on their mode of operation being either continuous or discontinuous. Discontinuous extruders are great for batch type processes such as injection moulding, whilst continuous extruders are used for processes like creating filament. Generally, discontinuous extruders use reciprocating motion and continuous extruders use rotating motion (Rauwendaal, 2014). Discontinuous extruders are not suited to this project as the designs are variable and therefore use different amounts of material.

The most common continuous type extruder is the screw extruder, an example of which is depicted in Figure 2-13. This type of extruder consists of a rotating screw that moves pellets down a barrel, past heating bands, and through a nozzle. Screw extruders can be classified as either single or twin screw extruders, depending on the number of screws used. Single screw extruders contain one screw within the barrel whilst twin screw extruders contain two rotating screws. These two screws can either be co-rotating or counter-rotating depending on the design.

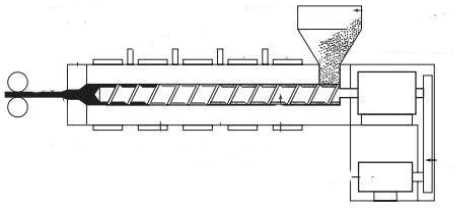


Figure 2-13: Single screw extruder (Douroumis, 2012).

Table 2-3: Pros and Cons of twin screw extruders (Chung, 2011).

Twin screw pros	**Twin screw cons**
• Positive conveying of polymer solid and melt along the screw channel • Better melting and mixing • Self-cleaning of the screw surface if fully inter-meshing • Fast pressure build-up along the screw channel • Narrow residence time distribution	• More expensive to purchase and maintain • Lower head-pressure capability because of the combined large channel area of two screws and the size limitation of thrust bearings • A metered feeding device required at high screw speeds to operate in a starved feeding mode

As seen in Table 2-3, twin screw extruders have many benefits, but the major benefit is that they provide a higher quality extrusion. This can easily outweigh the drawbacks of twin screw extruders in most instances. The decision to use single or twin screw extruders generally comes down to a question of application and cost.

2.3.2 Screw Design

Almost all screws are designed to have three stages: the feeding stage; the compression stage and the metering stage. The feeding stage is where pellets are loaded into the extruder. There is a large gap between the screw and the barrel. This is needed because, as the pellets are not yet molten at this stage, there are large gaps between the pellets.

The second stage is the compression stage. In this section, the screw is tapered from an initially small diameter to a larger diameter that progressively reduces the size of the gap between the screw and barrel. In this stage, the pellets are pre-melted and soft. As they travel down the barrel, the pellets are further melted and are compressed together so they form a molten polymer while the air bubbles within the melt are removed.

In the final metering stage, the screw diameter stays constant with the end of the taper but the flutes change pitch so that extra pressure is introduced to the melt to push it past the die at the right feed rate.

Many different types of screws have been developed to improve various aspects of the extruders. There are many different parameters within the screw design that can alter its performance, and that must be taken into consideration. There are four main types of high performance screws:

1. Flow Restriction/mixing
2. Barrier-to-Melt
3. Barrier between solid bed and melt pool
4. Solid/melt mixing

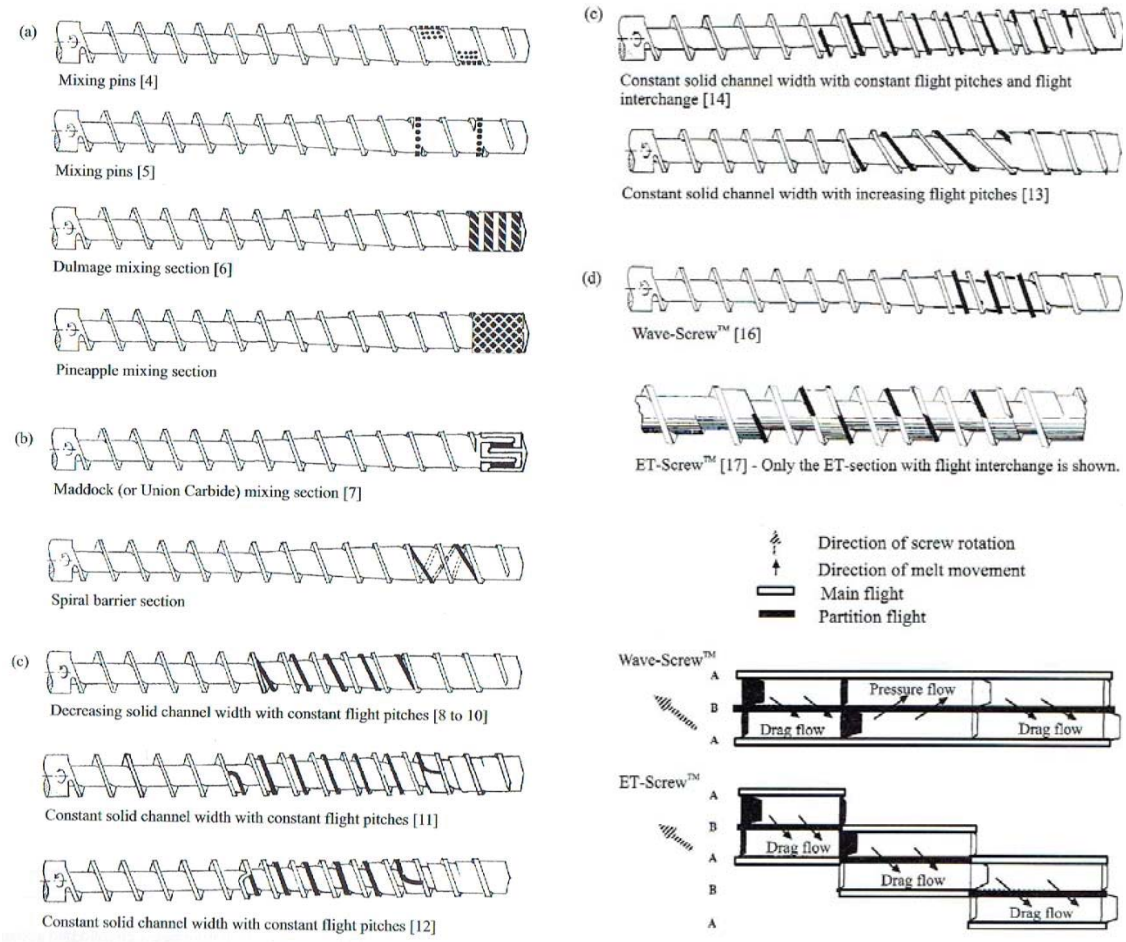


Figure 2-14: Schematics for screws: (a) Flow restriction/mixing type; (b) Barrier-to-melt type; (c) Barrier between solid bed and melt pool type; (d) Solid/melt mixing type (Chung, 2011)

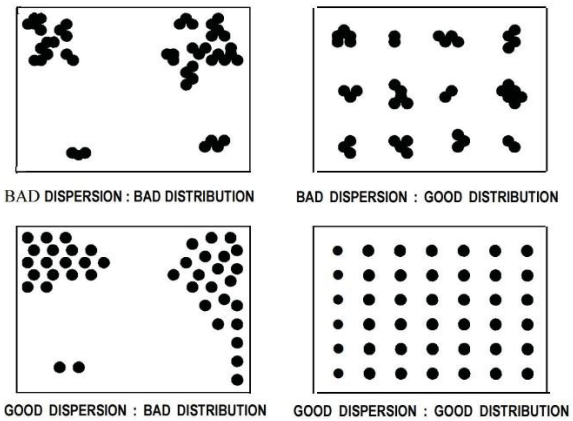


Figure 2-15: Illustration of dispersive and distribution mixers

Dispersive mixing involves reducing big clumps of polymer or additives from larger clusters into smaller components. Distributive mixing spreads the minor component of the polymer in order to create good spatial distribution of the additives within the polymer (Manas-Zloczower, 1997). The difference between dispersion and distribution mixing is shown in Figure 2-15.

2.3.3 Flow Restriction/Mixing Type

The Flow Restriction/Mixing Type screw gets its name from the pins, normally located along the metering section of the screw, which restrict the flow by splitting the melt. This was the first commercially successful high performance screw. The flow restriction/mixing type functions as a dynamic mixer. This means the polymer melt splits into many different layers to distribute the mix.

The advantages of this process include improved melt quality from mixing of the melt stream and improved pressure stability resulting from pressure build-up in the mixing section. The disadvantages of this process include little to no increase in the melting capacity of the screw and little or no increase of the dispersive mixing capability (Chung, 2011).

2.3.4 Barrier-to-Melt Type

In this type of screw, there is a very small barrier clearance between the barrier flights and the barrel surface. This allows the melt to flow through the barrier clearance at a high shear rate to blend the melt. The additives disperse through the melt as they travel over the barrier flight. The melt is split and mixed at the entrance and exit grooves of the mixing section where the barrier flight is removed to let the melt pass.

The advantages of this process include the blocking and filtering out of unmelted pellets, dispersion of the additives by the barrier flight, and dispersion due to the mixing of the melt. The disadvantages include little increase of the melting capacity of the screw, and possible degradation of the polymer in the slots or blind spots between the barrier and main flight (Chung, 2011).

2.3.5 Barrier between Solid Bed and Melt Pool Type

This type of screw has the barrier flight in the compression section. This separates the solid channel from the melt channel. As the solid pellets melt, they flow over the barrier flight into the melt channel. There is an inverse relationship between the size of the solid channel and melt channel. As the pellets move down the screw, the solid channel shrinks and the melt channel grows.

The advantages of this process include improved melt quality due to the separation of the solid and melt channels. There is also an increase in melting capacity and therefore output due to a

larger solid bed area. The disadvantages include little increase of the distributive mixing capability, possibility of the barrier flight restricting flow, and possible degradation of melt in the dead spots between the barrier and main flight or in the deep melt pool (Chung, 2011).

2.3.6 Solid/Melt Mixing Type

The Solid/Melt Mixing Type lets solid pellets mix with the melt. This is achieved by having a partitioning flight. These channels are divided into sub-channels that vary their depth. As the depth of one sub channel decreases, the next one increases. This forces the melt to flow from one channel to the next. As it flows over the partitioning flight, the melt is mixed and the core portion of the melt in one sub-channel goes to the outside of the next one. This makes for an even temperature flow throughout the melt.

The advantages of this process include increased melting capacity and output rate due to better heating through conduction from the melted pellets, lower melt temperature, and therefore higher efficiency, and uniform melt temperature from good mixing. The disadvantages include a possible stagnation of the melt in the deepest areas of the sub-channel, and little increase in the dispersive mixing capability.

2.3.7 Filament Extruder

Other continuous extrusion methods include filament-based extrusion where a spooled filament is gripped with the teeth of a gear and pulled through a heating element, turning it into a semi-liquid state that then gets pushed through a nozzle.

2.3.8 Gear Pump Extruder

Gear Pump Extruders are uncommonly used as extruders on their own but are generally attached to the end of a screw extruder to improve the quality of the extrusion. Figure 2-16 shows the gear pump works by moving a viscous fluid around a gear or gears. Generally, gear pumps work by having two gears counter-rotating. As they turn, they suck in melt between the teeth, which creates a pressure difference.

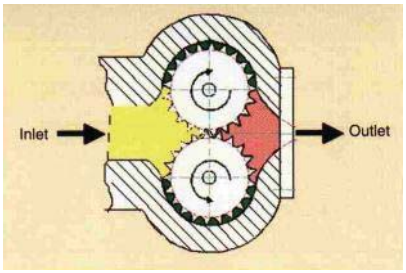


Figure 2-16: Diagram showing how a gear pump moves polymer through it (Skibba & Thoma, 2001)

This pressure differential pulls more melt in to fill the gap. Gear pumps can deliver a constant output rate of melt so are frequently used in conjunction with screw extruders (Rauwendaal, 2010).

2.4 Chapter Summary

The literature review shows additive manufacturing is a developing field with new technologies still emerging. The FDM process is well documented and research is currently being done regarding different methods, applications, and materials. There is currently little research around freeform printing in materials other than metal. The same is true for pellet-based 3D printers. There have been few 3D printers developed that use pellets rather than filament: some have been developed for research purposes, but most have been developed by hobbyist users to test new materials for their home 3D printers.

Industrial robots have been around for many years with most current research focusing on using robots for new applications and developing control algorithms. Extrusion systems are similar in that there has not been any recent radical development in this area. Lots of the research focuses on different screw designs or optimizing and improving parts of the process.

The following chapters show the development of a simple and cost effective extrusion system using biopolymer pellets in 3D printing, the development of a printer that blends the freeform and traditional 3D printing processes by printing using freeform techniques on a platform, and the development of software to create an object from a CAD model by controlling the printer.

Chapter 3: Mechanical design of the articulated platform and extruder

3.1 Platform

Current FDM 3D printing technology must deposit material onto something solid. If the model requires an overhang or cavity, the printer must build up support material for it to deposit the model material on. As the design provided is a paraboloidal shape, if printed using normal FDM techniques, it would require a support structure to print. Creating a custom platform removes the need for support material.

Having a custom platform limits the printer by restricting the designs it can achieve. It is possible to modify the look of the design, but the geometric shape of the model would stay the same as the shape of the platform, however it would be possible to create multiple platforms for different designs and swap them depending on what the user is printing.

Multiple manufacturing methods and materials were considered for creating the platform. When selecting a material and manufacturing method it is important to consider that the platform has an extremely low production run, must be able to withstand the temperature of the melted biopolymer, be cost effective to produce, and have a surface finish that allows the extruded biopolymer to stick to the platform while cooling.

Metals were considered to fabricate the platform from for a number of reasons. Firstly, they would be relatively easy to manufacture. Secondly, it would be possible to give it a surface finish that helped the deposited material to stick. It would be possible to use metal spinning rather than forming or CNC milling to reduce the amount of metal used and the cost. It would be possible to create an accurate mould with a good adhesiveness using this method. However, the metal platform could be too heavy for the 3-kg payload limit for the robot, and it is possible the platform could deform during operation from the heat.

Polymers were also considered for the manufacture of the platform. They are easy and cost-effective to manufacture. However, some polymers will deform under heat so the right selection must be made. Vacuum forming was strongly considered as a manufacturing method as it can create a near-net shape. However, the polymers available at the time were unsuitable, as they would not provide a good surface finish for the biopolymer to stick to. Ultimately, fibreglass was used because it is light and would not exceed the payload limit for the robot arm, quick to manufacture, would not burn, or deform under heat, is strong and keeps the shape over time.

The platform was designed using CAD by creating a spline that fitted the provided DXF model. This was revolved around the Z axis to create a 3D model that was exported into STEP format. From the CAD model, equally spaced points were taken along the curve. The resulting intersections provided discrete points that were placed into Matlab and used to approximate a polynomial. The higher the degree of the polynomial, the more accurately it represented the curve. The resulting polynomial equation for the platform is shown in Equation 3-1.

$$Y = (-3.309\times10^{-8})x^4 + (2.084\times10^{-21})x^3 + (-0.002343)x^2 + (-4.481\times10^{-17})x + 149.6$$

Equation 3-1: Equation of the platform

To generate the mould for the platform, ten 18mm thick sheets of MDF wood, as seen in Figure 3-1 (a), were glued together and put into the CNC machine to mill out the mould (Figure 3-1 (a) and (b)). MDF was used because it is relatively inexpensive, strong and provides a good base for a fibreglass mould. After removing the mould, it is sealed and buffed to create a smooth, waterproof finish. Mould release wax is put on so the fibreglass can be removed easily from the mould. The final mould is shown in Figure 3-1 (d).

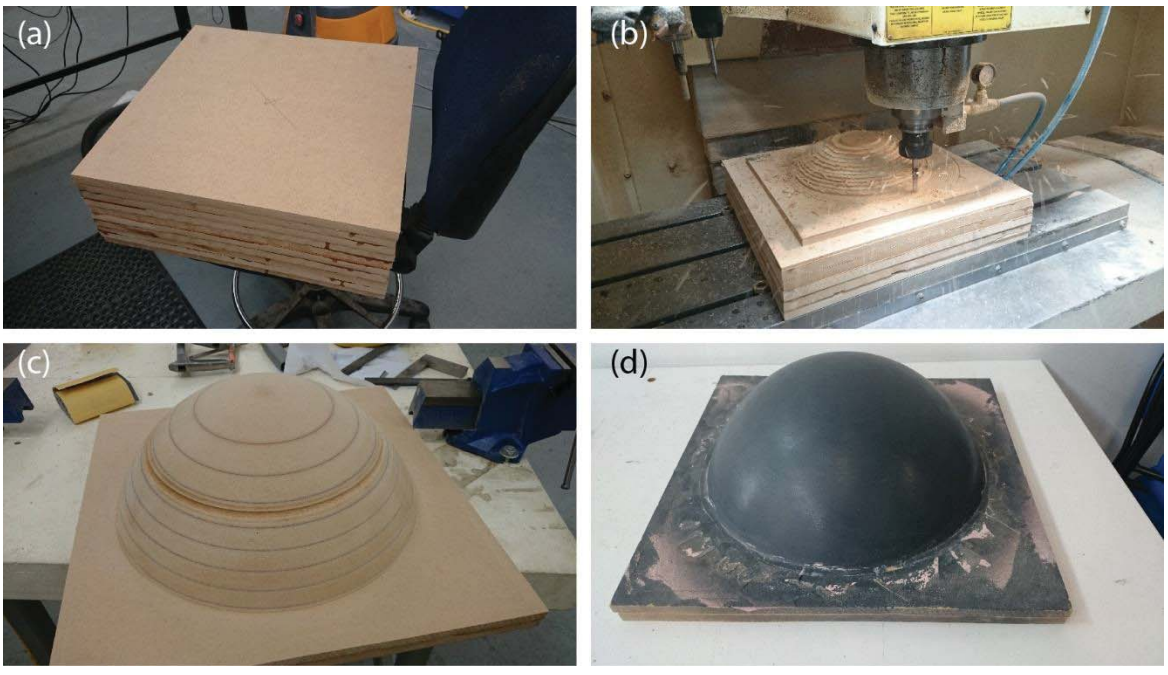


Figure 3-1: Process to create the mould. (a) Laminating MDF sheets, (b) CNC milling of mould, (c) sanding and filling required areas, and (d) sealed and waxed mould

The fibreglass sheets were draped over the mould and the resin applied. This was done multiple times as each layer increased the strength of the mould. The fibreglass was removed from the mould and excess bits cut off. The edges were then sanded smooth to stop the user from hurting themselves.

When pressure was applied to the fibreglass, there was localised buckling. This was resolved by inserting ribs, that can be seen in Figure 3-2 (b), to increase the rigidity of the platform. It was attempted to use expanding foam to help increase this effect. However, because it was found to be creating such a large pressure on the platform in some areas that it had changed the overall shape of it, the foam was removed. The final platform can be seen in Figure 3-2 (a).

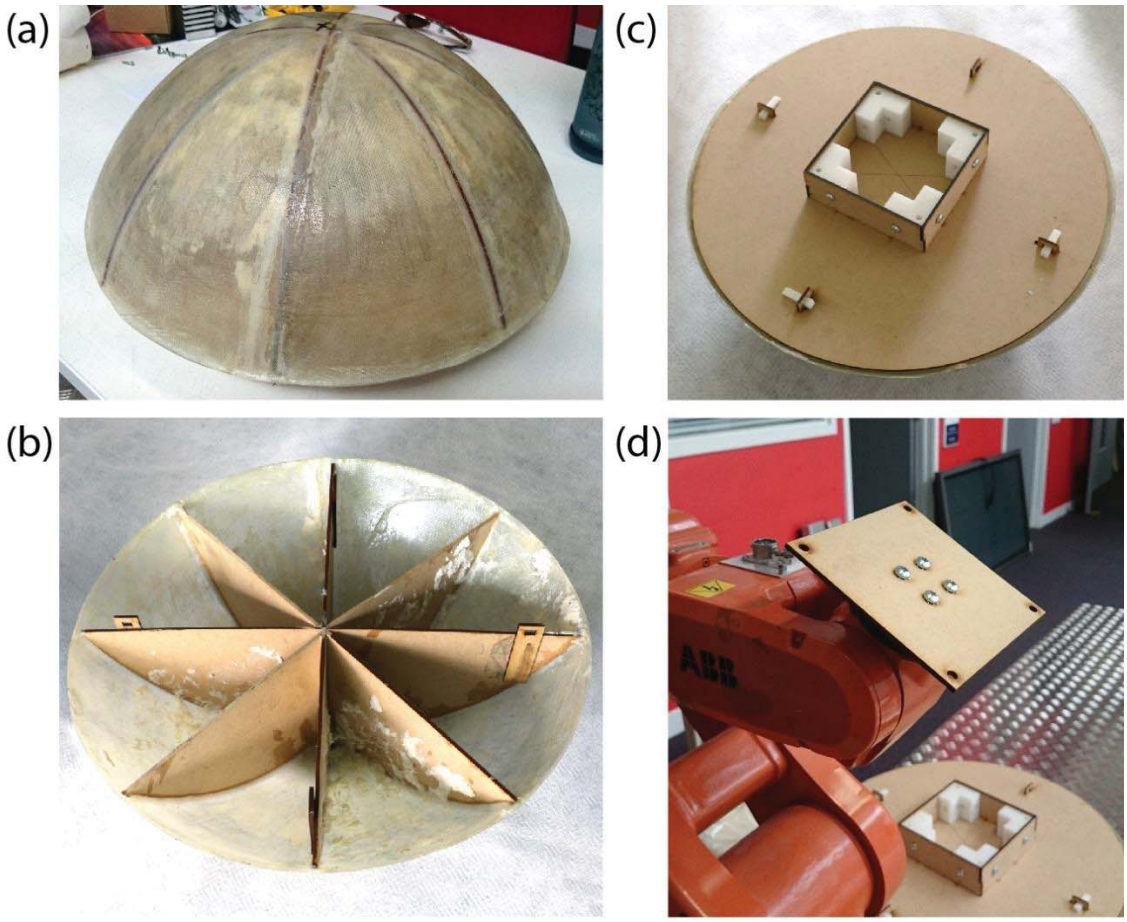


Figure 3-2: (a) Fibreglass platform, (b) ribs inside platform, (c) platform base with locking mechanism, and (d) bracket to attach platform to robot

The platform was then attached to a base consisting of laser cut MDF and 3D printed parts. These parts lock together using screws, wedges and 3D printed parts as seen in Figure 3-2 (c). This base is fixed to the robot's end effector with screws, using the attachment bracket seen in Figure 3-2 (d). The final platform attached to the robot is shown in Figure 3-3. The platform must be rigid and unable to wobble so that when the robot is moving, the platform does not shake, which could lead to small design changes. The platform can be seen attached to the robot in Figure 3-3.

Figure 3-3: Robot with platform attached

3.2 Robot arm

The IRB 120 Industrial robot, manufactured by ABB, is an articulated robot arm and currently the smallest robot arm ABB produces. The IRB 120 is a six degree-of-freedom robot with a payload of 3 kg. Having six degrees of freedom allows the IRB120 to have a full range of motion in all directions. The robot is attached to a table to secure it.

The second part of the robot is the controller, shown in Figure 3-4. This contains the power and signal distribution systems, and a programmable controller. This controller is programmed by the user to run the robot through logic operations. The controller interprets the user's program and works out the best path along which to move the robot. It then sends signals to the robot to turn the motors to the correct position at the right speed to deposit the correct amount of material on the platform.

Figure 3-4: Robot controller

The controller is connected to the computer through Ethernet. It is possible for this to either be direct from the controller to the computer, or through a router. The benefit of using a router is the controller can be accessed through Wi-Fi or off site. The benefit of having the computer connected directly is that it has much faster transmission speeds, and less likely to drop the data.

The Flexipendant (Figure 3-5) connects to the controller and consists of a touch screen interface, some buttons, and a joystick. It provides one of the ways the user can interact with the robot. Using the Flexipendant it is possible to move the robot through Cartesian space, or each axis individually. It is possible to start, stop, pause, and resume programs through the Flexipendant.

Figure 3-5: ABB's Flexipendant

The Flexipendant also provides feedback to the user. When running a program, it is possible to view the current XYZ position of the end effector, along with the quaternions or joint angles for each axis. This is helpful in debugging the software to make sure it is moving to the correct position.

This workspace is the volume of space the robot can reach and is defined by how long each link is and the joint range it can move to. The robot can reach every coordinate within the workspace. However, when a tool is attached to the end effector of the robot, the workspace changes. The IRB's workspace is shown in Figure 3-6. It was chosen to use the area marked in red for the printer to work in as it provides enough movement space, and a good configuration that will not need many changes as it moves.

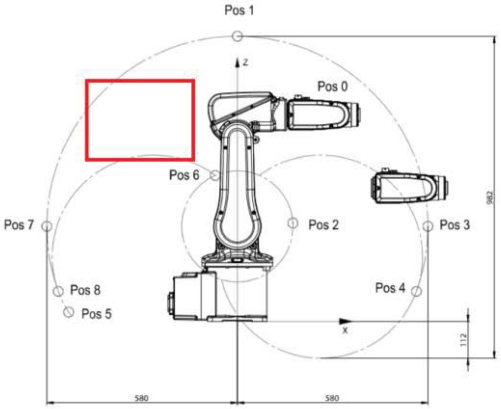


Figure 3-6: Workspace of IRB 120

The link lengths for the robot are:

Table 3-1: Link lengths for the IRB 120

Link	Length
1	103 mm
2	270 mm
3	70 mm
4	302 mm
5	72 mm

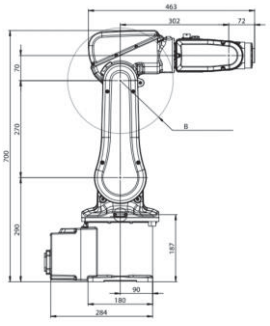


Figure 3-7: Link lengths of IRB 120

The maximum joint ranges and speeds for each joint of the robot are shown in Table 3-2. It is not possible for the robot to exceed the maximum joint range because it will cause damage to the robot.

Table 3-2: Joint ranges and Max speed for IRB 120

Axis	Maximum joint range	Maximum speed
1	+166° to -166°	250 °/s
2	+110° to -110°	250 °/s
3	+70° to -110°	250 °/s
4	+160° to -160°	320 °/s
5	+120° to -120°	320 °/s
6	+400° to -400°	420 °/s

ABB's IRB 120 has a repeatability of 0.01 mm meaning each time it moves to the same point it will be within 0.01 mm of its target destination. This is an incredibly important feature of the robot. As the application for this robot is a 3D printer, it will be working with fine tolerances. To repeat the design accurately for multiple prints many factors must be considered. However, if it is assumed that the extrusion is consistent over multiple prints, the major factor affecting the repeatability of the printer is the accuracy and repeatability of the robot. If the robot can move to the same point for every print, the design will be the same, even after thousands of runs. If the repeatability of the robot is low, the design will change very slightly from one to the next so after a large number of runs, it may be noticeably different from the original design. This can be countered by recalibrating the robot after a certain number of runs.

3.2.1 Robot control

RobotStudio is an application created by ABB to program their robots (Figure 3-8). Through RobotStudio, it is possible to simulate the robot's program before running it. This helps to avoid potentially dangerous situations for both the robot and the user as any issues can be rectified before applying the program to the robot.

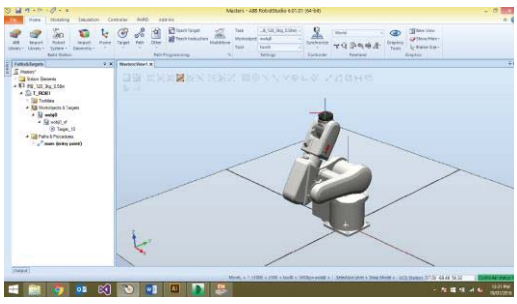


Figure 3-8: ABB's RobotStudio

ABB developed the programming language RAPID to control their industrial robots. The targets for the robot to move to can be set through RobotStudio, programmed manually through RAPID code or taught to the robot through the Flexipendant.

3.2.1.1 Robot positioning

Position data is used to define the position to which the robot will move to. ABB robots are positioned using either a robtarget or a jointtarget data structure. However, it is possible to convert between the two.

The jointtarget structure consists of the angles of each internal and external joint. External axis are ones connected to the end effector of the robot and can be controlled through robot studio. The jointtarget structure is:

jointtarget [robax, extax]

Where “robax” is an array consisting of the angles in degrees for all five robot joints and “extax” is an array of positions for each of up to six external joints. If the joint is rotational, it is the degrees of rotation, if the joint is linear, it is the distance for the joint to move to.

The robtarget structure consists of a Cartesian position of the tool and orientation of the tool expressed in the form of a quaternion. The robtarget also has a parameter for the configuration data of the robot and external axis. It is possible to reach some positions in the workspace while having the robots joints in different angles so it is necessary to specify the configuration the robot uses to reach these positions. The robtarget structure is:

robtarget [trans, rot, robconf, extax]

Where “trans” is the position data in the form of a 3D Cartesian coordinate of the tools centre point, “rot” is the orientation of the tool, “robconf” is the axis configuration of the robot’s axis, and the “extax” as defined in the jointtarget.

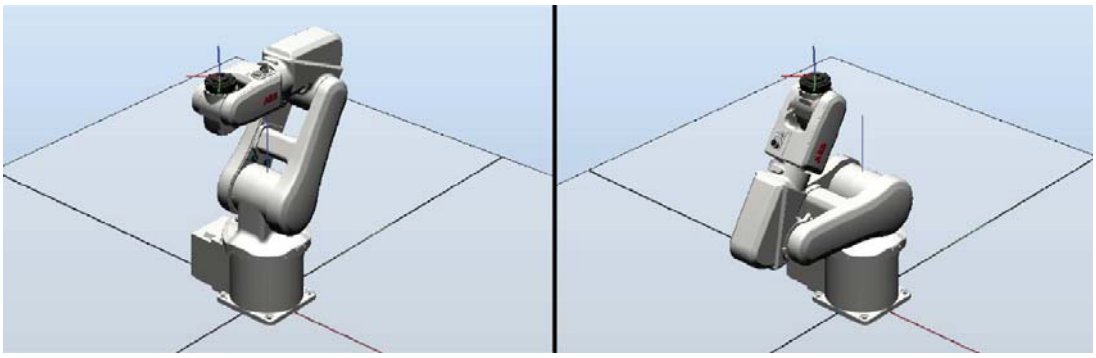


Figure 3-9: Different robot configurations for the same point

When using robtargets, the configuration of the robot's axis is important. As the robot can use multiple configurations (Figure 3-9) to reach the same point and the joint angles are calculated as the robot moves, it is possible for the robot to calculate a path that will lead to singularities. The configuration data is useful to stop this from happening. The configuration data tells the robot which way it should be oriented at a certain point. This stops the robot from large switches between orientations as it is moving. However, having the orientation set can limit the robot's movement. If the robot is moving and cannot reach a point with the specified configuration, the robot will stop and produce an error message.

The ConfL command specifies whether the robot monitors the configuration during linear or circular movement. If the configuration is not monitored, the configuration at execution time may differ to the programmed configuration. It can also result in unexpected motions as the robot switches to joint motion.

The ConfJ command specifies whether the configuration of the robot is controlled during joint motion. With this turned off, it will move to the closest axis configuration, which may be different from the programmed one. With this on, if the robot cannot move to the programmed axis configuration using the programmed position and orientation, the program stops.

Without using RobotStudio, the configuration data must be determined by using the inverse kinematics of the robot to get the joint angles for a certain position and orientation of the tool centre point and from that, determine the robot's axis configuration. It was chosen to turn off ConfJ and ConfL for two reasons – firstly, because the determination of the configuration data would slow down computation during the runtime of the printing software, and secondly, if the configuration was incorrect and being monitored during movement, the printer could stop part way during a print.

3.2.1.2 Robot Movement

The RAPID language includes multiple commands, seen in Table 3-3, for moving the robot. All of these commands have location, speed, zone, and tool parameters. The location parameter contains either the joint angles, or position and orientation data for the robot. The speed parameter contains the speed for the robot to move to the location. The zone parameter defines how close the robot will get to the point before moving to the next one. This application sets the zone data to fine so the robot will move to the exact point before moving to the next one. The tool parameter sets which tool is attached to the robot. For this application, it is set to default as the algorithms in Section Chapter 4: rely on calculating the end effector coordinates without anything attached to the robot.

Table 3-3: Table of RAPID movement commands

MoveL	*MoveL [\Conc] ToPoint [\ID] Speed [\V] | [\T] Zone [\Z] [\Inpos] Tool [\WObj] [\Corr]*
	To move the robot's end effector linearly between two points, the MoveL command is used. The movement is in a straight line, unless the robot must take evasive action for a singularity.
MoveJ	*MoveJ [\Conc] ToPoint [\ID] Speed [\V] | [\T] Zone [\Z] [\Inpos] Tool [\WObj]*
	The MoveJ function moves the robot quickly from one point to another along a nonlinear path so that all robot axis reach their destination position at the same time.
MoveAbsJ	*MoveAbsJ [\Conc] ToJointPos [\ID] [\NoEOffs] Speed [\V] | [\T] Zone [\Z] [\Inpos] Tool [\WObj]*
	The MoveAbsJ function is used to move the robot to an absolute joint position. The robot axes move along a non-linear path to the destination position, meaning they all reach their destination position at the same time. The difference between this function and MoveJ is that the tool, work object, or active program displacement does not affect the final position of the robot. Active program displacement is where the coordinate system is relative to the tool centre point, and changes as the robot moves. MoveAbsJ ignores this and uses a static coordinate system.
MoveC	*MoveC [\Conc] CirPoint ToPoint [\ID] Speed [\V] | [\T] Zone [\Z] [\Inpos] Tool [\WObj] [\Corr]*
	The MoveC command moves the tool centre point circularly to a given destination. The command uses three points to determine the circular motion to use. A start point, end point and a circular point. The circular point, generally chosen to be half way between the start and end point sets the curvature of the path.

3.2.1.3 Communication

Microsoft Windows provides mechanisms to facilitate communication between applications, called Interprocess communications (IPC). The ABB PCSDK utilises these communication mechanisms to send data or messages to and from the robot's controller (ABB Robotics, 2012).

On both the computer and the robot controller there is a message queue with a default maximum of five slots. It is possible to change this. The messages from the sender are sent to the queue of the receiver. The receiving client reads the message from the queue and the software and the user's software interprets the received data.

As the robot is connected through Ethernet to the computer, naturally it uses the TCP/IP network protocol to communicate between them. The TCP protocol uses a client-server model where one computer requests to connect and send information to the other. TCP includes methods for flow control, packet corruption detection, congestion control, and reliable transmission. This ensures that the data transferred arrives in the correct order, with no corruption or buffer overflows from receiving too much data or receiving it too fast.

ABB allows the robots to read messages using two methods. The first method is synchronous reading, which is where messages are added to a queue as they arrive. If the queue is full, the message is discarded. The messages are removed from the queue in the order they arrived by a read function called by the user. The messages are only read when this function is called. Because of this, there is the possibility of delays as the program has a set amount of time between checks. The second method is using interrupts. This is the preferred option as it does not require waiting between reads and each command is interpreted and run through an Interrupt Service Routine. The interrupt method was chosen to use as the method of reading commands because printing process is dependent on time.

The process to read messages through interrupts in RAPID code is to attach the ISR to the interrupt. Each ISR must be attached to a different interrupt. Two types of data are sent from this software, speeddata that sets the speed at which the robot will move, and robtargets for the robot to move to. Code Snippet 3-1 shows the control algorithm. The robot is acting as a slave, waiting for the interrupt to be triggered before doing any action.

```
Main()
      Connect moveRobot to interrupt 1
      Connect setSpeed to interrupt 2
      While(true)
            Do nothing
      End
End
```

Code Snippet 3-1: Main function of robot controller

The ISR to set the speed of the robot is shown in Code Snippet 3-2. The routine parses the message header and places the speeddata in a global variable to be accessed when moving the robot.

```
setSpeed
      Read Sent Message
      Extract Header from Message
      If (Header type == speeddata)
            Set Arm Speed
            Send Acknowledgement
      Else
            Send Error (Incorrect data)
      End
End
```

Code Snippet 3-2: Interrupt service routine for setting the robot's movement speed

The ISR to move the robot is shown in Code Snippet 3-3. This routine parses the message header. If the orientation is [0, 0, 0, 0], the robot moves to the home position. (0, 0, 0, 0) is an invalid quaternion and is used as a way to tell the robot to go to the home position instead of using a third interrupt. In all other cases, the robot moves to the provided robtarget.

The move takes place inside the ISR because if it were run in the main loop, there is the possibility of the ISR running and updating the target point before the robot is moved. By having it inside the ISR, it guarantees this will not happen.

```
moveRobot
      Read Sent Message
      Extract Header from Message
      If (Header Type == robtarget)
            Set Target Position
            Move Linear (position, speed)
            Send Acknowledgement
      Else
            Send Error (Incorrect data)
      End
End
```

Code Snippet 3-3: Interrupt service routine to move the robot

After the robot has moved, it sends an acknowledgement back to the software, which lets it know it is available for the next piece of data. This helps to keep the message queue from overflowing and losing data.

3.3 Hopper and Extruder design

3.3.1 Pellet versus filament extrusion

As discussed in 0 there are few pellet-based 3D printers, and currently none available commercially. Current direct extrusion 3D printers are modified versions of existing printers. Because direct extrusion negates the need for filament, it provides a number of advantages.

There is less cost involved as the pellets do not need to be turned into filament before extrusion. The polymer is heated a fewer number of times which reduces the thermal stress the material is subjected to. With each thermal process that the polymer is subjected to the chance of degradation of the polymer increases. With higher levels of degradation, the polymer breaks down and behaves unpredictably, giving different physical properties and potentially altering the quality of the print. If too much degradation occurs, the polymer can burn.

Using pellets also reduces the possibility of contaminants and impurities being introduced into the material. As creating filament is an extra process, it is an additional chance for contaminants to be introduced. If added, these can modify the structure, look, and integrity of the extruded material.

Figure 3-10: Pellets used for printing

Direct extrusion also allows the material to be changed easily. As the pellets, seen in Figure 3-10, are a mix of PLA polymer and harakeke (flax) fibres, the ratio is easily modified by adding more of one or the other. Currently in the biopolymer the ratio is 5 wt.% harakeke fibre. Changing the ratio gives different physical and aesthetic properties to the extrusion. With filament, it would not be possible to change the ratio without creating a new filament.

As the pellets are mixed as they are being extruded, the ratio of PLA to harakeke fibre can vary throughout the extrusion. This can lead to uneven distribution in fibres along the design. This gives the design a desirable organic look.

One of the advantages of filament is that there are no tolerance issues when extruding. When using pellets flow consistency can change while the extruder is running due to blockages, pressure changes, temperature and humidity differences, and many other factors. Not all of these can be controlled to the point required to get a perfectly constant flow, without increasing cost.

The use of filament also offers more control over the extrusion. The methods used to print from filament can be controlled to a finer degree than pellet extrusion methods. As a high pressure is not required to print using filament, when the extruder is stopped, very little material keeps flowing. This contrasts with a pellet-based extruder, which relies on pressure to compress and mix the pellets.

Ultimately, it was decided to use a pellet-based extruder because of the advantages it provides, along with the organic look of the material it produces.

3.3.2 Stationary versus moving extruder

The printer could be made with either a moving extruder with a stationary platform or a moving platform with a stationary extruder. With a stationary platform, the robot is moving rather than the extruder. The advantages and disadvantages of a moving extruder can be seen in Table 3-4 and the advantages and disadvantages of a stationary extruder can be seen in Table 3-5.

Table 3-4: Pros and cons of moving hopper

Moving Extruder	
Pros	**Cons**
• Easier to orientate the hopper than the platform	• Pellets move around • Wires and cables must run along robot with a high chance of tangling • Harder to refill the hopper during a print without a tube • Inconsistent flow of extrusion

Table 3-5: Pros and cons of stationary hopper

Stationary Extruder	
Pros	**Cons**
• Pellets don’t move around so extrusion stays relatively consistent • Easy to refill hopper during printing	• Platform is heavier • Larger so must be careful not to hit frame

Weighing up the advantages and disadvantages of both approaches, the stationary extruder will give a better continuous quality of extrusion.

3.3.3 Previous extruder design

As this project was taken on from a previous student, a pellet extruder had already been designed. The previous design, illustrated in Figure 3-11, consisted of a turned aluminium extruder with a brass nozzle attached. The hopper was laser cut from 3mm thick acrylic sheet. The screw was an M15 auger drill bit and turned by a 12 V motor. The extruder was heated with a single Watlow barrel heater with an integrated thermistor, driven by a Watlow EZ-Zone temperature controller. The extrusion speed was controlled through a 12V motor controller with a potentiometer providing manual speed adjustment via a knob that could be turned.

When testing, and reviewing this design, several problems became apparent:

- Pellets formed clumps inside the hopper chamber
- Pellets got stuck along the flat bottom of hopper
- Inconsistent flow from the extruder
- Hopper base warped from the heat

The following sections describe the processes and steps taken to solve the problems found and improve the quality of the extruder system.

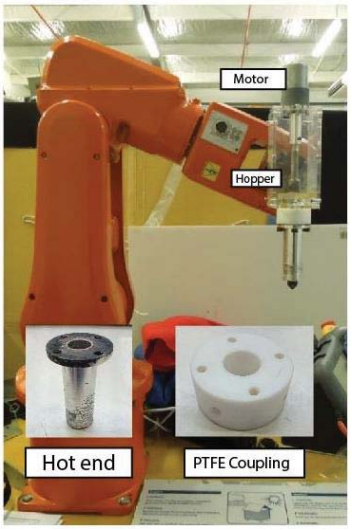


Figure 3-11: Previous student's extruder

3.3.4 First design

After finding the faults in the previous design, aspects of the design were modified to solve them. The hopper section was redesigned to prevent the pellets getting stuck on the bottom and therefore clumping. The hopper material was changed to stop it from warping due to the heat. The flow inconsistency was improved by replacing the motor and controller with a different system that involved feedback.

The initial redesign of the extruder used most of the original components. The heating band, temperature controller, auger, nozzle, and motor from the previous extruder design were used. The hopper, seen in Figure 3-12 and Figure 3-13, was redesigned to be 3D printed from nylon rather than acrylic to stop it warping from the heat. The new hopper also included an entry funnel to refill it easily with pellets while printing. It also featured a conical funnel that can be

seen in Figure 3-13, to channel the pellets from the hopper into the extruder. The idea behind the funnel was to stop the pellets from clumping by not letting the pellets stay at the bottom of the hopper.

Figure 3-12: CAD model of the extruder

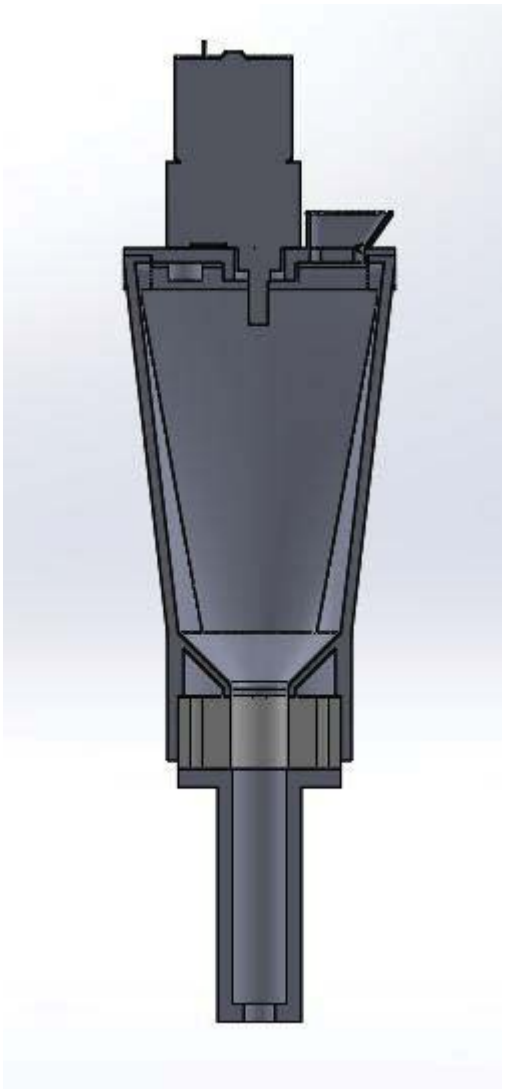


Figure 3-13: Cross section of hopper section with coupling ring

To drive the screw, a 200 RPM, 0.49 Nm motor, with a stall current of 5 Amps, was chosen because the original one did not have an encoder to provide feedback to measure the motor's speed, nor anywhere to attach one. As there were no specifications for the motor, these specifications were estimated by measuring the speed and current of the previous motor.

An Arduino Uno, like the one in Figure 3-14, replaced the original motor controller. The Arduino Uno was chosen because it is easy to program, relatively cost effective, with a serial communication bus and interrupt capability. Others considered were the raspberry PI and the Texas Instruments Tiva C. These were rejected because the application did not require the amount of processing power or high-level programming they offered.

Figure 3-14: Arduino Uno (Aduino Inc, n.d.)

To power and control the motors, a motor shield was chosen over creating a custom circuit. A shield provides better safety protection, smoother voltage flow to the motor, and reduces development time of the extruder.

Figure 3-15: Motor shield (RobotShop, n.d.)

The motor shield (Figure 3-15) provides an easy way to control the speed and direction of the motor. The speed is controlled by pulse width modification (PWM) from the Arduino. The direction is controlled by the setting the status of a specified Arduino pin. "Low" corresponds to one direction, while "High" corresponds to the other. The motor shield also provides power to the motor through a power pack. The board is rated to provide a maximum of 10 Amps, which lets a wide range of motors be used. The power pack supplies 12 V at 10 A so is able to provide up to the maximum current of the board.

The custom PCB, for which the schematic and PCB diagram can be seen Appendix B, is used to connect all the components together. It consists of header pins for the fan, motor and Arduino, terminal blocks to connect the power to the motor shield, and a jack for the power pack to plug into. They are built into a stack to create the final motor controller as seen in Figure 3-16.

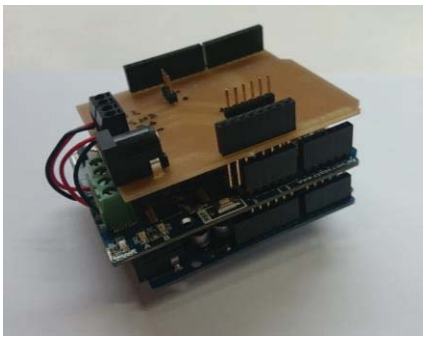


Figure 3-16: Extruder motor controller

Results

This extruder was tested and through observation, it was found that it solved the problem of not having a pellet feed but did still have the clumping problem. This showed the stuck pellets were not the cause of the clumping and the issue needed to be investigated further to find the cause. The flow consistency was increased with the new controller by utilising the feedback from the motor. However, the motor would occasionally stall because it would draw over 5A from the power supply when the torque required was too high. This sometimes led to inconsistent flow and burnt melt.

3.3.5 Second design

When the extruder was pulled apart immediately after being run, it was found the pellets were melting all the way up the screw, and even into the hopper. This was the cause of the clumping issue and was because the screw was getting too hot. The second design of the hopper was created to solve this issue.

Figure 3-17: Prototype heat sink

The heating band was moved closer to the nozzle but did not make much of an impact on the clumping problem, so a heat sink was designed that would draw the heat away from the screw. The heat sink was developed to use a 40 mm, 12 V fan with a flow rate of 15.96 CFM. This fan was chosen because 40 mm is small enough to fit on the project, 12 volts allows it to be powered from the same source as the rest of the system, and had a high flow rate that would allow it to draw more heat out. The heat sink was prototyped from ABS, a common polymer used for FDM 3D printing, to ensure sufficient airflow around the screw. This prototype is shown in Figure 3-17.

Using the ABS heat sink (Figure 3-17) gave a lower temperature at the same point on the screw, after the same amount of time, than with the first design. However, the airflow around the heat sink was restricted due to the support for the screws. A new hot end, seen in Figure 3-18, with an integrated heat sink was designed and tested. The tests showed a far greater improvement to the temperature reduction in the screw provided due to the heat sink. Figure 3-19 shows the progression of the process along the screw. At the top, the material is partially melted. As the material progresses down the screw, the material is further melted and mixed together. In the first design, the pellets would be clumping together at the top without moving through the screw.

Figure 3-18: Close up of hot end

Figure 3-19: Screw after being used

A new hopper with more volume, easier assembly and that is easier to fill was designed and 3D printed to fit onto the new hot end. The PTFE coupling ring was removed from the design as the heat sink provided better thermal insulation. The final design can be seen in Figure 3-20.

Two new motors were selected to try on the extruder: a 41 RPM, 0.25 Nm brushed DC motor running at 150 mA, and an 11 RPM, 0.93 Nm brushed DC motor running at 120 mA. The 41 RPM motor worked well overall but had some issues with consistency. When the 11 RPM motor was trialled on the extruder, the torque proved sufficient to replace the existing motor. This motor ran the extruder without any trouble with excellent flow consistency.

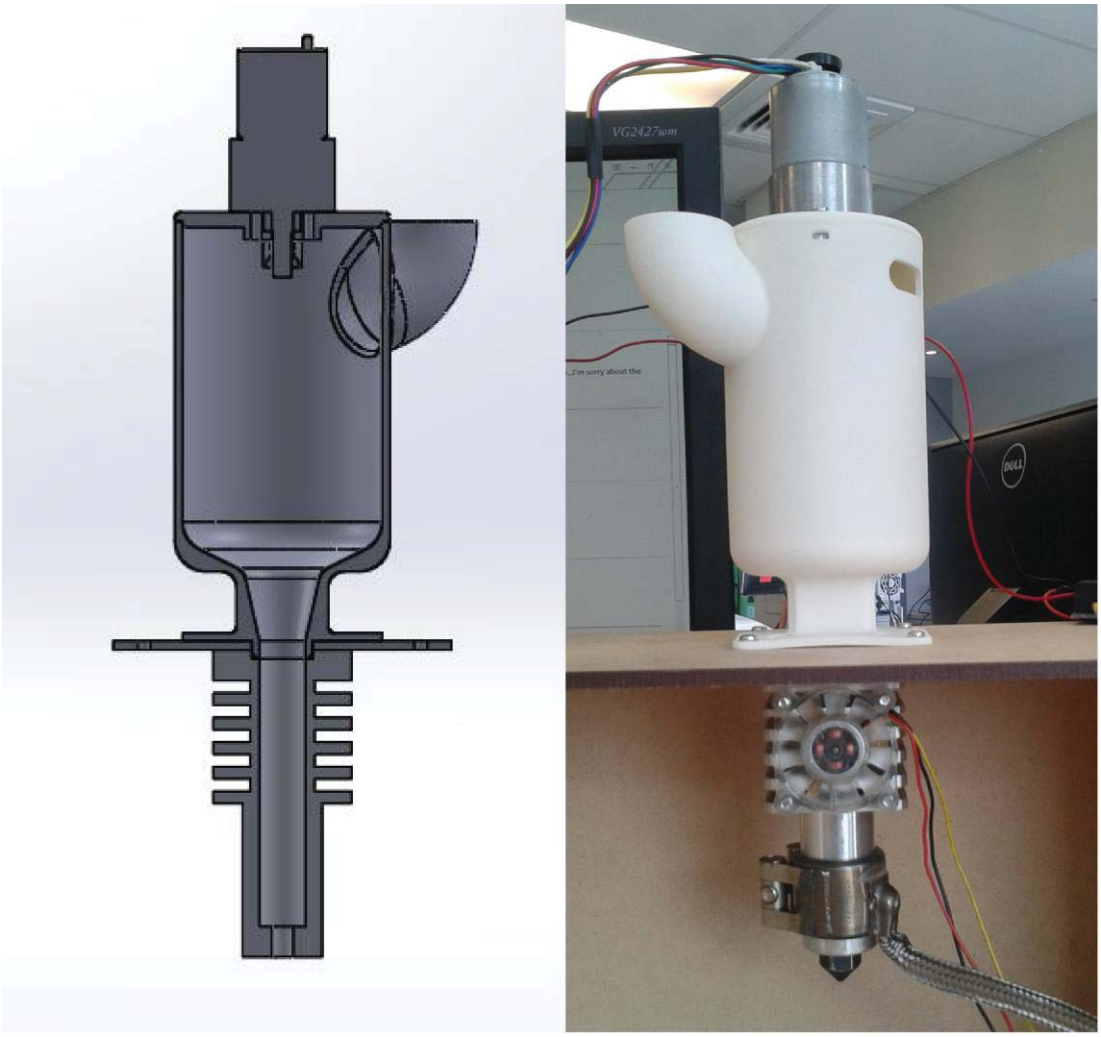


Figure 3-20: cross section of CAD model of extruder, and extruder attached to test platform

Results

This new design greatly increased the functionality of the hopper and completely removed the clumping problem from previous designs. The heat from the screw is transferred into the hot end body. The heat sink draws the heat out from the body and the fan passes air over it to cool it down. This process keeps the screw cool enough to prevent it from partially melting the pellets.

3.3.6 Minor improvements

After being attached to the frame, the printer was tested and minor adjustments were made to improve the hopper and extruder. The fan was insulated from the heat sink with thick cardboard as it was warping the fan's casing. Cardboard was chosen as it has a low thermal transfer rate and high self-combustion point making it an ideal insulator for this application. This slowed down the time taken to warp the fan, but did not prevent it from happening. A 3D printed bracket was created to move the fan away from the extruder but while still supplying a cooling effect. The bracket can be seen in Figure 3-21.

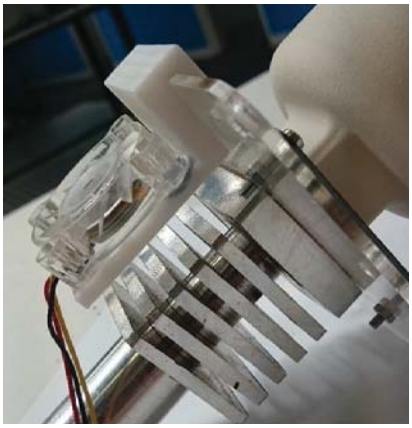


Figure 3-21: Fan bracket

Metal plates, seen in Figure 3-23, were laser cut and placed over the top of the section that joins the hopper to the extruder. The pressure build up was forcing the two apart as the 3D printed nylon is flexible. Adding these plates stopped the hopper and extruder from separating. A new 4 RPM, 1.4 Nm, worm drive brushed DC motor (Figure 3-22) was used to replace the 11 RPM motor. This gave much less aeration in the melt as it is extruded and excellent flow consistency.

Figure 3-22: Worm gear driven 4RPM motor

Figure 3-23: 2 mm laser cut steel plates

3.3.7 Heating

The heat is provided by a Watlow barrel heating band, attached to the extruder as shown in Figure 3-24 (b), controlled through a Watlow EZ-ZONE PM temperature controller shown in Figure 3-24 (a). This controller uses PID control to adjust the temperature of the band. All settings for the controller are set by using the buttons on the front. There is no way for this controller to be connected to the computer or a micro-controller. The temperature controller is powered by mains voltage. The heating band includes a thermocouple to send temperature information back to the controller.

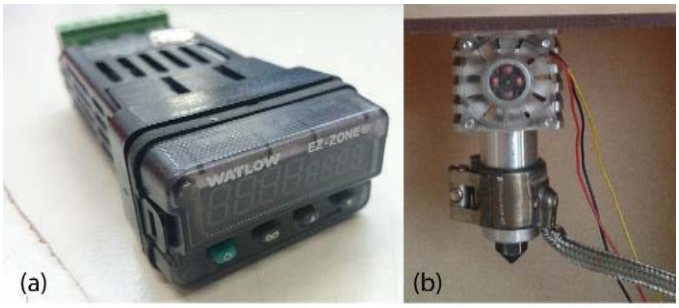


Figure 3-24: Heating components. (a) PID heating controller, and (b) Heating band

3.3.8 Extruder control

The software for the controller was written in the Arduino IDE and has three sections to it; decoding commands, feedback, and speed control. The software runs a main loop, after initialising everything, which checks if any commands have been sent, decodes them, calculates the speed and PID values, and turns the motor on at the correct speed.

3.3.8.1 Commands

Commands are sent from the computer software to the Arduino over the serial communication protocol. Serial communication involves sending one bit at a time, sequentially, until the message is completed. Arduino includes a serial library to communicate, read, and write messages between itself and the computer. A number of commands, listed in Table 3-6, were created for the extruder.

Table 3-6: Commands to control the extruder

Command	Description
“FORWARDS”	Sets the motor direction to forwards
“REVERSE”	Sets the motor direction to reverse
“START”	Starts the motor turning
“STOP”	Stops the motor turning
“RPM,X”	Sets the speed of the motor
“PULSEREV,X”	Reverses the motor at full speed for a specified amount of time before continuing.

The “FORWARDS” and “REVERSE” commands control the direction of the motor by setting the motor shield’s direction pin to the specified state.

The “START” and “STOP” commands toggle a software flag that is checked once every loop. If the flag is set, the motor is turned on. If the flag is not set, the motor is turned off.

The “RPM,X” command sets the speed of the motor. The X value in the command is the RPM to set the motor to. The maximum X value in this command is dependent on the motors max RPM and may be required to change if a different motor is used.

The “PULSEREV,X” command reverses the motor at full speed for a set amount of time before setting it back to its original forward speed. This function reduces the pressure built up inside the extruder so when it moves from the end of one path to the start of the next there is no excess material to remove at a later stage. The X in the command specifies the number of milliseconds to reverse the motor.

3.3.8.2 Feedback

Feedback from the extruder is important so the flow of the melt remains constant throughout the process. Having feedback allows the extruder to compensate for blockages, heat fluctuations, gas bubbles, and other factors that can affect the flow of the melt.

The motor has an encoder attached to it so the controller can determine the current speed of the motor. The encoder has a stationary hall sensor to detect the field of a magnet that is rotated by the motor. Each time the motor rotates a certain amount the hall sensor is activated and a pin on the Arduino is toggled to reflect the state of the sensor.

Each time a pulse is sent, a hardware interrupt on the Arduino is triggered. This interrupt is attached to an Interrupt Service Routine (ISR) that makes the program halt its current process, run the ISR, then continue with the original process.

Inside the ISR, the count of the number of pulses is incremented. The reason this is in an ISR is so no count is missed. If the status of the pin is read from the main loop and the loop takes longer than the time for the encoder to switch states more than once, counts will be lost, which results in the calculation of an incorrect speed. Having the ISR ensures no counts are missed.

In the main loop of the program, the algorithm, seen in Code Snippet 3-4 below, takes the current encoder count and converts it into the RPM of the motor using the number of counts per revolution of the encoder, gear ratio of the motor, and time between last speed calculations. The RPM is only updated if it has been 200ms after the last calculation. This stops the speed from updating too often and allows a reasonable time for the count to increase, which reduces the error in the calculations.

```
calcSpeed(void)
   time = Time Since Last Calculation
   If (time >= 200)
      RPM = count * 1000 * 60 / (Counts Per Revolution * Gear Ratio * time)
      Reset Encoder Count
   End
   Return RPM
End
```

Code Snippet 3-4: Algorithm to calculate speed from the encoder

When attaching a different motor, it may be necessary to change the counts per revolution and gear ratio variables to fit with the new motors properties. The code will also need to be modified to change the maximum RPM.

3.3.8.3 PID control

To control the motor's speed, proportional-integral-derivative (PID) control is used. PID makes use of the error between the set point and the measured value. Proportional control accounts for the current error, while integral control uses past error values and derivative control predicts future error values using the rate of change.

PID control is used rather than proportional control alone as it offers a smoother and more accurate output to the motor and therefore better flow control and quality of extrusion.

The proportional term produces an output directly related to the value of the current error. If the gain of the proportional term is too little, the rise time of the system to get to the setpoint can take too long. If the gain is too large, the system will oscillate around the setpoint. With proportional only control, there will always be a steadystate error so the system will never quite reach the setpoint.

Adding integral control will eliminate this steadystate error. However, as integral control relies on the accumulation of errors, there is a high chance of this overshooting and then oscillating around the setpoint. The potential for this can be reduced by not using a high value for integral control.

Derivative control can also be used to reduce any oscillation. Derivative control looks at the rate of change of the error over time. The larger the rate of change, the larger the derivative term is. Derivative control reduces the settling time and stability of the system.

Using all three types of control together generally provides the best system response possible. An external library for Arduino was used to do the PID calculations for the motor. Using an external library ensures fast and accurate calculation, and makes it easier to write the code.

The PID values were chosen by using an external library that automatically tunes the values for the system. These values are used as initial values and tweaked based on the response of the system. Each motor requires different PID values as they all have different system responses. The different responses in two of the same type of motor come from physical differences in the motors themselves, e.g. different resistance of the motor, differently wound stator or rotor, or different amount of backlash in the gear train. Other factors that can affect the response time are different motor types, loads, and gear ratios. Values of 2 for proportional gain, 0.5 for integral gain, and 2 for derivative gain gave a fast, accurate, and smooth response for the 11 RPM motor.

3.3.9 Limitations

The extruder must remain upright, or close to upright. If it is angled too far, it could modify how the melt flows, both through the extruder and after it has exited the nozzle. The extruder can manage slight variations in angle but if angled too far the melt can get caught on the outside of the nozzle, which would cause the print to fail.

The extruder's performance can be affected by fluctuations in the ambient environment. The two factors that affect the extruder the most are temperature and humidity. Both of these can

vary quite a lot from day to day. The temperature difference can be somewhat regulated through air-conditioning. However, the humidity is much harder to control without having a well-controlled, closed system.

Without modification to the code, printing on vertical or undercut surfaces is not supported. Neither are platforms with irregular shaped surfaces.

3.3.10 Future Improvements

3.3.10.1 Platform

The platform can be improved by modifying the material and method of manufacture. Currently making a platform takes a reasonable amount of work to create and it is easy for the platform to be different from the created model. Using a process such as metal spinning with aluminium would give a relatively fast, cost-effective, and lightweight platform. Some post processing or other material may be required for the material to be deposited effectively. However, this is something that should be investigated.

To keep the material on the platform slightly soft and malleable, and increase the adhesion of the material to the platform, the platform should be kept heated. This can be achieved by running some high resistance wire along the inside of the platform and supplying an electrical current through it. This will generate heat, which will transfer to the outside of the platform.

To control the ambient temperature around the printer, it is suggested to create a box that encloses it and keeps the heat around it at a specified temperature. Because the environment the printer will be run in can change in temperature and humidity from day to day, it is best to create a closed system that is controllable and can resist the change from external forces. A box around the printer would help to regulate these external forces.

3.3.10.2 Extruder

Through testing and general use of the extruder to create prints, it was found some improvements to the design of the extruder could be made. These improvements will make the quality of the extrusion much better.

Modifying the extruder to melt the pellets in multiple stages would improve the melt by allowing the process to be more controlled. For an effective extruder, a minimum of three stages are required. The first stage would be where the pellets enter the screw from the hopper would be

preheated. In the second stage, the pellets would be compressed and melted. The final stage would increase the pressure behind the melt and extrude it through the nozzle. Multiple heating and cooling bands would be used to help control the temperature of the barrel, which would keep the extrusion more consistent.

An auger drill bit from a hardware store will not provide the amount of shear required to effectively heat up, and mix the pellets within a multi-stage extruder to melting point. To generate the required amount of shear the extruder must use a properly designed screw. This will free up the heating bands from heating the pellets and let the shear force generated by the screw mix and melt the pellets, like most commercially available extrusion systems.

Using liquid cooling, rather than passive or fan cooling will let the extruder adjust the temperature of the melt much faster than it would with other forms of cooling. This allows the extruder to react faster to any temperature changes and adjust accordingly.

Having the temperature controller connected to the computer, along with more sensors, would provide much more feedback than currently. More feedback would allow the user to have greater control over the whole process as they would be able to monitor and change parameters to better the extrusion quality.

3.4 Chapter Summary

This chapter discussed the development of a pellet-based extrusion system for 3D printing from biopolymer pellets. The pellets for this project were developed and provided by Scion research institute and are a mix of PLA polymer and flax fibres. The extruder consisted of an auger drill bit turning in reverse to push pellets from a 3D printed hopper into the aluminium extruder. This extruder is controlled through an Arduino Uno microcontroller, which interprets commands sent to it from the computer over the serial communication protocol. The extruder is rudimentary, but works as expected to create a polymer melt with relatively even distribution of the fibres to give the desired aesthetic look. Many improvements can be made in the future to increase the efficiency and effectiveness of the extruder.

The following chapter discusses the software and algorithms developed to convert a file containing a CAD design into a path for the robot to follow including how the extruder is controlled and run from the computer in time with the robot to function as a 3D printer.

Chapter 4: Development and implementation of algorithms in a software application

Multiple languages were considered to write this software in. As the software deals with a real-world application, with many physical components, it was decided to use an object-oriented language. C# was chosen to write this program in as it is well supported, with many libraries that could make the algorithms used easier to write. Microsoft Visual Studio supports C# and provides the functionality to design user interfaces for applications.

Object-oriented design allows easy modelling of real world problems compared to conventional function-based design. Each class can represent a physical object and keeps the variables associated with that object nicely protected and organised. The classes used to define the objects in this application are found in Section 4.1.

Multi-threading is used to speed up operations through parallel processing, where data does not rely on sequential operations. For example, an array of data that has the same, complex operation done on each element can be processed in parallel to reduce the computation time. Multi-threading is also common where there are multiple sets of data that do not relate to each other and can be processed at the same time without relying on the result of one set for the processing of another. One of the most common reasons to use multi-threading is so the user can interact with the user interface while the application is performing an operation. This software uses multi-threading to keep the user interface active while simultaneously processing large amounts of data through complex operations.

External libraries can be imported while writing software to speed up development time, robustness, and increase speed of operations. This software makes use of the following libraries:

Library	**Description**
ABB PC SDK	ABB provides this library to help users create applications that communicate with, and can control, their industrial robots.
netDxf	netDXF is a library that is used to read, write and manipulate DXF files. It allows easy access to all of the DXF entities.
PolyLib	PolyLib is a library offering support for basic polynomial arithmetic, such as addition, multiplication, differentiation, integration, solving for roots, factorisation, and evaluation at a point.
Math.Net	Math.Net provides support for many linear algebra functions, such as matrix and vector arithmetic.

4.1 Coordinate systems

Coordinate systems are reference definitions for positions and locations. They provide a reference frame for spatial and geometric calculations and operations (Jazar, 2011).

There are many types of coordinate systems with different dimensions. As this project is working in physical space, only 2D and 3D coordinate systems apply here. Within the 2D and 3D space, all coordinate systems fall into two categories: orthogonal and non-orthogonal based systems. Orthogonal systems have two or three axes, all at 90° from each other, as shown in Figure 4-1. Non-orthogonal coordinate systems are generally used for specialised cases for mathematical problems, rather than modelling real world situations.

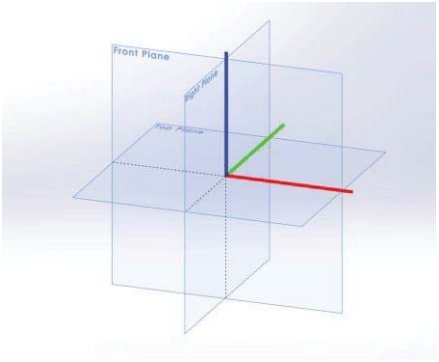


Figure 4-1: Cartesian coordinate system

Within orthogonal coordinate systems, the most commonly used system is the Cartesian coordinate system. This system in 3D defines a point using its position along each axis, giving it an X, Y and Z component. 2D Cartesian coordinate systems remove one of these axes so the point is only represented along two axes.

There are multiple other coordinate systems, generally based on the surface or solid they portray. These make it easier to represent coordinates that lie on these shapes. For instance, spherical coordinates represent points that lie on the surface of a sphere. Spherical coordinates of a point are defined with the radial distance from the origin, the polar angle between the zenith direction and the line segment from origin to point, and the azimuth angle measured from the azimuth reference direction to the orthogonal projection of the line segment from origin to point on the reference plane. Cylindrical coordinate systems represent points that lie on the surface of a cylinder. Points in a cylindrical coordinate system are defined with the radial distance from the Z axis to the point, the azimuth angle between the reference direction on the chosen plane and the line from the origin to the projection of the point on the plane, the height as the distance from the chosen plane to the point.

Some less common 3D coordinate systems include paraboloidal, elliptic cylindrical and ellipsoidal. It is possible and sometimes advantageous to convert between different coordinate systems, as some calculations are easier or require less computation in different systems.

In an application using multiple reference frames, it is possible to have multiple coordinate systems, i.e. one for each reference frame, or a global one that encompasses all frames. There are two coordinate systems used in this software. The first is the coordinate system for the platform and the points on it. The second is the coordinate system for the robot. ABB provides a number of choices for reference frames to use for the robot's coordinate system. These reference frames are described in Table 4-1.

Table 4-1: Different reference frames provided for programming ABB robots

System	**Description**
Tool centre point (TCP)	The tool centre point system uses the centre of the tool. As no tool has been specified, this would be the centre of Axis 6.
World coordinate system	The world coordinate system is based on the origin point as defined in RobotStudio.
Base frame	The base coordinate system is in relation to the base (axis 1) of the robot.
Task frame	The task frame is a real-world representation of the world coordinate system. The task frame can be used to align the robot with a pre-existing coordinate system used by other systems too.
Workobject system	The workobject coordinate system is used to represent a coordinate system based on an imported model. This is used when the robot will be acting upon the work object and makes defining a path around it easy.

The robot has been set to use the world coordinate system, making it use absolute positions, rather than relative ones. The coordinate system used in the platform and the robot have their axes aligned when in their home position. The coordinate systems used are shown in Figure 4-2.

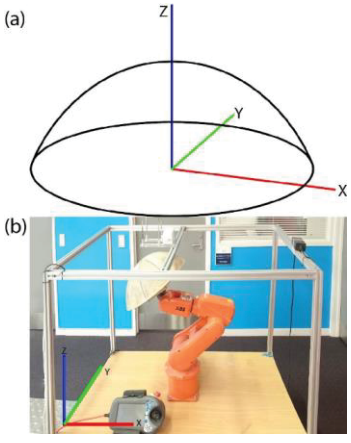


Figure 4-2: (a) Platform coordinate system, and (b) Robot coordinate system

4.2 Classes

Point2D

The Point2d class holds the coordinates for a point on the XY plane. It stores both Cartesian and polar coordinates. It has transformation functions to move and scale the point, and geometric functions to work out the distance or the angle between two points.

Point3D

The Point3D class holds both the Cartesian and spherical coordinates for a point in 3D space. The spherical coordinates are defined by using ISO notation, where R is the radius, θ is the inclination, and φ is the azimuth. This class includes a function to determine the distance between two three-dimensional points.

Path

The Path class stores a list of 2D and 3D points that define a path of points. The class contains functions to add to, clear, and reverse the lists. It also contains geometric functions to move and scale the path.

Entity

The Entity class contains four lists. Two contain the contour path, one in 2D and one in 3D, while the other two contain the points for the fill path in 2D and 3D. It also contains the centroid point, determined as the centre of mass of the entity.

This class contains functions to add points to each of the lists, rearrange, and clear the lists. The class also contains geometric functions to rotate the entity around the origin, move, and scale the entity, calculate the bounding box and centroid of the model, check if the path is clockwise or not, and determine the area of the entity. This class also contains a function to generate the fill path for an entity.

Model

The Model class contains the list of entity objects. This class contains functions to create the fill for each entity, and remove parts of entities that overlap. It also contains geometric functions to find the point furthest from the centre, calculate the centroid and bounding box of the entire model, scale, and move each point.

Dome

The Dome class contains the polynomial equation of the platform, and its derivative. This class holds the functions to project a point from the XY plane onto the surface of the platform, and to calculate the angle between the line tangent to the point on the surface and the X-axis.

PathFiller

The PathFiller class holds the functions to generate the fill path for an entity.

Interpolator

The Interpolator class holds the functions to convert DXF entities and interpolate between the points to generate the list of point2D objects.

Controller

The Controller class holds the list of available controllers as well as functions to find controllers for ABB robots, connect, disconnect, and send and receive messages to the robot's controller through ABB's PC SDK.

Robot

The Robot class holds the home position and orientation, the current end effector position and orientation, and the current speed at which the robot is moving. It holds functions to pause and resume printing, convert point3D objects to robtargets for the robot, and functions to print the contours and the fill.

Hopper

The Hopper class contains the current speed of the extruder, and a flag to tell if it is turned on or off. It also contains functions to connect, and to send and receive messages to the Arduino micro-controller using serial communication.

Other classes

The other classes that the software contains are:

- The Quaternion class that converts a rotation matrix to quaternion notation (Miyazaki, n.d.).
- The Retry class which implements retry logic for functions so they can be rerun after a time limit has elapsed (LBushkin, 2009).
- The NativeMethods class that prevents the computer from going into sleep mode during critical operations such as the printing phase (Hassel, 2012).
- The DistinctItemComparer class that compares two custom objects to see if they are equal or not (Hayter, 2009).

Some of these classes have been slightly modified by converting the code into C# or so they integrate with this software better.

4.3 Graphical User Interface

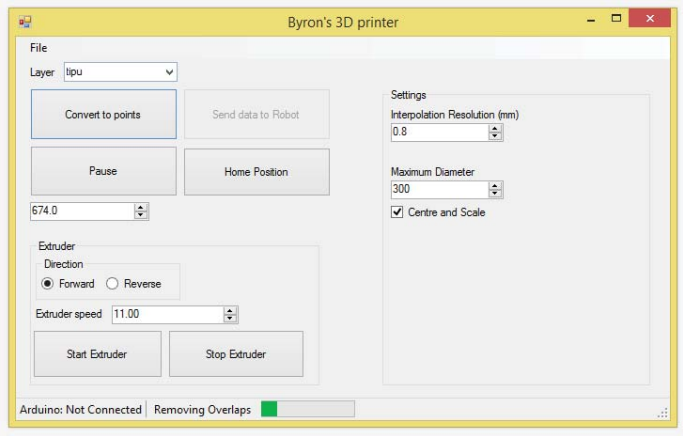


Figure 4-3: User Interface to control printer

The user interface, shown in Figure 4-3 is broken down into five areas:

1. Code run when form loads

When the application loads, two separate threads are set to start running. The first is repeatedly checking for the Arduino to be connected. Once the Arduino is attached to the computer, a serial connection is opened as described in Section 4.14.1. The second is checking if the controller is attached and active. Once a controller is detected, the thread tries to connect to it until a connection is made.

2. File manipulation

The files are loaded or saved through the file menu. From here, the user can open a DXF file or a previously saved B3D file. A B3D file can also be saved from this menu. A B3D file is a custom file format that contains the structure of a model class as a binary file. The value of each variable is written to the file using the binary serialisation class built into C#. A DXF can be saved using netDxf and the code shown in Code Snippet 4-1.

```
saveDXF
   Create New DXF
   For each point in the design
      If (Point is 2D)
         Add Point To DXF (X,Y)
      Else If (Point is 3D)
         Add Point To DXF (X,Y,Z)
      End
   End
   Save File (filename)
End
```

Code Snippet 4-1: Method of saving to DXF

The points generated by the software are converted into DXF points and the DXF saved. Viewing the DXF can be useful in finding and fixing problems in the software and to get the correct threshold settings.

3. Conversion

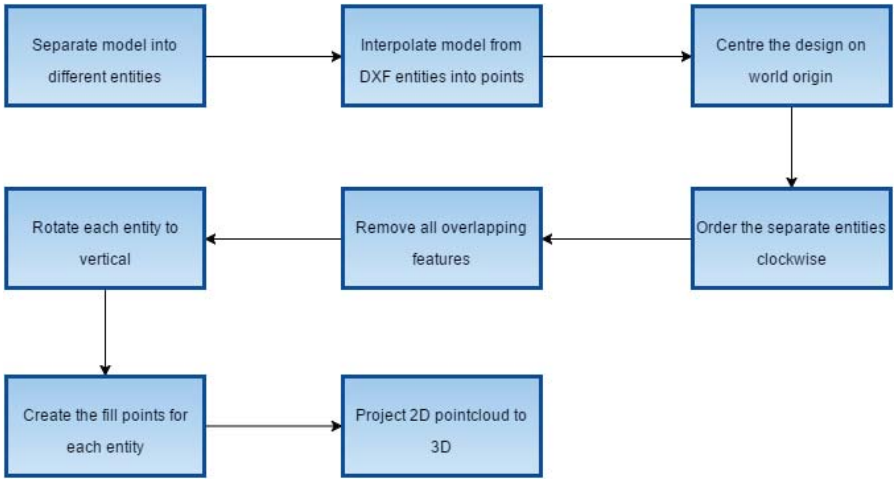


Figure 4-4: Process to convert DXF

When the “Convert to Points” button is pressed, a new thread is started that goes through the process of converting the DXF to multiple point3D objects. A new thread is used here so that the user can adjust the height of the platform, and control the extruder while the conversion is happening. The process followed when button is pressed is shown in Figure 4-4. The process is discussed in more detail in Section 0.

4. Settings

The settings section holds all the variables that are tweak-able by the user. These modify a settings file that is read when needed by the application. Only the nozzle height setting can affect the print during operation. The rest are used for the conversion process.

5. Running

These functions are those that can be used to manipulate the printer either before or during operation. They include the pause/resume button, the “Home Position” button, and the extruder controls. The pause and resume button toggles between two states. When the toggle is set to pause, the extruder is stopped and the software does not send any more points to the robot. When the toggle is in the running state, the commands are sent to the robot and the extruder is turned on.

The “Home Position” button moves the robot instantly to the starting point. This is useful in moving the platform away from the extruder when the software is paused, firstly so material does not build up underneath it, and secondly so the user can access the extruder or platform if needed.

The extruder controls allow real-time modification of the extruder. When a button is pushed, or a value changed, it sends a command to the extruder, which instantly adjusts. As this is a separate system to the robot, the extruder can be run independently and modified at any time.

4.4 Overall printing process

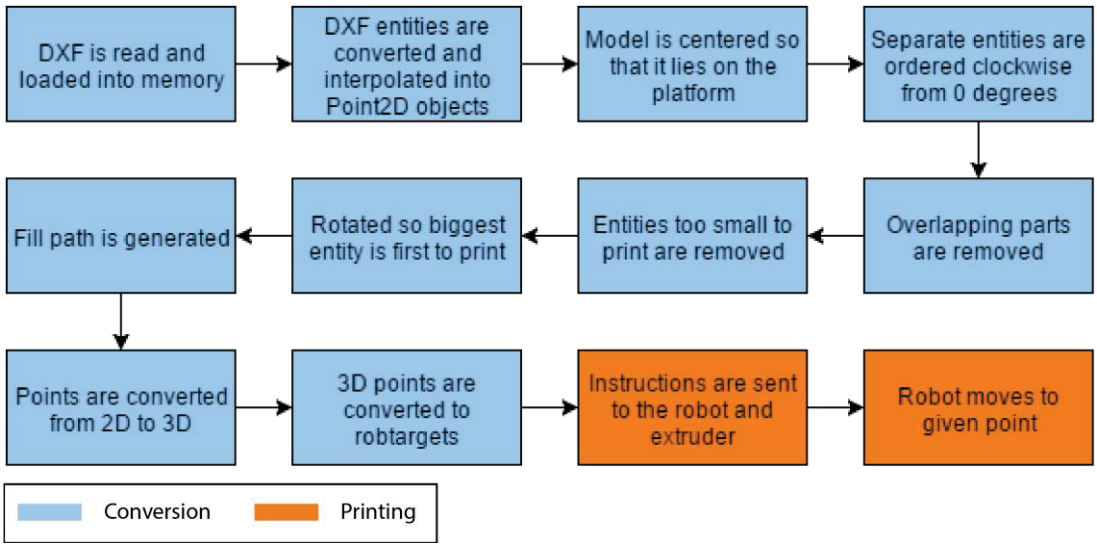


Figure 4-5: Flow diagram of the printing process

The process flow is shown in Figure 4-5. There are two stages, the model conversion stage, and the printing stage. The model conversion stage takes the user's file and turns it into robtargets for the robot. The printing stage uses these robtargets to move the robot while controlling the extruder to print the design.

The first step in the conversion process is to read the DXF file into memory using the netDxf Library. Secondly, the DXF entities are converted into 2D points. Next, the points are centred and scaled to fit on the platform correctly. After this, the individual entities are rearranged in order so they start from 0° for the overlap removal process, which takes place next. Following this, entities from the overlap removal that are too small to print are discarded. Next, the design is rotated so the biggest entity is at 0° ready to print. The fill path for each entity is then generated and all the contour and fill points are projected onto the platform to make them 3D points. Lastly, the 3D points are converted into robtargets for the robot.

During the printing stage, the extruder is manipulated to achieve the correct flow while the robot is streamed commands from the software. These commands are the speed for the robot to run at and the target position and orientation for the end effector.

4.5 DXF file is created from a CAD package

The design that was provided is a CAD model in the form of a Drawing Exchange Format (DXF) file format. The provided design is seen in Figure 4-6. DXF is a file format developed by Autodesk and can be created through most CAD software packages. Initially it was designed to enable the interchange of drawings between AutoCAD and other programs. DXF files are represented in both ASCII and binary formats, with the former being the most common. However, the binary format is smaller, faster to read and write, and preserves all the accuracy of floating point numbers. Both the ASCII and binary versions contain a complete representation of the AutoCAD drawing (Autodesk, n.d.).

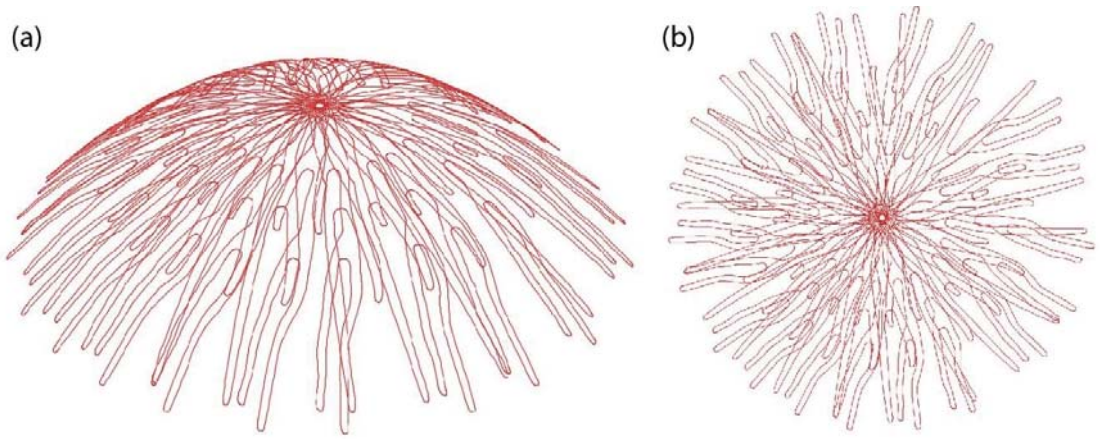


Figure 4-6: (a) Isometric view of the CAD design, and (b) top view of the CAD design

DXF files represents a CAD model as pairs of codes and associated values. The group codes indicate the type of value that follows. A DXF file is comprised of sections, which contain records, which consist of a group code and a data item. There are small differences in the codes and values used between ASCII and binary DXF files, but the general structure is the same. The group code and associated values define a specific aspect of an object or entity. The line immediately following the group code is the associated value.

The sections of a DXF file are:

Header section

Contains general information about the drawing, such as AutoCAD database version and system variables.

Classes section

Holds the application-defined classes, whose instances appear in the blocks, entities, and objects sections of the database.

Tables section

Contains definitions for the variables used in the DXF file. The tables are: APPID (application identification table), BLOCK_RECORD (block reference table), DIMSTYLE (dimension style table), LAYER (layer table), LTYPE (linetype table), STYLE (text style table), UCS (User Coordinate System table), VIEW (view table), and VPORT (viewport configuration table).

Blocks section

The blocks section contains the block definitions. Each block definition contains the entities that make up the block in the drawing.

Entities section

This section contains the graphical objects used for the drawing, including block references.

Objects section

The non-graphical objects are held in this section. All the objects that are not entities, symbol tables, or their records are contained in this section.

4.5.1 DXF benefits

DXF files offer a number of benefits over other CAD formats. DXF files are a simple format that is generally in 2D. However, when in 3D there is no thickness to it. Solid shapes are represented as lines, rather than surfaces or solid bodies. Most 3D printers use STL files, which represent solid bodies as many triangles. STL files would require complex algorithms to generate a path from them as they portray a solid body rather than a thin-shelled design.

DXF files offer designs with no wall thickness and can be exported from most CAD packages, as the file documentation is openly available to anyone. This also allows software to be easily written to interpret the file.

There are many types of entities, all of which can be seen in the DXF reference manual (Autodesk, 2011), but the most common are lines, points, arcs, circles, splines, polylines, and text. There is a code associated with each one to help the software differentiate between them, and the data following the code gives coordinates or relevant information. DXF files are generally used to express 2D designs, but do have the capability for 3D.

4.6 File loading

The DXF file is read into the software using an external library, netDxf, which is designed solely to read, write, create, and manipulate DXF files. Through the interface, the user provides the filename. The model is then read into memory through netDxf. This library stores the DXF file as an object in memory where each entity of the DXF is stored as a sub-object of the model. The layers are separated and the user selects the layer they want to print.

4.7 Contour point generation

To create the contour points, the entities of this DXF are converted into 2D points, stored as Point2D objects that hold both Cartesian and polar coordinates for each point.

For 3D DXF models, if the model has a different curvature to that of the platform, or is a 2D DXF, the points will not lie on the surface of the platform. By removing the Z component of the model, the points can be projected onto the surface of the platform as described in Section 4.11.

Each type of entity is converted and interpolated. Currently the software only handles the most common types of entities: Arcs, Lines, Splines, Polylines, and Circles. All these entities represent open contours. For each entity, the step size remains the same so the resolution of the points remains constant. The step size is set by the user through the user interface.

The entities are converted using parallel processing. Through testing it was determined that converting the entities was inefficient. To decrease the processing time, the function was changed so it processed the data using multithreading, which lets multiple entities be converted at the same time. Once converted, each set of points is stored in its own list to create a path.

Arcs are defined in the DXF with a centre point, radius, start and end angle. The Cartesian coordinates are used rather than polar coordinates as the latter relies on the centre point being on the origin. The X value of the point can be found from $X_c + R \times \cos\theta$ and the Y value from $Y_c + R \times \sin\theta$. The angle is increased by the step size each time and the point saved until the end angle is reached.

Circles are stored as a centre point and radius. The same equations as the arc can be used to interpolate the circle by using a start angle of 0° and an end angle of 360°.

Lines are stored in the DXF as a start point and an end point. Using the equation of a line is shown in Equation 4-1, the gradient is determined using $m = \frac{Y_2 - Y_1}{X_2 - X_1}$. The start point's values are placed into X_1 and Y_1, while the current step is placed into X_2. The result gives the current Y value for each step. The point is stored and the step is increased until it reaches the end point. If the line is vertical, the gradient is unsolvable using the equation above as the denominator is zero. In this case, the X value remains constants and the step is done in the Y direction.

$$Y_2 = m \times (X_2 - X_1) + Y_1$$

Equation 4-1: Equation of a line

Polylines are entities that consist of multiple lines joined together, end-to-end, to make complex shapes. They can use the same equation as the line to interpolate points.

Splines are curves defined by piecewise polynomial functions with a high degree of smoothness at the places where the polynomials connect. DXF's store splines as a list of knots and control points. The netDxf library includes functionality to convert splines to polylines at a specific resolution. The precision is determined by the number of control points divided by the step size. The resulting polyline is passed to the polyline interpolation function to provide the points.

After all the DXF entities have been converted into a list of point2D objects, the paths are merged together. The reason for this is that generally each DXF entity joints onto another one to create a closed path. Merging joins multiple paths that connect into one path. To merge the entities, the start and end point are analysed against the start and end points of the other paths. If they are within 0.005 mm from each other, it is assumed the points are meant to be merged. This tolerance is because the interpolation algorithms can give slightly different end point values for different types of DXF entities.

When merging the paths together there are five possible cases:

1. The end point of the current path is matched with the start point of the new path. The new path is appended to the current path.
2. The end point of the current path is matched with the end point of the new path. The new path is reversed and appended to the current path.
3. The start point of the current path is matched with the end point of the new path. The new path is inserted at the start of the current path.
4. The start point of the current path is matched with the start point of the new path. The new path is reversed and inserted at the start of the current path.
5. The points do not match so the next path is analysed.

When a path is appended, it is removed from the list. If all paths have been analysed for the current path, it is saved as an entity and a new entity is started. If the start point and end point of the entity match, the entity is defined as closed; otherwise, it is defined as open. This is repeated until all paths in the list have been removed.

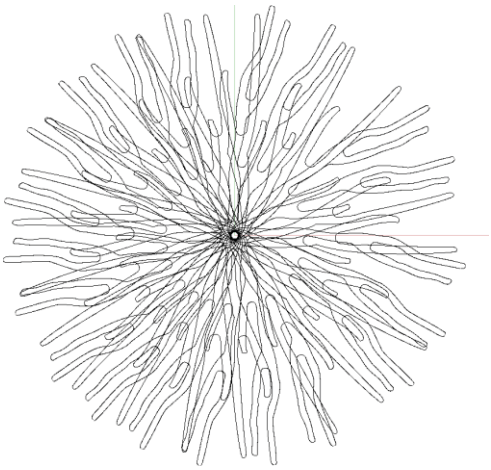


Figure 4-7: Interpolated design

It was considered to use a grid-based system to snap the points to predetermined positions. Even though it would make other computation, such as generating fill points easier, through trials it was found that the path became too staggered and would not produce a good quality print. These 2D points are interpolated at a specified resolution. This resolution does two things; firstly, it increases the accuracy of the printed design and therefore the quality. Secondly, it slows down the robot because it must move to more points to cover the same area. The interpolated design can be seen in Figure 4-7, while a close up of the interpolated points can be seen in Figure 4-8. Currently these points are being interpolated at a resolution of 0.2 mm.

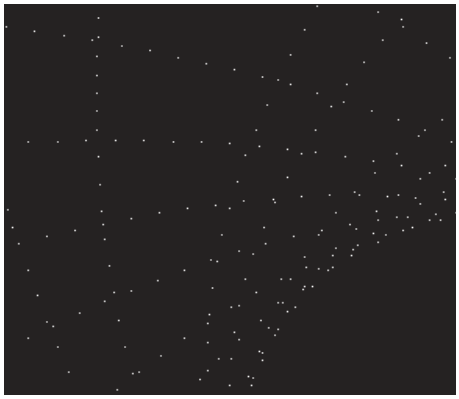


Figure 4-8: Close-up of interpolated design

4.8 Geometric operations

The next step is to centre and scale the design if the user desires. This ensures the correct placement of the design on the platform and safeguards against failed prints. The design is centred by taking a bounding box of all the points, calculating the centre of the bounding box and moving each point of the model (using Cartesian geometry as it is easier to work with than polar form), so that the centre of the bounding box lies on the origin point.

Once the points are centred, using polar geometry the design is scaled. Polar form is used because it consists of a radius and an angle rather than the X and Y values of the Cartesian form. To scale the points, the radius is changed and the angle stays the same. The points are scaled by a factor, alpha, which is determined by dividing the target diameter by the distance of the furthest point from the origin. All points are scaled by this value. The user sets the target diameter.

4.9 Overlaps are removed

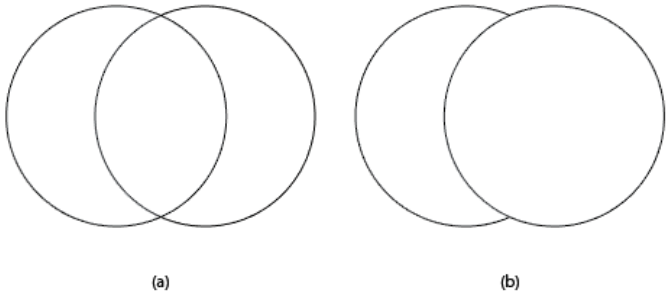


Figure 4-9: (a) Entities before overlap is removed, (b) Entities after overlap is removed

Currently, most 3D printers use a 2.5 process where a single layer is completed before moving to the next layer. In the FDM process, the reason for this is to stop the printer from colliding with the part. If each layer is completed before moving to the next, the printer will always be depositing material on top of the previous layer, rather than having to navigate around it.

Overlapping parts of entities are removed so that the design becomes a single layer and the printer will not deposit material over already printed areas. An example of this is shown in Figure 4-9. This stops the nozzle from colliding with already deposited material, which in turn stops the design from moving, the extruder from being bumped, and parts from being damaged.

As the design is stored as a set of points, if lines were drawn between each point and the next for each entity, it could be thought of as a polygon. There is a range of polygon clipping algorithms. The most established are Greiner-Hormann, Sutherland–Hodgman, Weiler–Atherton, and Vatti. Each of these algorithms works differently and has their drawbacks. The

Greiner-Hormann algorithm cannot handle degeneracies, such as sides of zero lengths or scanlines that intersect with a point. The Sutherland-Hodgman algorithm can return some overlapping edges on the polygons. The Weiler-Atherton algorithm occasionally cannot compute polygons with holes and requires the polygons to meet certain conditions first. The Vatti algorithm is the most generic algorithm available; however, it can be slower as it requires a large amount of processing and will not work where closed polygons intersect with open lines.

In 1992, Bala Vatti developed a generic algorithm to clip polygons. Previously existing solutions for polygon clipping were either limited to certain types of polygons or are complex and time consuming. Vatti (1992) describes an algorithm to perform intersection, union or difference operations on polygons. It was attempted to write the Vatti algorithm. However, it was more efficient and faster to use an external library that had already optimised the algorithm.

4.9.1 Polygon Clipping Library

```
removeOverlaps
   Create new model
   For each entity
      Create Polygon of Entity
      Remove part of entity that overlaps the previous entities
      Add modified entity to model
   End
   Return model
End
```

Code Snippet 4-2: Overlap removing algorithm

The General Polygon Clipper (GPC) Library was chosen to operate on the polygons as it provides four clipping operations: difference, intersection, exclusive-or, and union. To remove the overlapping sections, each entity is first converted into a polygon. The first entity is selected to be the base polygon. For all the rest of the entities, their polygons are compared with the base using a difference operation. The resulting polygon is saved and then added to the base with a union operation. This is repeated on all the entities. The original base polygon, along with all the saved difference results gives the final polygons with no overlapping areas. The algorithm is seen in Code Snippet 4-2. These are converted back into Point2D entities. The resulting design is shown in Figure 4-10.

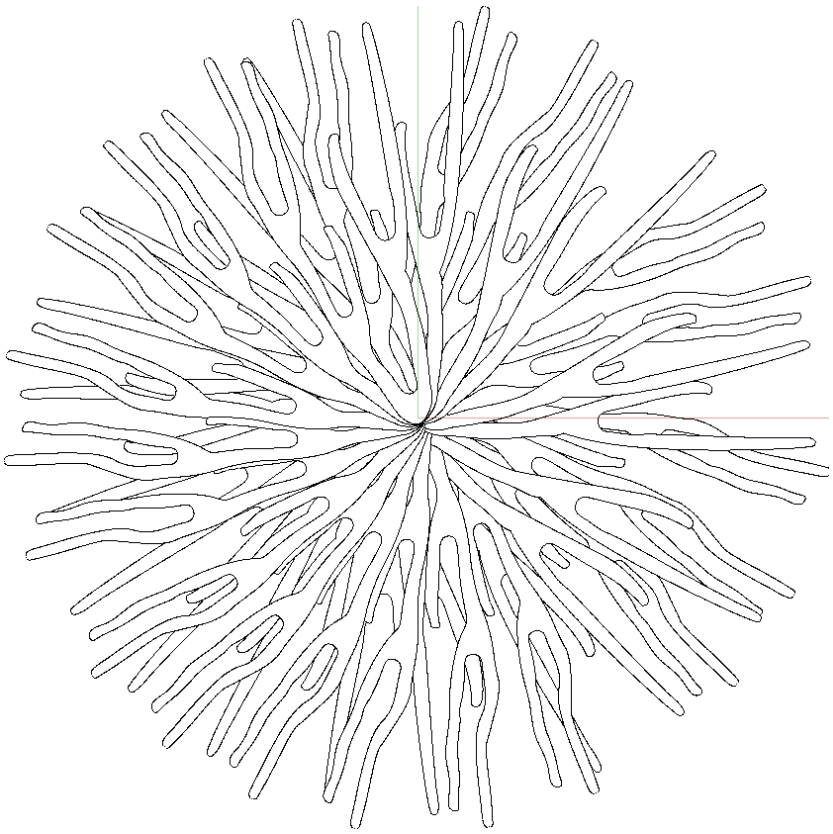


Figure 4-10: Design after overlaps are removed

4.10 Fill point generation

When the contour is printed, there are some gaps in the print. These gaps are not intended to be part of the design so require filling. The most common method of filling is to create adjacent linear lines. These can be oriented in any direction. It was decided to use a linear fill to close the gaps between the contours.

As the shapes in the design are irregular curves, forming a polygon with concave parts, rather than simple linear entities it is not easy to use a grid based system without distorting the design. This ruled out the possibility of using a flood-fill type algorithm to generate the fill points. Therefore, the points that make up the path for the robot to fill the entity are generated through a scanline algorithm.

With a scanline algorithm, it is possible to use interpolation along the scanline to find the start and end points and interpolate between these end points to generate the path. The scanline then moves along by a specified resolution and the process is repeated until it reaches the end of the entity. This process is done for each entity in the design.

Before the scanline is computed, the entity is rotated to the correct orientation. The user selects between a horizontal or vertical fill. If the fill is horizontal, the entities are rotated to 90° while for the vertical fill they are rotated to 0°. It is possible to configure any degree of fill at a later stage if required.

The entities are rotated by applying a rotation matrix in Equation 4-2 for each point, where θ is the angle to rotates to 0°.

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Equation 4-2: Rotation Matrix for 2D points

Next, the scanline algorithm is implemented to create the fill path, as shown in Code Snippet 4-3. A bounding box of the entity is taken to find the leftmost and rightmost point. This defines the first and last X values for the scanlines. The rest of the X values are interpolated from these points at a set resolution, which is determined by the user. For each scanline, the Y end points are determined by taking the value at which the scanline intersects the contour. These end points are interpolated using the line interpolator defined in Section 4.7.

```
createFillPath
   Create a horizontal line
   Get pairs of intersection points of line with polygon
   For (each intersection pair)
      If (pair doesn't have endpoint)
         Break
      End
      Generate fill points by interpolating between pairs
   End
End
```

Code Snippet 4-3: Scanline algorithm to create the fill path

When interpolating these points, it is assumed they each have a matching start and end point. While travelling along the scanline and counting the intersecting points, the points on the odd counts relate to the start point and the even counts relate to the end point of each section.

It is possible for the scanline to output more than one start and end point as seen in Figure 4-11 (a). When this is the case, each start point should have a matching end point to create a fill line as seen in Figure 4-11 (b). When there are an odd number of points, either from an open contour as seen in Figure 4-11 (c), where the scanline being perfectly tangent to the contour, or from

losing data through software bugs the interpolation will not work properly as it requires two points. If the missing point is from the middle of the scanline, the interpolation will be done between the end point of one section and the start point of the next, interpolating outside of the polygon rather than inside. This is illustrated in Figure 4-11 (d).

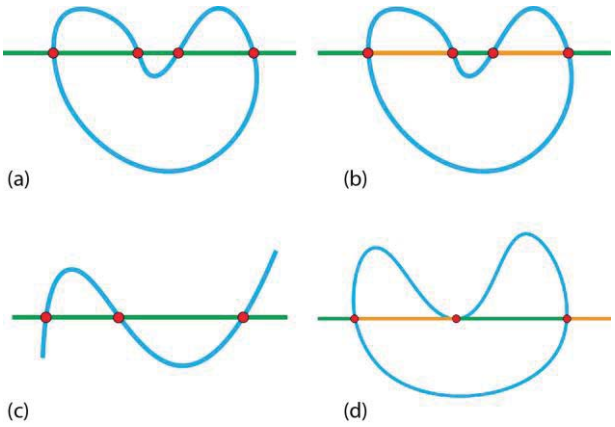


Figure 4-11: (a) Points of intersection along a scanline, (b) Fill line within polygon between start and stop points, (c) Points of intersection along scanline and open contour, and (d) Fill line with scanline intersecting polygon on a tangent.

The end points of each scanline are determined using the function in Code Snippet 4-4. This algorithm assumes that the contour is closed so there will always be an even number of end points. It works by looking at adjacent pairs of points in the contour. An end point is detected when the scanline's X value lies between X values of the pair of points. To find the Y value, the line-line intersection formula is used. This formula takes the end points of two lines and calculates the intersection point. The equations for the intersection point can be found in Equation 4-3 and Equation 4-4, where line one is defined by $(x1, y1)$ and $(x2, y2)$ while the second line is defined by $(x3, y3)$ and $(x4, y4)$.

$$X = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}$$

Equation 4-3: X value of intersection

$$Y = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}$$

Equation 4-4: Y value of intersection

```
getIntersections (Entity Polygon, Scanline)
   For (each point in the polygon)
      Find two adjacent points in the polygon that the scanline lies between
      Interpolate between the two points to get intersection point
   End
   Return endpoints
End
```

Code Snippet 4-4: Gets the points of intersection from the scanline and the polygon

Some points are removed if they are too close to the contour path. This is so that when printing, there is not too much overlap and so the material does not build up too much. This is achieved by finding the closest point of the contours to each fill point. If the distance is less than a set threshold, currently set at 0.8, the fill point is removed.

Once all the end points have been interpolated, the entities contour points and fill points are rotated back to their original position. This is the final stage of the point creation and produces the design shown in Figure 4-12.

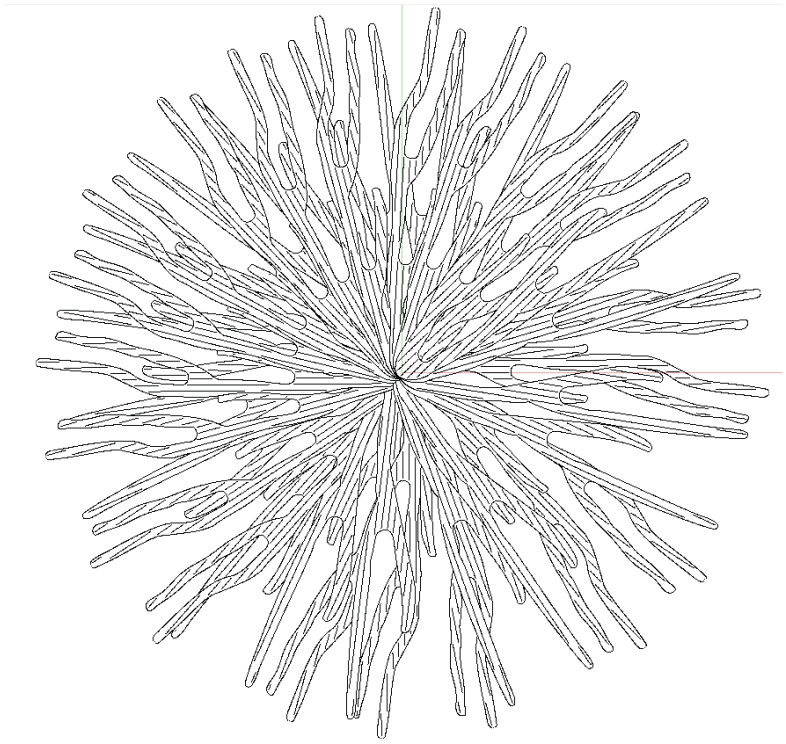


Figure 4-12: 2D design with fill points

4.11 2D to 3D projection

There are multiple ways to map 2D points to 3D. Many map projection algorithms take a flat surface and wrap it around a sphere. These could be modified to fit the platform. However, with most of these there is some distortion in the design. It was decided not to follow one of these methods, as the distortion would affect the design.

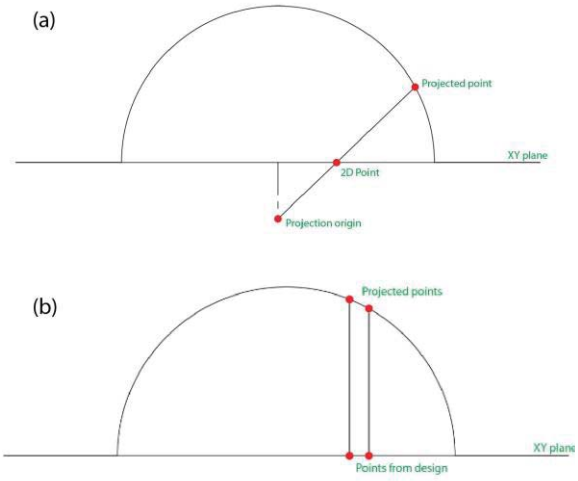


Figure 4-13: Side views of the (a) single point projection method, and the (b) vertical projection method

Another method considered was to take a line from a single point below the origin through each point that lies on the XY plane to where it intersects with the curve. This process is illustrated in Figure 4-13 (a). These intersection points create the 3D path of the design. This method was also rejected as it was found to distort the design slightly.

The method that was chosen in the end was to project each point vertically along the Z direction (Figure 4-13 (b)), normal to the XY plane the design lies upon, until it intersects with the platform curve. This has the benefit of keeping the design exactly as intended and wrapping it around the platform without any deformation.

The software uses a polynomial library to store the coefficients of the equation. Using a library makes it easy to work with polynomial equations. Code Snippet 4-5 shows the algorithm to project the points on to the platform.

The platform is modelled by rotating the polynomial around the Z-axis. This is the reason the X and Y value of the point stays the same, while the Z value is determined by solving the polynomial. The results create a 3D point with both spherical and Cartesian coordinates. The polar radius of the point is taken as the value to evaluate the polynomial at because if looking at a

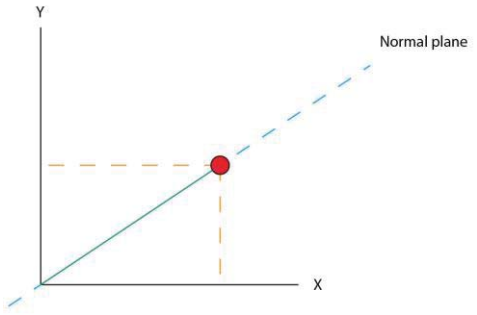


Figure 4-14: View of point normal to the XY plane

plane that lies normal to the XY plane along theta for a given point, the radius gives the distance from the origin. This is depicted in Figure 4-14.

```
projectPointTo3D(2D point Pt2D)
      Pt3D = new 3D point
      Pt3D.X = Pt2D.X
      Pt3D.Y = Pt2D.Y
      Pt3D.Z = Evaluate polynomial at distance from point to origin
      Return Pt3D
End
```

Code Snippet 4-5: Projection function

4.12 Saving/loading files

The 3D model of points can be saved at this point. This is done through binary serialisation of the classes. Binary serialisation saves all the data from the model class that holds all the 2D and 3D points. Saving the file lets the user speed up the printing process in the future. The file can be loaded and printed straight away so the conversion from DXF is only needed to be performed again if a new design is used.

4.13 Printing

The software keeps track of each point of the model that is printed. Each entity is searched to find a point that has been previously printed. The entity is rearranged so this point becomes the start point. This lets the print start each entity by attaching it to a previous one, which gives the material something to easily bond to which reduces the chance of a failed print. If there is no point in the current entity that has not been previously printed, the entity is rearranged to start at the point closest to 0°. Starting as close to zero as possible reduces the chance that axis six of the robot will go out of bounds.

All the contours are printed before filling them in. It was attempted to fill each contour after it had been printed, as well as printing the fill first. However, the prints failed far more often than they did when printing all the contours first. The contour and fill points are converted to robtargets (See section 4.15) and sent to the robot (See section 4.14). The fill points have their Z value modified beforehand by increasing them by 0.2mm, which gives the nozzle enough clearance to print over existing contours, while letting the fill print to bond correctly with the platform.

After each contour or fill path has been completed, the robot moves to the home position, then to the next starting point at the robot's max speed. This re-centres Axis 6 of the robot so it does not go out of range. It also creates tags, which are from excess material left from the extruder and must be removed after printing.

4.14 Communication to hopper and robot

4.14.1 Extruder communication

When the application starts up it creates another thread to check if the Arduino is plugged in so the software can connect to it. When the Arduino is detected, the software automatically connects to it. The software will not let the user print until the Arduino is connected to prevent the possibility of a failed print.

The thread is constantly searching for open COM ports that include the word Arduino. Once the port is found, the software opens a serial connection using predefined settings. Once the serial link is connected, the software is able to send and receive data from the Arduino through the USB. The data sent are the commands defined in Section 3.3.8.1.

4.14.2 Robot communication

The PCSDK that ABB offers provides the functionality for applications to interface with the robot's controller. When the software loads, it tries to connect until it is successful. As the controllers are connected by Ethernet, the first step is to scan the network to find all connected controllers. These are each checked to make sure they are not virtual controllers. Virtual controllers are not physical controllers and are used as simulation controllers. If it detects one physical controller connected, the software connects to it; otherwise, it prompts the user to select the correct one. When connecting, an Inter-process communication (IPC) message queue is initiated and the robot's software is started so it can listen for commands.

The Point3D's are sent to be encoded to robtargets along with the speed to move to them. The Point3D's are encoded on the fly as they are sent over, rather than all at once for two reasons. Firstly, the robot is unable to store all the points as RAPID code. For the design provided, there are 66,661 contour points and 27,758 fill points. To store each of these as a single file requires over 180,000 lines of code (two lines per point; one to store the robtarget, the other the move instruction). RobotStudio is not able to open the file, as it is too big.

Secondly, it allows the distance from the nozzle to be modified during runtime. By converting each point individually before sending it, if there are any changes in the platform height, this is updated before sending the data. This saves the user from having to convert and print the whole design again.

```
encode (Destination, Speed)
   If (speed needs changing)
      encode and send speed update message to robot
   End
     encode and send destination point
End
```

Code Snippet 4-6: Method to create the command for the robot

If the speed is different to the current speed the robot is moving at, the command to change the robot's speed is sent to the controller. If they are the same, it is ignored. If the point (0, 0, 0) is passed through, the point (0, 0, 0) generates a robtarget with an invalid quaternion of (0, 0, 0, 0) which is translated by the robot into the home position as described in Section 3.2.1.3. Otherwise, the point3d is converted to a robtarget, the method of which is detailed in Section 4.15, and is sent to the controller. The code for this process is shown in Code Snippet 4-6.

The points are sent to the robot using a third-party retry class and the PC SDK's IPC queue. The retry class will resend the same data until it receives an acknowledgment from the controller. The PC SDK has functionality to send the command to the controller, but the retry class and acknowledgment system is used as a safeguard against problems arising if something prevents the controller from receiving the data.

4.15 Converting the Point3D to robtarget format

Once the 3D points that make up the design have been generated, the coordinate and orientation of the robot's end effector must be determined so the plane that lies tangential to the point on the surface of the platform is normal to the extruder and a set distance from the end of the nozzle.

This process utilises linear transformations, specifically translation and rotation. The design can be thought of as an infinite number of 2D planes, normal to the XY plane, each rotated around a central axis, an example of which can be seen in Figure 4-15.

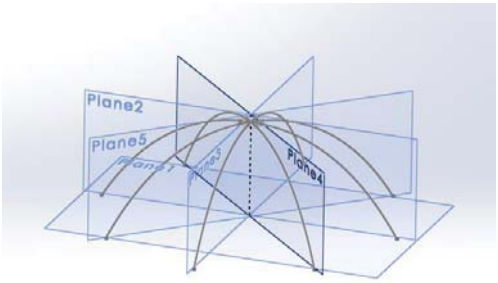


Figure 4-15: Planes normal to surface rotated around a central axis

Each plane holds points along the parabola on them. These planes are kept track of by recording the angle of the plane along the XY plane. This lets the points be visualised in a 2D space, when looking normal to the plane, which makes the linear transformations much easier.

4.15.1 End effector coordinate

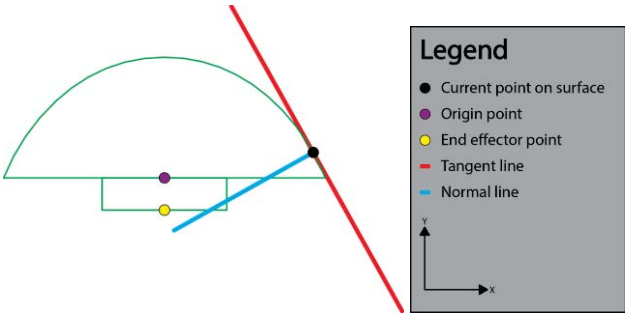


Figure 4-16: Starting point of diagram used for mathematics calculation

The target point is translated to the top of the platform, to point (0, 0, 150). This is the top of the platform and will be the point in contact with the nozzle. This is done by taking the point's spherical radius and projecting it onto the X axis. This specifies the amount to move the point by in the X direction. The Z amount is determined by taking the Z value of the point at the top of the platform in millimetres (150), minus the Z value of the point. Figure 4-16 views a single plane in the model with a point on it.

This gives the translational matrix:

$$T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -projX \\ 0 & 0 & 150 - pt.Z \end{bmatrix}$$

Equation 4-5: Matrix for initial translation

The end effector is stored as a vector of the point (0, 0, -40). This is because it is 40mm below the origin of the base of the platform, as shown in Figure 4-17. To translate the end effector, it is then multiplied by the matrix in Equation 4-5 to translate it by the same amount. The outcome of this translation is depicted in Figure 4-18.

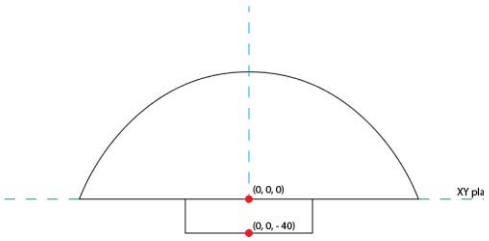


Figure 4-17: Endeffector vector in relation to origin

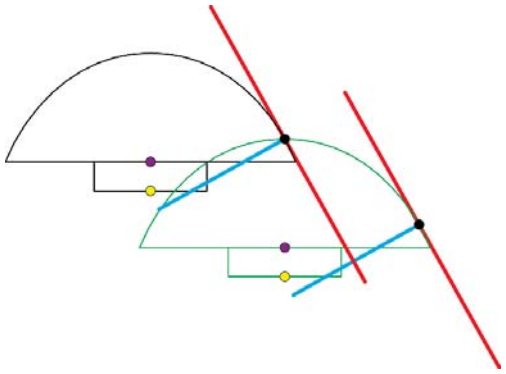


Figure 4-18: Black diagram is transposed to (0,0,150)

The end effector is then rotated around (0, 0, 150) until the line tangent to the point on the curve is horizontal. The method of rotating around a point other than the origin is used to translate all the points so the centre point (P_c) is on the origin. The points are then rotated by the desired angle and then all translated back so that P_c is in its original position. This process can be shortened into two equations, one for the X value (Equation 4-6) and one for the Y value (Equation 4-7). X' and Y' are the Cartesian X and Y values of the modified point, (X_c, Y_c) is the centre point to rotate the point about, P is the point to rotate and θ is the angle to rotate the point by.

$$X' = X_c + (P_x - X_c) \times \cos\theta - (P_y - Y_c) \times \sin\theta$$

Equation 4-6: X value for rotation about a point

$$Y' = Y_c + (P_x - X_c) \times \sin\theta + (P_y - Y_c) \times \cos\theta$$

Equation 4-7: Y value for rotation about a point

This angle is determined by taking the arctangent of the line normal to the gradient, found by the derivative of the polynomial equation at the X component of the point. The resulting points are shown in Figure 4-19.

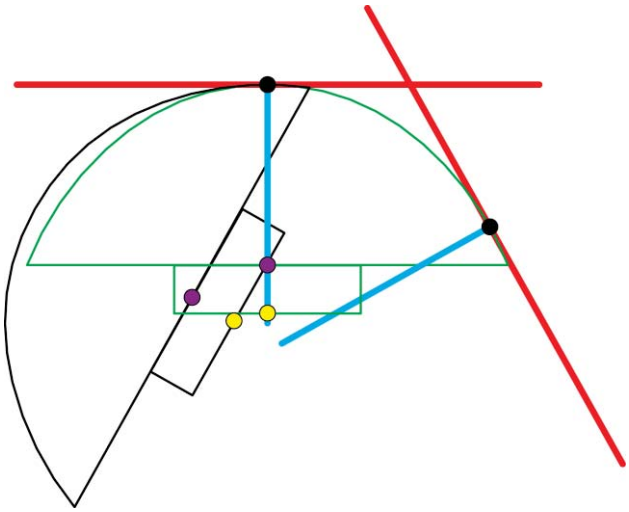


Figure 4-19: Black diagram has been rotated around (0,0,150)

Finally, the end effector location is translated to the home position of the robot to give the true position of the end effector in the robot's coordinate system. The robot's home position is defined as (0, -165, 674). The Z value of this is adjustable during runtime by the user. The home position is set as this point because it allows the robot to move the platform without any collisions.

4.15.2 End effector orientation

The end effector's orientation is defined as a quaternion in a robtarget. It is possible to translate between a rotation matrix and a quaternion. As the robot only moves in the YZ plane, only one rotation on each of two axes are needed to orient the end effector. The first is to rotate the platform to the correct angle as in Figure 4-19. The second is to rotate the platform so the line tangent to the point to be printed on the platform lies horizontal.

The rotation matrix for rotating the point around the X axis is generated by taking the same angle (θ) as used in Section 4.15.1 when determining the rotation angle, which gives the matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

The rotation matrix to rotate the point around the Z axis is created by taking the angle of the azimuth (φ) of the point on the platform. This gives the rotation matrix:

$$\begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

These matrices are multiplied together to provide a single rotation matrix, which covers both rotations. The order of multiplication is important as AB ≠ BA where A and B are both matrices.

In this case, the rotation around the Z axis must be computed first because after applying the rotation around the X axis, the orientation of the Z axis has changed.

$$R = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \cos\theta\sin\varphi & \cos\theta\cos\varphi & -\sin\vartheta \\ \sin\theta\sin\varphi & \cos\varphi\sin\theta & \cos\theta \end{bmatrix}$$

The values of θ and φ are entered and the matrix is converted to quaternion format to be stored in the robtarget.

Both the robot configuration and the external axis configuration of the robtarget are hard coded and not intended to be changed. There is no external axis so all the values are set to 9E9, which is the value used to indicate no axis is present. For the robot configuration data, the same configuration as the one when the robot is in its "home position" is used. As ConfJ and ConfL are set to off, the robot configuration during runtime can end up different to the one set.

4.16 Limitations

There are a few limitations of this software because it has been developed for a very specific application. Each limitation presented below is justified and as the project progresses in the future, these limitations will be reduced.

Currently this software can only convert DXF format files. Not every type of DXF entity, such as hatches or text, is supported in the point generation and interpolation algorithm. However, there are many entities, such as the dimension, hatch, and tolerance entities, which will not be useful in this application. If the input file format is not changed to a solid body format such as STL, the remaining DXF entities must be added to support point generation and interpolation.

The scale function currently only works if the design is centred on the origin. This is because it modifies the polar radius of the 2D points. To make this function work for any design, the whole design can be moved so it is centred on the origin, then scaled, and moved back to the original centre position.

The General Polygon Clipping Library used to remove the overlaps uses an extended version of the Vatti algorithm to increase its robustness. Because the Vatti algorithm uses scanlines, similar to the fill algorithm, it requires a start and end point along each scanline for the algorithm to use. This means it is unable to manage single intersecting line segments. All entities must be closed and open entities taken out before overlaps are removed.

The width between the fill paths is set by a threshold value. This can only be worked out through trial and error. The path width of the material can be used as a starting point, but must be tweaked so that it is small enough that no gaps are formed, but large enough not to have the material pushing up against previously printed paths.

Currently the printer will only deposit a single layer of material. To print more layers, the spherical radius of each Point3D object can be increased by a few millimetres per layer. The value of this is dependent on the height of each layer. If a layer with a different shape from the previous one were to be printed, it would require going through the whole conversion and projection process again. If support material is required, an algorithm to determine where and how to print the support is required.

After printing, the surface finish can be rough. The tags would need to be removed and surface smoothed down before printing a second layer. This would ensure the nozzle does not catch on anything as its printing the next layer.

Using the current method of projection, the algorithm is unable to project points onto vertical or non-equation defined surfaces. Currently the software supports only polynomial equations, but it could be modified to use other equation forms. This is because the projected point is determined by the intersection of two equations. The vertical surface is unsupported because the projection line is vertical and therefore parallel to the surface so it is impossible for the two to intersect.

4.17 Future improvements

The software manoeuvres the IRB 120 robot while controlling the flow of material from the extruder. The DXF is converted and interpolated into several 2D points. These points are projected vertically onto the platform to form a 3D model. Each point is translated to coordinate and orientation for the robot's end effector, while simultaneously controlling the extruder's operation. Future improvements can be made to increase the performance and quality of this software as outlined below.

4.17.1 General

Performance analysis can be run on the software to determine where the computational bottlenecks are. This will show functions that would benefit from optimisation. Generally, they are ones that deal with operating on large amounts of data and involve multiple nested loops or complex operations.

If it is possible, it is suggested to run these loops in parallel so they utilise multiple threads. This speeds up computation by allowing the dataset to be broken into multiple streams that are processed at the same time. However, if the data is reliant on the order of the dataset, care must be taken so the order is preserved through each process.

If running the loops in parallel is not possible, it is suggested to look at methods of optimising the algorithm. Many times, it is possible to speed up a function by removing unneeded code, using a different method such as lower level code with less overhead or an already-optimised third party library, which will provide the necessary functions, or by changing the algorithm to achieve the same result more efficiently.

If the slow function is provided by a third-party library, it is suggested to look at either finding a different library with the same functionality but that is more efficient, or writing an optimised version of the algorithm.

Improving the robustness of the software would be beneficial because this would reduce the possibility of the print failing for different designs or unexpected conditions. Currently the software only supports the most common DXF elements. If a DXF is loaded with unsupported entities, those sections will not be printed. Increasing the robustness of the software will stop occurrences of this. There are many chances for errors to occur throughout the processes within this software. These can be avoided with exceptions and try-catch statements, which could be added to the code, as well as enabling the software to provide error messages to the user.

The format of DXF is an unusual file format for 3D printers. The standard file format is a Stereolithography (STL) file, which consists of triangular surfaces. Currently the designs are developed in Rhino, a 3D CAD modelling package. Rhino and many other CAD packages export STL format files, so the current designer will not need to modify their design process, but could rather choose to export the file in a STL format. The benefit of STL file formats is that the design will be able to convey the thickness of the design, as well as being widely supported through CAD design packages and software libraries.

4.17.2 Path planning

Point clusters must be avoided. Clusters appear when multiple points are located in a small area as shown in Figure 4-20. While printing, these areas are built up with material and if too much is built up, it can affect the print to the point that the nozzle catches on the material, which can lead to the print being dragged or the robot stopping due to it detecting a collision. It would be possible to avoid this by detecting clusters and removing or reducing them.

Figure 4-20: Clustering of points around centre

The algorithm for generating the fill path can be improved. Currently there are some limitations with it, which can be overcome using different algorithms. The path can be optimized to reduce the distance travelled between start and end points of each fill line. This will speed up the print, as it will have less distance to travel. Other methods of fill, such as scaling down the entity and travelling around those contours to fill them could be tested as replacement algorithms for the current method.

One major improvement would be to develop an algorithm that does not rely on the assumption that entities are all closed or all open paths. This assumption restricts the type of design that can be used. All closed entities follow the same assumption when filling them, laid out in Section 4.10, which is that there are start and stop points to define closed contours, while no open contours require any filling. When there is a mix of open and closed contours in the design and the open contour intersects with the closed one, the assumption is invalid and will not work correctly. A new algorithm could be developed to solve this.

4.17.3 Projection

Having a polynomial equation for the platform means to change the shape of the platform, the polynomial must be modified. Without modifications to the algorithm, it is not able to project designs onto complex shapes. It also means the polynomial must be known or worked out before the printer is used which makes changing the platform slightly harder.

To create complex geometry on different platforms, such as a printing on multiple faces of a cube or trapezoidal prism, or shapes that are not revolvable around a single axis, the projection

algorithm will need modification. It may be possible to use the same projection method, but rather than using an equation to define the platform, it could be possible to have a CAD model of the platform, such as a STEP file, that can be loaded and used as the surface to project the points onto.

If the user supplies a 3D design, the points do not need projecting. However, it would be prudent to check that the design will match the platform so a failed print does not occur. If there is a disparity between the design and the platform or the model is designed to protrude from the platform, it may be necessary to develop support material to help build this structure. Support structure could be built either using the same nozzle, or a second one. The second one may be filament based and could use water-soluble filament for support structure for easy removal. Having support structure greatly increases the type of designs this printer can print.

4.17.4 Communication

Developing the software to use the message queue, rather than sending and receiving messages one at a time, would allow the robot to build up a buffer. In the event that something delays the transmission of data from the computer, or the cable being temporarily unplugged, the buffer would be able to compensate and reduce the possibility of a failed print. This could be achieved by having a queue in the computer software, keeping track of the messages sent and the points printed.

The robot's controller is able to store files. A cache feature could be added to it where it writes the points and speed commands to a cache file on the controller as they are sent from the computer, without the robot moving. Once all the points have been written to the file, the robot uses the commands from the cache file instead of streaming them. By doing this, the robot can be disconnected from the computer while printing and it would not affect the print. It would also let the user print the same file again without having to load the file. This is useful for printing the same file multiple times. However, this would require a different method of controlling the extruder and nozzle height distance because the endeffector's position would be stored rather than being streamed in real-time. This new method could include sending commands to the extruder's controller from the robot's controller, and having a manual way of adjusting the nozzle height in addition to controlling it through software.

4.18 Chapter Summary

This chapter discussed the development of software algorithms used to convert the design to a workable format and project it onto the platform. The provided CAD design is imported and converted into 2D points, which are then projected onto the platform to form a 3D point cloud. Part of this conversion process involves generating the algorithms to remove parts of the design that overlap and create a fill path for the printer to follow.

This chapter also covered how these points, once projected are converted into a path for the robot to follow. A target and orientation for the robot's end effector is determined for each point, which is transferred to the robot, interpreted, and causes the robot to move to the specified location. All this software is controlled through a graphical user interface that is simple, but allows full control over all parameters.

The following chapter describes the process of the concurrent development of the hardware and software, tweaking of the parameters and design improvements influenced through testing of the printer.

Chapter 5: System integration and design implementation

5.1 Development

5.1.1 Extruder

When developing the extruder, it was attached to a testing platform rather than the frame. This led to easy access to parts of the design so troubleshooting and analysis could take place. The extruded material was tested for quality and aesthetics properties, ability to join onto previously printed material, and flow consistency. These factors were very important to get correct.

The quality and aesthetics of the extrusion is important because these define the look of the print. As this printer is designed to create prints that have a stronger focus on aesthetics than strength, a heavy emphasis should be placed on the aesthetics and the quality of the extrusion.

As the print is not one continuous path, the material must be able to form a bond with previously deposited material. If the paths were not able to join with enough strength, when touched or moved, the design would have the potential to break. Figure 5-1 shows the material being tested for bonding.

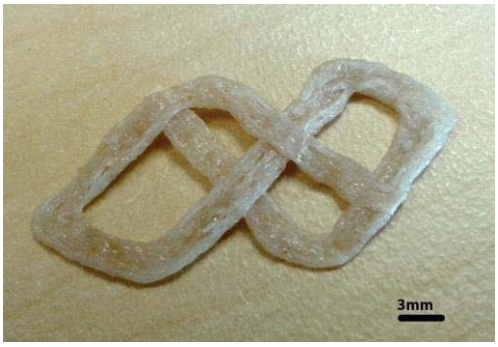


Figure 5-1: Bonding test for overlapping material

The flow consistency is one of the biggest factors of the extruder. If the flow is not consistent, it is not possible to create a reliable print. Some of the issues it could lead to are broken pieces, too much material, weakened structure, and holes in the design. Having one of these would mean the print would need to be thrown away. Figure 5-2 shows the effect of inconsistent flow. The right-hand side has a thicker extrusion than the left-hand side. This affects the fill path, as the same fill path will not work for the left-hand side and the right-hand side.

Figure 5-2: Flow inconsistency of extrusion

The extruder was designed as described in Section 3.3. The extruder was tested and the temperature and speed of the auger were tweaked to get the correct values. The extruder started with the 1.2 mm diameter nozzle, 200 RPM motor, and the temperature controller set at 230 °C. This worked well until the pressure got too much and the motor would cut out. The 11 RPM motor was added and run between 2 and 4 RPM, with the temperature controller set at 300 °C. This is very high for PLA polymer. However, the motor was unable to turn more slowly without overheating, as it could not generate the torque required to generate the pressure to extrude the material. The high temperature allowed the melt to extrude without causing the motor to stop.

5.1.2 Software algorithms

Before running the software, the algorithms are tested using simulation methods to ensure they are correct and will need as little tweaking as possible during testing.

The algorithms used to convert the DXF format to 3D points, were tested, and verified by converting the points back into DXF form and viewing them in a DXF viewer. By comparing this to the original design, it is possible to see if an algorithm is working correctly and modify the software until it is.

The algorithms to transform Point3D objects to robtargets were simulated in RobotStudio. The robtargets for an entity were generated and exported into a text document. To test this algorithm, a rectangle, circle, and hexagon (Figure 5-3) design were used because the robot is only able to store a certain number of targets, and using the provided design would generate too many.

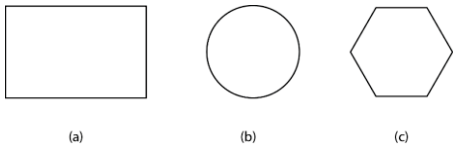


Figure 5-3: Test designs. (a) Rectangular design, (b) circular design, (c) hexagonal design

The robtargets generated from the software were copied into RobotStudio so they could be simulated. This simulation is used to check the robot will move along the correct path, and not make any unexpected movements that will cause collisions. It is also used to check that unexpected errors, such as “joints out of range” do not occur. Any time changes were made to the transformation algorithm; this process was used to make sure no damage to the hardware, or injuries the user occurs. Figure 5-4 (a) shows the simulation of the contour path of a single entity. It shows the end effector moves along the Y-Z plane while rotating axis 6 around the Z axis as expected.

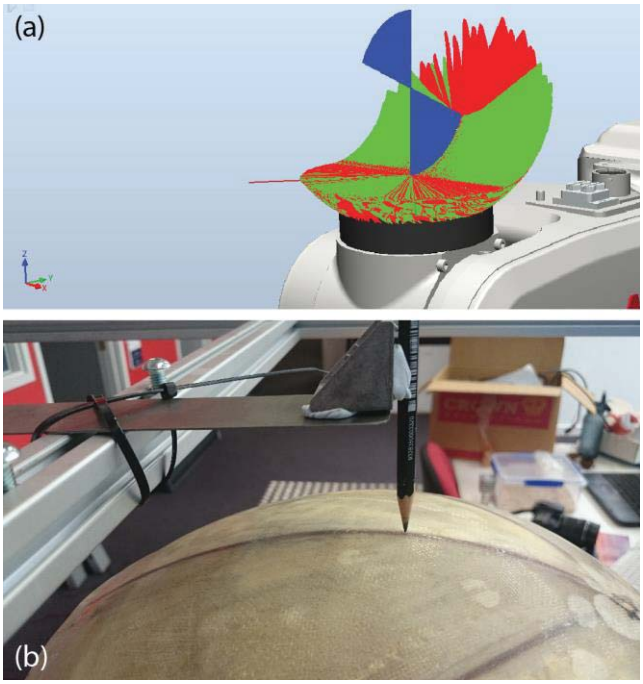


Figure 5-4: Methods of simulation. (a) Through RobotStudio, and (b) with a pencil

As it is not possible to see the output of the printer on the platform during simulation, a pencil was attached to the frame in the position of the extruder to simulate the path that would be printed (Figure 5-4 (b)). By using the pencil rather than the extruder, it was possible to reduce the number of variables in the system and focus on troubleshooting the software algorithms for path generation.

As there are two rotation matrices used together to transform the point3D object to a robtarget, this method was used with different designs to test each of them. The circle model was used to check if there was any problem in transformation around the Z axis. If the circle was drawn with the correct radius and no distortion, the transformation around the Z axis was correct.

The square and hexagon models were used here because using straight lines was an effective way to determine if the algorithm was correct. If the lines had any deformation in them, they could be used to determine where the problems were occurring in the algorithm that transforms around the X axis.

5.2 Testing

As the printer was developed, each new stage was tested to make sure it was working correctly. During testing, many problems occurred which were investigated to find the cause.

First, the extruder was designed and tested until it was running for extended periods with no issue. Next, the software was developed to the point where it generated robtargets. Then the hopper was placed onto the frame and run to find issues. A reference design was 3D printed using SLS technology to get an accurate geometric representation of the design that could be used to test against the results.

5.2.1 Testing of the Point3D to robtarget conversion algorithm

Initially, the incorrect value for the rotation about the X axis was used. Originally, when calculating the angle of the tangent line, the point's spherical radius was used with a different formula. This problem was not picked up when testing with the pencil as the pencil was able to move in the vertical direction and still produced a shape similar to the one wanted. This incorrect value gave results that were close while the end effector was vertical, but as the point moved further from the origin, the radius increased and the platform moved closer to the nozzle. There was a height increase of 3mm at the furthest point. As the platform moved closer to the nozzle, it deposited less material until they made contact and no material was deposited. As the extruder was still running, this increased the pressure and when the nozzle moved further away, the pressure was released quickly and lots of material was deposited at once.

It was attempted to rectify this by creating a guide system to keep the nozzle the correct distance from the platform. These guides, seen in Figure 5-5, used a ball to roll across the surface of the platform and to lift the extruder nozzle and keep it the same distance from the platform. These guides worked to keep the distance between the nozzle and platform at a constant distance. One guide was on either side of the nozzle as the platform is not a true hemisphere with a constant radius.

Figure 5-5: Guides resting on platform to keep the nozzle height constant.

When running over recently deposited material the guide picked it up, dragging it across the platform and deforming the print. Because of this, the guides were removed and the software was reviewed. It was attempted to lower the platform by a percentage based on the spherical radius of the point. i.e. $Z = Z - 0.02 \times radius$. This helped, but it was not very effective, as it would raise the platform for every point, including the ones close to the XY origin. While reviewing the software, it was noticed the formula for calculating the angle of the tangent line was incorrect and once rectified by using the value of the radius projected on the XY plane, the nozzle distance stayed correct and constant.

5.2.2 Print failing due to over extension of axis 6

During testing, it was found that the printer would stop and the robot would give the error "Joint Out of Range", which indicated axis 6 was going outside the -400° to +400° working range. As the first entity encompasses the origin, axis six rotates a full 360°. When printing the subsequent entities, axis six only has 40° to rotate in the same direction before exceeding the working range. This problem occurs because ConfJ and ConfL are turned off. Without monitoring the configuration of the robot, there is no way to differentiate between angles with aliasing: e.g. -350°, 10°, and 370° all lie on the same point.

This was resolved by returning the robot to the home position every time it finishes printing an entity. This resets axis six so it has enough motion to reach each point. It also proved helpful to implement a function to arrange entities by their angle around the platform so the printer prints entities sequentially rather than jumping between entities that do not lie next to one another.

5.2.3 Entities not joining to each other

Another challenge faced was entities printed after the first one not sticking to the platform. The further away from the centre of the platform the entity started, the more likely this was to occur. It was attempted to move the platform closer to the nozzle by 0.2mm for the first five points printed. This occasionally made a difference, but sometimes the platform was too close to the nozzle for the material to form properly on the surface of the platform.

The solution to this was to start each entity from a point on a previously printed entity. Initially, each entity started printing at the point closest to 0°. Starting the print from a point on a previously printed entity gave the material something to bond to and helped to stop the print from failing.

5.2.4 Tags left over from printing

Because the robot prints an entity, then stops, and prints the next one rather than a continuous path between entities, each time the robot moves to the next entity, the hopper continues to extrude even when turned off. This happens due to the backpressure built up in the extruder, which keeps forcing out material after the extruder has been turned off. This would leave tags, like the ones shown in Figure 5-6 (a), on the design that would need to be removed by hand later. In most FDM 3D printers that use filament, it stops extruding faster almost instantly, and follows a continuous path so tags are not produced.

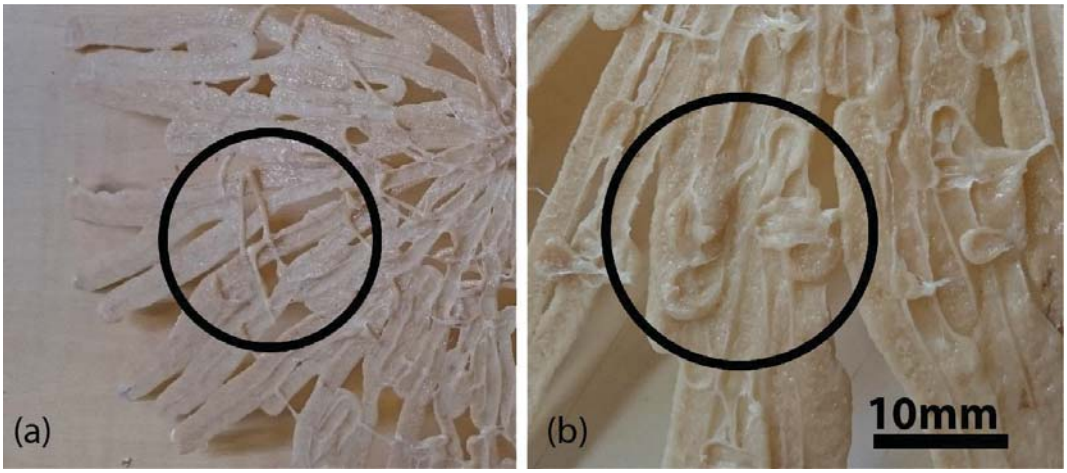


Figure 5-6: (a) Tags and (b) artefacts left by the printing process

Multiple methods were implemented to solve this. Firstly, it was attempted to stop the extruder before the entity had finished so the backpressure was used to print the rest of the entity. This method was rather unreliable as the pressure in the extruder was variable. Next, instead of stopping the extruder, reversing it was trailed. This improved the backpressure drop, but took a while. The nozzle was kept on the design while it was reversing, which worked to create smaller tags, but left other artefacts, such as the ones shown in Figure 5-6 (b) on the design.

Moving the robot to the home position very quickly while the extruder was reversing helped to take the tags off, as they were broken when the platform moved. This could leave drag lines on the designs that would need to be broken off later. Lastly, the 11 RPM motor was switched for a 4 RPM worm drive motor. There has been limited testing with the new motor. However, current results look promising and seem to solve this issue entirely, with no need to reverse the motor.

5.2.5 Horizontal vs vertical fill

Horizontal fill is material that is deposited on the platform in lines parallel to the base of the platform, within the boundaries of the entity, while vertical fill is material deposited in lines normal to the base of the platform, within the boundaries of the entity.

When filling the entities in a horizontal fashion there are many start and stop points. At each of these, a tag can be generated. To reduce the size and number of tags the robot also moves very quickly between the stop point of one fill line to the start point to the next. This can leave very small strands between each line that can be removed after printing. These strands can be seen in Figure 5-7.

Figure 5-7: Horizontal fill path trial

It was also found when filling in the gaps of the entities, the extruder would deposit much more material than was required, raising the height of the print in localised areas. This happened due to the pressure building up while the extruder was moving over previously deposited material. These areas of excess material can also be seen in Figure 5-7. To try to lower this pressure, the speed of the extruder was reduced while printing the filling. While this reduced the amount of overflow, because the extrusion was not exiting the nozzle so quickly, it was not as aerated which lead to a smaller path width.

Because the speed of the extruder was reduced but the pressure was still built up from printing the contour, this lead to very compact fill lines at the start of the entity and gaps between them at the end of the entity. The distance between the fill lines was tweaked to try to solve this. Unfortunately, this did not make any positive difference to the fill. To reduce the backpressure in the extruder, the extruder can be stopped or reversed until the flow stops. This still leaves some material being extruded that must be removed manually. It may be possible to automate this process in the future. However, it does add extra time to the print.

It was decided the look of the horizontal fill was too rough and would need a large amount of post processing to rectify this. The fill algorithm was changed so the printer was filling the entity vertically. This greatly changed the quality of the printed fill. When printing vertically on this design, it was noted a lot of the fill was going over previously deposited material. To solve this, the algorithm was modified to print only in areas where it was needed. Tweaking the threshold for this algorithm greatly increased the effectiveness of the fill path and deposition of the fill material.

5.2.6 Path width

The width of the path is rather wide. When comparing the reference design with the printed design, the printed design is roughly twice the width of the reference design.

It appears this is because the melt is extruded from the nozzle rather quickly. This aerates the melt, which, along with the die swell effect that causes it to expand as it exits the nozzle, causes the material to expand beyond the diameter of the nozzle orifice. Because the material is deposited while it is expanded, when it cools and the air escapes, the deposited material contracts onto the platform. As it has already been deposited, the relatively high friction between the platform and the material means it does not contract sideways, but rather flattens itself towards the platform. Because of this, the path cools down and sets wide. The aeration also gives it the bubbly texture as it cools down.

To rectify this, the speed of the extruder was turned down. Occasionally this worked. However, even with the temperature turned down, often the material would come out burnt, thin and with an inconsistent flow. The motor also got very hot while running like this. It is thought this occurred because the material was spending too much time in the chamber being heated, as the motor did not have enough torque to push it through as it melted.

5.2.7 Nozzle

Smaller nozzle diameters of 0.8mm and 1.2mm were tried. The 0.8mm diameter nozzle had problems with the fibres fitting through it. Because the fibres are too big to fit through the nozzle, the print fails. Once it fails, the nozzle must be detached and the fibre creating the clog removed. The fibres fit through the 1.2mm nozzle. However, it requires more pressure to extrude the melt. The 11 RPM motor does not provide enough torque to deliver this pressure.

5.2.8 New motor

A 12 V, 4 RPM, worm-gear-driven motor with 1.37 Nm of torque (AliExpress, n.d.) was recently chosen to replace the 11 RPM one. The 4 RPM motor hasn’t yet been tested thoroughly. The temperature was lowered to stop the material from burning as it spent longer in the extruder barrel. The resulting melt when using the 1.8 mm nozzle was not aerated and did not expand as much as previously. With this motor, the path width is narrower, averaging 3.2 mm, making the print of the design much more accurate in comparison to the reference design.

As Figure 5-8 shows, when the extruder is run at 4 RPM with the temperature controller set at 220 °C, the quality of the print is much better. There are few to no tags, bulges, or other artefacts from the filling.

However, because the flax fibre stays in the barrel for longer, even with the lower temperature, it is extruded as a deeper colour. The 4 RPM motor has not yet been tested with the 1.8 mm diameter nozzle but it is suggested to trial this as it could reduce the path width even further.

Figure 5-8: Extrusion quality of 4 RPM motor

5.3 Chapter Summary

This chapter described the process gone through to create a functional design by testing and concurrent development and adjustments to the 3D printer extruder and software. Initially the extruder was tested on its own to get the right values for extrusion without burning the fibres or degrading the polymer. Next, the software was tested through simulation, both in RobotStudio, and using a pencil to simulate the extruder, to get the software algorithms correct.

When starting testing on both the extruder and software concurrently, only the contours were printed so the fundamental principles of printing the design could be tested, such as joining entities, reducing the path width, and printing a full design without receiving errors from the robot or software. Next, the method of filling in the contours was tested and the parameters and algorithm tweaked and improved until a full design was printed. Finally, developments were made to increase the accuracy of the design and improve the aesthetic quality of the design to reduce the amount of post-processing required.

The next chapter discusses the results obtained from the functional design against the objectives of the project.

Chapter 6: Results and discussion

6.1 Develop a pellet-based extruder system for 3D printing biopolymers

The extruder system created takes the mixed PLA polymer and flax pellets from the hopper, 3D printed from nylon powder using a SLS machine, through a machined aluminium extruder with heat sink and out of a 1.8 mm nozzle by running an auger drill bit in reverse. The heat is provided by a barrel-heating band, which is controlled through its own PID temperature controller. The auger is turned by an 11 RPM motor, which is controlled with a motor shield and custom PCB attached to an Arduino controller connected to the computer.

The extruder system was thoroughly tested before integrating it into the 3D printer to make sure it provided a constant flow of material. Once it was working proficiently and the software had been developed, it was attached to the frame and tested by printing contour lines. When the tuning of the parameters was correct, the fill lines were also printed and parameters tweaked until an acceptable print was made.

The extrusion system is a rudimentary design which effectively 3D prints the provided design. Currently the print has a very wide path. However, the addition of the worm gear driven motor significantly reduced the path width in recent tests and shows promise in making the print more accurately represent the design in terms of dimensional accuracy and aesthetics as it will reduce the amount of post processing required.

Many variables affect screw extruders including: temperature, screw speed, ambient temperature, humidity, pressure inside the system, material makeup, etc. Through testing, several areas that require improvements have been identified, and some possible solutions proposed in Section 3.3.10.2.

6.2 Develop software and parameters to convert the design into commands for the robot

The software currently can convert multiple designs into a path for the robot to move along to create a print of the design. It was important for the software to be generic enough to be able to convert a wide range of designs, as it will be used as a commercial 3D printer for a variety of concepts.

The adjustable parameters in the conversion process are easily accessible to the user through the GUI. This gives the user full control over the process to improve their prints. The conversion

process is relatively quick due to the use of multiple threads, allowing certain operations to be done concurrently. There are a few improvements that could be made such as changing the file type to a more generic 3D printing one (such as STL, which would allow the design more complexity, as it would accurately represent a 3D body) improving the fill algorithm to optimize both the printing path and the computation of creating the path.

6.3 Develop software to control the robot

The robot is controlled through commands sent to the robot's controller. These commands are generated by using affine transforms to convert the points from the generated point cloud to a location and orientation for the robot's end effector.

The method of streaming the points to the controller is effective and has many benefits to it, the main one being the speed of the application as the user does not have to wait for a large number of commands to be sent and stored on the robot. There are some drawbacks to this method such as the robot stopping if the communication between the robot and computer is interrupted.

This aspect of the software has many improvements that can be made which will improve the robustness of it. By improving the robustness of the software, it increases the viability for it to be used for commercial application as it increases the reliability of it being able to print an exact replica without errors.

6.4 Build a functional design using the developed 3D printer

The printer and the software created to control it worked to create the print shown in Figure 6-1. This was printed using the 11 RPM motor turning at 2.5 RPM and a temperature of 220 °C on the temperature controller. Currently this design takes 2 hours and 43 minutes to print. This will be reduced as improvements are made.

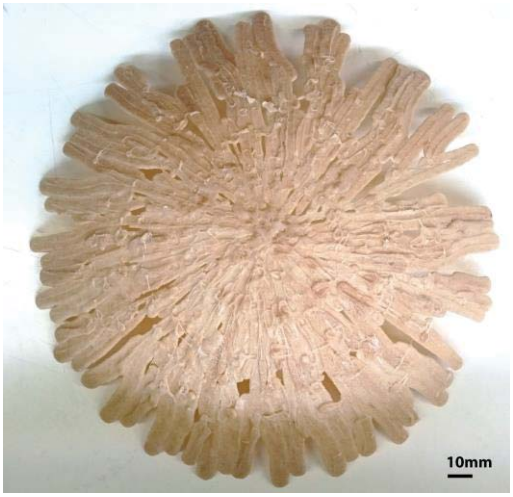


Figure 6-1: 3D printed design

6.4.1 Visual aesthetics

The aesthetics of the print are concerned with the look of the overall item.

Figure 6-2: Close-up of surface finish

The surface finish, which can be seen in Figure 6-2, shows the design after the tags have been taken off by hand using a pair of side cutters. It is possible to see the spots where the extruder has stopped and left an imprint, as well as lines where the nozzle has gone over previously printed material and dragged it slightly. It is also possible to see a few drag-lines that need to be removed, and fill paths that were printed mostly on top of existing material.

For the initial development of a printer, the resulting prints are very good in terms of visual aesthetics. Because it is a completely new process, with a relatively basic extrusion method, and there are many variables that require controlling, the print is considered a success. With more time and testing, as the variables are tuned, the results will only get better.

There are some parts of the surface finish that could be improved by removing the imperfections. Sanding down the print would smooth the surface and remove imperfections. This would greatly improve the visual aesthetics of the print. It would also make it easier to print a second layer.

It would be possible to replace the extruder with a sanding tool, after a print had been finished, and run the robot around the print again, or spiral from the centre outwards until all the print has been sanded.

The extrusion melt is well mixed and has fibres evenly dispersed. This is verified by the constant colouring of the material throughout the print as different ratios or mixing would lead to changes in the colour. The fibres can be seen, with very few bubbles, and a shiny surface finish.

6.4.2 Dimensional accuracy

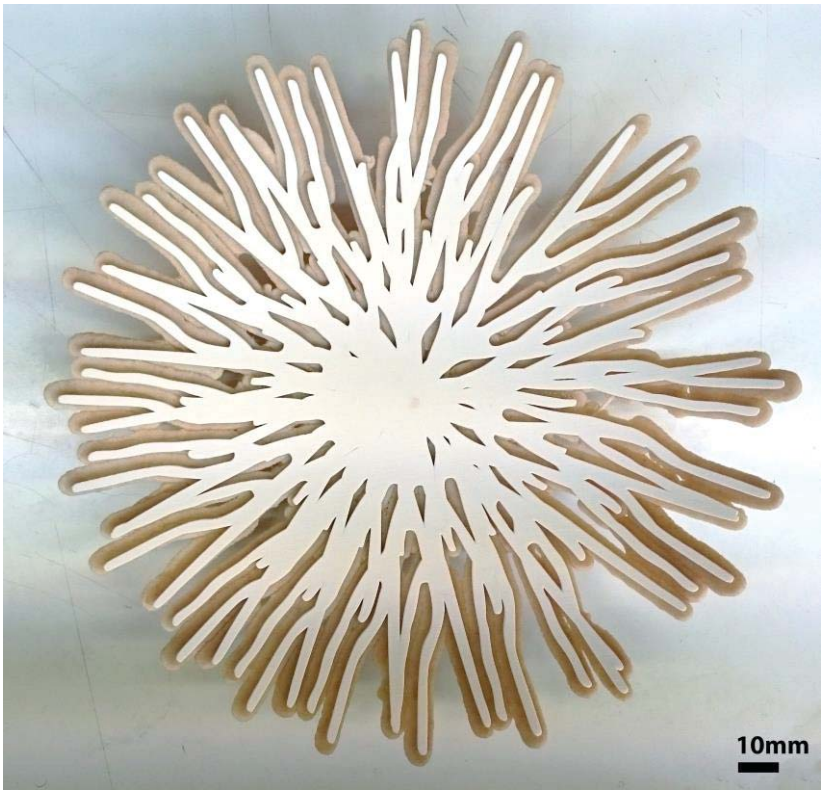


Figure 6-3: Comparison of reference design and print

It is important to get the contours of the design correct because they create the shape of the print. If the contours are not printed correctly, the print won't accurately reflect the design.

As seen in Figure 6-3, the contours of the design follow the shape of the reference one as when they are overlaid, the print, although thicker than the reference design, follows the same shape and path. If the path width were smaller, the contours would match even more accurately. The geometric accuracy of the print shows the software algorithms and extruder used to create the print are accurately transforming the provided design into a functional print.

As seen in Figure 6-1 and Figure 6-2, the gaps in the entities are filled where required. The reference design cannot be seen behind the print. Apart from the height added to the layer from the fill path, the fill material covers all the gaps without too much excess material.

The curvature of the print is very accurate. When placed against the reference design, the curvature of the design and reference are the same. This is because the design is printed on a platform with the correct curvature already and there is no deformation after the design has been removed from the platform.

The average path width of the reference design, as in Figure 6-4 is 4.2 mm, while the average width of the print is 13 mm.

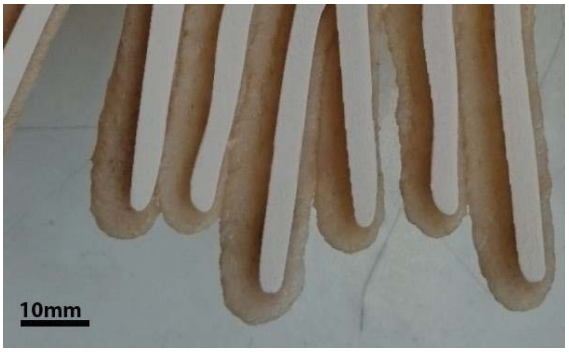


Figure 6-4: Comparison of print and reference design

The print is 3.1 times wider than the reference design. The width of a single path is 6.4 mm. This could be greatly improved. By using a different algorithm that takes into account the actual width of the path and travels inside the contours of the design (Figure 6-5 (b)) instead of along them (Figure 6-5 (a)), the print's path would be much more accurate, giving a print much closer to the intended design. Using a smaller nozzle along with this, could give optimal results.

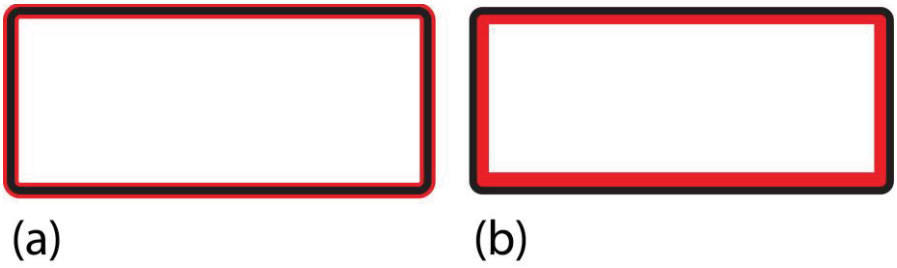


Figure 6-5: Printing along a contour. (a) Over the path. (b) Within the path.

The results obtained from testing the printer with the 4 RPM motor (Figure 6-6) show a significant improvement in the reduction of the path width, and the quality of the surface finish. As the figure shows, the path width is much thinner (3.2 mm), which makes the print represent the design more accurately. There are also no artefacts from printing which greatly increases the surface finish. However, the print is thinner than previously, which weakens the overall design and shows a requirement for additional layers or structural reinforcements. As mentioned previously, this motor has not been fully tested but the results are promising and indicate they should be investigated further.

Figure 6-6: Result from testing the new motor

6.4.3 Structure

The entities are joined together very well. When holding the design, it feels sturdy and unlikely to break. Looking at Figure 6-7 it is possible to see many of the paths have a large amount of surface area bonding them to each other.

Figure 6-7: Underside of 3D printed design

The outer surface of the print feels very rough due to the artefacts discussed in Section 0. Because of these, the product does not feel finished, sufficiently refined to a commercial standard. A post processing operation to smoothen the outside surface, while keeping a slightly organic feel, would help to make this design commercially ready.

6.5 Mr Trubridge's perspective

Mr Trubridge had some positive comments about the process and print. Overall, he is pleased with the printing process and the resulting print and wishes the project to continue. He feels that the dimensional accuracy, together with the resulting aesthetics of the design is great but there is room for improvement. To help with the accuracy of the print, he will review the design so it can be made with single contours without the need for any filling. He also drew inspiration from the printed designs to create designs that follow one continuous path, something that will require extra software development to print correctly. It was agreed that, with further development of the extruder, the prints would more closely represent the designs.

Mr Trubridge did request the print time be reduced to make it more commercially viable. Following the recommendations set out in the previous chapters, along with the designs being reviewed to suit the printing process better, it would be possible to reduce the time taken to create the print.

Chapter 7: Conclusion

The material was deposited using a pellet-based extrusion system, constructed with a 15mm auger drill bit inside an aluminium barrel with heat sink, and a 3D printed hopper from nylon. The auger was turned by an 11 RPM, 0.93 Nm motor, turning at 2.5 RPM. An Arduino micro-controller controls this, while a Watlow heating band and temperature controller provide the heating, set at 300 °C. This extruder worked very well for a relatively simple design. With most extrusion systems, there are many variables that affect both function and aesthetics of the extruded material. This design can be further developed to improve the control in these factors. When the design was printed with the 11 RPM motor, the extrusion width was overly wide compared to the reference design. The new, 4 RPM motor shows some promise in reducing this and should be developed further.

The software accurately converts the design from a DXF format to points on the platform. This is evident by turning the projected points back into DXF format and comparing the results. The results show no difference in the shape of the design. The matching shape of the print against the reference design is also testament to this. The software has been designed specifically to make it easy to continue its development.

The mapped 3D points are successfully converted to targets for the robot to move to. This conversion is correct as the nozzle stays a set distance from the platform while moving. The robot is able to adjust its speed and position based on the commands sent. This is verified by the accuracy of the final printed design.

This 3D printer successfully prints Mr Trubridge's design. There are many further improvements which can be made, as outlined in the previous chapters. Overall, this project is a success and serves as an excellent basis for further development.

Chapter 8: References

ABB. (n.d.-a). IRB 360 - Indistrial Robots (Robotics). Retrieved from http://new.abb.com/products/robotics/industrial-robots/irb-360

ABB. (n.d.-b). IRB 1520ID - Industrial Robots (Robotics). Retrieved from http://new.abb.com/products/robotics/industrial-robots/irb-1520id

ABB Robotics. (2012). Application manual - PC SDK.

Aduino Inc. (n.d.). Arduino - Windows. Retrieved from https://www.arduino.cc/en/Guide/Windows

Ahn, S.-H., Montero, M., Odell, D., Roundy, S., & Wright, P. K. (2002). Anisotropic material properties of fused deposition modeling ABS. *Rapid Prototyping Journal, 8*(4), 248-257.

Aized, T. (2010). Flexible manufacturing system: hardware components. *Future Manufacturing Systems*, 1-16.

AliExpress. (n.d.). Wholesale 6v 12v 24v JGY-370 worm reverse gear motor 1-375rpm low speed motor High Torque Dc Motor Low Noisy For Diy Experiment. Retrieved from http://www.aliexpress.com/item/Wholesale-Jgy-370-12V-Small-Motor-1-375Rpm-Gear-Motor-6V-High-Torque-12V-Dc-Motor/32373494696.html?spm=2114.01010208.3.37.LBLDCb&ws_ab_test=searchweb201556_9,searchweb201644_2_505_506_503_504_301_502_10001_10002_10016_10017_10010_10005_10006_10011_10003_10004_10009_10008,searchweb201560_5,searchweb1451318400_-1,searchweb1451318411_6449&btsid=0aa696d0-ce68-4b29-89da-87a6becb1f20

Altintas, A. (2015). A new approach to 3-axis cylindrical and cartesian type robot manipulators in mechatronics education. *Elektronika ir Elektrotechnika, 106*(10), 151-154.

Anitha, R., Arunachalam, S., & Radhakrishnan, P. (2001). Critical parameters influencing the quality of prototypes in fused deposition modelling. *Journal of Materials Processing Technology, 118*(1), 385-388.

Autodesk. (2011). DXF Reference.

Autodesk. (n.d.). Drawing Interchange File Formats. Retrieved from http://www.autodesk.com/techpubs/autocad/acadr14/dxf/drawing_interchange_file_formats.htm

Biggs, G., & MacDonald, B. (2003). *A survey of robot programming systems.* Paper presented at the Proceedings of the Australasian conference on robotics and automation.

Carneiro, O. S., Silva, A. F., & Gomes, R. (2015). Fused deposition modeling with polypropylene. *Materials & Design, 83*, 768-776. doi:10.1016/j.matdes.2015.06.053

Chung, C. I. (2011). *Extrusion of polymers : theory and practice*: Munich : Hanser Publishers ; Cincinnati : Hanser Publications, c2011

2nd ed.

Craig, J. J. (2005). Introduction to Robotics.

Diegel, O., Singamneni, S., Huang, B., & Gibson, I. (2011a) Curved layer fused deposition modeling in conductive polymer additive manufacturing. & U. Guangxi, T. Guilin University of Electronic, W. University of, U. Korea Maritime, & C. Hong Kong Industrial Technology Research (Vol. Ed.)*: Vol. 199-200. 2nd International Conference on Manufacturing Science and Engineering, ICMSE 2011* (pp. 1984-1987). Guilin.

Diegel, O., Singamneni, S., Huang, B., & Gibson, I. (2011b) Getting rid of the wires: Curved layer fused deposition modeling in conductive polymer additive manufacturing. & S. National University of, S. International, & C. Engineering (Vol. Ed.)*: Vol. 467-469. 2011 International Conference on Materials, Mechatronics and Automation, ICMMA 2011* (pp. 662-667). Melbourne, VIC.

Donelan, P. (2007). Singularities of robot manipulators. *Singularity Theory*, 189-217.

Donghong Ding, Z. P., Dominic Cuiuri, Huijun Li. (2015). Wire-feed additive manufacturing of metal components: technologies, developments and future interests. *International Journal of Advanced Manufacturing Technology, 51*(1-4), 465-481. doi:10.1007/s00170-015-7077-3

Epson. (n.d.). Epson G3 SCARA Robots. Retrieved from http://robots.epson.com/product-detail/2

Groover, M. P. (2015). *Automation, Production Systems, and Computer-Integrated Manufacturing*: Pearson Education.

Harris, I. D., & Director, A. (2011). Development and Implementation of Metals Additive Manufacturing. *DOT International, New Orleans*.

Hassel, B. (2012). Prevent Computer Sleep in C#. Retrieved from http://www.brianhassel.com/2012/08/i-am-wrapping-up-development-on-backup.html

Hayter, C. (2009). C# - Remove duplicates in the list using linq. Retrieved from http://stackoverflow.com/questions/1606679/remove-duplicates-in-the-list-using-linq

Hudson Robotics. (n.d.). PlateCrane EX Microplate Robot Arm. Retrieved from http://www.hudsonrobotics.com/products/microplate-handling/platecrane-ex/

Isaksson, M., Brogårdh, T., Watson, M., Nahavandi, S., & Crothers, P. (2012). The Octahedral Hexarot—A novel 6-DOF parallel manipulator. *Mechanism and machine theory, 55*, 91-102.

Jazar, R. N. (2011). *Advanced dynamics: rigid body, multibody, and aerospace applications*: John Wiley & Sons.

LBushkin. (2009). .net - C# cleanest way to write retry logic? Retrieved from http://stackoverflow.com/questions/1563191/c-sharp-cleanest-way-to-write-retry-logic

Low, C. Y., Chong, K. H., Salleh, K., & Johnny, K. S. P. (2010, 13-14 Dec. 2010). *Path optimization using genetic algorithm evolution.* Paper presented at the Research and Development (SCOReD), 2010 IEEE Student Conference on.

Manas-Zloczower, I. (1997). Analysis of mixing in polymer processing equipment. *Rheology Bulletin, 66*(1), 5-8.

Miyazaki, D. (n.d.). Convert from rotation matrix to quaternion. Retrieved from http://www.cg.info.hiroshima-cu.ac.jp/~miyazaki/knowledge/teche52.html

Montero, M., Roundy, S., Odell, D., Ahn, S.-H., & Wright, P. K. (2001). *Material characterization of fused deposition modeling (FDM) ABS by designed experiments.* Paper presented at the Proceedings of Rapid Prototyping and Manufacturing Conference, SME.

MX3D. (2015a). Metal - MX3D. Retrieved from http://mx3d.com/projects/metal/

MX3D. (2015b). MX3D Bridge. Retrieved from http://mx3d.com/projects/bridge/

Rauwendaal, C. (2010). *Understanding extrusion* (2nd ed.). Munich Cincinnati, Ohio: Hanser.

Rauwendaal, C. (2014). *Polymer Extrusion*: Carl Hanser Verlag GmbH & Co. KG.

Reddy, B., Reddy, N., & Ghosh, A. (2007). Fused deposition modelling using direct extrusion. *Virtual and Physical Prototyping, 2*(1), 51-60.

Robotics Bible. (n.d.). Robot Drive Systems - Robotics Bible. Retrieved from http://www.roboticsbible.com/robot-drive-systems.html

RobotShop. (n.d.). 10A DC Motor Driver Arduino. Retrieved from http://www.robotshop.com/en/10a-dc-motor-driver-arduino-shield.html

Sahari, K. S. M., Weng, K. H., Han, Y. W., Anuar, A., Baharuddin, M. Z., & Mohideen, S. S. K. (2012). Design and development of a 4–dof SCARA robot for educational purposes. *Jurnal Teknologi, 54*(1), 193–215.

Schilling, R. (2013). Fundamentals of robotics.

Schunk. (2015). Depanelling Technology - The fastest Depanelling Technology in Electronic Production.

Sood, A. K., Ohdar, R., & Mahapatra, S. (2010). Parametric appraisal of mechanical property of fused deposition modelling processed parts. *Materials & Design, 31*(1), 287-295.

Standardization, I. O. f. (2014). ISO 13482:2014 *Robots and robotic devices*.

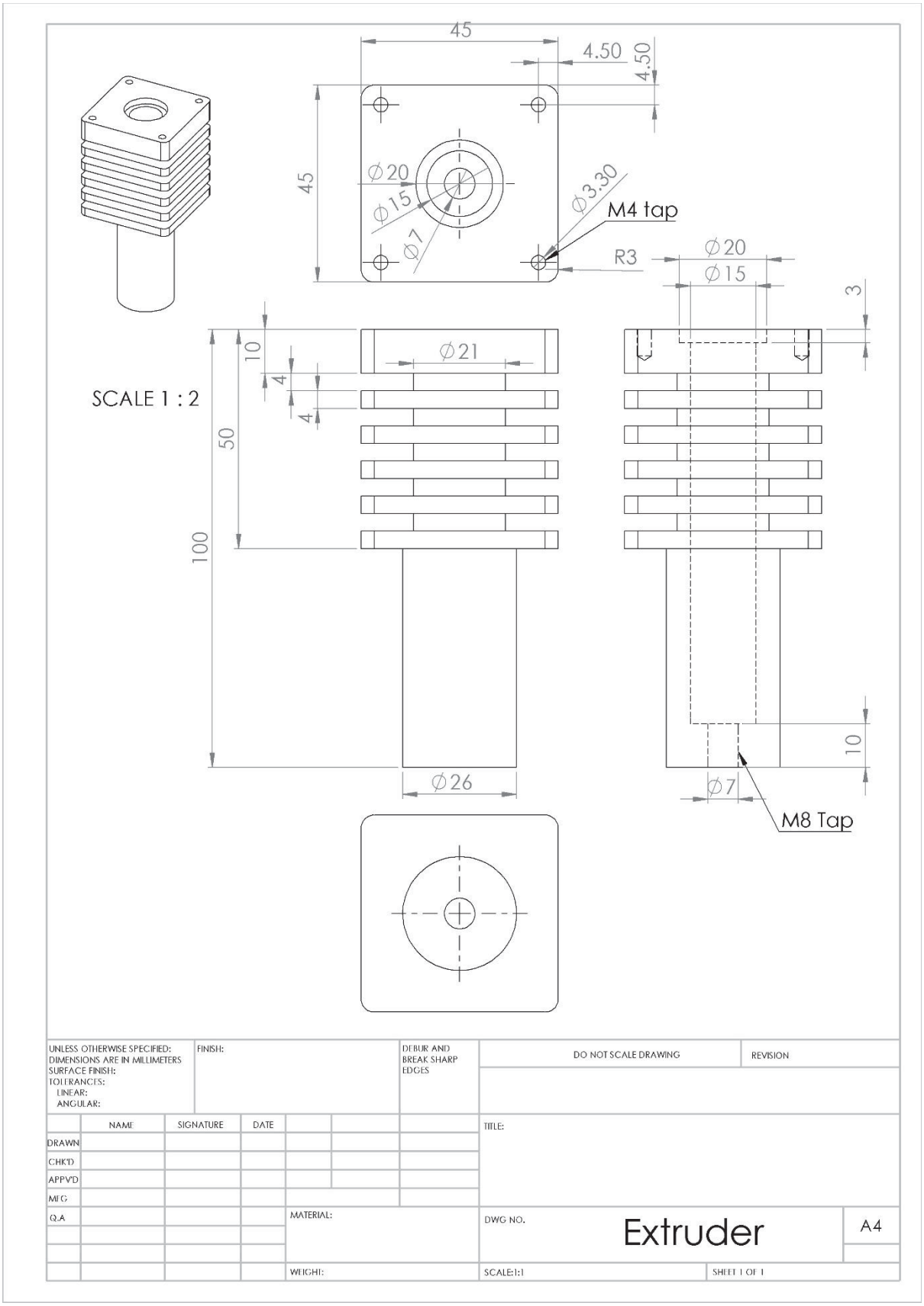
Tseng, A. A., Lee, M., & Zhao, B. (2001). Design and operation of a droplet deposition system for freeform fabrication of metal parts. *Journal of engineering materials and technology, 123*(1), 74-84.

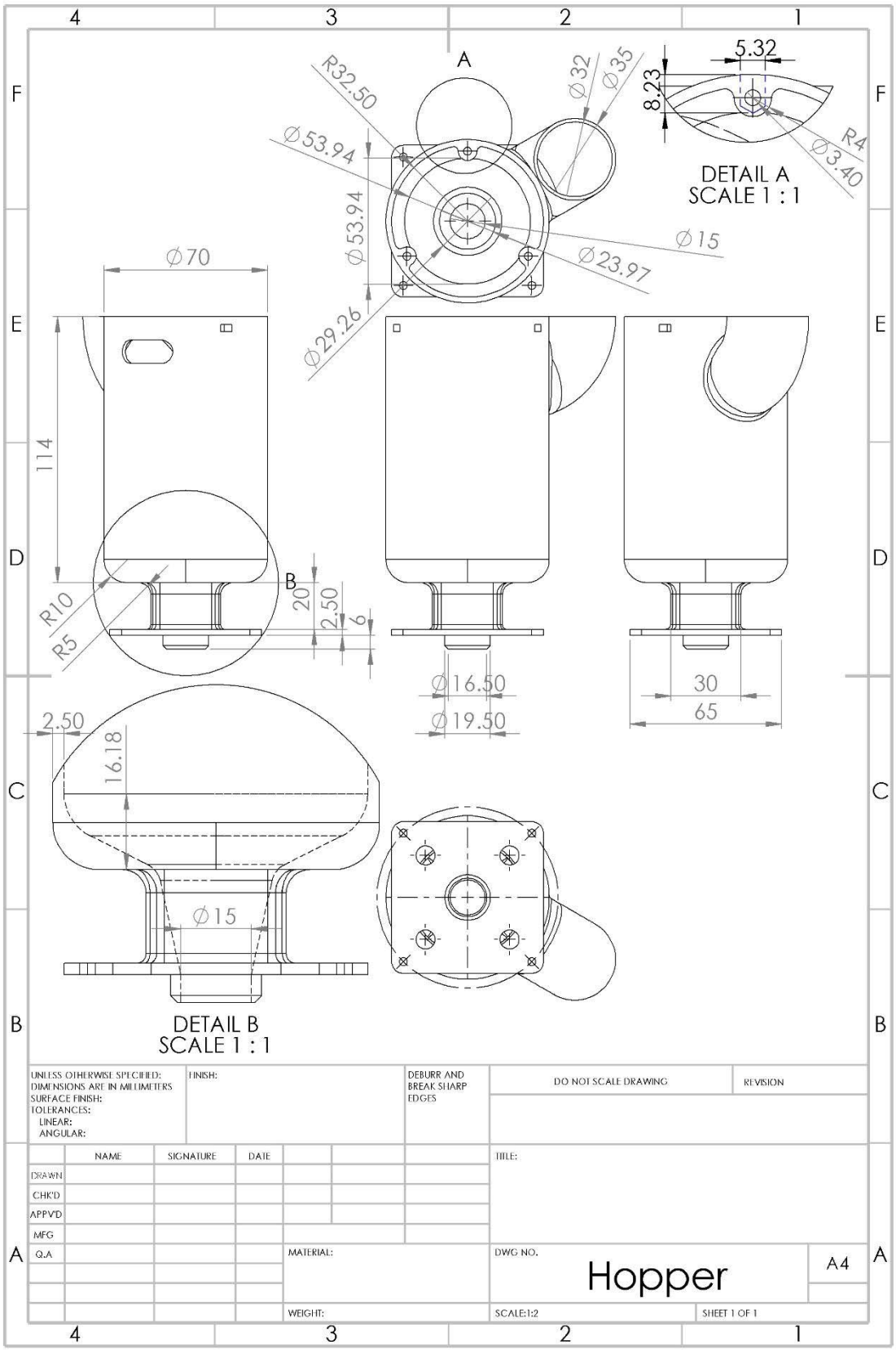
Valkenaers, H., Vogeler, F., Ferraris, E., Voet, A., & Kruth, J.-P. (2013). *A novel approach to additive manufacturing: screw extrusion 3d-printing.* Paper presented at the Proceedings of the 10th International Conference on Multi-Material Micro Manufacture.

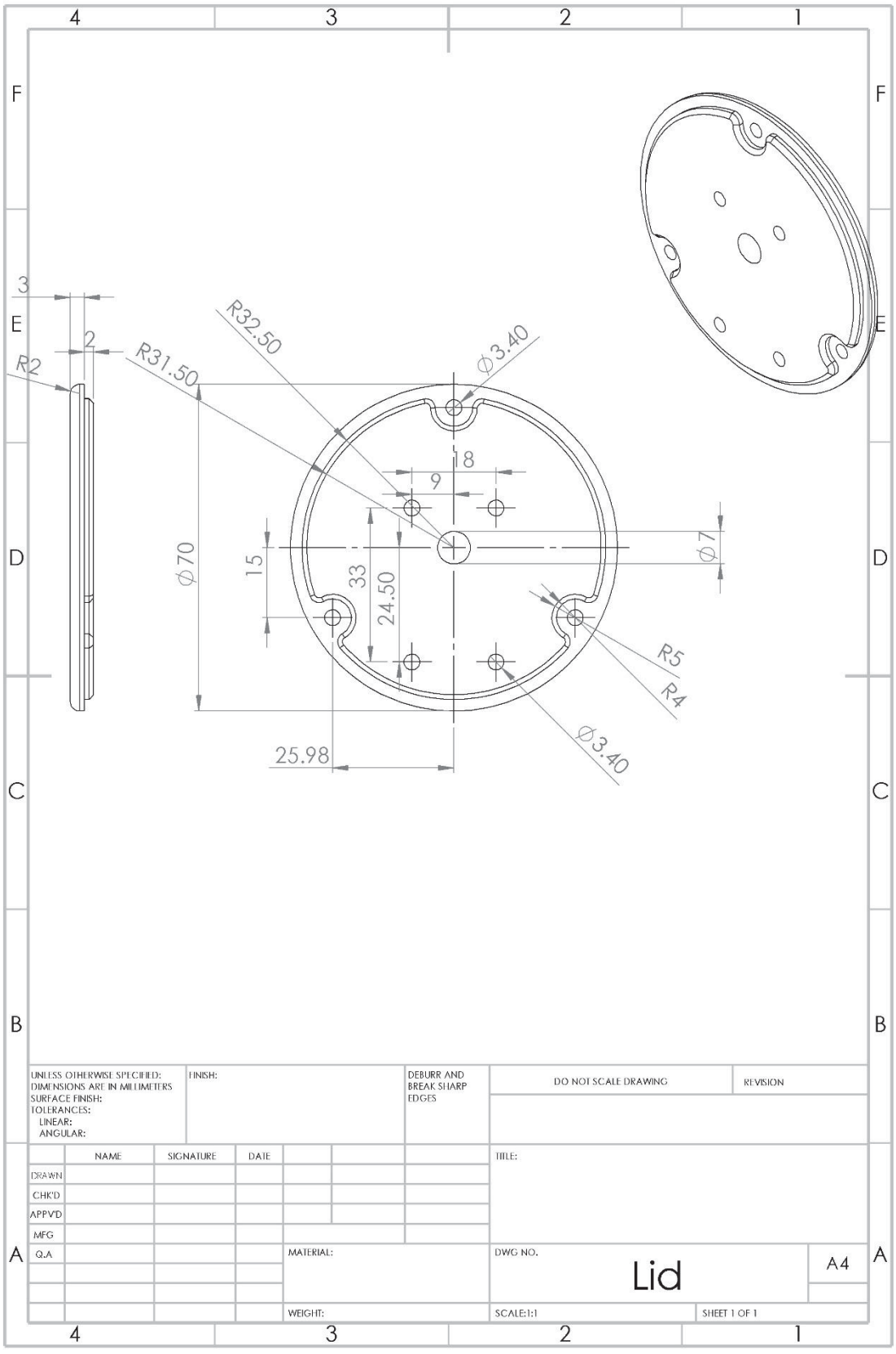
Vatti, B. R. (1992). A generic solution to polygon clipping. *Communications of the ACM, 35*(7), 56-63.

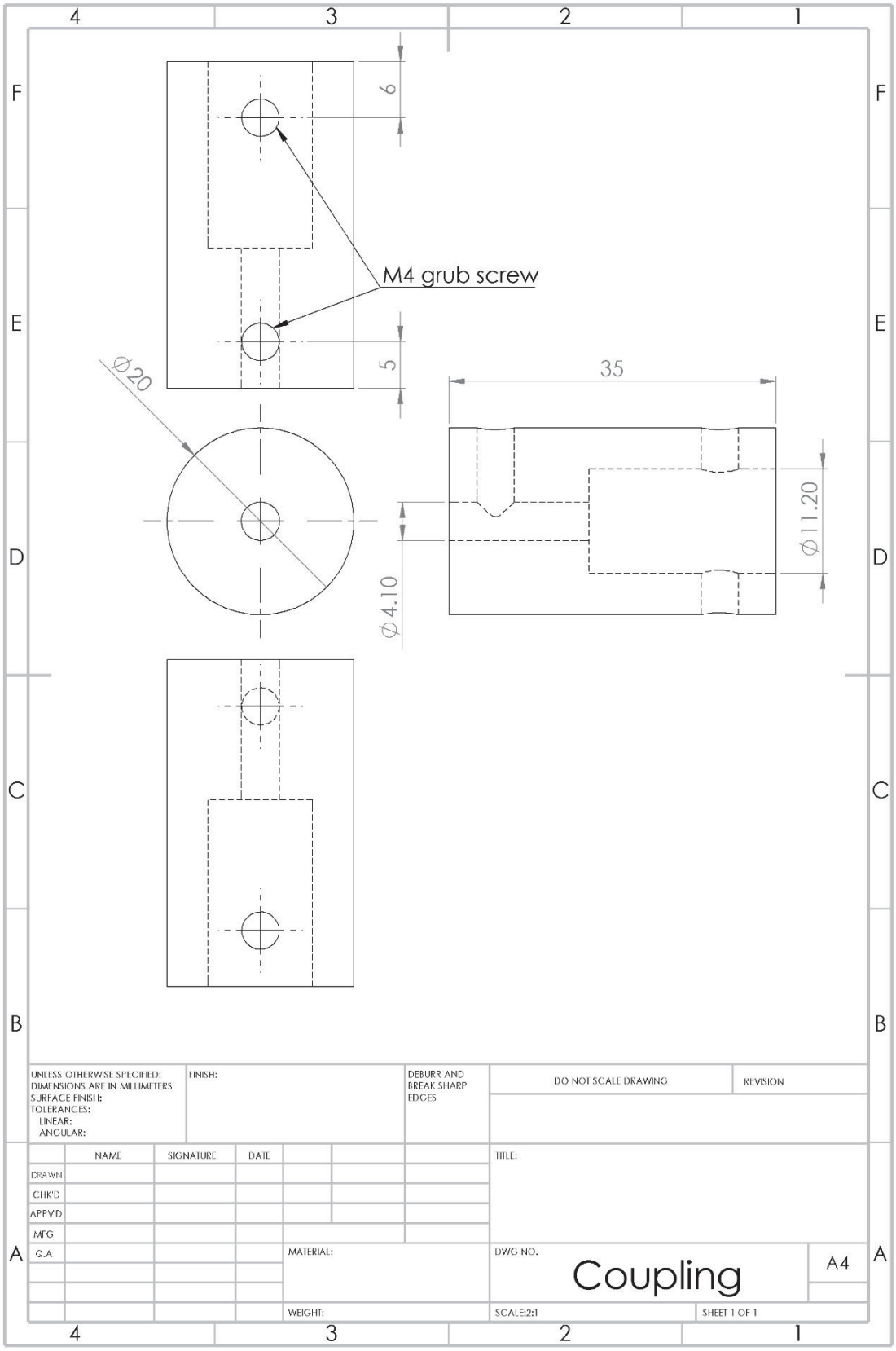
Wilson, M. (2014). *Implementation of Robot Systems: An introduction to robotics, automation, and successful systems integration in manufacturing*: Butterworth-Heinemann.

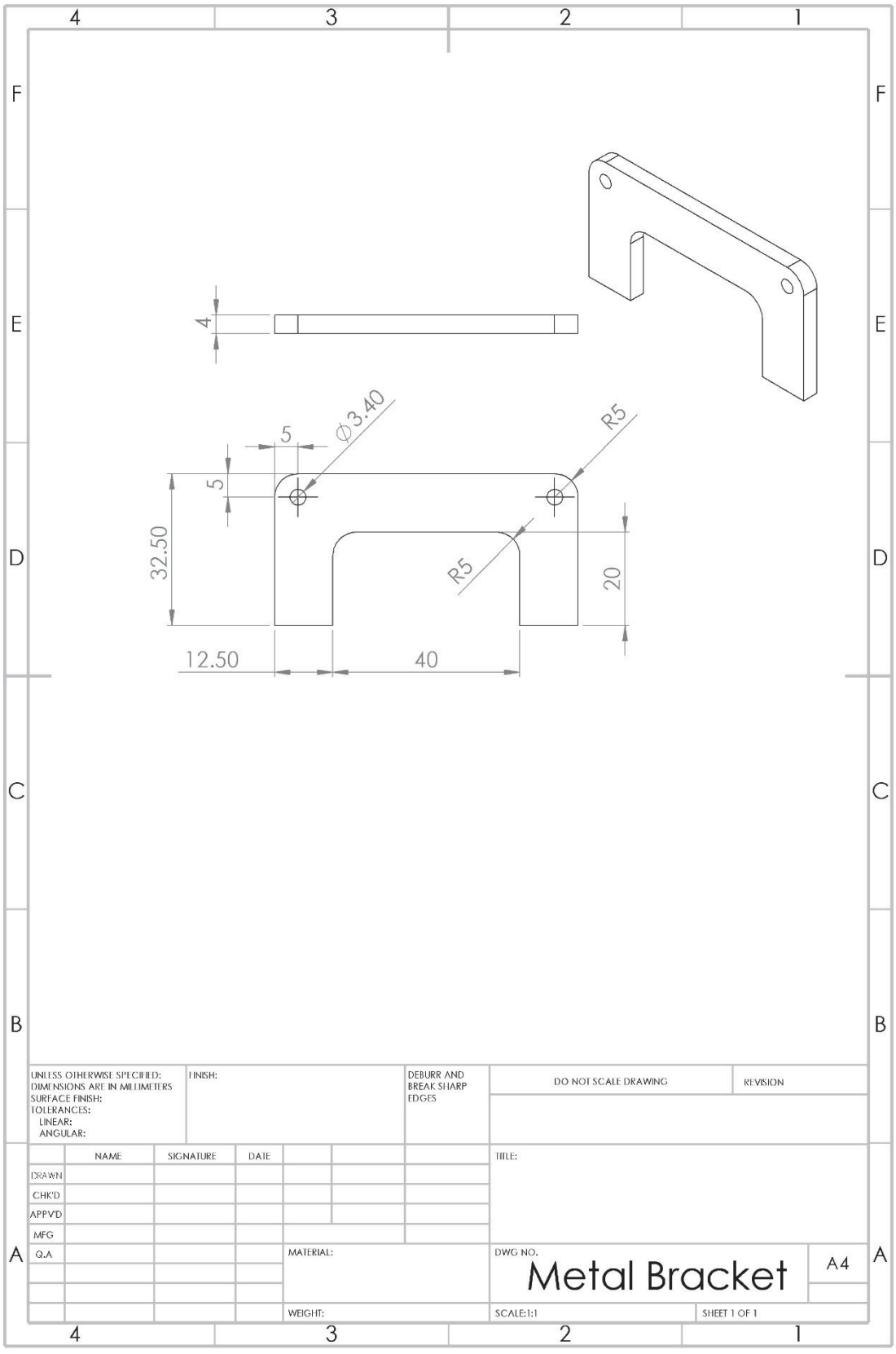
Appendix A – Design drawings

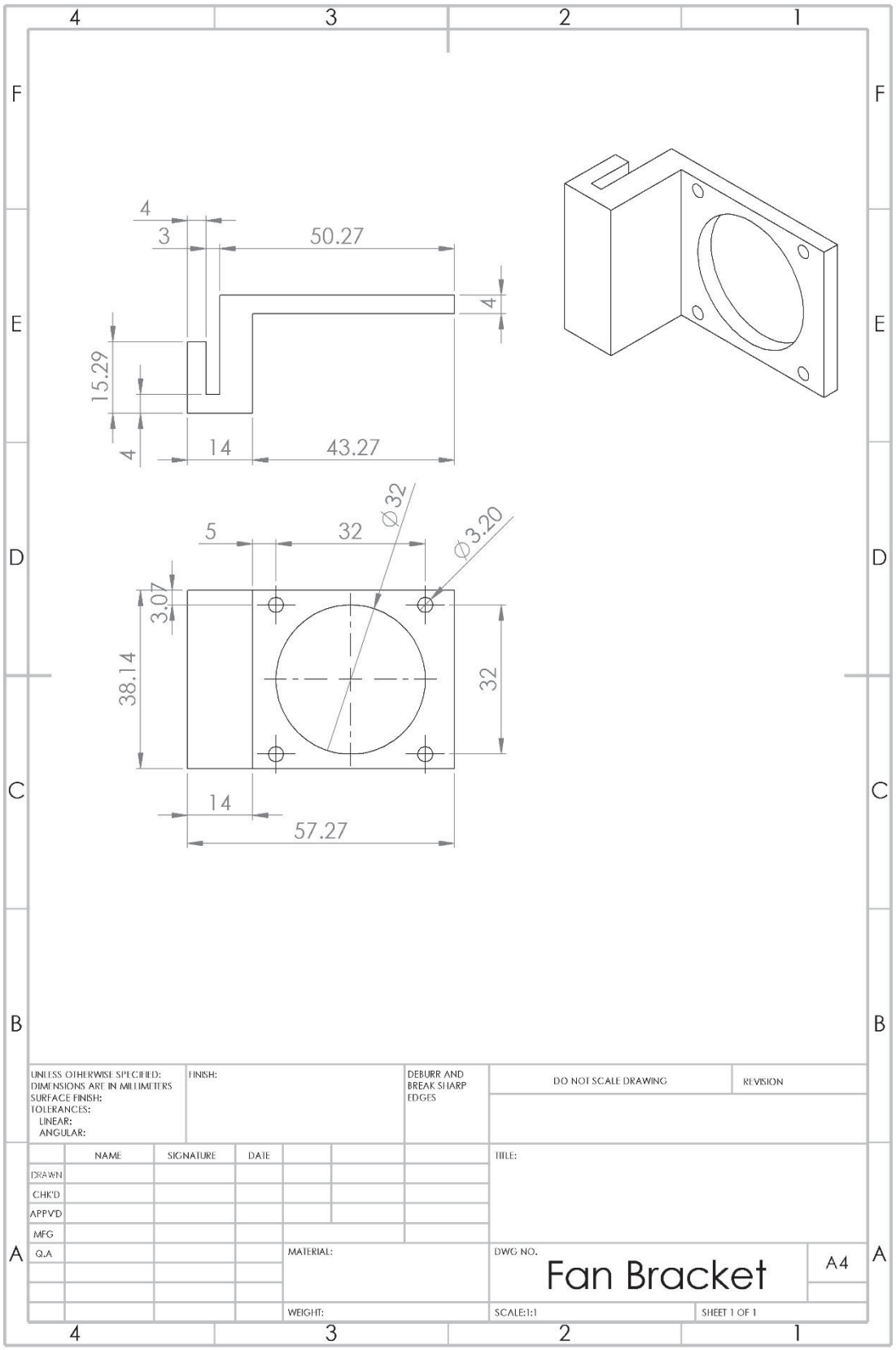



R32.50
Ø53.94
Ø53.94
Ø29.26
Ø32
Ø35
Ø15
Ø23.97
5.32
8.23
R4
Ø3.40
DETAIL A
SCALE 1 : 1
Ø70
114
R10
R5
20
2.50
6
Ø16.50
Ø19.50
30
65
2.50
16.18
Ø15
DETAIL B
SCALE 1 : 1
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
MATERIAL:
WEIGHT:
TITLE:
DWG NO.
Hopper
A4
SCALE:1:2
SHEET 1 OF 1

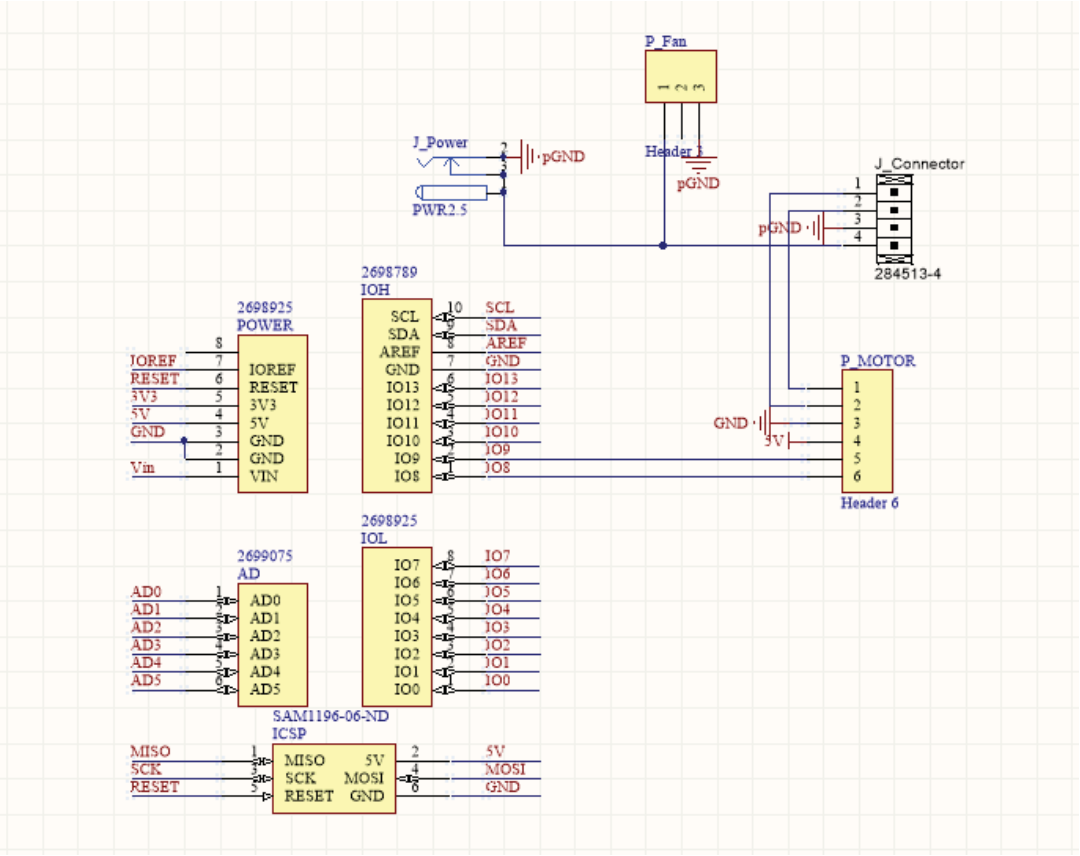

4
3
2
1
F
E
D
C
B
A
3
2
R2
R31.50
R32.50
Ø3.40
8
9
Ø7
Ø70
15
33
24.50
R5
R4
Ø3.40
25.98
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
TITLE:
MATERIAL:
DWG NO.
Lid
A4
WEIGHT:
SCALE:1:1
SHEET 1 OF 1

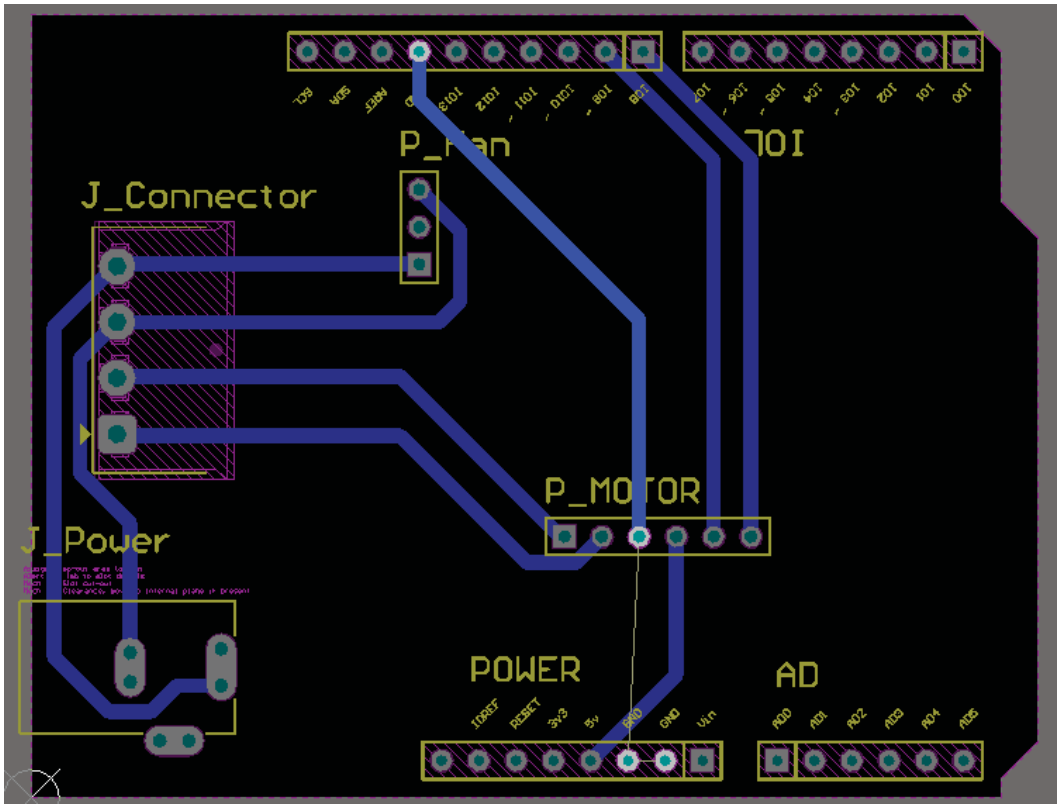

6
M4 grub screw
5
35
Ø20
Ø11.20
Ø4.10
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
TITLE:
DRAWN
CHK'D
APPV'D
MFG
Q.A
MATERIAL:
DWG NO.
Coupling
A4
WEIGHT:
SCALE:2:1
SHEET 1 OF 1


4
5
Ø3.40
R5
5
32.50
R5
20
12.50
40
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
MATERIAL:
TITLE:
DWG NO.
Metal Bracket
A4
WEIGHT:
SCALE:1:1
SHEET 1 OF 1


4
3
50.27
4
15.29
4
14
43.27
Ø32
Ø3.20
5
32
3.07
38.14
32
14
57.27
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
MATERIAL:
WEIGHT:
TITLE:
DWG NO.
Fan Bracket
A4
SCALE:1:1
SHEET 1 OF 1

Appendix B – PCB drawings





Appendix C – Journal Article

Robot Assisted 3D Printing of Biopolymer Thin Shells

Byron James Brooks · Khalid Mahmood Arif · Steven Dirven · Johan Potgieter

Received: 31/03/2016 / Accepted: date

Abstract The design, development, and testing of a robot-assisted biopolymer thin shell free-form printing system is presented. This fused-deposition style printing system directly extrudes pellets of biomaterial and is capable of printing directly on organically-shaped 3D curved surfaces. The screw extrusion method allows direct printing from pellets. The printed structure is supported by a pre-built base (a mandrel), which is manipulated by a six degree-of-freedom industrial robot arm, an ABB IRB120. This robot is used to manipulate the orientation of the support mandrel surface. The print method works by projecting a desired 2D image onto a mathematical model of the pre-built mandrel surface. This produces a 3D point path for the system to follow. These points are then converted into vectors for the robots pose and orientation of the end effector, which ensures that the extrusion remains normal to the mandrel surface. Inverse kinematics is applied to convert the trajectory into joint positions for the robot to follow. This paper demonstrates the utility of the developed system through simulation and printing of concave surface designs.

Keywords Biopolymer 3D printing · Thin shell printing · Pellet extrusion · Articulated robot · Fused deposition modeling

K. M. Arif
Tel.: +64 9 414 0800
E-mail: k.arif@massey.ac.nz

Center for Additive Manufacturng, School of Engineering and Advanced Technology, Massey University, Auckland 0632, New Zealand.

1 Introduction

Additive manufacturing (AM), or 3D printing, has been around since the 1980s; since then, different materials and methods have been developed to exploit mechanical characteristics, and layer adhesion. Whilst there are a range of processes available [1], each of these processes have their benefits and drawbacks. In general, they typically work by printing a layer of material and then building upon the previous one in a 2.5-D build strategy. The building process, particularly for Fused Deposition Modeling (FDM) can produce any geometrical shape, but it is not efficient when building thin, tall, convex, or webbed features. This motivates the requirement for a printing process that can print directly in 3 dimensions.

While the general aspects of FDM printing have been heavily developed, there has been little development on printing thin-shelled parts. The most widely used technique, in this regard, uses a scaffold beneath the part, which is removed afterwards, leaving curved print rather than staggered layers [2,3]. The scaffold is particularly necessary for polymer processes due to their inability to retain printed shape instantly in the free space and for the process to finish in a reasonable time [4]. This is in contrast to metal extrusion by weld type processes [5], which can build parts without any support. However, the major limitation of using current cartesian print systems with their orthogonal axes is the inability to tilt, or orient, the scaffold normal to the 3D scaffold surface. This limits the maximum gradient and height of the surface, as the print head should not contact previously printed surfaces. To overcome this limitation this paper proposes the use of an industrial robot arm (ABB IRB120) to reorient the support-

ing scaffold structure and conduct all motion of the 3D printing process.

A number of industrial robot assisted AM processes have been presented in the literature [6–8]. For example, Zhang *et al.* [6] has recently reviewed the use of industrial robots in AM. To use a robot in conjunction with a 3D printing process, it must be programmed to undertake surface-scanning tasks. However, the data normally used in 3D printers is not useful for this purpose and it must be converted to other suitable formats. This topic has been treated quite extensively in literature; for example [9] used an adaptive compensation algorithm for printing letters on a curved B-spline surface, [10] utilized point cloud slicing for a spray painting robot, and [11] split CAD models into distinct parts for cooperative tool path planning of curved AM.

Articulated robots (or serial robots) are used in many different applications; they are very versatile as they typically have a large dextrous workspace. Their movement closely resembles that of a human arm, though they are typically designed with 6 degrees-of-freedom (DOF) [12]. The dextrous capability facilitates complex print profiles. This is in contrast with alternative robotic configurations that have been used in AM technologies, such as a cable-suspended Delta Robot to print large foam structures [7] and a SCARA robot that was used in [13] to develop a laminated object manufacturing system.

Although industrial robots can be used in any AM process, extrusion based printing methods, such as FDM, are more suitable because the extruder operates in a synonymous, vector-based, manner that is common among other machining tools (e.g. end mill [14,15], weld gun [16], etc.). Biopolymer materials such as PLA with a composite fiber can be thermoplastically extruded through a die [17]. The pellet-based method facilitates printing with any thermoplastic material, so long as it can be continuously melted and passed through such an orifice [18].

In addition to the printing method, new materials such as carbon nano-fibre reinforced filament [19] and biopolymers [20] have been considered as future print materials. Both filaments and pellets can be used in extrusion processes; however, pellets offer a number of advantages. For example, new materials can be quickly tested by purging the extruder and filling it with the new pellets, or additives can easily be added to the polymer to change its properties. Moreover, there will be reduced chances of having degradation in the material as it will go through less thermal processing compared to filament methods. Prior work by Reddy *et al.* [21] and Valkenaers *et al.* [22] both used custom designed pellet extruders to replace filament extruders on current 3D printers. However, this architecture is not currently commercially available.

Fig. 1 Blended pellets of biopolyesters containing approx. 5 wt.% of Phormium tenax fibre. Smaller units on the scale = 0.5 mm.

In this paper, we present the design and development of a robot-assisted system for thin shell printing on a complex 3D surface. A mandrel is used as a supporting surface, which the print head extrudes the thermoplastic polymer on to. The material is presented to the machine in a pellet form, which is a proprietary mixture of biopolyesters and fibers from the native New Zealand flax plant (Phormium tenax). The supporting mandrel is fixed to the end-effector of an articulated industrial robotic arm to print on its predefined surface.

2 System Design and Implementation

2.1 Hopper and Extruder

Extruder design is fundamental to any FDM process as the continuity and integrity of the material bonding depends on the quality of the extruded material exiting the nozzle. While filament extruders are widely used in the FDM process, there are few pellet-based extruders commercially available for 3D printing purposes. Pellet extruders (also called direct extruders) , are generally found on modified versions of existing printers [23]. The advantages of using pellets include: the possibility to change the formulation online; fewer thermal processing cycles; and less contamination of base materials (less processing). However, the large advantages of filament extrusion such as achieving tight deposition tolerances is lost. Changing the ratio gives different physical and aesthetic properties to the extrusion. The pellets are 3 mm in diameter, and are 1 - 3 mm long. The material feedstock is obtained by blending a masterbatch of

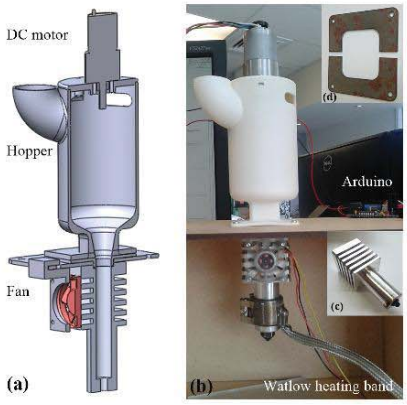


Fig. 2 Pellet extruder with hopper. (a) Cross-section of the CAD model, (b) Assembled extruder with hopper on testing rig with heating band around extruder barrel.(c) Close-up of hot end (d) Fan bracket, a 2 mm laser cut steel plate.

flax pellets (brown pellets) with a blend of co-polyesters (white pellets) (Fig. 1).

In order to preserve the stability of pellet extrusion, the printing system was developed with a stationary extruder, and a mobile support mandrel. The extrusion screw forces pellets vertically downward through the heating band (Fig. 2b). A heatsink and fan ensure that pellets in the hopper do not agglomerate and the heating band (Watlow) is closed-loop controlled with an EZ-ZONE PM. The machine is operated in an air-conditioned laboratory to ensure that performance fluctuations do not occur due to changes in the ambient environment such as temperature and humidity.

2.2 Articulated Platform

Due to the requirement of printing directly in 3 dimensions, traditional planar XY print deposition systems cannot be used for the current application. Therefore, a six degree of freedom articulated industrial robot (IRB 120, ABB) with a 3 kg payload capacity is used in this work. The robot has a repeatability of 10-μm, which facilitates printing with fine tolerances.

Conventional FDM technologies use a flat platform to build parts, which requires wasteful support material if the intended design requires large convex surface profiles such as shell-like objects. In order to overcome this challenge, a mandrel is used as an exemplary 3 dimensional support surface. The surface can be updated depending on the application.

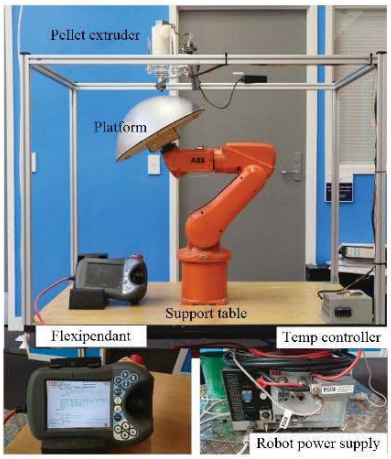


Fig. 3 Robot with platform (or mandrel) attached. The lower side show robot controller, and ABB's flexipendant.

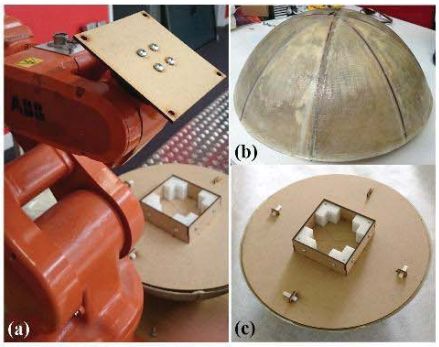


Fig. 4 (a) Robot attachment bracket for platform (b), Fiberglass platform (before painting), and (c) Base of the platform.

An exemplary concave platform (Fig. 3b) was developed to demonstrate the mandrel printing principle. It was manufactured from fibreglass to ensure that it is light weight, quick to manufacture, does not burn or deform under heat, is strong, and keeps its shape over time. This mandrel (base) is fixed to the robot's end effector using an attachment bracket (Fig. 4).

3 Print Data Format Conversion

Due to the unique nature of the proposed printing process, existing data formats such as STL, or its variants,

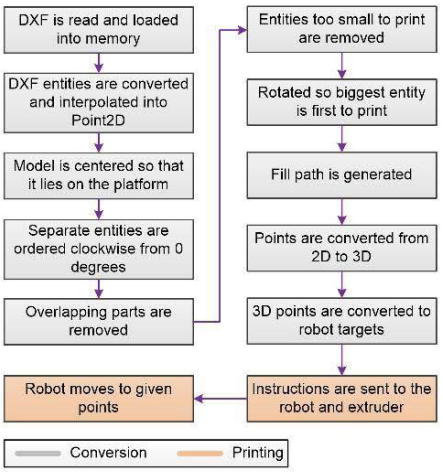


Fig. 5 Flow diagram of the proposed robot-assisted 3D printing process.

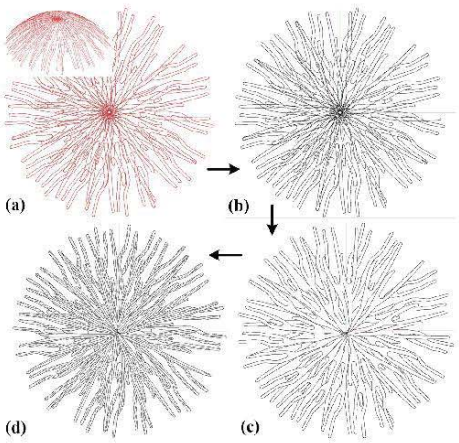


Fig. 6 Conversion of the file format to 3 Dimensional Mandrel (platform) contour. The inset in (a) shows 3D view of the original design. Overlapping lines (b) are removed (c) and shape filled (d) for scanning.

cannot be used directly. The data is formatted into a robot trajectory format by the process shown in Fig. 5.

There are two stages in this process: the model conversion stage, and the printing stage. The model conversion stage takes the users file and turns it into targets for the robot. The printing stage uses these targets to move the robot along the trajectory while controlling the extruder to print the design. The first step is to convert the raw design from planar DXF entities into an array of 2D points. Next, the points are centered and scaled to fit on the support structure. After this, the individual entities are rearranged so they start from a polar angle of 0 for the overlap removal process. Entities that are too small to print are discarded. The fill paths for each entity are then generated and all the contour and fill points are projected onto the platform to describe their position in 3 dimensions. Lastly, the 3D points are converted into targets for the robot. During the printing stage the extruder speed is manipulated to achieve the correct flow while the robot is streamed trajectory commands to achieve target position and orientation for the end-effector.

Whilst the technique is general; we demonstrate the principles through an example of lampshade designs that have been commissioned by an artist (Fig. 6a). The designs are converted to 3D points that conform to the surface of the exemplary domed mandrel (Fig. 6b).

The robot uses a world coordinate system, making it use absolute positions, rather than relative ones (Fig. 7). The coordinate system used in the platform and the robot have their axes aligned when in their home position. The coordinate system used for the platform (Fig. 7a) and base (Fig. 7b) are only translated in this position.

3.1 Contour Point Generation

Currently the software handles the most common types of DXF entities; Arcs, Lines, Splines, Polylines and Circles. All of these entities represent open contours. For each entity, the step size remains the same, so the resolution of the points remains constant. The step size is set by the user through the user interface. Once converted, each set of points is stored in its own list to create a path.

After all the DXF entities have been converted into a list of point2D objects, the paths are merged together. This is in order to create closed paths. To merge the entities, the start and endpoint are analysed against the start and endpoints of the other paths. If they are within a distance of 0.005 mm, it is assumed the points are meant to be merged. This tolerance overcomes any discrepancy in the interpolation algorithms, as they can give slightly different endpoint values for different types of DXF entities.

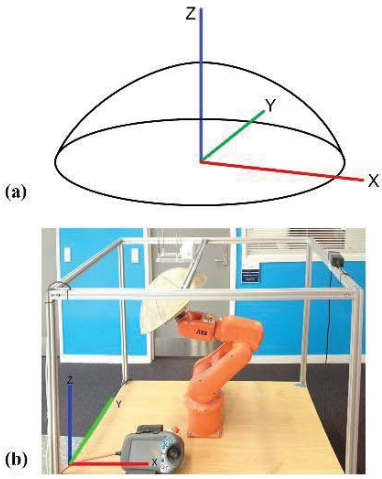


Fig. 7 (a) Platform coordinate system, and (b) Robot coordinate system.

The next step is to centre and scale the design if the user has selected to do so. This ensures the correct placement of the design on the platform. The design is centered by taking a bounding box of all the points, calculating the centre of the bounding box and moving each point of the model using Cartesian geometry so the centre of the bounding box lies on the origin of the mandrel surface.

It was considered to use a grid-based system to snap the points to. However, it was found that the path became too staggered and would not produce a good quality print. These 2D points are interpolated at a user-specified resolution. Currently these points are being interpolated at a resolution of 0.2 mm.

Four path clipping operations are used to generate the path: difference, intersection, exclusive-or, and union. To remove the overlapping sections, each entity is first converted into a polygon. The first entity is selected to be the base polygon. For all the rest of the entities, their polygons are compared with the base using a difference operation. The resulting polygon is saved and then added to the base with a union operation. This is repeated on all the entities. The original base polygon, along with all the saved difference results, gives the final polygons with no overlapping areas. These are converted back into Point2D entities. The resulting design is shown in Fig. 6c.

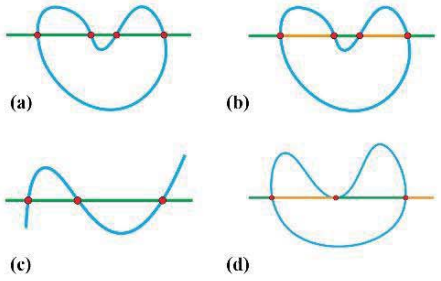


Fig. 8 (a) Points of intersection along a scanline, (b) Fill line within polygon between start and stop points , (c) Points of intersection along scanline and open contour , and (d) Fill line with scanline intersecting polygon on a tangent.

3.2 Fill Point Generation

When the contour is printed, there are some gaps that require filling. A parallel-line hatch was used to fill these voids, which can be oriented in an angular manner from 0 to 90 degrees.

The points that make up the path for the robot to fill the entity are generated through a scanline algorithm [24]. In order to find the fill geometry, a scanline is projected across the mandrel surface. Where this line exists within an entity the fill will follow its trajectory. Filling involves using multiple parallel scan lines with equidistant horizontal resolution. This process is repeated until it reaches the end of the entity. This process is conducted for each entity in the design.

The scanline algorithm is as follows: A bounding box of the entity is taken to find the leftmost and rightmost point. This defines the first and last X values for the scanlines. The rest of the X values are interpolated from these points at a set resolution, which is determined by the user. For each scanline, the Y endpoints are determined by taking the value at which the scanline intersects the contour.

When interpolating these points, it is assumed they all have an equal number of start and end points. While traveling along the scanline and counting the intersecting points, odd points represent an entry to the entity, and even points represent exiting the entity. The process is represented in Figure 6c. The scanline technique is demonstrated in Fig 8, and the resulting fill lines in Fig 6d.

It is possible for the scanline to output more than one start and endpoint as seen in Fig. 8. When this is the case, each start point should have a matching endpoint to create a fill line as seen in Fig. 8b. When there is an odd number of points, either from an open contour

as seen in Fig. 8c, or the scanline being perfectly tangent to the contour, the interpolation (8d) won't work properly as it requires two points. If the missing point is from the middle of the scanline, the interpolation will be done between the end point of one section and the start point of the next, interpolating outside of the polygon rather than inside. This is illustrated in Fig. 8d. To find the end points, the algorithm assumes that the contour is closed so there will always be an even number of endpoints.

Some points are removed if they are too close to the contour path. This ensures that there is not too much overlap and so the material does not build up too much when printing. Fill points are removed if less than a threshold (currently 0.8 mm) from an external contour surface.

Once all of the endpoints have been interpolated, the entities contour points and fill points are rotated back to their original position. This is the final stage of the point creation and gives the design shown in Fig. 6d.

3.3 2D to 3D Projection

There are multiple ways to map 2D points onto a 3D surface. Many map projection algorithms take a flat surface and wrap it around a mandrel shape by skewing the image. These algorithms could be modified to fit the platform. However, with most of these, there may be unpredictable distortion in the design. The current projection algorithm stretches the design only in the Z dimension (eg. A vertical projection).

The exemplary domed platform is modeled by rotating a polynomial (Fig. 9a) about the Z-axis. The X and Y value of the points are fixed, while the Z value is determined by solving the magnitude of the polynomial in a polar manner. This results in a list of 3D point targets, which are expressed in both Cartesian and Polar coordinates.

3.4 Printing Trajectory

Each entity is searched to find a point that has been previously printed. The entity is rearranged so this point becomes the start point. This lets the print start each entity by attaching it to a previous one, which gives the material something to easily bond to. If there is no point in the current entity that touches another that is already printed, it starts at the point closest to the origin of the mandrel surface.

All of the contours are printed before filling them in. The fill points have an additional offset of 0.2 mm from the mandrel surface to ensure that the print head does not contact with previously printed contours.

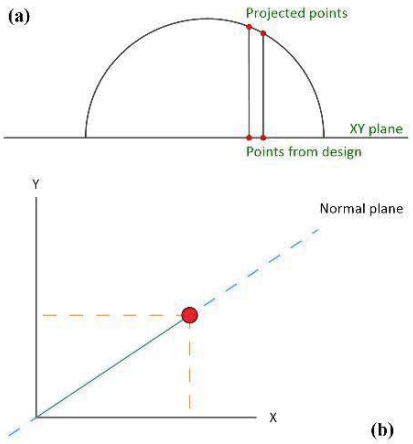


Fig. 9 (a) Side view of vertical projection method, and (b) View of point normal to the XY plane.

3.5 Point3D to robtarget

Once the 3D points that make up the design have been generated, the coordinate and orientation of the robot's end effector must be determined so the plane that lies tangential to the point on the surface of the platform is normal to the extruder and a set distance from the end of the nozzle.

This process utilises linear transformations, specifically translation and rotation. The design can be thought of as an infinite number of 2D planes, normal to the XY plane that holds points on them rotated around the Z-axis of the origin point.

3.5.1 End-Effector Coordinate

The target point is translated to the top of the dome, to point (0, 0, 150). This top of the dome becomes the point in contact with the nozzle. This is done by taking the projection of the points spherical radius and projecting it onto the X axis. This gives the amount to move the point by in the X direction. The Z amount is determined by taking the Z value of the point at the top of the platform (150) minus the Z value of the point.

The end effector is stored as a vector of the point (0, 0, -40). This is because it is 40 mm below the origin of the base of the platform, as shown in Fig. 10a. The

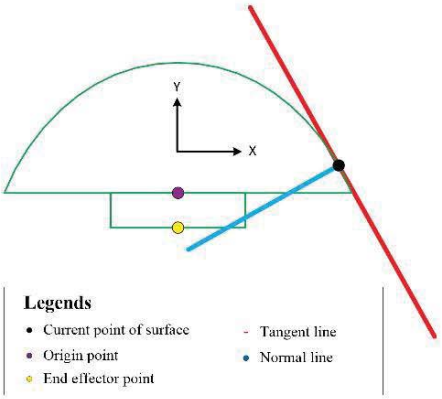


Fig. 10 Starting point of diagram used for calculation and figure legends.

end effector is translated and rotated as from Fig. 10a through Fig. 10b to 10c.

4 Print Communication and Implementation

To communicate with the robot, ABB's proprietary PC SDK is used to interface with the controller. The Point3Ds are sent to be encoded to robtargets along with the speed to move between them. The Point3Ds are encoded on the fly as they are sent over ethernet, rather than all at once. This is due to two reasons. Firstly, the robot is unable to store all the points as RAPID code. For the example design discussed above, there are 66,661 contour points and 27,758 fill points. To store each of this as a single file requires over 180,000 lines of code. Secondly, it allows the distance from the nozzle to be modified during runtime. By converting each point individually before sending it, if there are any changes in the platform height, this is updated before sending the data.

5 Results and Discussion

5.1 Pellet Extruder

The extruder was tested for quality and aesthetics properties, as well as its ability to print material that can join onto previously printed material. The quality, aesthetics, and flow consistency of the extrusion is important because these define the look of the print. As the object of the printer is for print path generation and

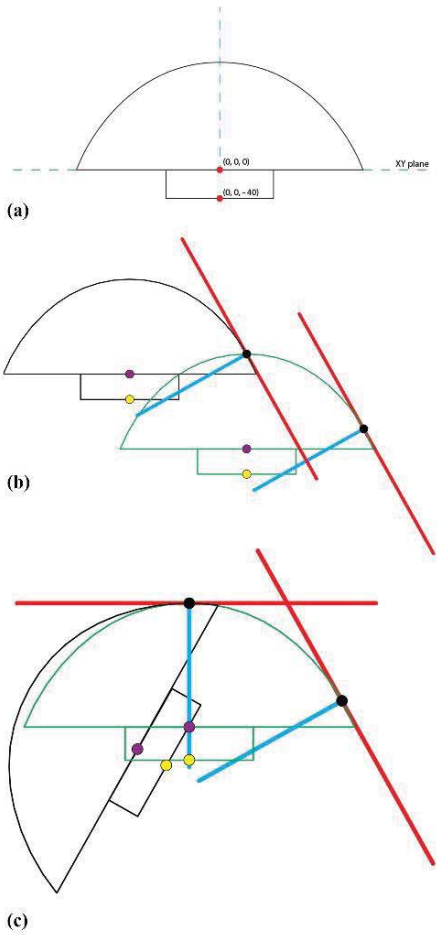


Fig. 11 (a) Robot end-effector vector in relation to origin, (b) Black diagram is transposed to (0,0,150), and (c) Black diagram has been rotated around (0,0,150).

implementation for an artistic application, a heavy emphasis has been placed on the aesthetics and the quality of the extrusion.

As the print is not one continuous path, the material must be able to form a bond with previously deposited material (Fig. 12). If the paths aren't able to join with enough strength, when touched or moved the design would have the potential to break. In order to test the quality of this bond, small demonstration prints were

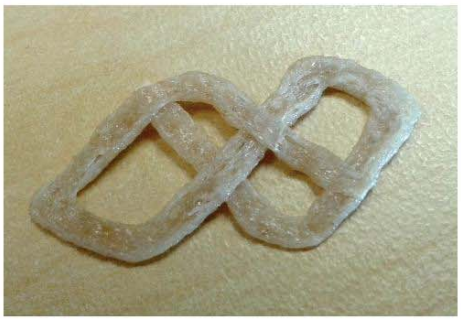


Fig. 12 Bonding test for overlapping material.

initially conducted to demonstrate the bonding capability (Fig. 12).

5.2 Software Algorithms

Before running the printing software, the algorithms are tested using simulation methods to ensure they operate correctly. The algorithms used to convert the DXF format to 3D points, were tested and verified by converting the points back into DXF form and viewing them in a DXF viewer.

The algorithms to transform Point3D objects to robtargets were simulated in RobotStudio. The robtargets for an entity were generated and exported into a text document. To test this algorithm, rectangle, circle, and hexagon designs were used because the robot is only able to store a certain number of targets, and using the intended final design would generate too many.

From the text document, the robtargets were copied into RobotStudio so they could be simulated. This simulation is used to check the robot will move in the correct path, and not made any unexpected movements that will cause collisions. It is also used to check that no unexpected errors, such as joints out of range occur.

Fig. 13a shows the simulation of the contour path of a single entity. It shows the end effector moves along the Y-Z plane while rotating axis 6 around the Z-axis. When the simulation is run, no errors occur.

As it is not possible to see the output of the printer on the platform during simulation, a pencil was attached to the frame in the position of the extruder to simulate the path that would be printed (see Fig. 13b). As this was to get the path planning correct, the pencil tip represented a reference point in 3D space.

As there are two rotation matrices used together to transform the point3D object to a robtarget, this method was used with different designs to test each

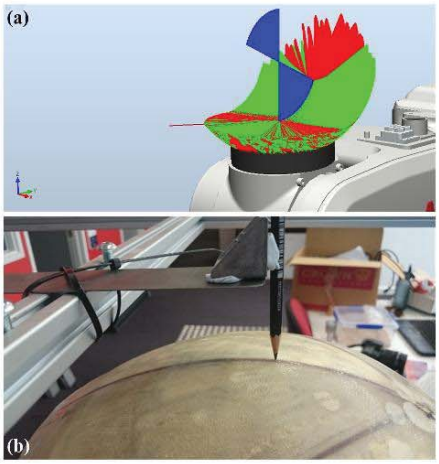


Fig. 13 (a) Simulation of entities in RobotStudio, (b) Simulation using a pencil.

of them. A circle model was used to check if there was any problem in transformation around the Z-axis. If the circle was drawn with the correct radius and no distortion, the transformation around the Z-axis was correct. Square and hexagon models were used to confirm correct transforms around the X-axis.

5.3 Preliminary Printing Results

It is challenging to begin printing an entity, as this is when adhesion between the extruded material and the mandrel is weakest. The further away the entity starts from the origin of the mandrel the more challenging this is to address. This was overcome by starting each entity from an intersection with a previously printed entity, or nearest the mandrel origin.

Another challenge that needed to be addressed was that the pressure in the extruder caused it to continue extruding when the auger was stationary. This is a common occurrence in most FDM 3D printers, even with filament. However, it is faster to remove this pressure in a filament extruder by extruding the filament slightly. Until this was addressed in the current print method, it would leave tags, like the ones shown in Fig.14a, on the design that would need to be removed by hand later.

Multiple methods were implemented to solve this. Firstly, it was attempted to stop the extruder before the entity had finished so the backpressure was used to print the rest of the entity. This method was rather

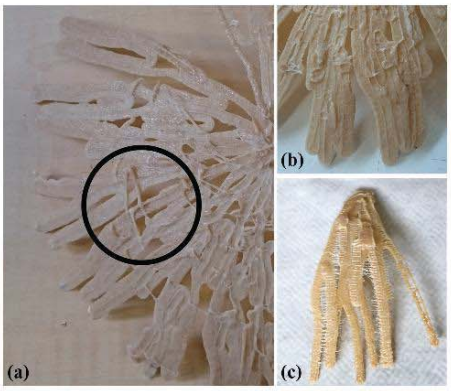


Fig. 14 (a) Artefacts from 3D printing in black circle, (b) Magnified view of artefacts, and (c) Horizontal fill path trial.

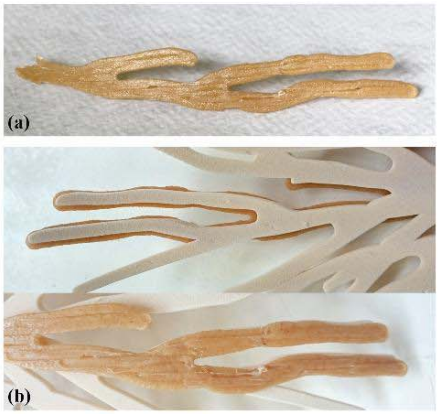


Fig. 15 (a) Extrusion quality of 4 rpm motor, (b) Comparison of prints against reference design. The reference design was printed on an SLS machine.

unreliable as the pressure in the extruder could change from each print. Next, instead of stopping the extruder, reversing it was tried. This improved the backpressure drop, but took a while. The nozzle was kept on the design while it was reversing, which worked to create smaller tags, but left artefacts, such as the ones shown in Fig. 14b on the design.

When filling the entities in a horizontal fashion there are many start and stop points. At each of these, a tag is generated. To reduce the tags the robot is programmed to move very quickly between each fill line. This does reduce the tags but leaves very small strands between each line that can be removed after printing. These strands can be seen in Fig. 14c. The extruder was run at a lower speed when filling, which led to less aerated material, and therefore a more narrow print width.

To reduce this backpressure in the extruder, the extruder can be stopped or reversed until the flow stops. This still leaves some material being extruded which must be removed manually. The width of the path is currently 3.2 mm. When comparing the proposed printing method with the reference design it is clear that the swelling slightly oversizes parts (Fig. 15b). This swelling effect is due to aeration that expands the material as it leaves the die. In order to alleviate this issue, the rotational speed of the extrusion screw was reduced to 4 rpm.

Smaller nozzle diameters of 0.8 mm and 1.2 mm were tried. The 0.8 mm diameter nozzle had problems with the fibers fitting through it. The fibers fit through the 1.2 mm nozzle. However, it requires more pressure to extrude the melt. A higher rpm (e.g. 11 rpm) motor does not provide enough torque to deliver this pressure.

With 4 rpm motor, the path width is narrower, averaging 3.2 mm, and making the print of the design much more accurate in comparison to the reference design. As Fig. 15a shows, when the extruder is run at 4 rpm with the temperature controller set at 220 °C, the quality of the print gets better. There are little to no tags, bulges, or artefacts from the filling.

However, because the flax fibers stay in the barrel for longer, even with the lower temperature, it is extruded as a deeper color.

5.4 Full Design Printing

The printer and software created to control it worked to create the prints shown in Figs. 15 and 16. The design shown in Fig. 16 was printed using a geared motor turning at 2.5 rpm and a temperature of 220 °C. Under these conditions, the printer delivered multiple full designs on the platform.

For the initial development of a printer, the parts have a good aesthetic appearance. The extrusion melt is well mixed and has fibers evenly dispersed. This is verified by the constant coloring of the material throughout the print as different ratios or mixing would lead to changes in the color. The fibers can be seen, with very few bubbles, and a shiny surface finish.

It is important to get the contours correct because they give the overall shape to the design. As seen in

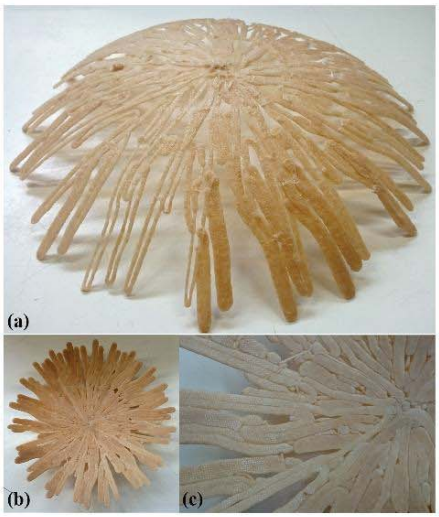


Fig. 16 Different views of the 3D printed design. (a) Isometric view showing stability of the structure ,(b) Underside showing conformity to the base platform, and (c) Magnified view of the underside of the object.

Fig. 16a, the contours of the design follow the shape of the original design, although thicker at certain regions, follows the same shape and path. The curvature of the print is very accurate. This is because the design is printed on a platform with the correct curvature already. There is no deformation after the design has been removed from the platform. Furthermore, the structure is strong and mechanically stable under its own load (Fig. 16a and b). A close-up of the underside shows smooth printing profile that resulted from the shape of the mandrel.

Fig. 17 shows examples of thinner profiles - a spiderweb and a mesh design - which were printed by altering the print nozzle and screw speed. The nozzle diameter of 3.2 mm and extrusion auger speed of 4 rpm allowed printing of these parts. Apart from some variability in the extruded profiles, the structures show excellent mechanical strength and aesthetic feel.

Overall the developed 3D printing system has demonstrated very encouraging results and opened up a possibility for further investigation robot-assisted printing of biopolymers directly from pellets.

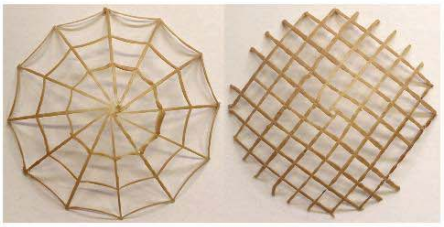


Fig. 17 A spiderweb and a mesh design printed with the developed system.

6 Conclusion

We have presented the design and development of a robot-assisted biopolymer printing technique for thin shell like objects. The system has been tested with printing experiments and very encouraging results. It is found that printing open paths could lead to more intricate detail being kept, by deprecating the need for a return path.

The software accurately converts the design from a DXF format to points on the platform. This is evident by turning the projected points back into DXF format and comparing the results. The results show no difference in the shape of the design. The matching shape of the print against the reference design is again, testament to this.

The mapped 3D points are successfully converted to targets for the robot to move to. This conversion is correct as the nozzle stays a set distance from the platform while moving. The robot is able to adjust its speed and position based on the commands sent. This is verified by the accuracy of the final printed design.

The presented system is only first prototype and many improvements can be made to enhance the quality of the print. For instance, having multiple stages in the extruder would improve the melt by allowing the process to be more controlled.

The software only handles DXF format files. Not every type of DXF entity is supported. However, many such as the dimension, hatch, tolerance, etc., will not be used. If the input file format stays as DXF rather than changing to STL or another, more standard, 3D format, the required entities can be added easily to the software.

The printer will only deposit a single layer. To print more layers, the spherical radius of each Point3D object can be increased. The value of this is dependent on the height of each layer. If a layer with a different shape from the previous one were to be printed, it would require going through the whole conversion and projec-

tion process again. If support material is required, an algorithm to determine where, and how, to print the support is required.

Acknowledgments

This research was part of the Extrusion Plus program led by Scion and Dr. Dawn Smith, and funded by the Ministry of Business, Innovation, and Employment funding under High Value Manufacturing and Services (HVMS) Enabling Technologies investment contract. The authors thank Dr. Marie-Joo Le Guen for providing the materials and related expertise. Special thanks are also due to Mr. David Trubridge, a New Zealand based artist, for providing the artistic design shown in Fig. 6.

References

1. H. Bikas, P. Stavropoulos, G. Chryssolouris, The International Journal of Advanced Manufacturing Technology pp. 1–17 (2015)
2. O. Diegel, S. Singamneni, B. Huang, I. Gibson, in *Advanced Materials Research*, vol. 199 (Trans Tech Publ, 2011), vol. 199, pp. 1984–1987
3. O. Diegel, S. Singamneni, B. Huang, I. Gibson, in *Key Engineering Materials*, vol. 467 (Trans Tech Publ, 2011), vol. 467, pp. 662–667
4. S.H. Ahn, M. Montero, D. Odell, S. Roundy, P.K. Wright, Rapid Prototyping Journal **8**(4), 248 (2002)
5. I.D. Harris, A. Director, DOT International, New Orleans (2011)
6. G.Q. Zhang, X. Li, R. Boca, J. Newkirk, B. Zhang, T.A. Fuhlbrigge, H.K. Feng, N.J. Hunt, in *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of* (VDE, 2014), pp. 1–6
7. E. Barnett, C. Gosselin, Additive Manufacturing **7**, 27 (2015)
8. K.M.M. Tam, J.R. Coleman, N.W. Fine, C.T. Mueller, in *Robotic Fabrication in Architecture, Art and Design 2016* (Springer, 2016), pp. 350–361
9. Z. Wang, R. Liu, T. Sparks, F. Liou, International Journal of Computer Applications in Technology **53**(2), 183 (2016)
10. G. Wang, J. Cheng, R. Li, K. Chen, in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2015), pp. 1717–1722
11. C. Kim, D. Espalin, A. Cuaron, M.A. Perez, M. Lee, E. MacDonald, R.B. Wicker, Journal of Mechanisms and Robotics **7**(2), 021003 (2015)
12. M. Wilson, *Implementation of Robot Systems: An introduction to robotics, automation, and successful systems integration in manufacturing* (Butterworth-Heinemann, 2014)
13. K.M. Arif, Z. Aftab, M.T. Najam, in *International Conference on Manufacturing Automation (ICMA'2007)* (2007), pp. 616–625
14. Y. Hu, Y. Chen, The International Journal of Advanced Manufacturing Technology **15**(9), 624 (1999)
15. Y. Hu, Y. Chen, The International Journal of Advanced Manufacturing Technology **15**(9), 630 (1999)
16. Y.M. Zhang, P. Li, Y. Chen, A.T. Male, Mechatronics **12**(1), 37 (2002)
17. B.N. Turner, S.A. Gold, Rapid Prototyping Journal **21**(3), 250 (2015)
18. S. Masood, W. Song, Materials & Design **25**(7), 587 (2004)
19. B.G. Compton, J.A. Lewis, Advanced Materials **26**(34), 5930 (2014)
20. X. Li, R. Cui, L. Sun, K.E. Aifantis, Y. Fan, Q. Feng, F. Cui, F. Watari, International Journal of Polymer Science **2014** (2014)
21. B. Reddy, N. Reddy, A. Ghosh, Virtual and Physical Prototyping **2**(1), 51 (2007)
22. H. Valkenaers, F. Vogeler, E. Ferraris, A. Voet, J.P. Kruth, in *Proceedings of the 10th International Conference on Multi-Material Micro Manufacture* (Research Publishing, 2013), pp. 235–238
23. C. Rauwendaal, *Polymer extrusion* (Carl Hanser Verlag GmbH Co KG, 2014)
24. D. Schweitzer, E.S. Cobb, in *ACM SIGGRAPH Computer Graphics*, vol. 16 (ACM, 1982), vol. 16, pp. 265–271