

Bachelor's Thesis

**Optimization and Control of 5D (Bio-)printing**

Stephan König

Spring 2024

Supervisor: Prajwal Agrawal

Prof. Dr. Daniel Ahmed

Acoustic Robotics Systems Lab

Swiss Federal Institute of Technology Zurich (ETH Zurich)

Abstract

Multi-axes additive manufacturing, also known as 5D printing has been a significant focus of research in the past few years. The benefit of combining additive manufacturing and improving on existing systems by adding additional degrees of freedom to optimize topography, strength and manufacturability has been researched in different ways. The technical challenge of the added complexity is one focus as well as the benefits to be exploited in different industries such as bioprinting.

The goal of this thesis is to enable the printing process in the project ‘SonoBone’. It aims to print in-vivo and in-situ bone structure using an acoustically assisted 5D-printing process. The work done as part of this thesis is the creation of a program with a graphical user interface that can interpret sliced .gcode files, connect the program to the robotic arm as well as an Arduino and then synchronously print 3D structures using Acrylite.

List of Contents

[Abstract 2](#_Toc170548206)

[List of Contents 3](#_Toc170548207)

[Acknowledgements 4](#_Toc170548208)

[Abbreviations 5](#_Toc170548209)

[List of Figures 6](#_Toc170548210)

[01 Introduction 7](#_Toc170548211)

[1.1 Additive manufacturing overview 7](#_Toc170548212)

[1.2 Bioprinting 7](#_Toc170548213)

[1.3 5D (bio-) printing 8](#_Toc170548214)

[02 Goal 9](#_Toc170548215)

[03 Methods 9](#_Toc170548216)

[3.1 Robotic Arm 9](#_Toc170548217)

[3.2 Bioink and UV-LED 10](#_Toc170548218)

[3.3 Printhead 10](#_Toc170548219)

[3.4 Rotating Base 10](#_Toc170548220)

[3.5 Extrusion 11](#_Toc170548221)

[3.5 Slicer Settings 11](#_Toc170548222)

[04 Project work and results 12](#_Toc170548223)

[4.1 Meca500 handling 12](#_Toc170548224)

[4.1.1 Robot Workspace and Singularities 12](#_Toc170548225)

[4.1.2 Vertical arrangement 13](#_Toc170548226)

[4.1.3 Perpendicular arrangement 13](#_Toc170548227)

[4.1.4 Tilted needle arrangement 14](#_Toc170548228)

[4.1.5 Rotating base 14](#_Toc170548229)

[4.1.6 Varied base mounting 14](#_Toc170548230)

[4.2 Plotting 14](#_Toc170548231)

[4.3 GUI 15](#_Toc170548232)

[4.4 Code Structure 16](#_Toc170548233)

[4.4.1 gui.py 16](#_Toc170548234)

[4.4.2 globals.py 17](#_Toc170548235)

[4.4.3 gcode\_translator.py 17](#_Toc170548236)

[4.4.4 utility\_functions.py 17](#_Toc170548237)

[4.4.5 stepper\_control.py 18](#_Toc170548238)

[4.4.6 Arduino code 18](#_Toc170548239)

[4.5 Acrylate printing using a lead screw extruder 19](#_Toc170548240)

[05 Discussions and Conclusions 19](#_Toc170548241)

Acknowledgements

I would like to thank the ETH Zurich and the Acoustic Robotics Systems Lab for the opportunity work on my bachelor’s thesis. This entire project has been intense work over the past months. I want to thank Prof. Dr. Daniel Ahmed and my Supervisor Mr. Prajwal Agrawal for supporting me to the best of their abilities during this time and having the vision to start this endeavour.

I would also like to thank the entire team of ‘SonoBone’ for the collective work and cooperation as well as our friendship during the time of this project which made difficult times enjoyable.

Abbreviations

CAD

List of Figures

[Figure 1: Meca500 6-axis robotic arm [Source] 10](#_Toc170569998)

[Figure 2: IP address setup window [Meca500 user manual] 11](#_Toc170569999)

[Figure 3: CAD render of the rotating base assembly 12](#_Toc170570000)

[Figure 4: Robot wrist workspace [Meca500 user manual] 14](#_Toc170570001)

[Figure 5: testing of the availlable workspace in different configurations using an adjustable nozzle size 15](#_Toc170570002)

[Figure 6: Plotting images with the meca500 17](#_Toc170570003)

[Figure 7: Screenshot of the SonoBone control interface 19](#_Toc170570004)

[Figure 8: printing one layer with ecoflex without heat curing 22](#_Toc170570005)

[Figure 9:printing attempts using ecoflex material while heat curing 22](#_Toc170570006)

[Figure 10: Robot setup to print acrylate using lead-screw extruder and UV lights 23](#_Toc170570007)

01 Introduction

1.1 Additive manufacturing overview

Additive Manufacturing, commonly referred as 3D printing, is a transformative approach to traditional industrial manufacturing techniques as well as prototyping which has been gaining massive interest in the past 10 years. While operating on different principles, all AM technologies utilize a bottom-up approach by adding material selectively and fusing it to existing geometry while more traditional manufacturing employ a subtractive process. Both the consumer and industrial sectors have benefited from the development of different technologies in AM. The ability to quickly produce very intricate models into the physical space from just a digital model is a powerful ability for all prototyping. While not optimal for large batch sizes, there exist also mass production facilities that employ AM for smaller volume parts as well certain complicated parts. There exist many techniques for AM such as SLA, FDM, Binder Jetting, SLS and more with their respective applications and advantages. Commercially available machines for consumer focus mostly on SLA and FDM principles. [1]

SLA is a photopolymerization technique, which generally refers to the curing of a liquid or viscous photoreactive polymer resin using a laser. By having a laser cure an entire planar layer at once, this technique can print quickly for high volume parts. The main downsides to this is the requirement for post-processing such as cleaning and post-curing. It is possible to achieve very high details using SLA.

The other commonly used technique is FDM. With FDM, the printer deposits material at the desired places using a gantry that is able to move in 3D-space. Commercial systems use solid filament that is melted in a hot end so that is viscous and able to fuse to the previous layers. The main downside is the slow printing speed and low details. Newer developments have been improving the speed and details. The advantage of this method is the rather simple utilization and the lack of post-processing needed. This has been the main factor in the commercial success of FDM machines.

1.2 Bioprinting

Bioprinting is an emerging technology that is used to patch dead cells or even print entire new biostructures such as organs and bones. It is an advanced form of Am that that can create complex biological structures by depositing different bioinks to mimic living structures. Bioinks are a combination of cells and bio-compatible cell-supporting materials. Although bioprinting is still in its infancy, its versatility has accelerated the applications in tissue engineering. Further use can be gained in the development and testing of drugs, especially for personalized medicine.

With the emergence of 3D-Bioprinting the scientific community has gained a higher degree of control using multiple bioinks to compile increasingly complex structures. The main bioprinting methods can be classified into laser-assisted bioprinting, inkjet bioprinting/droplet bioprinting and extrusion bioprinting.[2] explain different conecpts

In many bioprinting processes, scaffolds are required. They can be temporarily used until the main body becomes self-sustaining or permanent depending on the requirements. One main challenge is the utilization of the correct bioink. There exists a vast amount of bioinks, optimized for the type of cells, type of printing process and target tissue. The main metrics are the structural and functional integrity as well as cell survivability. While it is easier using 3D-Bioprinting, vascularization, the development of a network of blood vessels in the cell structure as well as the complexity and scale of the printed structure remain a challenge.

1.3 5D (bio-) printing

The classical cartesian based systems face various downsides such as the effect of staircasing, which leads to a lack of feature size in the vertical direction. The main way to combat this is the reduction of the layer height to the desired size of the features which reduces the printing time drastically. Another downside is that the strength of parts is highly dependent on the orientation of layers. While the part strength in the direction of the layers is strongest and dependent on the material strength. The part strength across layers is significantly less than inside a layer and depends on the quality of layer adhesion.

Another downside of classical FDM approaches is that layers of filament or ink are deposited and supported by previous layers. This makes printing overhangs difficult as there is no supporting material beneath. A certain overhang (classical FDM machines can handle 45 ° well) is possible as the previous layer does offer some structural support, even if not completely. This can lead to sagging in the affected parts depending on the print settings. When working with liquid or viscous materials overhangs are even more critical.

Methods like the S^3 slicer have tackled these problems by employing a multi-axis approach to AM. The proposed method introduces additional degrees of freedom to classical 3D printers to achieve various optimizations. This leads to a variety of advantages compared with the planar layer-based 3D printing including the reduction of supporting structures (SF), enhanced mechanical strength (SR) by printing in curved lines perpendicular to projected stresses and improved surface quality (SQ) by avoiding classical stepping effects. [3]

Current bioprinters mostly utilize modified versions of cartesian 3D printers with a layer-based approach to stack cells on simple-shaped flat planes with the assistance of polymerized biomaterials. [4] This method has multiple downsides such as the inability to provide nutrients during the printing process and the glue-like biomaterial inhibiting cell-to-cell contact.

5D Bioprinting has the potential to tackle these challenges in a similar way that new multi-axis methods have enabled the recent advances in AM. By employing the extra degrees of freedom, it will be possible to print more complex and detailed geometries. Combined with multi-material printing intricate details such as vascularized tissues and curved anisotropic shapes can be created. By adding the precision of highly accurate machinery and increasing structural integrity and stability, cell viability can be improved by precisely positioning cell material.

02 Goal

The goal for this thesis is the enablement of 5D bioprinting in the project ‘SonoBone’. This encompasses the handling of the robot arm including print-assisting components. Also, part of this thesis is the collaborations of the other team members in their advancements and coordination to achieve the aforementioned goal.

‘SonoBone’ is a cutting-edge research project set to develop an acoustically assisted 5D (bio)printer for bone (bio)applications [X-prajwal]. It aims to print in-situ and in-vivo bone structures using a multi-axis multi-material extrusion based system that aligns cells acoustically. The goal is to develop a workings prototype of the SonoBone-Bioprinter.

03 Methods

3.1 Robotic arm

This project is based on the Mecademics 6-axis industrial robot arm Meca500 [5 mecademics]. This robotic arm claims to have a precision of 1 µm and a repeatability of 5 µm. It weights less than 5 kg and is rated for a payload up to 500 g. It comes with an integrated controller in the base of the robot that connects to a PC via an ethernet cable. The Meca500 supports Ethernet/IP, TCP/IP, Ethercat and PROFINET communication. For this project TCP/IP was used.



Figure 1: Meca500 6-axis robotic arm [Source]

The integrated controller features a web-based interface called the MecaPortal, the files for which reside in the robot’s controller. This interface allows the writing for short scripts and control over individual joints or axis. Although it does not support any variables, it is a useful tool to directly control the robot.

For this project, the mecadmicpy API [X] has been used. It features a range of commands that work very similarly to the web interface. Once sent, the robot will use its own controller to fulfil the command, such as reaching a certain location and orientation, also called a pose.

Starting the robot requires the following procedure:

1. Confirm that the emergency stop is not activated.
2. Connect the power cables and turn on the power.
3. Press the Reset button on the robot’s power supply and wait for approximately 30 seconds. The lights on the base will flash until the robot is ready. Only then will the mecaportal be ready. To access it one needs to open a web browser and type in the IP address 192.168.0.100. The connecting computer needs to be setup correctly to be able to connect.

A screenshot of a computer

Description automatically generated

Figure 2: IP address setup window [Meca500 user manual]

1. Activate and Home the robot (either via the Mecaportal or the API).

3.2 Ink

For the material to be able to solidify after a short curing process, but still be printable we used a highly viscous ink. The material used for the prints was the resin di-pentaerythritol pentaacrylate mixed with a photo initiator that is mixed in a ratio of 1g of resin to 1 mg of photo initiator. This was first preheated to just below 100° C and then mixed in a magnetic stirrer at 500 rotations per minute at 100° C. After that it was cooled slightly and then filled up into the syringe from above. It is important to reduce bubbles as much as possible because they greatly disturb the printing process.

3.3 Printhead

A custom designed fixture that attaches to the print head by clamping holds 4-6 UV LED lamps of type LST1-01G01-UV04-00. The LED lamps emit a wavelength of 405nm and have a luminous flux of 930 mW. They operate at 500mA and 3.6 V. They are all attached to a potentiometer and connected to the 5V power and ground pin of an Arduino. The rest of the printhead was kindly provided by Mr. Sven Gautschi who has been working on this for his thesis.

A black object with a hole

Description automatically generated

Figure 3: Render of the UV fixture

3.4 Rotating base

Because of the very limited reach and workspace of the Meca500 with a long end effector and the flexibility requirements to the printing process so that the full capabilities of this multi-axis approach can be utilized, the print bed is not fixed in space but mounted on a FDM 3D-printed rotating bed. This is attached to a NEMA 13 Stepper motor and can be spun around. To make sure the bed is level in all position, the mount features 4 bearing mounted rollers that stabilize the base in all directions. This is necessary so that the base plate is perfectly level. If that would not be the case, then not only bed adhesion for the first layer would be impossible but also, if the bed would rotate, then alignment of the parts would not be proper and it might crash into the printed structures.

The rotating base, as well as all other components aside from the robotic arm, is controlled by an Arduino UNO R4 MINIMA with a CNC stepper shield v3 and stepper drivers A4988 with micro-stepping enabled (1/16) and the EN/GND shorted. The Arduino receives position data via a serial connection from the main controller, which in this case it a windows laptop. The instruction message is always attached to an index, which will be in turn be replied to once the position has been reached.

A white and black device

Description automatically generated

Figure 4: CAD render of the rotating base assembly

3.5 Extrusion

Extrusion is done by a NEMA 13 motor connected to a custom linear mechanism using a lead screw. The stepper is also connected to the previously mentioned Arduino and is controlled in unison with the base.

Attached to the end of the rig is a 3Dprinted mount and a PTFE Tube that then leads into a silicone tube of higher diameter. This is then lead into the printhead where its attached using a reverse chamfer and the flows entered the 3D printed head. In the other hand there is a needle of 0.4 mm diameter that is attached using a Luer lock connection. The entire length is UV shielded using aluminium tape, so that the curing process does not start before the ink reaches the needle tip.

3.5 Slicer settings

Ultimaker Cura was used as a slicer. As the only relevant part of the print is the extrusion information and the print path, there are some relevant settings.

* The base printer settings were the Cura default for the Creality CR10 printer
* Layer height of 0.3 mm with a nozzle size of 0.6mm
* The wall layers were set to 9999 so that there would be no infill. The reason for that is that the infill is printed differently to wall layers, especially in regard to extrusion.
* The top layers were set to 9999 because Cura tends to make the top layer of a higher quality. By setting every Layer to a top layer, the detail of the print should be higher
* Retraction and z-hop when retracted is enabled. The retraction distance is 0.1mm and z-hop height is 0.4mm
* The ‘skin overlap’ setting is set to -0.1mm. This will make it so that the needle tip does not collide into other already printed structures and potentially get stuck. This setting might need some tweaking depending on the extrusion.
* The ‘wall ordering setting’ was set to ‘Inside to Outside’

04 Project work and results

4.1 Meca500

4.1.1 Robot workspace and singularities

A common description of 6-axis robots is the reach and some sort of working area. The reach is the the maximum distance between the robot’s wrist center, which is defined as the intersection point between the axis of the last three joints of the robot and the first, also called base, joint. Figure [X] displays the area of attainable poses for the robot wrist center. The workspace itself, which is the actually space of reachable poses however is highly non-linear and highly dependent on the end effector configuration. The actual description of the workspace is 6 dimensional and generally impossible to display graphically.

A drawing of a mechanical arm

Description automatically generated

Figure 5: Robot wrist workspace [Meca500 user manual]

When handling the robot, one needs to make the distinction between the base reference frame (BRF) and the tool reference frame (TRF). The BRF is the static global coordinate system with origin in the base. The TRF however the coordinate system associated with the actual end effector and origin in the tool reference point (TCP). It moves with the end effector and the software allows for it to be set up dynamically.

Because of the mechanical layout of the robot, it is often the case that the inverse kinematics solution is ambiguous. This means there are different combinations of joint positions or configurations, that can achieve the same pose. These positions are characterized by three distinct properties:

1. The shoulder configuration parameter which describes if the wrist center is on the “front” or “back” side of the plane consisting of joints 1 and 2.
2. The elbow configuration parameter which describes if joint 3 is above or below -72.43°
3. The wrist configuration parameter which describes if the joint 5 is above or below 0°.

So technically every position has 8 different potential configurations, of which not all are attainable though. The limit conditions of these three configuration parameters act as singularities. It possible to move through but one must be especially careful with it.

In every six-axis robot, there exist certain points where the robot end-effector loses certain degrees of freedom, for example it can’t move in a certain direction. They are critical in the utilization of robot arms. They are not a mathematical construct but the real physical restriction of certain configurations [example sketch]. The existence of singularities can lead to a loss of control, theoretical infinite velocities and therefore unpredictable behaviours. These considerations must be considered when doing the path planning and need to be included in control algorithms. Theoretically they can be combatted by introducing further actuators to create redundancy.

Because the Meca500 is a small robot with limited reach and the setup required for the print of structures is large in comparison, it would not be possible to print very large structures and reach all the poses necessary.

To tackle this challenge, it was necessary to determine the limits of operation and gain an intuitive understanding by trying to reach the extreme positions necessary for the 5D printing process. Because this was done in the early stages of the project, there were various uncertainty which had to be taken into account such as the size and length of the printhead, the available space in the chassis, the mounting and position of the base in relation to the print bed. For this reason, a modular printhead was designed and produced with the goal of gaining a higher intuitive understanding. The following description are by no means a complete discussion but a best-effort analysis of the various approaches considering uncertain conditions.

A robotic arm with a black and silver arm

Description automatically generated

Figure 6: testing of the availlable workspace in different configurations using an adjustable nozzle size

4.1.2 Vertical arrangement

This arrangement is the most employed arrangement for 5D printing which has already been used in previous research and was the most straight forward approach. However, the problem, especially with longer printheads, is that the available print size shrinks to zero, or even becomes negative as soon as the length of the printhead reaches half of the reach of the Meca500. And even if it would only approach the length of the reach, it would mean that the size and height is severely limited. The main problem with the Meca 500 is that the robot size and reach is very small compared to the necessary size of the printhead as well as the limited joint angles. The joints are not continuously turnable and are also mechanically limited by the arm structure. One cause for headache was also the often-occurring collision with the base when trying to reach poses with high angles. Because of these reasons, this arrangement was deemed infeasible.

4.1.3 Perpendicular arrangement

The next arrangement that was tested is one where the nozzle is oriented perpendicular to the axis of joint 6. In this case collision with the base would be very marginal, when printing with high angle poses. For these kinds of poses it would work well and the reach of the robot would not pose a big restriction. However, the most concerning problem in this arrangement was in fact the low angle poses, meaning the poses that would be printing close to planar layers. In these cases, it was necessary to double-twist the arm to get into the right positions. This is not necessarily an issue. However, the configurations that allow the robot to reach these poses from different sides are in different configurations so that to move from one of them to the next, it needs to move through a singularity. While this is generally possible with the Meca500, it is still unreliable and not possible from every start to target position. Therefore, this arrangement was discarded.

4.1.4 Tilted needle arrangement

As an option in between the vertical and perpendicular arrangement, tests have been made to find out if tilted needles would perform better. Three needle angles were explored at 60°, 45° and 30°. However, none of those arrangements performed satisfactorily, enhancing or diminishing problems with the previously explored options. Collision with the base would be one of the biggest problems.

4.1.5 Rotating base

Inspired by the paper of [] reinfocedFDM, this approach was developed. Their system utilized two rotation axis around one horizontal rotation axis and one vertical axis relative to the previous rotation. Due to the flexibility of the Meca500 the decision was made to simulate one of the rotation axis by keeping the arm tilting movement in one axis and then rotating the base in a way that the real orientation plane would coincide with the robot tilting plane. This design went through a few iterations but in the end this concept seems feasible to print all kinds of poses up 80° rotation from the vertical axis and therefore selected.

A diagram of a circle with lines and a circle in the center

Description automatically generatedA drawing of a robot

Description automatically generated

Figure 7: Scheme of the coordinate transformation for the rotating base

ENTER EXPLANATION

4.1.6 Varied base mounting

Other options including the mounting of the robot on the wall or the ceiling of the chassis were also explored. Even though not selected due to structural and vibrational reasons, the wall-mounted robot might also work. The ceiling-mounted robot will not work because of the joint limitations of the robot. As can be seen in [previous sketch], the reachable points for the wrist are severely limited in the space above the robot and behind, which makes it infeasible.

4.2 Plotting

4.2.1 Slicing

Conventional 3D printers can print highly complex structures. These are usually designed in CAD programs such as *Autodesk Fusion 360* and provide a 3D mesh file in the *.stl* format. These mesh files are then sliced using a slicing software into planar layers and a path that the 3D printer has to follow. For these tests *Ultimaker Cura was* used to slice all the objects and generate the paths in a *.gcode* file. The information from the gcode was then extracted and printed using a custom python program.

4.2.2 gcode translation and 2d plotting

The first step in order to start the project was to be able to control the robot using the mecademicpy API and a custom program to read *.gcode* files and translate it into positions that the robot should follow. To test this out, a custom designed pen holder end effector was designed using *Autodesk Fusion 360* to plot a single planar layer using a white board and marker. These tests were successful, and this was the basis for the subsequent layers of code that were integrated into the system.

A machine with a black and white object

Description automatically generated with medium confidenceA drawing of a robot arm

Description automatically generated

Figure 8: Plotting images with the meca500

4.3 GUI

The next step was to create a graphical user interface that could be used to control the robot. It consists of a few general robot control operations such as initialization and calibration as well as real time control options for the print such as the z-offset and speed control. Additionally there is a terminal, that displays relevant information and a status bar that displays the current printing state.

* Initialization: This button connects the controlling PC to the robot arm via TCP/IP as well as the Arduino via a COM port. The program itself searches for the correct COM port to use. Both of those need to be synchronized and respond with a confirmation for the program to proceed. The Arduino makes the rotating base turn in quick succession and command the robot arm to reach a starting position. A Remark shall be made that Ultimaker Cura can interfere with the connection to the Arduino. If there is an error in the connection, then it might be because of that. To mitigate one needs to close Cura and retry.
* Select File: By pressing this button, the file explorer window opens, allowing the user to select a file to be printed.
* Start printing: This will start the printing process. Firstly it reads the .gcode file line by line to extract the relevant information about the model. Then it will activate the ‘preview print’ button and start the main printing loop. Only then will the robot arm move to start printing.
* Pause Printing: When trying to pause the printing loop, the robot will not react immediately but finish the commands stored in the motion queue and then turn into a position that presents the print until further input. The button itself will change label into “resume printing”. If clicked again, the robot will recommence printing at the same point in the print as it has been stopped before.
* Stop Printing: This button stops the print and slowly moves the printhead back into the starting position. After this, one can reselect a file and start printing again.
* Calibrate: This calibration pose is used to determine the z-position. In order not to collide with the base, if it is slightly too high, the calibrate button moves the robot arm into a position 10mm above the print surface. One can the use the z-offset buttons to correct the orientation and then stop the calibration again by repressing the same button.
* Status: The status field shows relevant information about the printing state such as the file that has been selected, the print progress and the printed structure.
* Print info: This works similarly to a terminal and displays some information about the system and helps the user keep track of what happened.
* Z-offset: The two arrow buttons allow the user to control the z-offset in increments and decrements of 0.1mm. For more control, the field that displays this offset is writable. By writing a number and pressing enter, a finer control is possible.
* Speed: This modifies the printing speed at a percentage of a base speed in increments of 5%. The field is writable in the same way as the Z-offset. Note that even though the system theoretically allows very high-speed modifiers, there is a physical limit to what the system can achieve. Note also that with the pressurized extruder setup in place, there is no way to control the extrusion speed so the user should refrain from modifying carelessly
* Extrusion: This allows the user to modify the amount of extrusion that is done. The program takes the extrusion information from the .gcode instructions and multiplies it with a constant, which can be modified by using this button. Note also here, that with the pressurized valve, one cannot control the extrusion amount. In this case, this button has no effect.
* Reset System: This button aims to reduce errors that might happen with the robot arm. By pressing this button it send error resetting commands to the robot. This is by no means a complete reset. For a complete reset the user should restart the software.
* Preview print: This button gets activated once the printing process is started and plots the extracted coordinates from the .gcode file and plots it in a separate window.

While this system is not perfect by any means, it allows the user to use the SonoBone setup without diving into the code.

A screen shot of a computer

Description automatically generated

Figure 9: Screenshot of the SonoBone control interface

4.4 Code Structure

4.4.1 gui.py

This is the main part of the code which handles the entire graphical user interface. It consists of the aforementioned buttons and functions. It also starts all the functions as threads that need to allow the GUI to function even while running such as a *progress\_update*() thread or a *status\_update()* thread. Generally the program is defined into a few functions that define all the elements, a few functions that handle button presses as well as a few functions that are being run as a thread as to not interrupt that code. After every button is pressed, the program first disables the button and then checks if any other function is currently running. Another button shall not begin their function, if there is one active already. After that come a few button specific checks. E.g. The “stop print” button can only run, when there is a print active. This is done by checking GlobalState.printing\_state variable. The main function, that can also be called in another file, is the init\_gui() function.

4.4.2 globals.py

There are any global variables that are used in the entire program and they are defined in the globals.py file. It is split into two singleton classes that can only be created once. The RobotStats class includes all the invariant data such as the build volume, minimum coordinates. The GlobalState class acts more as an information carrier between different parts of the program. It contains real-time user inputs such as the user\_z\_offset or file path information. The information in this class can be changed throughout the program.

4.4.3 gcode\_translator.py

This file is the core of the printing process. The first function is *extract\_gcode()* which reads the file information from the filepath location specified by the user and iterates over all the lines of the *.gcode* file. This reads all the G0 and G1 commands which specify movement and includes assembles this into an array of coordinates. Simultaneously it keeps track of the maximum and minimum values in each direction so that it can later be used to move the position of the print to the middle of the printbed by applying an offset.

The coordinates then go through a function called *modify\_coordinates().* This function takes the orientation data from the gcode to apply the algorithm described in a previous chapter to extract the robot tilt angle as well as the base rotation angle. Consequently, those coordinates will be saved in the *GlobalState* class.

The whole printing process is coordinated in the *start\_print()* function, where all the input and data and checks happen. The main printing happens in the *write\_coordinates()* function. This will first set a few things to be correct such as the printing speed and resets the Arduino position.

For every position there is an iteration count assigned to which will require a confirmation from the Arduino and the robot arm, so that the program can continue. Because the robot function is based on a motion queue, i.e. the robot receives commands and fulfils them on his own without control from the outside, a “checkpoint system” is employed. This system ensures that the extrusion and the turning of the robot is in sync. This is a simple system but given the limitation in the communication speeds over serial, this works better than other approaches. For some continuous motion to happen, the code only requires every n-th checkpoint to be synchronized, where n is an arbitrary number. In the current version of the code n is set to 4. After 100 steps, one counter resets to 0 and starts counting again. The reason behind this is that the Meca500 arm supports only a finite number of checkpoints.

The last function in this file is the *display\_preview()* function which opens a second window with matplotlib and displays all the coordinates of the print.

4.4.4 utility\_functions.py

The *utility\_functions* file includes a lot of functions, not all of them are used in the current version of the program. The most interesting and useful functions are the following:

*GetPose()* is a function that gets the real-time data object from the robot and extracts the real time position of the current time step and outputs this information. The option to record the position data is not on by default but is done in this project in the *initsquence()* program.

*Remark*: The extraction of the real-time data objects sometimes returns a *null* object. It was not possible to determine why this happens. However, it is consistent in terms of where in the program it is called. If it works for a single, it works consistently and if it does return the null object then it does so always, and the position of that code needs to be changed.

*Checklimits()* takes the input coordinates and checks whether they are in the bounds of the globally defined limits.

*Cleanpose(), endpose(), startpose(), callibrationpose()* are all predefined locations which can be changed collectively depending on the needs.

*Initsequence()* is the starting the entire robot and Arduino connection, does some first tests and takes the *init\_position.*

*Commandpose()* is the main function to make the robot achieve a pose. Instead of directly sending the command to the robot, this function employs the *checklimits()* function and then overrides any infeasible position, so that the robot doesn’t encounter any error.

4.4.5 stepper\_control.py

The entire communication to the Arduino is done by this program. It includes a connection function to search for the correct COM port to connect to the Arduino, all the messages that are encoded and sent via serial as well as the reading of the serial response. The function *read\_steppers()* runs in a separate thread and constantly reads the serial port, unless a writing action is done. If a new message shows up, it is saved into the global message array that is then used to check if a certain response has been received. This is done, so that no messages get lost if the check for a message is done after the message has been received.

The message format is very specific and needs to be specified in the Arduino code as well. The feed rates and speeds depend on many global variables and constants and need to be adjusted with any change in the print setup.

4.4.6 Arduino code

The Arduino code consists of multiple parts. The first part is the reading of the serial port for messages that may come from the main python program. This message is then analysed and depending on the format, it can be handled in different ways such as the turning of certain motors or the initialization of the connection.

If the command is to move the stepper motors, then the *halfPulseDuration* is calculated based on the speed. Stepper motors work based on a rapid switching of the stepping pin on the motor driver depending on the switching time or *PulseDuration*. The calculated number of steps is also calculated separately for each motor. Then the stepper driving function is called. This handles both motors in a loop until both have reached their desired step count. The last switching time for both motors is recorded and used to switch each stepper motor individually at the corresponding time.

After the motor is finished, they will stop and a confirmation reply including the iteration number will be sent back via serial.

4.5 Printing with Ecoflex material

The first tests conducted using the robot with this setup were done only with a single layer to investigate the behaviour of the material after deposition and was also not cured. In a second step the material was cured and multiple layers were attempted. A significant challenge was the manufacture of a heat resistance printbed that was heatable and compatible with the robot arm. This setup was less than optimal and not perfectly level. Also the high temperature of the printbed sometimes cause uneven curing already near the needle exit which caused blobs to form. Another problem was the high viscosity and lack of fine control of the print, which sometimes caused the material to flow out without actively being extruded. This problem was even more significant in later experiments using acrylate resin. The printing experiments with acrylate followed quickly after this because it was suspected that a non-heat-based material curing process would be easier to achieve.

A robotic arm with a green needle

Description automatically generatedA close up of a logo

Description automatically generated

Figure 10: printing one layer with ecoflex without heat curing

A close up of a device

Description automatically generatedA close up of a glass

Description automatically generated

Figure 11:printing attempts using ecoflex material while heat curing

4.6 Printing with acrylate

To create 3-dimensional structures we collaborated in the SonoBone team to create a setup that can print and cure di-pentaerythritol pentaacrylate and cure it using UV lamps to produce a solid structure. The main component of the extruder and print head were provided from a previous setup and adapted for the robot arm. The addition needed for the curing process consists of six UV lights in a fixture designed for the robot arm. This design was also shared with other team members for their setups. The methodology has been described previously. The code used in this setup was solely planar.

The results with this setup have been very inconsistent. The main challenges were the lack of control of the extrusion flow. Even though the program is able to send precise position commands to the stepper motor controlling extrusion, the real flow is dependent more on the pressure in the fluid channel. In some cases, the flow would continue to print multiple layers without the motor even being connected to the power supply. Under these conditions it was hard to adjust the flow to not over-extrude. Both under- and over-extrusion are significant problems. If the printer extrudes too much ink in a short amount of time, a puddle of ink forms, not only destroying the print structure and precision but also trap the needle in a well after curing. Because the curing process was done continuously, the needle would get stuck either shortly after overextruding, or if it continued the print and printed the next layer, it would get stuck.

A robotic arm with a blue light

Description automatically generatedA robot arm with wires and cables

Description automatically generated

Figure 12: Robot setup to print acrylate using lead-screw extruder and UV lights

Another issue was an inconsistent curing. In some cases, the ink would cure very shortly after leaving the nozzle and then form a blob that would stick to the nozzle and attach more ink to it. Once it received enough weight to fall off or some collision made it detach, it would drop into the previously printed part and adhere to the previous layers. This would have a similar effect as the previously described challenge. In some rare cases the ink would cure even faster and lead to a clogged nozzle. In all three cases the print is unusable and needs to be aborted.

A close-up of a square

Description automatically generatedAdd acrylate print and results

A close up of a metal object

Description automatically generated

A close-up of a glass

Description automatically generated

A round object on a surface

Description automatically generated

A glass plate with a hand imprint

Description automatically generatedA white object on a white surface

Description automatically generated

05 Discussions and Outlook

**Sources**

**Image sources**

Picture meca 500 <https://th.bing.com/th/id/OIP.2bD5vTrLOSWMq2bM0pfIzQAAAA?rs=1&pid=ImgDetMain>