Closed-Loop Control of Robotic 3D Clay Printing Using Machine Learning for Overhang Structure

Xiaochen Ding

Building Technology Graduation Thesis

Xiaochen Ding

TU Delft
Faculty of Architecture and the Built Environment
Master of Science in Architecture, Urbanism, and Building Sciences
Building Technology Track

Date: 25 June 2025

Thesis mentors:
Dr. Serdar Aşut First Mentor
Dr. Charalampos Andriotis Second Mentor

Abstract

This research presents the development and validation of a machine learning-assisted closed-loop control system designed for robotic 3D clay printing, with a particular focus on enhancing print quality and structural stability of complex overhang geometries. Unlike traditional open-loop manufacturing processes, the proposed framework integrates real-time visual feedback captured by dual Raspberry Pi cameras and leverages a multi-objective neural network model to dynamically adjust the robotic printing speed. This enables the system to detect and correct extrusion anomalies such as over-extrusion and under-extrusion, thereby significantly improving the printability and quality of challenging overhang structures.

Two advanced deep learning architectures, ResNet-56 and a DINOv2-based hybrid network, were systematically evaluated to determine their effectiveness in defect detection and prediction of overhang success. The system was implemented using a UR5 robotic arm equipped with a clay extruder, demonstrating practical feasibility within a laboratory environment. Experimental results show that the closed-loop control approach substantially enhances print consistency, reduces structural failures, and maintains geometric accuracy compared to baseline open-loop methods.

This foundational work lays the groundwork for future scaling to construction-scale additive manufacturing, highlighting the potential to extend this system to other construction materials such as concrete. The study discusses key challenges including material variability, sensor integration, real-time control complexity, and robotic motion planning, and provides strategic recommendations for future research aimed at achieving robust, adaptive, and scalable additive manufacturing systems for complex architectural applications.

Keywords: 3D Printing, Additive Manufacturing, Clay, Robotic Fabrication, Machine Learning, ResNet-56, DINOv2-based hybrid model, Computer Vision, Real-time closed-loop control, Quality Monitoring, Defect detection, Overhang structures

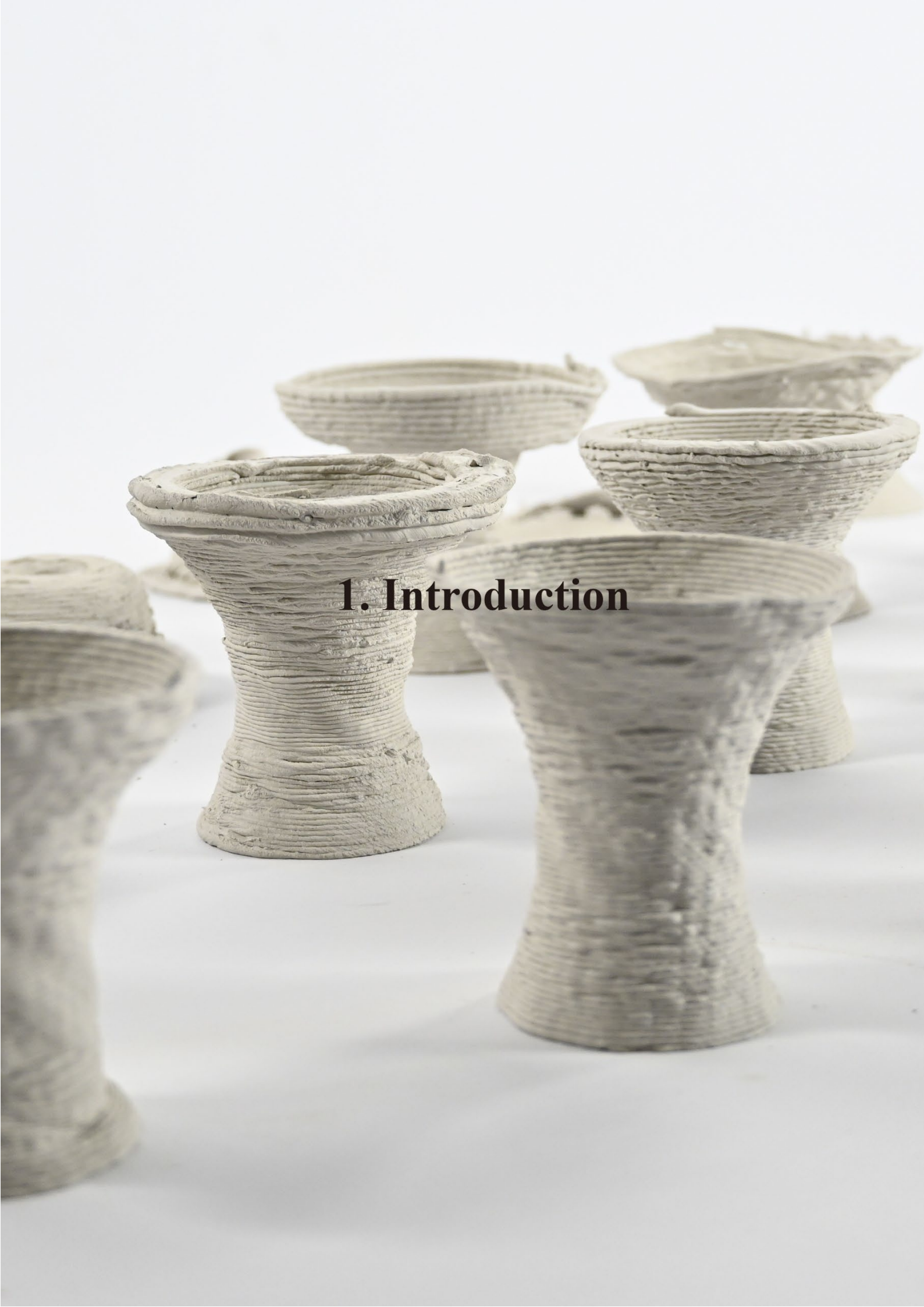
Research Context

This graduation project is a direct continuation of the previous group research project, MudTracker 3D, conducted from September 16, 2024, to November 6, 2024. During that group project, I was primarily responsible for machine learning model training and fine-tuning. Over the course of nine main rounds of fine-tuning, various strategies were explored to improve model performance, including adjusting hyperparameters, applying class balancing techniques, extracting balanced datasets from initially imbalanced data, refining the labeling standards for greater accuracy, redesigning prototypes, modifying print board colors, and adjusting lighting conditions during image collection.

Despite these efforts, the machine learning model predictions remained unstable and inconsistent. Key findings from that phase indicated that class balancing techniques are critical for model accuracy and generalization. Balanced datasets yielded the best results, while upsampling and downsampling techniques were more effective than simple class weighting when perfect balance could not be achieved. Training on imbalanced datasets without proper adjustment significantly reduced accuracy, especially for minority classes. Additionally, consistent lighting conditions were found essential during data collection to ensure reliable training outcomes. Incorporating diverse lighting conditions into the dataset was also recognized as a potential way to improve model robustness in real-world scenarios.

The motivation behind continuing this topic is to overcome the limitations encountered in the MudTracker 3D project, particularly the instability of ML predictions and the open-loop nature of the system. MudTracker 3D was an open-loop calibration system implemented on a WASP printer, using a single side-view camera for image collection. In contrast, the current graduation project develops a closed-loop control system integrated with robotic fabrication and enhanced visual feedback from dual top-view cameras. This setup comprehensively captures geometric features and focuses more on architectural geometry characteristics. By employing more advanced neural network architectures, the project seeks to enable precise and robust 3D clay printing of complex overhang structures, ultimately contributing to more efficient and automated construction processes.

Contents



1. Introduction

1.1 Background

Additive manufacturing (AM), widely known as 3D printing (3DP), allows to construct complex geometries by depositing material layer by layer according to digital models (Lee & Park, 2025). Recent advancements in AM are pushing the boundaries of real-world design possibilities, allowing for mass customization, the fabrication of intricate structures, and the reduction of material waste, while also enabling rapid prototyping (Delgado Camacho et al., 2018). AM technologies are increasingly being applied across diverse scales and industrial fields including healthcare (Placone & Engler, 2018), medical devices (Haghiashtiani et al., 2020), robotics (López-Valdeolivas et al., 2018), aerospace (Froes et al., 2019), engineering (Singh et al., 2020), and construction. As noted by Ngo et al. (2018), a diverse range of materials is utilized in 3DP, including metals, concrete, ceramics and polymers. Among polymers, Acrylonitrile Butadiene Styrene (ABS) and Polylactic Acid (PLA) are commonly used for composite 3DP, while advanced alloys and metals are favored in aerospace applications due to the limitations of traditional manufacturing processes. Ceramics are often used for 3DP scaffolds, and concrete is primarily used in 3DP for construction.

Although 3DP has obtained significant attention in recent years, it is still in the early stages of adoption for construction industry (Wu et al., 2018). Nonetheless, there has been a rapid growth in case studies focused on production of full-scale houses and structures (Parkes, 2021a, 2021b; Teizer et al., 2016). These studies highlight several advantages of 3DP, including the ability to create complex, custom geometries without the need for molds (Delgado Camacho et al., 2018), reducing errors during the building process, and minimizing construction time, material waste and cost (Nematollahi et al., 2017). As a result, 3DP has become increasingly relevant in the construction industry, particularly for its potential to revolutionize how buildings are designed and constructed.

Several AM processes are already in use, including Stereolithography (SLA), Fused Deposition Modeling (FDM), and Selective Laser Sintering (SLS) (Fastermann, 2016). These processes differ based on the material used and the deposition technique. One method within this field, Liquid Deposition Modeling (LDM), involves extruding viscous material through a nozzle to build structures in a continuous, layered approach. LDM is currently being explored in several manufacturing applications(Delgado Camacho et al., 2018; Jang et al., 2020; Klug et al., 2022; Rosenthal et al., 2018). When this process is applied to clay or other earthen materials, it is referred to as 3D Clay Printing (3DCP). 3DCP is emerging as a promising solution for overcoming the intrinsic limitations of ceramics, for example, formability and processability (Jang et al., 2020). Clay, a naturally malleable and adaptable material, serves as the primary

medium in LDM (Yang et al., 2023). Due to its low embodied energy and recyclability, clay is increasingly considered a viable material for sustainable construction (Oti, 2010). Its malleability and plastic behavior facilitate the formation of continuous, organic geometries, which are increasingly relevant in the design of biophilic and non-standard architectural forms (Grigoriadis & Lee, 2024). The benefits of clay in construction are further amplified by its availability, cost-effectiveness, and sustainability when employed in an LDM system (Kontovourkis & Tryfonos, 2020; Wolf et al., 2022).

However, clay is also a highly variable material. Its performance depends heavily on water content, mixture consistency, and environmental conditions such as temperature and humidity. These factors can lead to extrusion instability, weak layer adhesion, and deformation during printing (Ding et al., 2025; Gürsoy, 2018; Witte, 2022). As a result, 3DCP presents unique challenges, particularly in terms of material consistency and the need for constant monitoring of printing parameters (Aşut et al., 2025). These challenges can result in printing anomalies such as perforations and material overhangs, which can compromise print quality, lead to failures, or require the use of extensive support structures that must be manually removed after printing (Gürsoy, 2018). The need for such support structures restricts design freedom and limits the ability to experiment with form and geometry, which needs to be carefully considered in design process (Bhooshan et al., 2018). Therefore, unsupported overhangs present a significant challenge in maintaining both structural integrity and aesthetic quality during the printing process.

1.2 State of the Art

Machine learning (ML) has become an integral part of the AM field, enabling significant advancements in defect detection, process optimization, and real-time error correction (Jiang et al., 2022; Khan et al., 2021; Li et al., 2019; Ramiah & Pandian, 2023; Zhang et al., 2019). There is growing research on the use of ML for quality monitoring and adaptive control in AM (references). Most of the existing work has focused on materials with relatively stable properties, such as thermoplastics, resins, and metals, where defect detection and parameter optimization are more predictable (references).

Most research focuses on correcting under-extrusion and over-extrusion issues to improve print quality. Brion and Pattinson (2022) developed a real-time defect detection and correction system for extrusion-based 3DP using a multi-head residual attention neural network named Resnet 56. Trained on over 1.2 million automatically labeled images collected in thermoplastics 3DP process, the model predicted deviations in parameters including flow rate, Z-offset, lateral speed, and hotend temperature. An overall classification accuracy of 84.3% is obtained, with individual parameter accuracies exceeding 85%, and demonstrated robust performance even on unseen materials like ketchup.

Building on similar ideas, Fu et al. (2025) proposed a real-time feedback system combining EfficientNetB0, a lightweight Resnet 56 network, with camera-equipped extruders, which only predicts and adjusts two printing parameters including flow rate and nozzle offset to correct extrusion defects during printing process, improving print quality for polylactic acid (PLA) materials. Avro et al. (2024) developed a deep learning framework that integrates YOLO (You Only Look Once) and VGG-16 for detecting under-extrusion anomalies in Fused Filament Fabrication (FFF) for thermoplastics, achieving a 97% detection accuracy. Similarly, Goh et al. (2022) introduced a YOLOv3 and YOLOv4-Tiny ML models for extrusion defect detection system in PLA printing with 89.8% accuracy, enabling real-time correction of over- and under-extrusion issues. Jin et al. (2021) employed YOLOv3 and DeepLabv3 for semantic segmentation of over- and under-extrusion zones for PLA materials in FFF, enabling layer-wise quality assessment and correction. Point cloud data was utilized by Akhavan et al. (2024), who established a hybrid convolutional autoencoder (HCAE) to classify under- and over-printed regions in PLA and dynamically modified G-code to improve subsequent layers.

In parallel, some studies focused on improving surface roughness in 3DP. Lee and Park (2025) introduced a Variational Autoencoder (VAE) to detect surface defects in additive-lathe printing for PLA and applied corrective strategies such as ironing and speed tuning, significantly enhancing surface quality in situ. Toorandaz et al. (2024) used Random Forest (RF) and XGBoost with photodiode sensors to predict surface roughness in titanium alloy prints, achieving high accuracy at microscale resolution. In bioprinting, Jin et al. (2023) used convolutional neural networks (CNNs) for anomaly detection in hydrogel-based systems, achieving an F1-score of 0.955 by classifying discontinuities and irregularities layer-by-layer.

Addressing misalignment in prints in 3DP, Zubayer et al. (2024) applied YOLOv8 to detect fiber misalignments in carbon fiber-reinforced polymer (CFRP) 3DP, integrating nozzle temperature adjustments for closed-loop correction, and reaching 94% detection accuracy. Kim and Park (2023) combined a VAE for anomaly detection with Proximal Policy Optimization (PPO), a reinforcement learning algorithm, to adjust print speed for polyvinyl alcohol (PVA) filament. The system achieved over 99% accuracy in detecting defects such as layer shift, and successfully optimized speed to minimize deflection and improve productivity, enabling the reliable fabrication of unsupported high-aspect-ratio and overhang geometries. Lu et al. (2023) also used YOLOv4 in a real-time feedback system for CFRP printing to detect misalignment and abrasion, achieving real-time adjustment of printing parameters to improve surface finish.

Surface deformation has also been a key focus in 3DP. Ansari et al. (2022) developed a CNN model for identifying surface deformation defects including distortion and warping in laser powder bed fusion (LPBF) of AlSi10Mg metal with 99.3% accuracy. Brion et al. (2022) integrated YOLOv3 with heuristics corrections to detect and correct warping in extrusion-based

AM for Acrylonitrile Butadiene Styrene (ABS), adjusting bed temperature and print speed to reduce deformation. Gunasegaram et al. (2021) proposed Digital Twin integration with surrogated model in metal AM, using ML-driven sensor feedback to optimize process parameters and ensure consistent part quality and reduced defects. Finally, Paraskevoudis et al. (2020) used Single Shot Detector (SSD) networks for real-time video-based detection of stringing defects in PLA prints, achieving notable improvements in material efficiency and print quality.

Together, these studies reflect a diverse and rapidly evolving body of work demonstrating the potential of ML-based frameworks, ranging from object detection (e.g., YOLO, SSD) to generative modeling (e.g., VAE), for improving print reliability, surface quality, and adaptive parameter control across a variety of AM materials and systems.

Despite these advancements, a major gap remains in applying ML for real-time error correction in 3DCP. Existing studies have focused primarily on materials such as thermoplastics, metals, and composites, which exhibit more predictable behaviors during the printing process. Clay, on the other hand, presents unique challenges for its unpredictability due to its highly variable properties, such as viscosity, malleability, and moisture content, making 3DCP more prone to printing issues. Although ML techniques from other materials provide valuable insights, they need significant adaptation to suit clay material behavior.

Because of the inherent variability and ununiform clay mixture due to non-standard mixing process of clay, the printing issues are amplified during 3DCP than other predictable materials. Variations in printing quality can occur across different layers due to changes in the clay mixture even within a single prototype. Traditional fixed-parameter methods fail to adequately accommodate these rapid and unpredictable shifts in material state, typically requiring human intervention during the printing process, which is not always possible or effective enough. Traditional open-loop approaches operate based on pre-set parameters without real-time feedback, assuming constant material properties and environmental condition (Ding et al., 2025). Așut et al. (2025) developed a computer vision-based open-loop system for pre-printing parameter calibration in 3DCP, aiming to address material variability and printing defects. The system employs a two-head Attention-56 model (based on ResNet-56) to predict optimal layer height and extrusion settings. The study demonstrated the effectiveness of automatic parameter calibration compared to manual calibration, showing improved efficiency in pre-printing preparation. However, this open-loop system cannot dynamically adjust to fluctuations in extrusion consistency and inter-layer adhesion during real-object printing processes, which makes formal printing process especially prone to printing errors caused by the natural variability of the material. Without real-time sensing and adjustment, these systems struggle to compensate for changes in material behavior and it often leads to inconsistent results and defects (Ding et al., 2025).

Unlike open-loop systems, closed-loop systems can process sensory data by recognizing changes in material properties, environmental disturbances, and calibration errors in real-time. They can continuously update the printing parameters and allow real-time correction of extrusion inconsistencies and printing errors (Zhu et al., 2021, as cited in Ding et al., 2025). Thus, a notable gap remains in the application of closed-loop calibration systems for 3DCP.

Furthermore, layer adhesion and overhang structures continue to present significant challenges in 3DP from architectural and engineering aspects, especially in materials like clay that have limited interlayer bonding due to cold extrusion. Successful printing of overhangs with minimal layer contact requires optimizing the print parameters to ensure stability. While ML models have been used to optimize print errors in simpler geometries, there is limited research on their architectural application to complex geometries with minimal contact areas, such as overhangs in clay-based 3DP.

In summary, this literature review underlines the potential of ML models in advancing AM, particularly in defect detection, error correction, and process optimization. However, there is a clear gap in research regarding the application of ML in real-time adjustment systems for 3DCP, where material-specific challenges need to be addressed. This research aims to fill this gap by exploring methods for achieving stable, accurate overhang structures in 3DCP without the need for excessive support materials, and with minimal manual adjustments to the printing parameters. This will involve addressing the inherent variability of clay and developing strategies to enable the automatic calibration of printing parameters in real-time, thereby reducing the reliance on manual intervention during the printing process.

1.3 Problem Statement

The use of clay in AM presents unique difficulties that are not well addressed by current systems. Most 3DP workflows rely on open-loop control, where printing parameters are set before the process begins and remain unchanged (Aşut et al., 2025). This method assumes that material properties and environmental conditions stay constant, which is rarely the case in 3DCP. As a result, deviations such as layer shifting, over-extrusion, or incomplete bonding are common, especially in geometrically complex features like overhangs.

Manual adjustments can sometimes compensate for these issues, but they are time-consuming and difficult to scale (Gürsoy, 2018). For construction-scale applications, such delays or inconsistencies can lead to structural weaknesses or even failed builds. Therefore, a more adaptive system is needed, which can detect problems as they arise and automatically adjust printing parameters in response during printing process.

However, the method is largely focused on thermoplastic materials, where extrusion consistency

is more predictable, making its application to clay, which has a highly variable viscosity, less straightforward.

1.4 Objectives

Real-time monitoring allows continuous acquisition and analysis of p process data during manufacturing, allowing immediate adjustments to printing parameters in response to material variability. This is particularly important in 3DCP, where fluctuations in clay consistency can significantly affect print quality. A closed-loop control system builds on this by using feedback from sensors or vision systems to automatically modify process parameters such as extrusion rate or nozzle speed in real time, thereby reducing defects and maintaining print consistency (Zhu et al., 2021, cited in Ding et al., 2025).

ML is a subset of Artificial Intelligence (AI) that has an increasing potential to develop the capabilities and efficiency of AM, and it can handle the challenges and optimize the various aspects of AM processes by extracting patterns, learning from data, and building effective predictions (Ukwaththa et al., 2024, cited in Ding et al., 2025). As outlined in the literature review, ML can play a crucial role in real-time defect detection, process optimization, and adaptive control in 3DP and help to ensure more precise and reliable printing outcomes. In deep learning-based image classification (a subset of ML), models such as Residual Attention Networks such as ResNet 56 and DINO v2 can enhance defect detection by focusing on relevant image regions while minimizing background noise. So, they can be particularly effective for identifying fine-grained defects in 3DP (Wang et al., 2017; Zhao et al., 2017, cited in Ding et al., 2025).

Robotic 3DP utilizes industrial robotic arms equipped with specialized extruders to enable flexible, multi-axis motion control during material deposition (Farahbakhsh et al., 2021). Such systems integrate advanced toolpath planning and robotic control algorithms within parametric design environments such as Grasshopper, allowing the digital translation of 3D models into optimized robotic motions for material extrusion (Kontovourkis & Tryfonos, 2020).

Compared to conventional 3-axis clay printers such as the WASP system, robotic arms offer significantly more spatial flexibility. This enables the fabrication of complex geometries such as overhangs and freeform curves, without extensive reliance on support structures. The multi-degree-of-freedom movement enables non-planar layer deposition and dynamic toolpaths that are difficult or impossible to achieve with fixed-axis systems. This capacity for real-time trajectory updates allows in-process correction strategies, such as modifying toolpaths in response to detected anomalies. However, this project will focus specifically on adjusting printing speed to correct failures in overhangs using a UR5 robot due to time constraints. Furthermore, current

robotic arm-based 3DP serves as a scalable prototype system for future construction-scale manufacturing using 6-axis gantry systems, providing a more compatible and transferable workflow compared to desktop-scale 3-axis WASP printers. This makes it a more suitable platform for developing adaptive, closed-loop fabrication techniques aimed at architectural-scale applications.

Thus, this research aims to develop such a real-time, ML-assisted closed-loop control system specifically for robotic 3DCP, with a focus on challenging overhang structures. Overhangs are structural elements that extend horizontally without direct vertical support. It poses a particular challenge in 3DCP due to the limited interlayer adhesion and the deformable nature of clay. Printing such structures requires precise control of printing speed and extrusion to prevent sagging or collapse. By targeting overhangs, this research intends to address a critical barrier in expanding the design freedom and structural stability of 3DCP in construction applications.

A comparison will be conducted between two deep learning models, ResNet-56 and DINO v2, to determine which is more effective in detecting failure. The goal is to achieve automatic and dynamic calibration of printing speed during fabrication, ensuring stable extrusion and improving the overall quality and reliability of clay-based overhang structures.

The key objectives are:

- Robotic 3DP
    - Set up a UR5 robotic 3D clay printing system integrated with Raspberry Pi cameras for visual monitoring.
    - Establish real-time communication between the UR5 robot, the Raspberry Pi modules, and the main control computer.
- ML:
    - Collect a dataset of overhang prints with varying inclination angles and extrusion conditions.
    - Develop a multi-objective ML model capable of detecting extrusion anomalies and predicting overhang printability in real time.
    - Compare ResNet-56 and DINO v2 in terms of prediction accuracy, reliability under 3DCP conditions.
- Close-loop
    - Integrate the selected ML model into a closed-loop system that continuously adjusts printing speed based on live visual feedback.
- Validation

- Test and evaluate the system's performance across various geometries and material conditions to verify its effectiveness in maintaining consistent, high-quality clay prints.

1.5 Research Questions & Sub-questions

Main Research Question:
How can a machine learning-assisted closed-loop system enable real-time anomaly detection and correction in robotic 3D clay printing of overhang structures?

Sub-questions:

1. How can real-time visual data be used to detect extrusion anomalies, such as under-/over- extrusion, or deformation, during the printing of overhang structures in 3DCP?
2. Between ResNet-56 and DINO v2, which machine learning model performs more effectively in classifying extrusion failures and predicting the printability of overhang geometries in 3DCP?
3. How can a trained ML model be integrated into a closed-loop system that dynamically adjusts robotic printing parameters, such as nozzle speed, based on continuous visual feedback?
4. How can a live communication link between the PC, ML model, and UR5 robotic arm be established to support real-time control and feedback exchange?
5. How can the movement speed of the robotic arm be smoothly and safely adjusted in real time via the PC, without disrupting the continuity of the toolpath?
6. What are the implementation constraints and scalability considerations when applying this closed-loop robotic system to construction-scale 3D clay printing using multi-axis platforms?

1.6 Methodology

To address the limitations of current open-loop 3DCP systems and to enable responsive adaptation to material variability, this research proposes a data-driven, closed-loop control system integrating ML and robotic fabrication. The methodology builds upon established approaches in computer vision-based feedback control (Aşut et al., 2025; Brion & Pattinson, 2022; Fu et al., 2025), adapting them to the unique challenges of clay extrusion and overhang geometry in robotic 3DP.

Specifically, the methodology shown in Figure 1.1 follows a five-step framework aimed at developing, training, and validating a real-time feedback loop that can detect extrusion anomalies and automatically adjust robotic movement speed to maintain consistent print quality. The system is built around the use of a UR5 robotic arm equipped with Raspberry Pi cameras and a WASP clay extruder, enabling synchronized image capture, parameter logging, and dynamic control updates.

The full workflow ranges from prototype fabrication and dataset generation, to machine learning model comparison and final system validation. It is designed to establish a replicable and scalable pipeline for adaptive control in 3DCP.

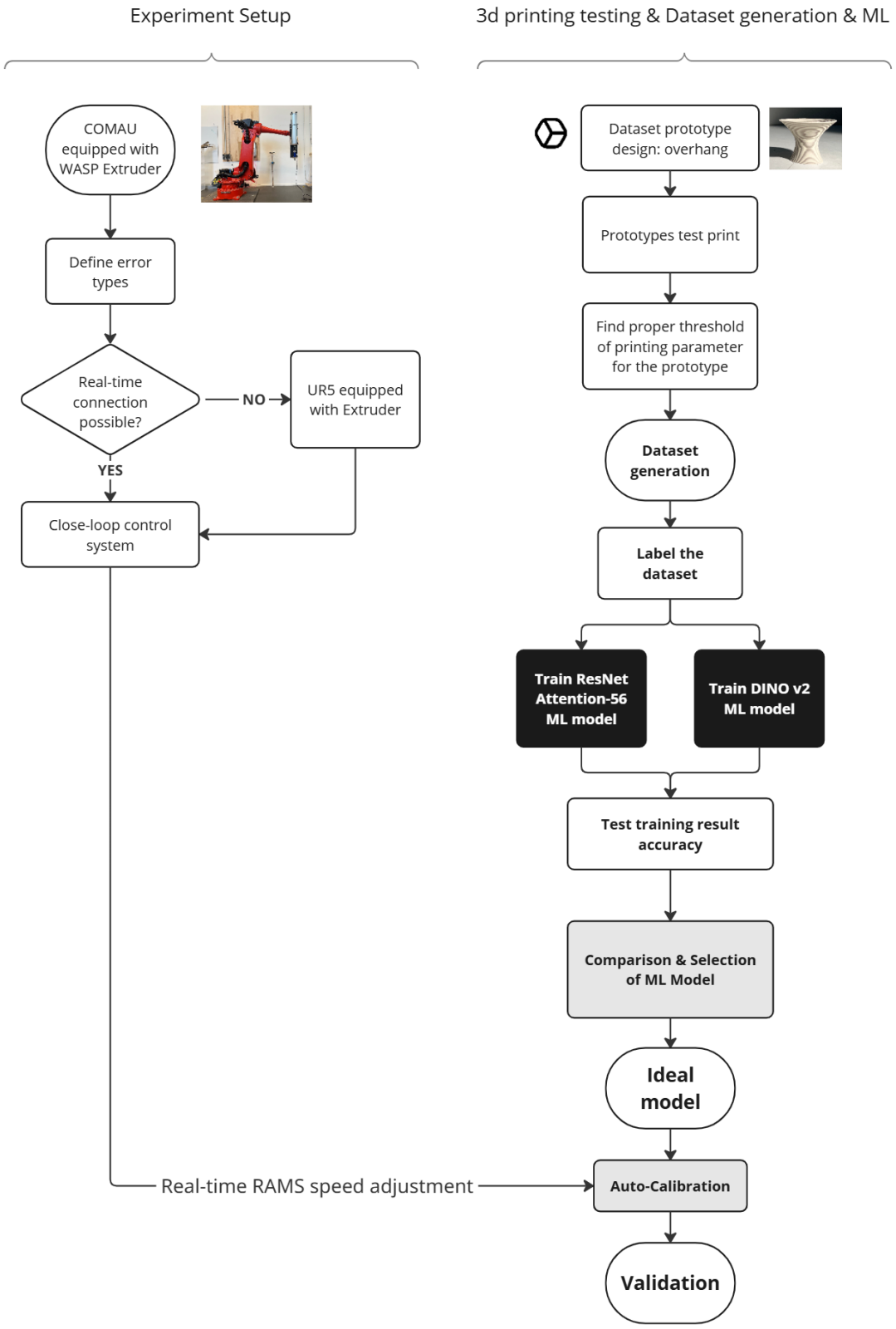


Figure 1.1: the workflow diagram of methodology

Step 1: Data Collection through Prototype Printing

A series of overhang prototypes are printed at varying speeds using a WASP LDM extruder (3 mm nozzle diameter) mounted on a UR5 robotic arm as dataset. Two Raspberry Pi Camera Module 3 units are mounted on either side of the extruder, focusing on the nozzle to capture pairs of top-view images at regular intervals as shown in Figure 1.2.

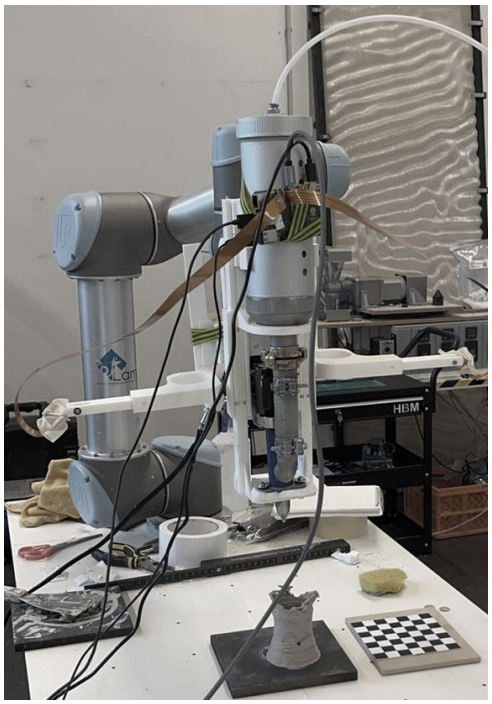
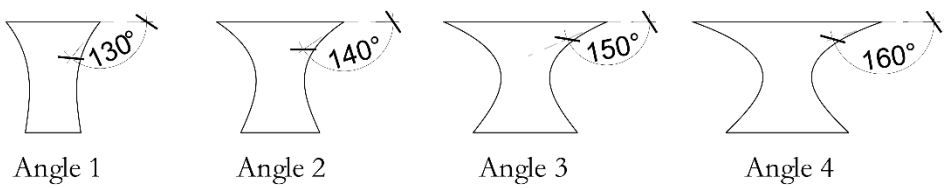


Figure 1.2: Experiment setting of 3DCP.

To generate a diverse dataset, two parameters were systematically varied during printing: the robotic arm's movement speed (RAMS) and the overhang inclination angle (Table 1). The speed was adjusted to 100%, 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, and 10% of the base speed. For each speed setting, overhangs were printed at angles of 130°, 140°, 150°, and 160°. These combinations resulted in a various extrusion outcomes, including both successful and failed prints, which were used to train the ML model.



Speed / Angle	S1 0.1	S2 0.2	S3 0.3	S4 0.4	S5 0.5	S6 0.6	S7 0.7	S8 0.8	S9 0.9	S10 1
A1 130°	√	√	√	√	√	√	√	√	×	×
A2 140°	√	√	√	√	√	√	×	×	×	×
A3 150°	√	√	√	×	×	×	×	×	×	×
A4 160°	√	√	√	×	×	×	×	×	×	×

Figure 1.3: A dataset that needs to be collected with varied printing speeds and inclined angles.

Step 1.1: Evaluation and Benchmarking

The printed prototypes are evaluated based on two criteria: extrusion quality and overhang success. Key indicators include shell thickness and the horizontal interlayer contacting area.

Extrusion quality was assessed by shell thickness (Figure 1.4). It was classified into three categories: *under extrusion* (label 0), *good extrusion* (label 1), *over extrusion* (label 2). The prototypes that matched the nozzle diameter (3 mm) were used as the standard benchmark and labeled as the optimum (*good extrusion*) extrusion. The remaining ones were sorted from very thin (*under extrusion*) to very thick (*over extrusion*) (Ding et al., 2025).

Figure 1.4: Extrusion label categories demonstration according to shell thickness (Top view)

Figure 1.5: Extrusion label categories demonstration according to general quality (elevation view)

Overhang success was evaluated based on the printing completeness and form accuracy of the printed prototypes (Figure 1.6). The outcomes were also categorized into three classes. A print was labeled as *safe* (label: 0) if it completed successfully and maintained its intended shape without noticeable deformation. Prints that were completed but showed signs of sagging or visible distortion in overhang curvature were classified as *at risk* (label: 1). If the structure collapsed during the printing process or failed shortly after completion, it was labeled as *unsafe* (label: 2). This qualitative classification helped differentiate overhang performance levels based on observable physical outcomes.

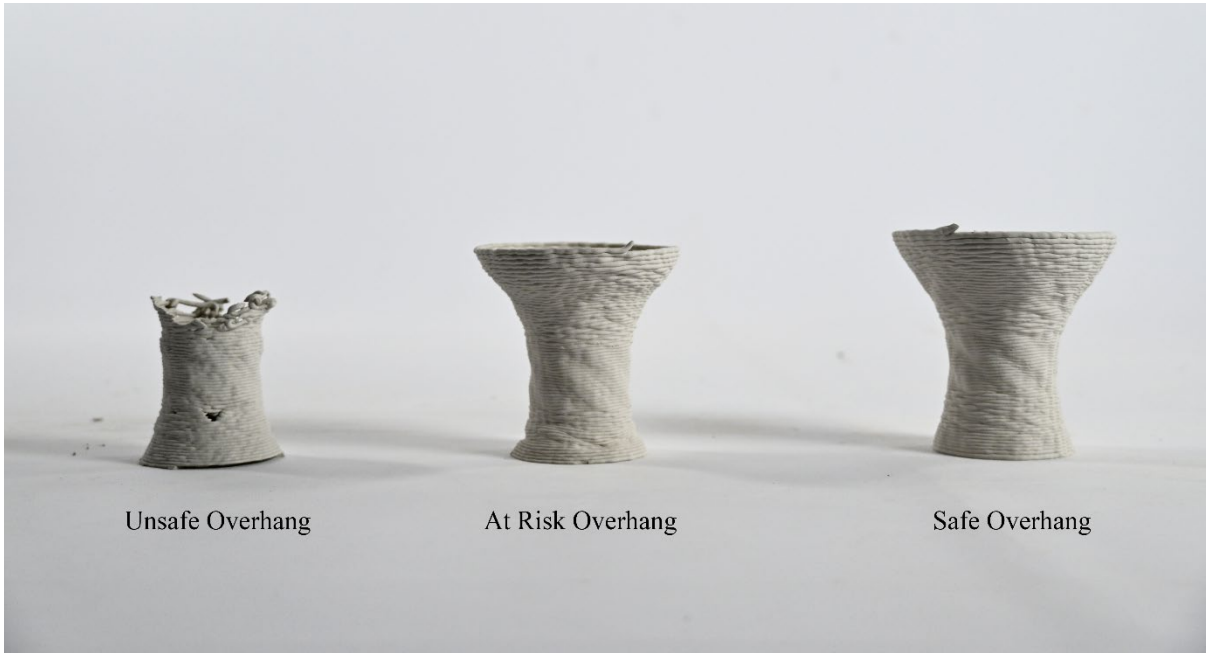


Figure 1.6: Overhang label categories demonstration

To refine this classification and support the visual evaluation, the horizontal interlayer contacting area served as a key visual indicator of overhang success. This area, defined as the horizontal overlap between two adjacent layers, especially in the unsupported segments of the inclined overhangs. It is expected that the ML model will rely on such visual cues to distinguish between *safe, at-risk,* and *unsafe* overhang categories. Based on visual estimation and cross-referencing with physical printing outcomes, the classification is further guided by the ratio between the horizontal interlayer contacting area and the shell thickness. When this ratio falls below 50%, the print is typically categorized as unsafe due to insufficient bonding between layers. Ratios between 50% and 60% indicate a higher risk of deformation, placing the print in the at-risk category. A ratio exceeding 60% generally suggests sufficient interlayer adhesion, corresponding to a safe and stable overhang print.

Step 1.2: Image and data Collection

During the printing process, images from both cameras are recorded and synchronized with relevant G-code parameters, printing speed. Image pairs are corrected for perspective distortion and merged into single composite images. These are then augmented using techniques such as rotation, mirroring, scaling, brightness variation, and normalization (Figure 1.7). Corresponding CSV files store the synchronized visual data and G-code settings.

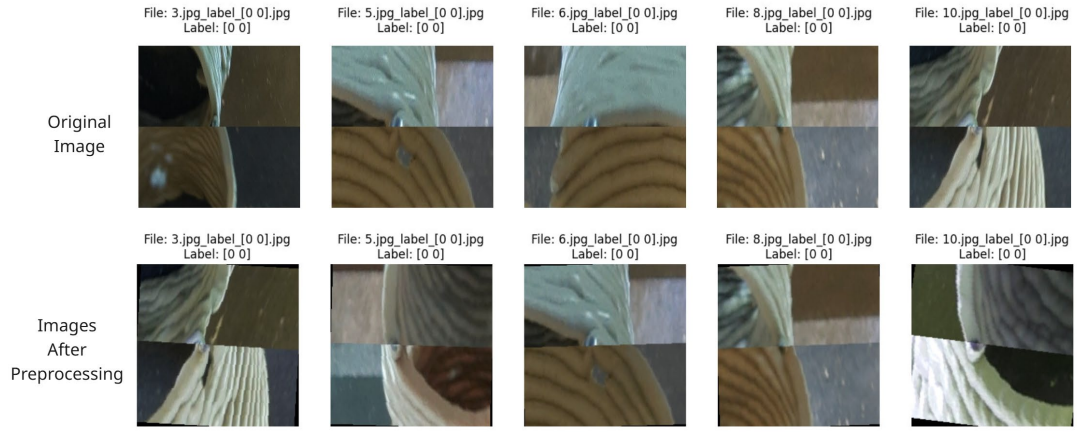


Figure 1.7: Examples of images after augmentation

Step 2: Dataset Labelling

The resulting dataset is labeled according to extrusion quality (three classes: under-extruded, well-extruded, and over-extruded) and overhang safety (three classes: safe, at risk and unsafe). These labels are derived from physical measurements and visual assessments performed in Step 1.1.

Step 3: ML Model Training and Evaluation

Two ML models are trained on the labeled dataset from Step 2 and evaluated to identify the most suitable approach for defect detection in clay-based overhang printing.

The first model is based on the ResNet-56 architecture with integrated residual attention modules

(Wang et al., 2017), previously used in AM for flow control (Brion & Pattinson, 2022). The second model is a hybrid architecture that replaces the front portion of ResNet-56 with a pre-trained DINOv2 ViT-S/14 vision transformer backbone from Meta AI, retaining the original ResNet structure for the final layers. Both models are trained using identical datasets to ensure a fair comparison in performance.

To support multi-objective learning, both models adopt a two-head neural network architecture, in which a shared feature extractor branches into two separate output heads. One head is trained to classify extrusion quality (under-, good-, or over-extrusion), while the other predicts overhang success (safe, at risk, or unsafe). This structure allows the model to jointly optimize predictions for both extrusion and overhang performance based on shared image features.

To enhance ML model's interpretability, Gradient-weighted Class Activation Mapping (Grad-CAM) is applied. This visualization tool focuses the most influential regions in each image used for prediction. It provides transparency in the model's decision-making process and enables users to understand whether the system is focusing on relevant visual cues.

In this context, extrusion quality directly impacts overhang stability. Excessive extrusion can lead to sagging or a bloated overhang shape with inaccurate curvature. While it may increase the horizontal contact area between layers, it also raises the risk of deformation or even collapse due to the added weight and uncontrolled material spread. Conversely, under-extrusion results in insufficient material deposition, weakening interlayer adhesion. In extreme cases, this leads to premature failure, as freshly extruded material can no longer adhere to the preceding layer. Therefore, maintaining an appropriate extrusion rate is critical not only for achieving good surface quality, but also for ensuring the structural integrity of inclined or unsupported overhangs.

The ultimate goal of the ML model is to identify visual features for both shell thickness and horizontal interlayer contacting area, which can then be used to infer whether the print parameters need adjustment. The contacting area is influenced by both shell thickness and the inclined angle of the overhang. To simplify real-time correction during actual printing, this study limits the number of controllable variables by keeping both the layer height and the inclined angle constant. Under the assumption that both the extruder motor speed and the air pressure also remain constant, the shell thickness becomes a function of the extruder nozzle movement speed, which is the RAMS. The relationship is defined as follows:

$$V = Q \cdot \Delta t = h \cdot w \cdot L = h \cdot w \cdot (v \cdot \Delta t)$$

Where:

V = volume of extruded material per unit time

Q = extrusion rate (constant)

Δt = unit printing time

h = layer height (constant)

w = shell thickness

$L = v \cdot \Delta t$ = print length in unit time

v = robotic arm movement speed (RAMS)

Rearranging the equation yields:

$$w = \frac{Q}{h \cdot v}$$

This shows that shell thickness w is inversely proportional to the RAMS v, assuming a fixed extrusion rate Q and layer height h. However, in real-world printing scenarios, the effective volume of deposited material also depends on material intensity, which is a property affected by clay density, viscosity, and moisture content. A denser or more viscous clay mixture may reduce the actual spread or deposition volume even if the extrusion rate remains constant. To account for this conceptually, a material-dependent correction factor ρ_m could be introduced, yielding a more general form:

$$w = \frac{Q \cdot \rho_m}{h \cdot v}$$

Where ρ_m represents the material intensity factor.

Nevertheless, due to the absence of real-time tools for measuring this factor in the current experimental setup, this study adopts a simplified model by assuming $\rho_m = 1$. This allows shell thickness to serve as a reliable visual proxy for deposition performance. Therefore, the ML model is trained to learn the visual relationship between RAMS and shell thickness. This enables the system to adaptively adjust RAMS in real time based solely on image-based feedback, thus maintaining print quality and overhang structural printability.

Step 4: Implementation of a Closed-Loop System
Based on the trained model's predictions, a real-time correction mechanism is developed to dynamically overwrite the RAMS within the URScript.

URScript is the scripting language used by Universal Robots to control robotic arms. It manages movement commands, I/O operations, and basic logic execution. In this study, URScript is used to modify the RAMS in real time based on feedback. Its flexibility allows speed adjustments to be applied directly during the printing process, which is essential for implementing responsive control.

The core logic of this closed-loop control system is built around maintaining an appropriate shell

thickness, which serves as a proxy for ensuring both structural stability and material efficiency. When the ML model detects extrusion anomalies, such as over-extrusion or under-extrusion, it sends adjustment commands to update the RAMS accordingly, allowing the system to correct the print in real time without manual intervention.

To enable such real-time corrections, the system must establish stable and low-latency communication between three key components:

- Vision System: Two Raspberry Pi Camera Module 3 units are responsible for capturing top-view images during printing. These units continuously stream image data to PC via Ethernet.
- ML Model Host (PC): A local PC receives the image data, performs inference using the trained ML model, and determines whether a correction in speed is needed based on detected deviations in shell thickness or interlayer contact.
- Robotic Control Environment (UR5 Teach Pendant): The teach pendant runs the URScript and interfaces directly with the UR5 robotic arm. It receives updated RAMS values from the PC via TCP/IP socket communication.

To support this closed-loop workflow, the communication pipeline is implemented using TCP/IP socket programming. Initially, the PC sends the URScript to the UR5 controller to initiate printing. During the print, images are captured approximately every 10 seconds and sent to the PC. The ML model processes each image on PC, and if necessary, sends updated velocity commands from PC to the robot controller. A wired LAN network connects all components, optimized to reduce latency and packet loss to ensure reliable operation.

Together, this setup forms the foundation of the real-time closed-loop calibration system, allowing visual feedback to directly influence robotic motion in a continuous and responsive manner.

Step 5: Final Validation

Once the closed-loop system is in place, its effectiveness will be validated by printing a new overhang prototype and, later, applying the method to different overhang design or larger-scale geometries. This step will verify whether the adaptive model can generalize to different design configurations and maintain reliable performance under changing conditions.

2. Research By Design And Experimentation

2.1 Experiment setup

This study was conducted using a robotic 3DCP setup composed of a Universal Robots UR5 arm integrated with a WASP LDM extruder (3 mm nozzle diameter). A custom Grasshopper-based slicing workflow was used to generate printing path, which were executed by the UR5 robot via TCP sockets through URScript commands.

A custom slicing tool was developed in Grasshopper, a visual programming environment embedded within Rhinoceros 3D software. The printing path generated by slicing tool was converted into URScript commands and executed on the UR5 via TCP socket communication. TCP sockets provide a reliable protocol for data exchange over local networks, allowing the PC to communicate directly with the UR controller by sending movement commands real time.

For visual monitoring and dataset generation, two Raspberry Pi Camera Module 3 units (max resolution: 4606 × 2592 pixels) were mounted on a lightweight, adjustable frame positioned on either side of the extruder at an angle of approximately 10° from the horizontal. The cameras were connected to a Raspberry Pi 5 and synchronized to capture images at fixed 0.7-second intervals. The PC is used to download logged data for database construction and transfer real-time images during the calibration process. All devices are interconnected via a local network through a router as shown in Figure 2.1. The entire setup, including extruder mount and camera frame system, was custom fabricated by PLA 3DP to ensure adaptability for experimental control (Figure 2.2). The setup design emphasized experimental flexibility, allowing quick adjustments to camera angles, camera height, or mounting positions.

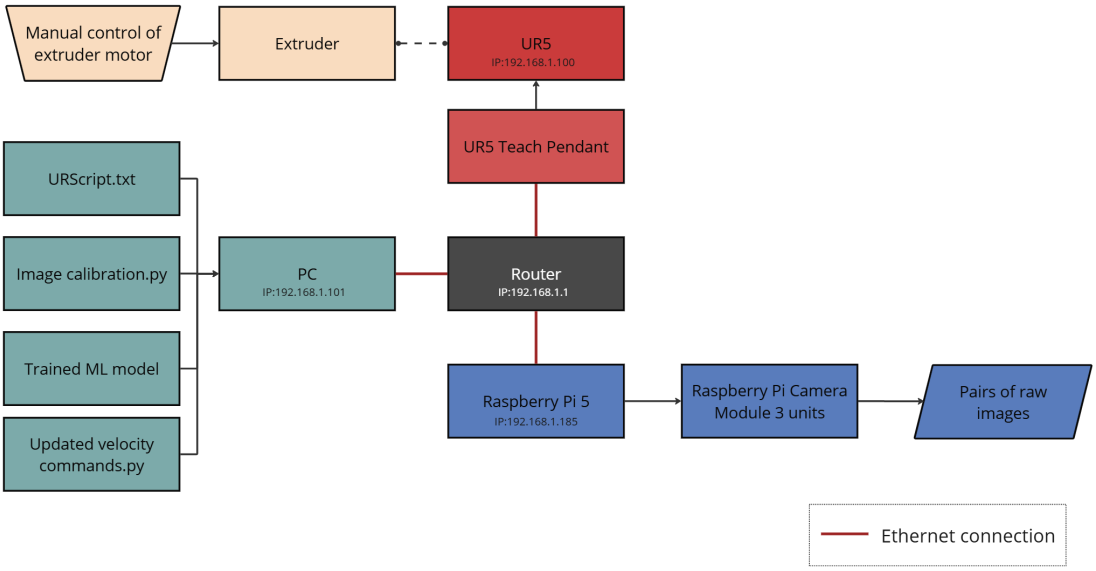


Figure 2.1: Diagram of the hardware connection architecture.

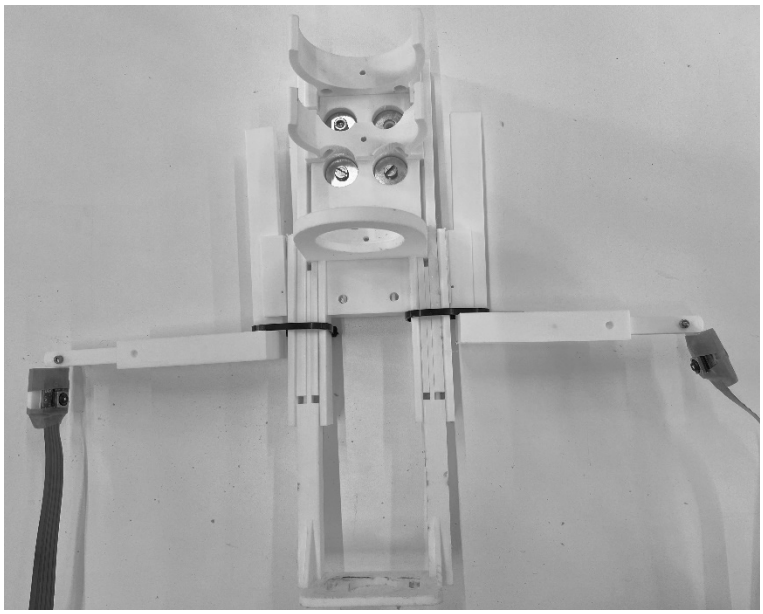


Figure 2.2: extruder mount and camera frame system, custom-fabricated by PLA 3DP

2.1.1 Hardware setup

2.1.1.1 Material

Clay Preparation and Moisture Control

In this study, the preparation of clay was a crucial step directly affecting the stability and quality of the 3DCP process. The clay material used was PRAI 3D (a stoneware formulated for AM), supplied in 5 kg packages with a manufacturer-indicated water content of approximately 22% by weight. Despite this nominal ratio (equivalent to 1.1 kg of water per 3.9 kg of dry clay), the actual moisture level required further adjustment based on its extrudability under the specific conditions of my printing setup.

To improve consistency and flow through the extruder, additional water was incrementally added following guidelines provided in the LAMA Lab in-house manual (LAMA lab is a laboratory for Additive Manufacturing in Architecture within the Faculty of Architecture and the Built Environment in Delft University of Technology). In general, adding 100g of water to the base mixture proved effective in enhancing the material's workability. However, precise moisture control remained a persistent challenge throughout the experiment. Due to the clay’s natural variability and sensitivity to environmental factors (such as ambient humidity and storage duration), it was nearly impossible to replicate identical mixture conditions across different batches (Figure 2.3).

Figure 2.3: clay preparation with manual mixing process

Small deviations in the water ratio had noticeable effects on the printing process (Figure 2.4). When the clay mixture was too dry, it resulted in increased internal resistance, leading to clogging of the extrusion nozzle and occasional motor stalls due to excessive backpressure. Conversely, an overly wet mixture reduced the material's shape stability, causing the printed layers to collapse or deform under their own weight. These failures often required disassembling and cleaning the extruder, manually remixing the material, and re-calibrating the system—operations that were both labor-intensive and time-consuming.

Figure 2.4: clay extrusion changes in one prototype printing due to slight clay mixture changes

Through empirical testing, it was determined that an optimal water-to-clay ratio ranged between 23% and 24% by weight, depending on environmental conditions and time since packaging. This range corresponded to an additional 100–150 grams of water per 5 kg batch (Figure 2.5) and provided a

balance between ease of extrusion and shape stability after deposition. Therefore, maintaining an optimal water-to-clay ratio was identified as a key parameter for achieving reliable and high-quality prints. While efforts were made to keep this ratio as consistent as possible, the inherently variable nature of clay posed ongoing limitations to repeatability and process control.

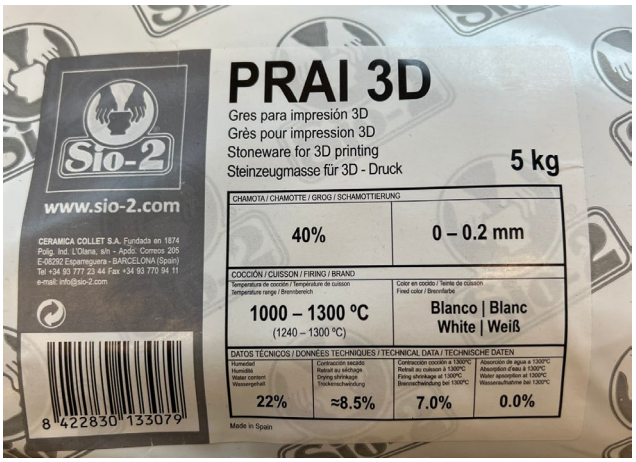


Figure 2.5: Clay mixture package information

2.1.1.2 Robots selection & Connection

A key component of the methodology is the selection and configuration of suitable hardware to support the intended system responsiveness, adaptability, and scalability.

Printer Selection: Delta WASP 40100 LDM printer vs. Robotic Arm 3DP

Initial experiments were conducted using the WASP 40100 Delta printer (Figure 2.6), a three-axis LDM 3D printer, due to its ease of setup and adequate build volume for early-stage prototyping.

However, this system imposed significant limitations during the transition toward real-time parameter control. Specifically, the WASP's closed firmware, which refers to proprietary software that cannot be modified or accessed. This issue prevented direct intervention in print parameters such as printing speed during runtime, making integration with an external feedback system unfeasible.

Additionally, the WASP printer operates with only three degrees of freedom (DOF), allowing movement along fixed X, Y, and Z axes without any rotational control. This restriction limits toolpath customization and dynamic nozzle orientation. As a result, the nozzle cannot be reoriented in response to complex geometries or toolpath requirements, which reduces the system's flexibility and monitoring accuracy, particularly in printing overhangs or curved features.

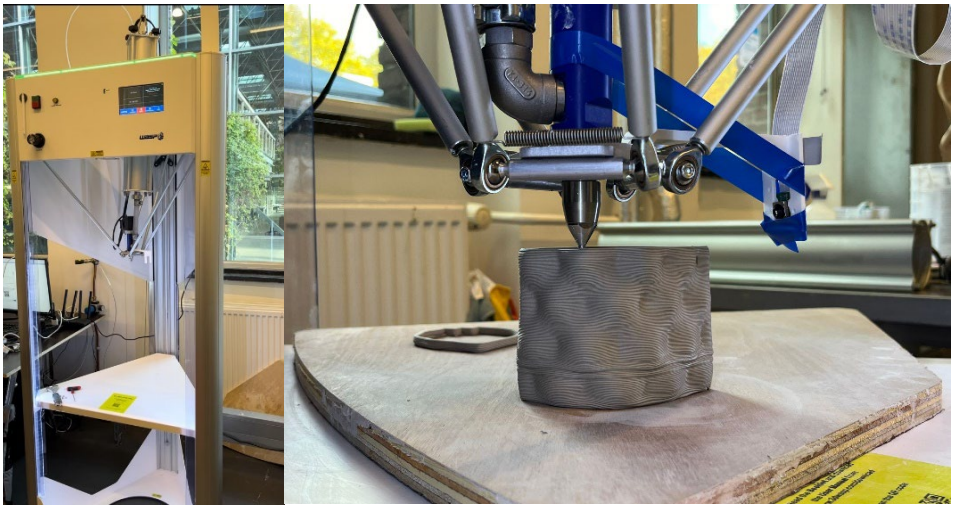


Figure 2.6: WASP 40100 Delta printer (left) & printing set up (right)

To overcome these constraints, the research transitioned to a six-degree-of-freedom UR5 robotic arm paired with the WASP LDM extruder. This setup allows for both translational and rotational motion, providing greater freedom to adapt the toolpath and nozzle orientation dynamically. The ability to control nozzle orientation helps maintain tangency along complex circular paths, which is critical for monitoring curved overhang printing. This strategy will be discussed further in Section 2.1.2.2.

Although the UR5 system does not increase the physical print volume, it offers motion capabilities that closely resemble those of future construction-scale 3DCP systems, such as six-axis gantry robots. This makes the current setup a more scalable and forward-compatible platform for developing closed-loop fabrication workflows.

Robot Selection: COMAU NJ60 vs. UR5

COMAU NJ60:

Initially, the COMAU NJ60-2.2 industrial robot was selected for its robust load capacity and extended reach, suitable for medium-to-large-scale prototypes.

(1) Extruder frame design for COMAU

A custom clay tank frame was designed in four configurations as shown in Figure 2.7, with the final version utilizing a dual-metal plate connection, fabricated through the Maquettehal workshop (The model hall with model-making facilities in the Faculty of Architecture and the Built Environment in Delft University of Technology).

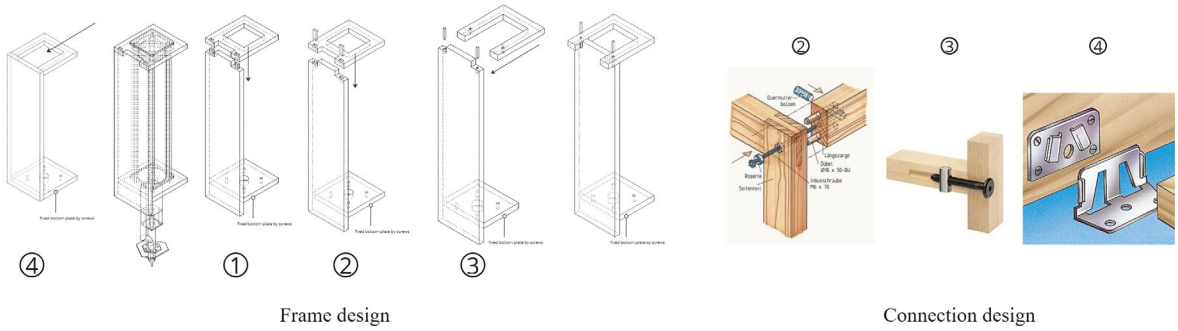


Figure 2.7: Four frame designs and related connection designs for clay extruder

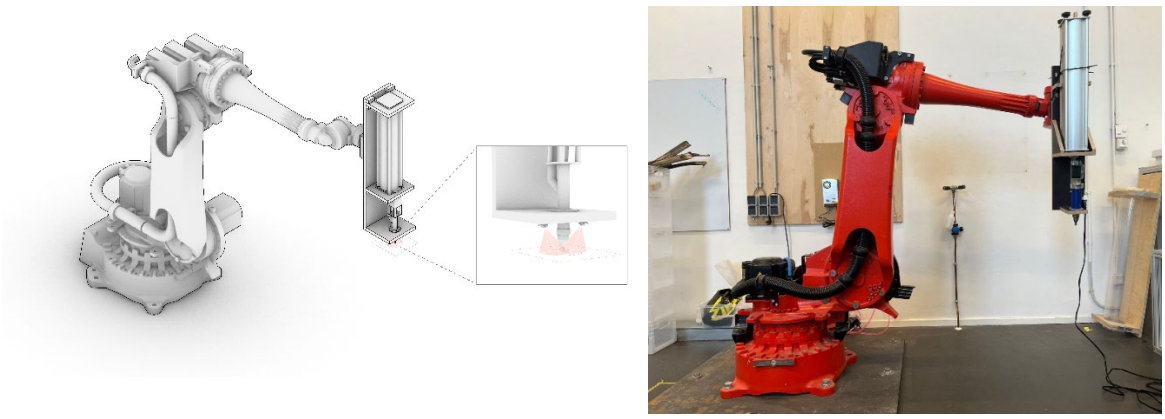


Figure 2.8: Extruder frame design and real-life setup for COMAU

(2) Robot Control for COMAU in 3DP

A custom slicing script built in Grasshopper was used to convert overhang geometries into 6-axis toolpaths. The slicing workflow begins with defining the overhang geometry and layer height as input parameters. The model is then sliced horizontally to generate contour lines along the Z-axis. Using the Termite plug-in, these contour lines are converted into a continuous spiral path to avoid layer seams, meanwhile exporting this spiral path into G-code (Figure 2.9). G-code is a standardized numerical control language used to instruct the 3D printer or robot on precise movements, speeds, and toolpath trajectories during the printing process.

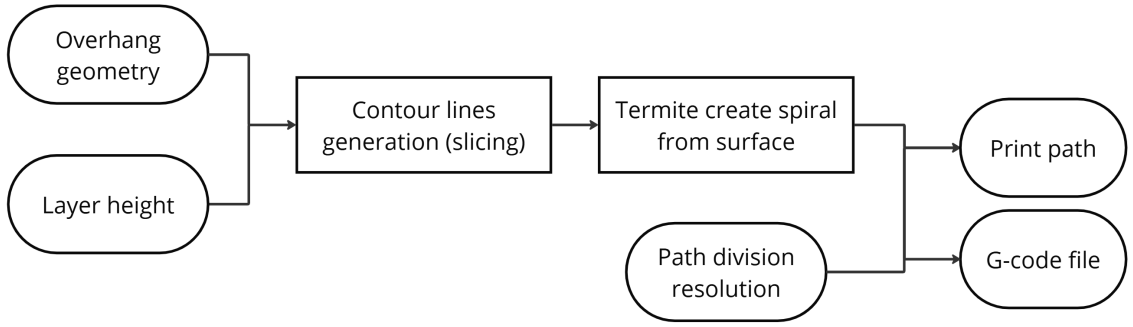


Figure 2.9: A custom slicing script built in Grasshopper to generate G-code

The G-code file was then imported into *RoboDK*, which generated corresponding motion trajectories and converted them into COMAU-compatible PDL2 code for execution by the robot controller (Figure 2.10). PDL2 code, is the native language required by COMAU controllers.

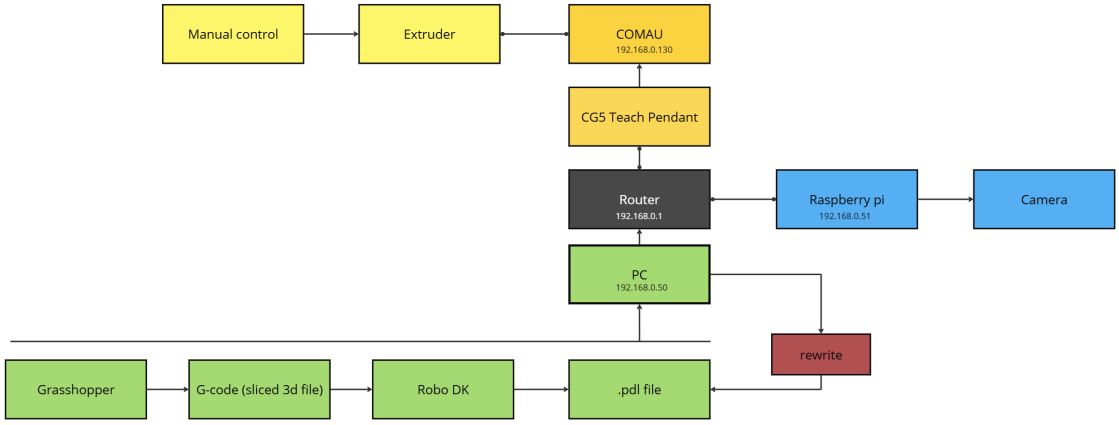


Figure 2.10: Diagram of the proposed connection and data exchange workflow

Attempt to Enable Closed-Loop Real-Time Speed Control

To enable a closed-loop control system with real-time speed adjustment, it was necessary to establish a stable, two-way communication channel between the PC and the COMAU C5G controller. An Ethernet cable connected the user's computer to the internal PC of the COMAU controller, specifically through the ETH2 port located on the Advanced Processing Controller (APC) module. However, establishing this live connection for dynamic speed adjustment proved to be a complex and ultimately unresolvable challenge. Three attempts were made to solve this problem, but all ultimately failed due to the constraints of the available hardware and software.

Plan 1: RoboDK + TCP/IP Communication

The first method involved using RoboDK as the communication interface between the PC and the COMAU controller. The system was configured by setting the robot's IP address (192.168.0.130) within RoboDK and matching it with the settings accessed via the Teach Pendant. The default port

used was 21 (FTP), and the remote path was set as /UD:/usr, using apicomau.py as the driver script. A successful ping confirmed that the network configuration was correctly established as shown in Figure 2.11.

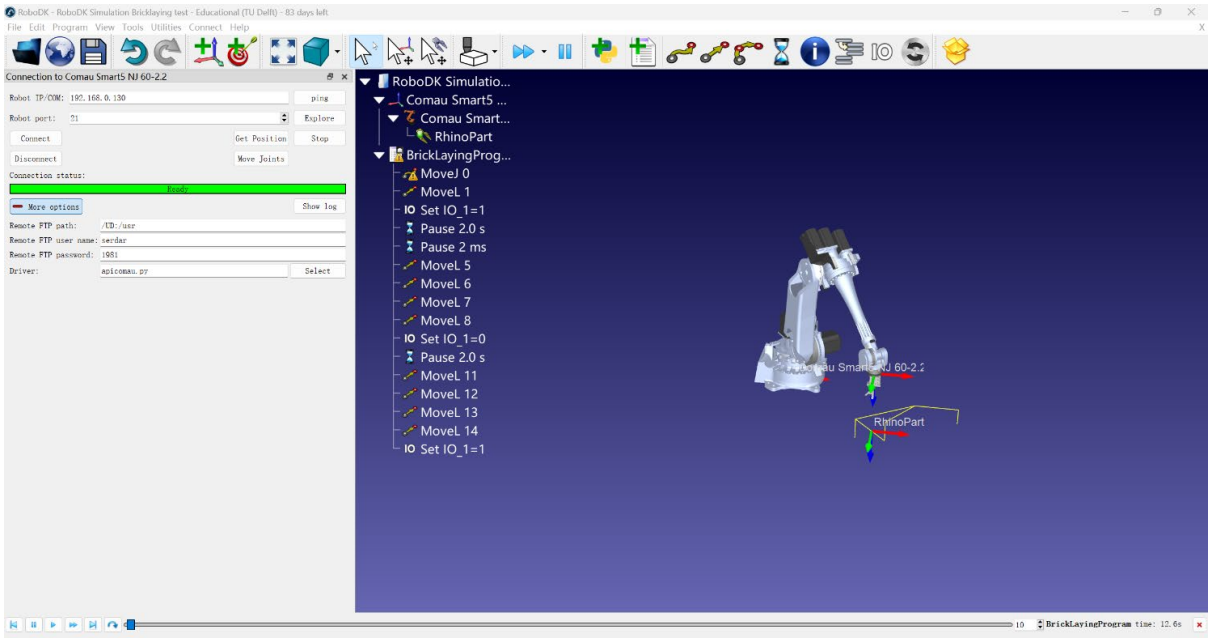


Figure 2.11: the screenshot from Robo DK shows successfully connection via port 21

While FTP-based file transfer allowed for uploading and downloading .PDL and .COD files, it did not support real-time control or feedback. More advanced capabilities—such as live position tracking and motion override—require connection via a TCP/IP control port (typically port 1301 or 5025), which utilizes the DV_CNTRL API for command execution.

Attempts to activate this functionality using RoboDK's provided service driver failed. Running the driver resulted in the following error on the Teach Pendant:

40040/4: RoboDKdriver(101)/; DV_CNTRL operation error 30971

According to RoboDK's technical support and documentation, this error indicates that the controller does not have the required network communication license to use DV_CNTRL. This is a hardware-locked feature that cannot be enabled by the user and requires assistance from COMAU technical representatives. Communication with COMAU support confirmed that the controller did not support real-time DV control in its current configuration. Their internal tests also confirmed that remote control via RoboDK was not available unless the controller was upgraded with a new cabinet (C5G+) and the relevant software license—a process estimated to cost approximately €13,000 for the hardware and an additional €2,500 for the software.

As a result, while FTP-based deployment was successful (i.e., programs could be saved directly to

/UD:/usr), the workflow still required manual loading of the files into RAM (Prog environment) via the Teach Pendant. This manual step can't meet the requirements for real-time, programmatic control necessary for a closed-loop system.

Plan 2: Direct PDL2 Programming with TCP/IP

The second approach explored the use of native PDL2 scripting with TCP/IP communication, bypassing RoboDK entirely. This method involved writing custom routines in PDL2 that leveraged the DV_CNTRL function to open and manage TCP sockets for live data exchange.

The following sample code illustrates the intended communication logic:

```
ROUTINE OpenTcp
VAR
BEGIN
  OPEN FILE LunTcp ('NETT:', 'rw')
  DV_CNTRL(30, (LunTcp), '192.168.0.68', 20001)
END OpenTcp

ROUTINE CloseTcp
VAR
BEGIN
  DV_CNTRL(31, (LunTcp))   -- Disconnect TCP
  CLOSE FILE LunTcp
END CloseTcp
```

Despite the simplicity of the routine, successful execution again depended on the DV_CNTRL function being available. As with Plan 1, this function was disabled on the current C5G cabinet, and activation required additional licensing. Without the ability to open TCP connections natively, this method was also deemed infeasible.

Plan 3: COMAU Open Controller Software

A third option involved using COMAU's proprietary Open Controller software, which is specifically designed to support external real-time control and advanced robot-program interaction. However, after consultation with the local COMAU representative, it was confirmed that the existing cabinet (C5G) was too outdated to support the Open Controller platform. Upgrading to the compatible C5G+ cabinet, along with the purchase of the Open Controller license, would require substantial financial investment, beyond the available budget.

Given the consistent failure of all three methods to enable real-time control on the COMAU system within the available resources and timeline, the project switched to use a **UR5 robotic arm**. Although it has a smaller reach and payload capacity compared to COMAU (which limits the size of printed geometry), the UR5 offers built-in support for remote TCP/IP communication, easier manual control via teach pendant, and real-time script execution compatibility, making it more suitable for research

and iterative testing.

UR5:

(1) Extruder frame design for UR5

A fully customized, lightweight frame was 3D printed in PLA to mount the clay tank and camera system. The frame includes an adjustable sliding hook mechanism that enables flexible tuning of the camera angle and height to accommodate the need for initial adjustment based on nozzle position.

(2) Robot Control for UR5 to 3DCP

To implement robotic 3DCP with real-time process control, a customized workflow (Figure 2.12) was developed based on the Universal Robots UR5 robotic arm.

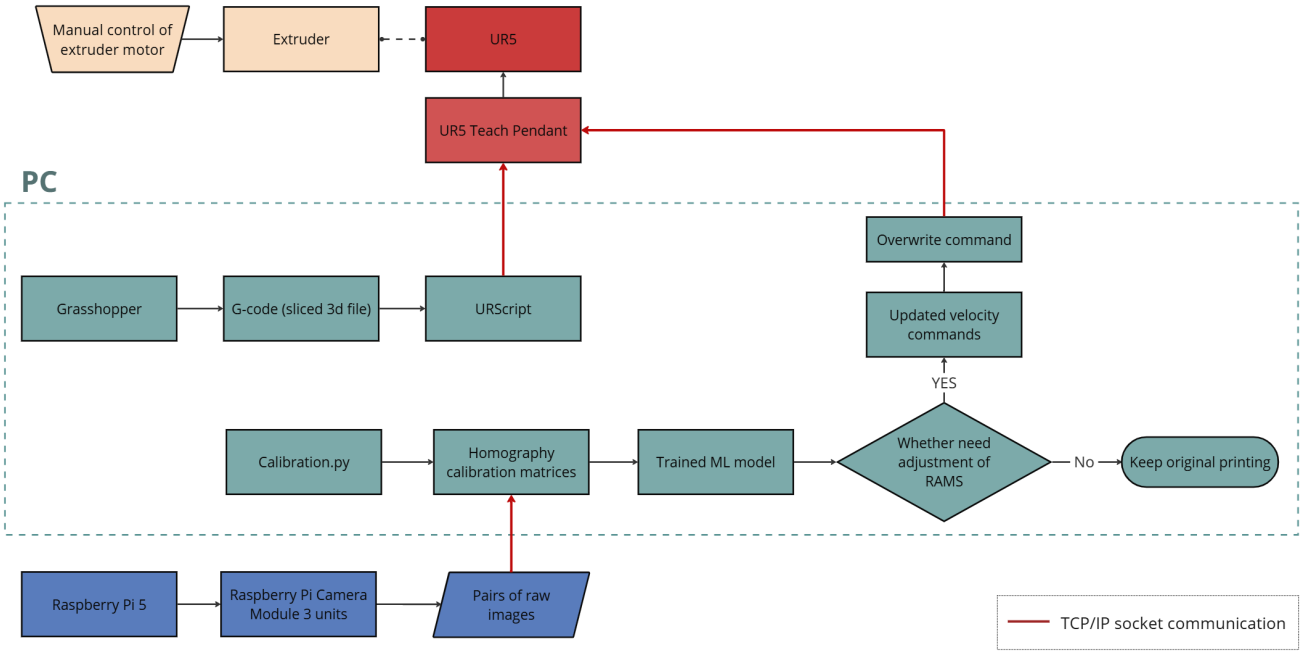


Figure 2.12: Diagram of the real-time calibration system program structure with hardware setting.

The slicing of prototype geometries was performed using a custom Grasshopper script (Figure 2.13). The generated paths are then segmented into motion planes and converted into URScript commands using the Robots Plugin. The final output is a .script file in URScript format, which defines a series of *Move j* instructions that govern the movement trajectory of the UR5 robot during the printing task. To implement real-time control of the robot's speed during execution, three strategies were investigated in combination with TCP/IP socket communication between the PC and the robot controller.

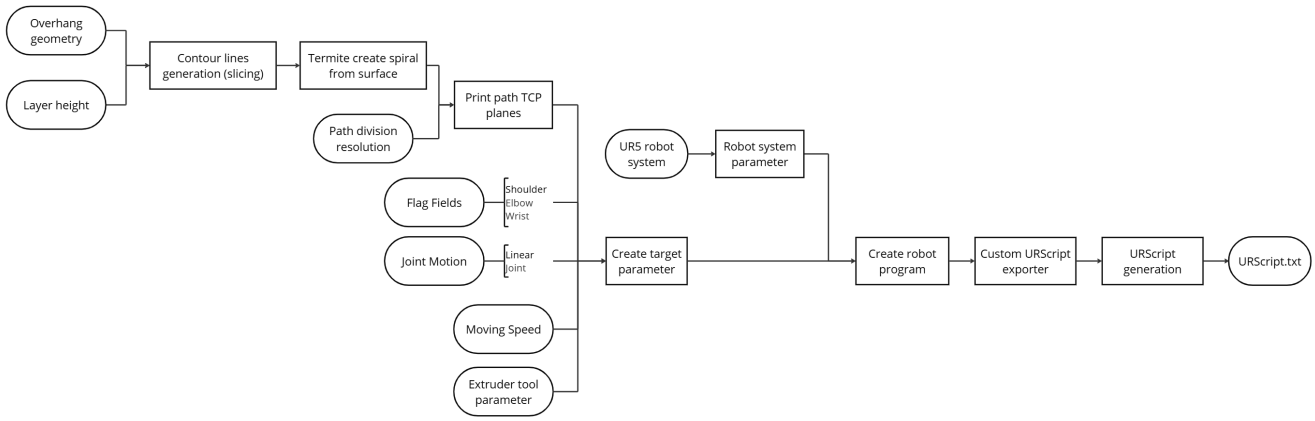


Figure 2.13: Workflow diagram showing custom slicing and URScript generation in Grasshopper

Challenges and Real-Time Control Solutions

(1) Challenge 1: URScript Size Limitation

During initial testing, short programs written in UR Script containing a limited number of *Move j* commands could be successfully transmitted and executed by the UR5. However, once the number of motion commands exceeded approximately 2,500 lines, the robot stopped responding without displaying any error, giving the false impression that the script had been uploaded and initiated correctly.

Two primary causes are suspected: (1) the total script size may exceed the UR5's internal buffer capacity (estimated at ~75 KB), and (2) the number of sequential *Move j* instructions may exceed the permitted execution queue size, especially when streamed over Port 30003.

To address this issue, two script optimization strategies were explored:

Plan 1: Use of a *For Loop*

This approach aimed to reduce the script size by using a *for loop* inside the URScript. Rather than sending each motion command individually, all target positions were stored as a list in a CSV file, and then iterated using a *for loop* structure. This significantly reduced the number of lines in the script while preserving full control of the printing geometry.

However, this method was unsuccessful. The script still failed to upload properly, and the UR5 remained unresponsive. It is suspected that the for loop is processed on the PC (Python) side before the entire script is transmitted to the UR controller, meaning the UR still receives a long script in full, defeating the purpose of the loop-based optimization.

Plan 2: Script Chunking with Delayed Transmission

The successful method involved breaking the script into smaller chunks and transmitting each one with a brief delay in between. This kept the size of each transmitted segment within a few kilobytes

and allowed the robot sufficient time to receive and process each chunk before the next one arrived.

The delay was introduced to give the UR5 controller adequate time to parse and buffer the script content, avoiding packet loss or timeouts caused by overwhelming the system with large scripts all at once. This reduced both network and processing loads by transmitting the script gradually and smoothly, rather than overwhelming the controller with a large file in a single burst.

Importantly, this delay was implemented only on the Python client side and had no impact on URScript execution timing. The robot begins execution only after receiving the complete script along with the program end marker. Even when transmitted in chunks, the UR5 caches the entire script before execution begins.

Therefore, this chunked transmission method, which executed successfully in the experiment and does not affect the continuity of 3D printing, was adopted as the final and effective solution.

(2) Challenge 2: Real-Time Speed Control

A series of methods were tested to achieve real-time adjustment of the robot's movement speed during the printing process. In the field of robotics, particularly with Universal Robots (UR), real-time control of robot velocity has been explored in several domains beyond AM, including surgery, polishing, and human–robot interaction (HRI).

For example, Fontúrbel et al. (2023) used URScript's *speedL* command to send velocity vectors to a UR3e robotic surgical assistant at 125 Hz, allowing smooth real-time TCP speed adaptation based on force feedback. Similarly, Muñoz et al. (2021) demonstrated that *speedJ* (joint-space) and *speedL* (Cartesian-space) commands were effective for real-time control in an endoscopic surgery assistant. They streamed velocity updates every 8 ms to achieve adaptive motion.

In the HRI domain, Van Oosterwyck (2018) adjusted a UR10's speed in real time via the speed slider rather than resending *Movej* commands, which could cause abrupt motion. By programmatically setting the speed slider fraction between 0 and 1, smooth transitions were achieved without interrupting the trajectory. In a polishing task on a UR3, Pérez-Ubeda et al. (2020) used velocity-mode control with periodic updates to maintain stable force contact, showing how speed can be adapted mid-process for continuous material processing.

These examples suggest that frequent velocity updates, either by direct *speedL/speedJ* commands or by adjusting the global speed scaling, are effective approaches for maintaining smooth and responsive robot motion in real time, a principle this study adopts.

Plan 1: Speed Control via Grasshopper-Robots Plugin

In this approach, the printing script was generated and executed using the *Robots Plugin* within Grasshopper. Speed values were defined as variables within the toolpath generation, and could

theoretically be adjusted in real time during printing. However, due to latency in command transmission from the PC to the UR controller, each speed update caused the robot to revert to a previous waypoint and restart the movement with the new speed. This unexpected behavior resulted in overlapping motion and damaged printed layers, making this method unsuitable for continuous printing.

Plan 2: Python Script + Socket Communication (speed_slider_set_override)

To avoid dependency on Grasshopper's plugin, a new method was implemented using Python scripting. The entire *Move j* toolpath, initially generated in Grasshopper, was exported into a .script file. This URScript file was loaded and executed by the UR controller via TCP port 30003. In parallel, a Python script on the PC was connected to the robot via dashboard port 30002, from which real-time speed adjustment commands were sent using the speed_slider_set_override() function.

Although the script executed successfully and the PC could send and receive basic commands (e.g., stop), the robot did not respond to the dynamic speed change signals. This indicated that while the connection was established, the command interface lacked the authority or synchronization to update the controller's internal state mid-execution.

Plan 3: Real-Time Script Segment Streaming

A third method attempted to bypass static script uploading by streaming smaller script segments to the robot in real time, thereby allowing updated speed values to be embedded directly before each motion instruction. However, this approach conflicted with UR's internal motion planning mechanism, which requires pre-loading a complete trajectory for smooth execution. As a result, the robot exhibited jerky, intermittent motion, pausing between each new command batch, which is incompatible with the smooth, continuous motion required for AM.

Plan 2+: Python Script + Socket Communication via runSlow()

As a final and successful method, the system retained the basic structure of Plan 2 but modified the speed control mechanism. Instead of using the dashboard speed_slider_set_override() function, the control was shifted to the Teach Pendant interface, where the Python script sent the runSlow(speed) function via TCP port 30002. This function directly controls the speed slider on the pendant, allowing real-time adjustments to the robot's velocity during program execution.

This method proved effective. When the user changes the speed percentage in the Python script running on the PC, the robot controller's speed slider reflects the updated value instantaneously, and the UR5 adjusts its motion speed accordingly without interrupting the ongoing print. This solution was ultimately adopted as the core mechanism for implementing the closed-loop real-time calibration system in the UR5-based clay printing workflow.

Challenge 3: Joint motion Mode Selection for Consistent TCP Speed & circular path

In robotic 3DCP, the selection of Joint motion mode in URScript plays a critical role in maintaining consistent tool center point (TCP) speed and path fidelity, These are both essential for ensuring high-quality extrusion and avoiding defects such as over-extrusion or under-extrusion. Unlike other robotic tasks, 3DCP requires smooth, uninterrupted motion at a constant speed to guarantee even material deposition and layer adhesion. Therefore, evaluating different UR motion modes was a key step in developing a reliable printing workflow. URScript supports several motion types, including *Movej* (joint motion), *Movel* (linear Cartesian motion), *Movep* (constant-speed Cartesian process motion), and *Movec* (circular Cartesian motion), each with distinct characteristics.

Plan 1: *Movej* (Joint Motion)
The *Movej* command moves the robot by interpolating joint angles. It defines robot poses via six joint positions and moves the robot through a nonlinear path between waypoints.

Syntax:

movej(q, a, v, r=0)

q: list of 6 joint angles [rad]

a: joint acceleration [rad/s²]

v: joint speed [rad/s]

r: blending radius [m] (optional, for smoothing)

It is the fastest and most commonly used motion type, suitable for free-space movement. While it does not guarantee a constant TCP velocity, *Movej* was adopted in this project due to its robustness and execution reliability.

To improve path smoothness, the project used a "*Move Arch*" strategy, applying *Movej* with blend radii between waypoints.

Syntax:

movej(joints, a, v, r=blend)

This allowed the robot to create smoothed transitions between points, emulating arc-like curves without switching to Cartesian motion modes. Among all tested methods, this plan consistently produced successful prints.

Importantly, *Movej* does not maintain a constant TCP speed by default, with slight speed change between certain segments. However, it was considered the most reliable option under current time and system constraints in this project.

Plan 2: M*ovep* (Process Motion)

The *Movep* command is designed for applications where the robot must maintain a fixed TCP speed through a series of Cartesian waypoints, which is ideal for 3DP.

Syntax:

movep(pose_to, a, v, r)

pose_to: target TCP pose [x, y, z, rx, ry, rz]

a: Cartesian acceleration [m/s²]

v: Cartesian speed [m/s]

r: blend radius [m] for smoothing transitions

However, *movep* requires Cartesian pose inputs (not joint angles), so a forward kinematics model using UR5's Denavit–Hartenberg (DH) parameters was implemented in Python to convert joint angle CSV files into [x, y, z, rx, ry, rz] poses.

Despite the technical setup, this plan failed during execution. The converted paths often rotated out of the desired print plane or collided with the print surface, indicating either misalignment of coordinate frames or incorrect TCP offset transformations. These issues prevented successful deployment of *Movep* under project time constraints.

Plan 3: *Movec* (Circular Motion)

Movec enables circular path interpolation between two TCP poses (via and to). It's ideal for drawing arcs or curved paths with a constant TCP speed.

Syntax:

movec(pose_via, pose_to, a, v, r)

pose_via: intermediate TCP pose [x, y, z, rx, ry, rz]

pose_to: target TCP pose [x, y, z, rx, ry, rz]

a: Cartesian acceleration [m/s²]

v: Cartesian speed [m/s]

r: blend radius [m] for smooth transition

Similar to *Movep*, it requires Cartesian input and relies on accurate spatial alignment. A Python

function was developed to pair waypoints and output *movec*(via, to) commands with specified speed and blend radius.

However, testing revealed unexpected behavior: the robot's tool rotated incorrectly or followed circular paths in the wrong orientation. This likely stemmed from inconsistent pose transformations or extruder frame offsets, making the method impractical within the available timeframe.

Conclusion and Adopted Strategy
After evaluating all options, *Movej* with blend radius (*Move Arch*) was selected for this project. It offered sufficient control and smoothness while avoiding the Cartesian conversion errors encountered with *Movep* or *Movec*. However, the disadvantage of slight change in TCP speed still remains.

If time permits in future work, future development should revisit *Movep* or *Movec* with a refined kinematics pipeline, allowing the robot to follow accurate Cartesian paths with true constant velocity, which remains the ideal approach for high-fidelity AM.

2.1.1.3 Camera setup

Two Raspberry Pi Camera Module 3 units were simultaneously connected to a Raspberry Pi 5 and synchronized to capture images at the same timestamp. Additionally, data of the robotic arm's movement speed during image capture was recorded, ensuring accurate matching between image timestamps and movement data.

The cameras were mounted on a custom-designed PLA arm fabricated using FDM 3DP. Each camera was angled from above to view the printing area on either side of the nozzle tip (Figure 2.14), enabling coverage of both left and right sides of the printing process.

This top-down, oblique mounting configuration was intentionally selected to ensure that at least one camera maintains visibility of the freshly printed structure regardless of the nozzle's direction of movement. Specifically, to monitor interlayer adhesion and overhang characteristics, a top-down view is essential. Side-view configurations would result in visual blocking when the nozzle moves behind the printed part, obstructing critical regions of interest during extrusion.

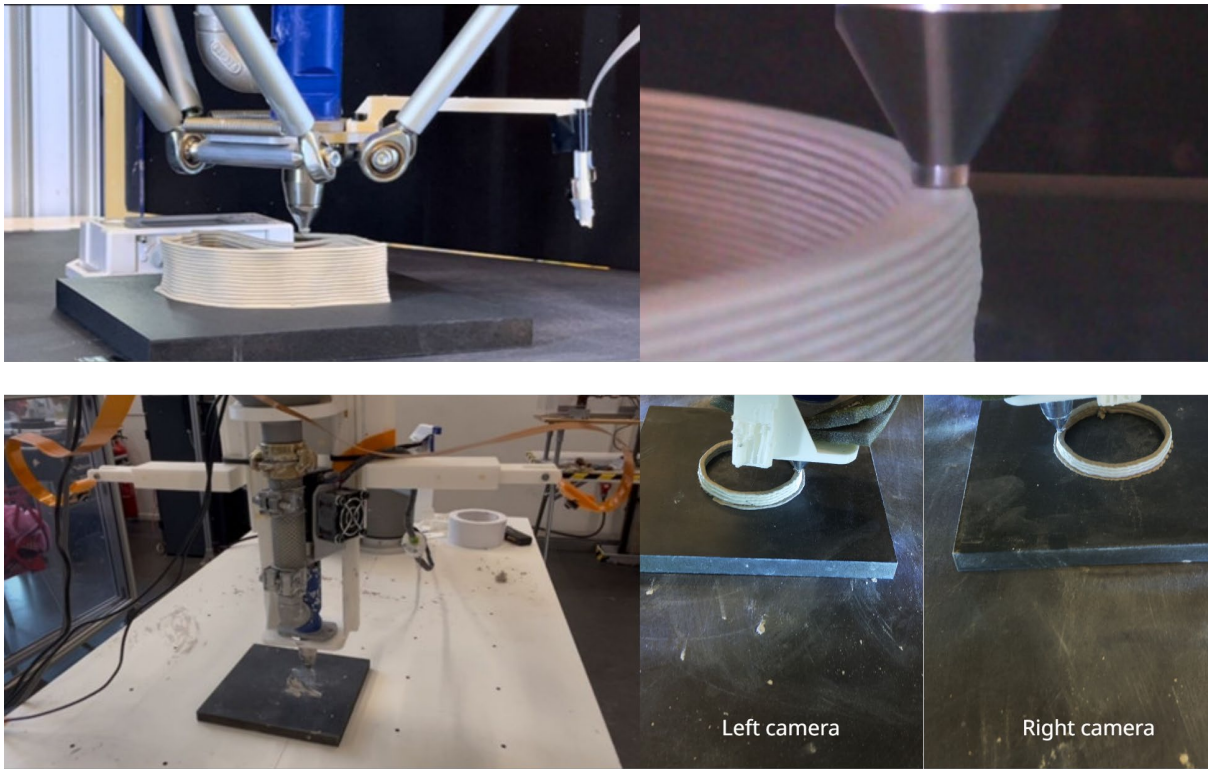


Figure 2.14: A comparison between the experimental setups and the images captured during the pre-research (up) and current research (down), showing side and top views respectively.

2.1.2 Software setup

2.1.2.1 Camera Calibration and Image Rectification

To ensure the consistency and generalizability of the dataset used for training ML models, the two images captured simultaneously by the raspberry pi camera system were geometrically transformed and merged into a unified overhead view with corrected perspective distortion. This rectification process standardizes all collected data into a consistent spatial layout, compensating for the original tilt angles and mounting positions of the cameras to be approximate a top-down orthographic view (Figure 2.15). It minimizes perspective effects that could otherwise bias feature detection and geometric interpretation.

Figure 2.15: explanation of top-down view rectification

Such transformation is essential for reproducibility. If raw perspective images were used directly, any future replication of the system would require the cameras to be placed in exactly the same positions and orientations to maintain visual consistency across datasets, which is hard to achieve. By contrast, applying homography-based rectification ensures that new users can reproduce the dataset structure by simply applying the same calibration parameters, without needing to replicate the exact physical setup. This approach significantly enhances the portability and reusability of the trained ML model across slightly different position and angle of camera setup.

To achieve pixel-level alignment of dual-camera images captured from top and bottom views, a custom calibration procedure was implemented using OpenCV library(*OpenCV*). A physical checkerboard pattern with known geometry (7×3 inner corners, with each square measuring 50 mm) was used as the calibration target. Calibration images were taken simultaneously from both Raspberry Pi cameras.

Chessboard Corners finding and image Warping

To geometrically align images captured from different viewpoints, a homography-based calibration was conducted. First, subpixel-accurate checkerboard corners were detected using OpenCV's *findChessboardCorners* and *cornerSubPix*. For each camera view, a homography matrix *H* was estimated via *cv2.getPerspectiveTransform*, mapping the detected 2D image points to an ideal rectified plane. This homography defines a projective transformation between two planes, expressed as:

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{H}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

To ensure that the rectified image occupied a valid region in the output coordinate system, the warped boundary was computed and shifted using a translation matrix. The combined homography was then applied to the full image using *cv2.warpPerspective*, followed by cropping based on the warped checkerboard region. This calibration step yields a consistent transformation matrix and scale factor (Figure 2.16), which are later reused to process new images without checkerboards by warping, scaling, aligning, and merging them into a unified top-down view.

Image Cropping

To isolate the region containing only the checkerboard, a second perspective transform was applied based on extended corner points, estimated by shifting the outermost corners outward by one grid width. This crop ensured that only the valid, rectified region was retained.

Image Aligning

The grid size in pixels was then computed for both views, and the left image was scaled accordingly to match the grid size of the right image. The two cropped and scaled images were finally merged into a single top-down composite by aligning a reference point (nozzle tip point coordinate) between the two views.

Save Calibration parameters

This calibration procedure yields not only rectified and aligned images but also reusable transformation parameters (.npz file), including homography matrices, crop regions, and scale factors. These parameters were subsequently applied to new images without checkerboards, enabling consistent alignment across the dataset.

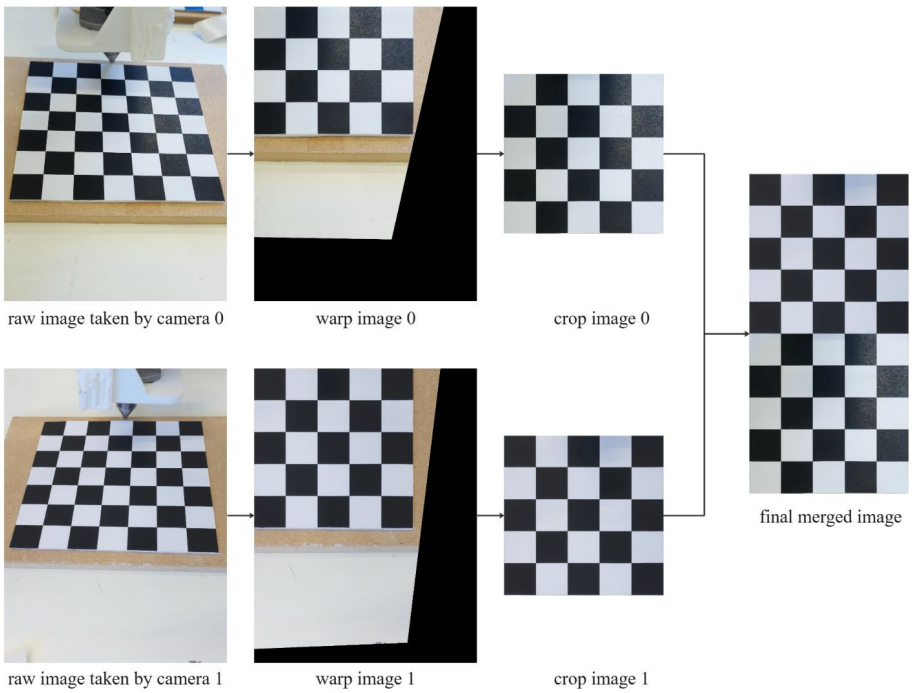


Figure 2.16: Demonstration of the steps for stitching images

In this study, two calibration target patterns and OpenCV detection methods were evaluated to determine the most robust approach under the specific imaging conditions produced by the Raspberry Pi Camera Module 3.

Grid Pattern Selection and Detection Method Evaluation

Two calibration target types were tested: a standard checkerboard grid (square corners) and a circular dot grid. Correspondingly, two OpenCV functions*: cv2.findChessboardCorners* and *cv2.HoughCircles*, were compared. In initial trials, the findChessboardCorners function failed to consistently detect all corners in images captured by the Raspberry Pi camera. In many cases, only a partial grid (e.g., 3×3) could be detected, while full pattern detection was only successful in images taken with a mobile phone camera under similar lighting conditions. Conversely, when using a circular dot grid, the HoughCircles function reliably detected all circle centers in Raspberry Pi images.

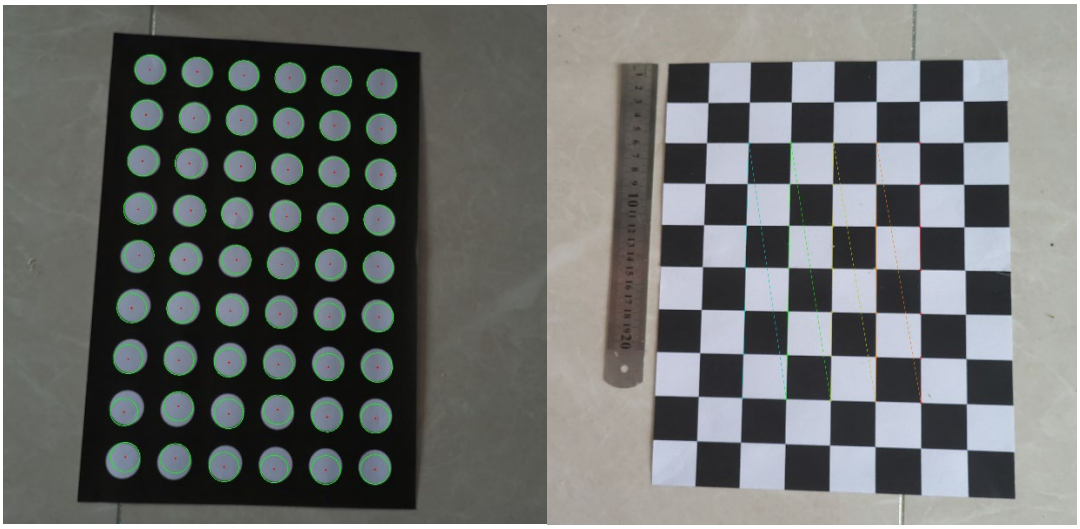


Figure 2.17: comparison of corner detection accuracy between *cv2.HoughCircles*(left) and *cv2.findChessboardCorners* (right) with detection color circles/ lines generated by code

This discrepancy likely stems from the inherent robustness of circular feature detection, which depends on approximate parameters such as circle diameter and spacing. In contrast, checkerboard detection relies on precise localization of corner intersections and the regularity of grid geometry. While the circle grid allowed for more consistent detection, the geometrical accuracy of the detected points was relatively low, leading to unacceptable distortions in the rectified images. Despite incomplete detection in some cases, the checkerboard provided higher positional accuracy for detected points, which is essential for accurate warping. Therefore, *cv2.findChessboardCorners* was ultimately selected for the calibration and rectification pipeline.

Image Quality Comparison and Hardware Limitations

Substantial variation was observed between images taken by the Raspberry Pi camera (right image in Figure 2.18) and those captured by a smartphone (left image in Figure 2.18). Smartphone images consistently produced better results due to their superior sharpness, higher contrast between black and white squares, reduced glare, and more even lighting. These advantages stem from the smartphone's optical and image processing hardware: high-quality lenses with anti-reflective coatings, advanced ISP modules, dynamic exposure control, and high-dynamic-range (HDR) techniques. By contrast, the Raspberry Pi Camera Module 3 is a low-cost component with limited lens coating, basic image signal processing, and a global exposure system. As a result, its images frequently suffer from glare (particularly in the upper portion of the checkerboard), shallow depth of field, and image deformation due to skewed perspectives, especially when the camera is placed close to the checkerboard.

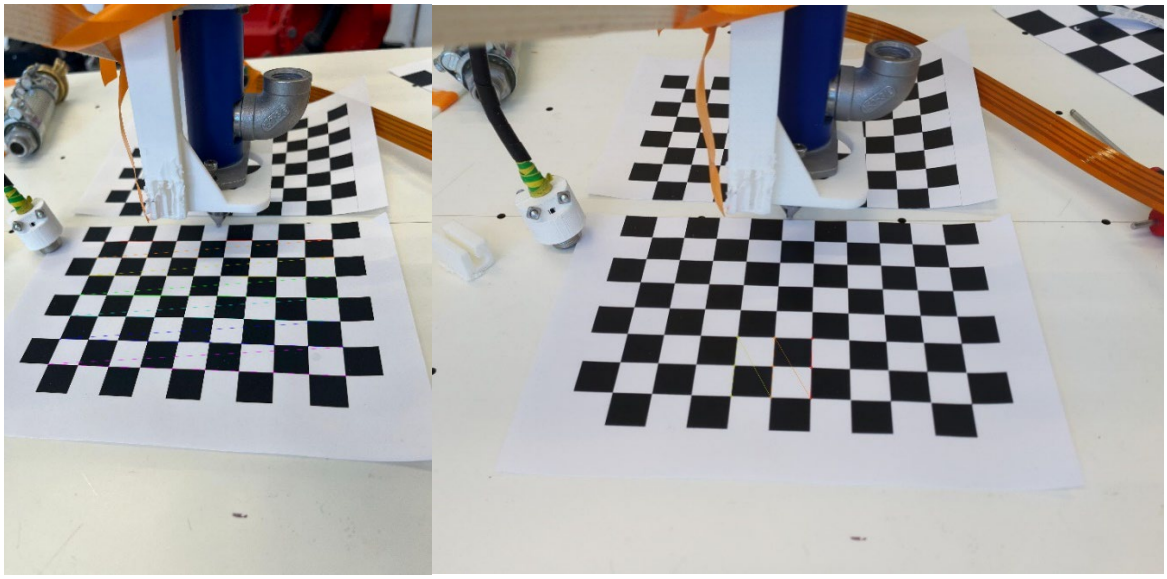


Figure 2.18: corner detection test images taken by iphone(left) and by raspberry pi(right) with detection color lines generated by code

Attempts to compensate for these issues included image sharpening, use of CLAHE (Contrast Limited Adaptive Histogram Equalization), lighting adjustments, and autofocus/focal length tuning (e.g., using AfModeEnum.Continuous and LensPosition). However, these modifications provided only marginal improvements, and in some cases (e.g., over-sharpening), they caused the sub-pixel refinement step (cornerSubPix) in the OpenCV pipeline to fail entirely. Additionally, hardware constraints limited further enhancements, as the Raspberry Pi camera's native resolution (4608 × 2592 pixels) had already been set to its maximum.

Rectification Strategy and Experimental Adjustments

Despite hardware limitations, adjustments to the experimental setup improved detection consistency. It was observed that portrait-oriented images provided better detection results than landscape ones, due to the reduction of extreme perspective distortion at the image bottom, where deformation is most severe. Moreover, reducing the checkerboard's cell size and positioning the pattern in the center-upper region of the frame further mitigated issues with lens distortion and field-of-view compression.

By carefully controlling the camera angle to minimize tilt and keeping the nozzle tip visible, it was possible to ensure detection of at least a 3×3 corner grid. While using only a small portion of the checkerboard may reduce rectification accuracy—since the perspective transform is calculated based on a limited number of reference points—our experiments showed that 4×4 corner detection was sufficient to achieve acceptable top-down warping (see Figure 2.19).

Figure 2.19: Corner point visualization and debugging on the image taken by raspberry pi

Implications for Multi-Camera Image Stitching and ML Integration

The stereo camera system introduces additional complexity for ML applications. Images captured from the left and right cameras must be accurately aligned and stitched to form a single top-down view. However, any inconsistency in rectification, especially at the seam between the two views, may affect the quality of the visual features used by ML models. This becomes critical if the stitched image is adopted as the input for a trained model, as the performance is highly dependent on maintaining consistency in camera positioning and transformation parameters.

Thus, while the current stitched image approach provides good spatial coverage and nozzle visibility, it introduces a potential domain shift problem if the system is replicated under different physical or

geometric configurations. This may affect model generalization and suggests that additional robustness measures, such as domain adaptation techniques or spatially aware data augmentation, may be required for future deployments.

Vibration-Induced Image Misalignment and Merging Errors

During image preprocessing, motor-induced vibrations were identified as a significant factor affecting image clarity and consistency (Figure 2.20), which in turn influenced the accuracy of image merging from the two Raspberry Pi cameras. Although the cameras were calibrated correctly using a flat checkerboard target, periodic vibration, particularly during nozzle movement, caused minor shifts in camera position and focus.

Figure 2.20: Example of a blurred image caused by motor vibration

Initial attempts to stabilize the camera system included several strategies:

1. Mount relocation: Cameras were relocated from the clay tank (PLA frame) to the robot arm to reduce direct vibration from the extruder motor.

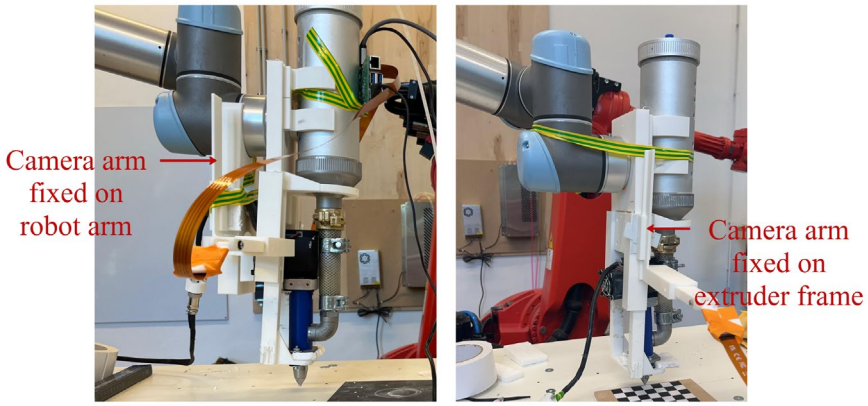


Figure 2.21: Demonstration of different camera arm mounting configurations.

2. Foam insulation: Foam padding was added between the motor and its frame and between camera joints to dampen mechanical resonance.

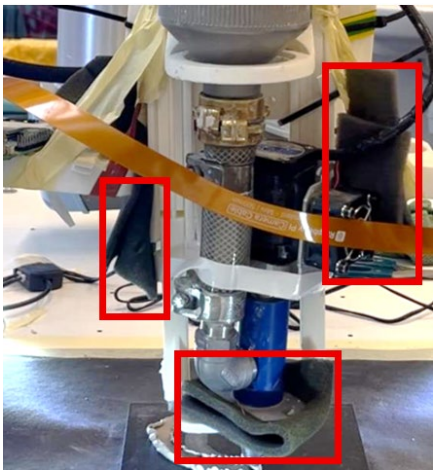


Figure 2.22: Foam padding position

3. Counterweights: Additional mass was added to both camera arms to absorb vibration.

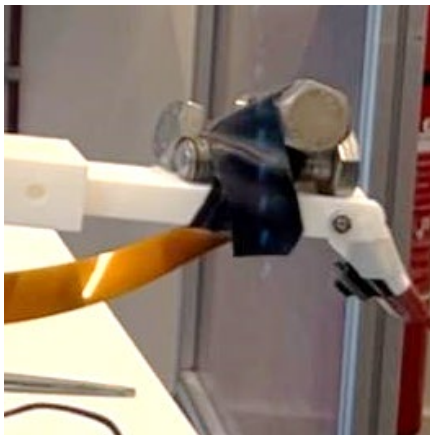
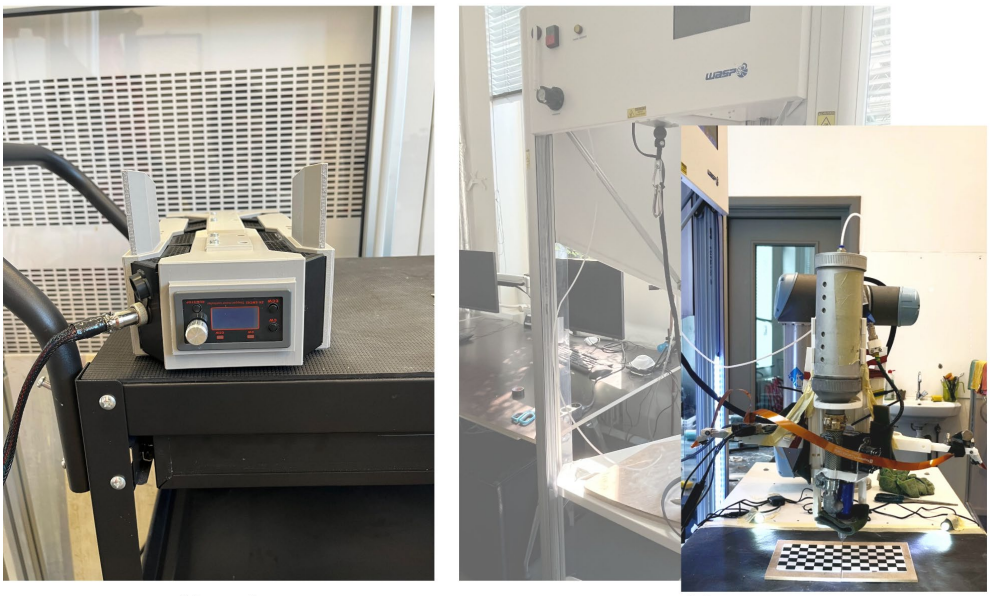


Figure 2.23: Additional mass position

4. Motor tuning: The extruder motor's rotational speed was carefully adjusted to minimize vibration; interestingly, some higher speeds led to reduced vibration depending on resonance behavior.
5. Motor replacement: Finally, the original self-built motor was replaced with a commercial WASP extruder motor, which provided significantly better mechanical stability and reduced vibration.



self-made motor connect with WASP motor

Figure 2.24: Motor change comparison

While solutions 1–4 contributed to partial improvement, it was the implementation of solution 5 that led to a major reduction in image distortion. However, some residual periodic vibration remained, occasionally leading to small vertical misalignments between the two camera views. This resulted in occasional inconsistencies in the merged images.

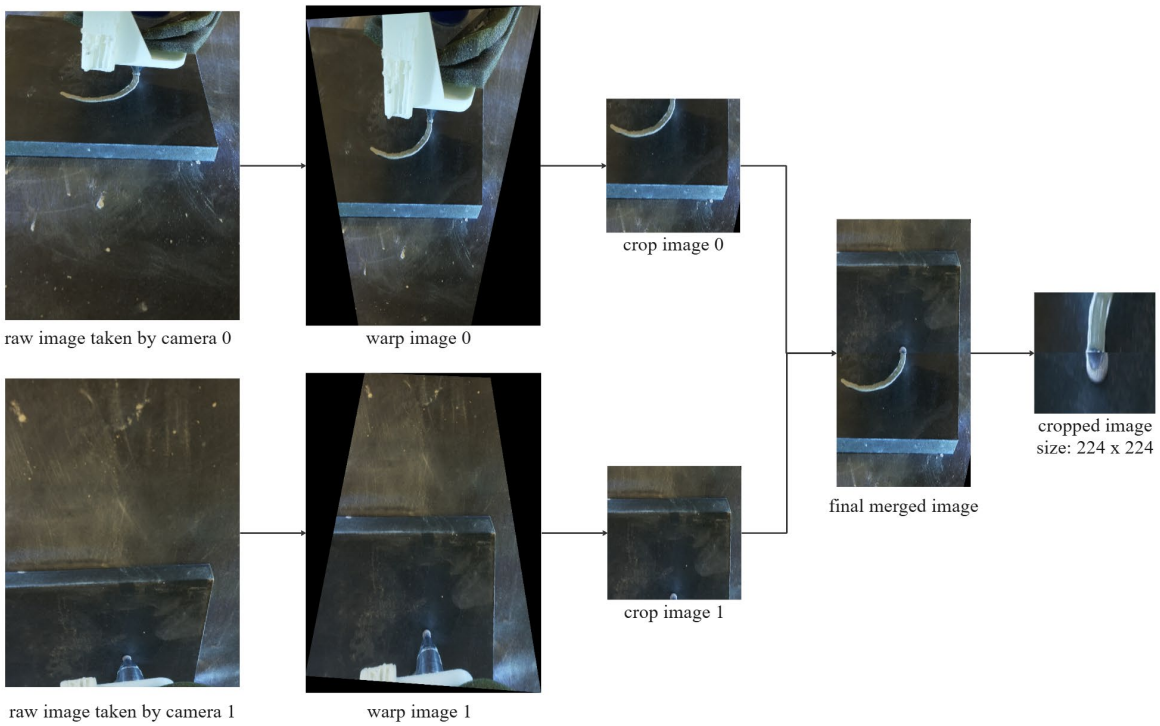


Figure 2.25: Good example of calibration parameters applied to dataset image

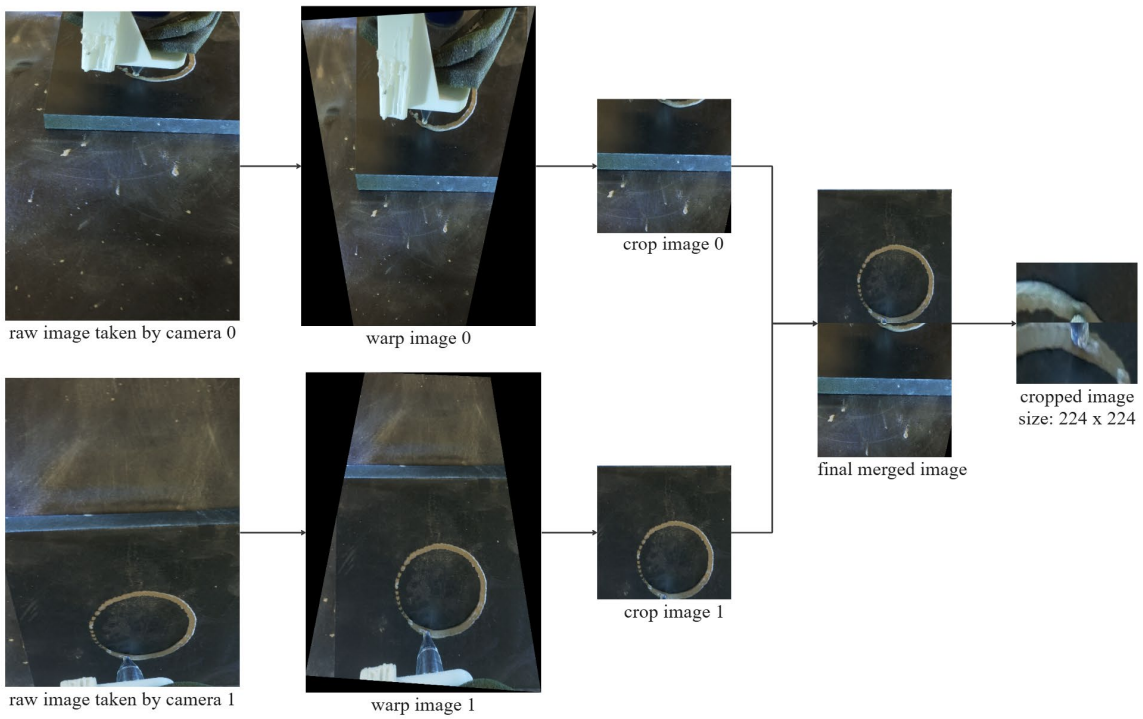


Figure 2.26: Bad example of calibration parameters applied to dataset image

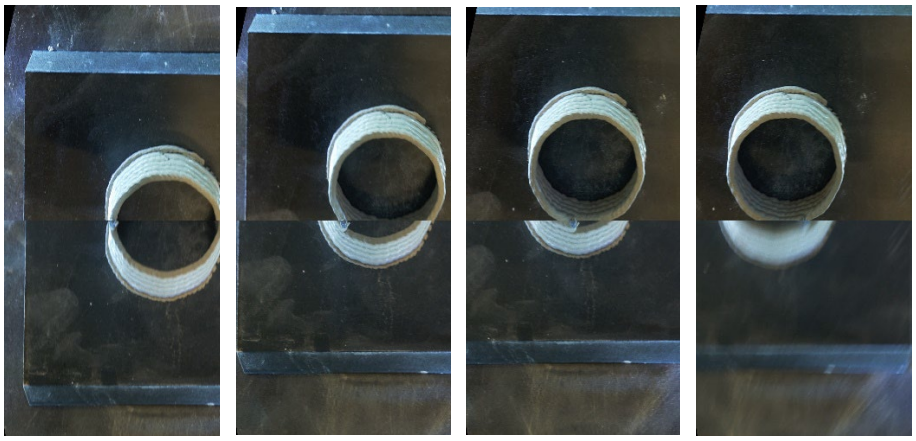


Figure 2.27: Example of a series of misalignment caused by vertical position change

Figure 2.25 shows a good example of successful image rectification, where the warped and cropped images from both cameras align correctly to produce a unified top-down view. In contrast, Figure 2.26 shows a bad example where overlapping content appears in the merged output, suggesting misalignment between the two input frames. This typically occurred when the extruder vibrated during image capture, slightly altering the apparent vertical position of the nozzle or print geometry (Figure 2.27). Although rare, these misalignments compromise dataset consistency and can affect ML training if not handled.

Camera Reassembly Sensitivity

One recurring issue in the experimental workflow is that each time the setup is reassembled, the position of the cameras may shift slightly. Such variations, though minor, can affect the accuracy of the image-stitching process, potentially degrading the quality of the merged top-view image used for monitoring and model inference. Therefore, camera calibration must be repeated prior to each print session following reassembly, using the procedure described in Section 2.1.2.1.

In future iterations, a more precise method could be implemented by calibrating the camera-to-nozzle tip distance using laser range sensors or similar depth-measuring tools. This spatial data, in combination with the known TCP coordinates, could be incorporated as auxiliary input to the machine learning model. By doing so, the model would be able to compensate for geometric discrepancies caused by reassembly or mechanical variation, thereby improving the model's robustness and reducing the reliance on manual calibration or alignment procedures.

2.1.2.2 Robot program

Two nozzle control strategies were tested to evaluate the effect of tool center point (TCP) orientation on both printability and image acquisition quality:

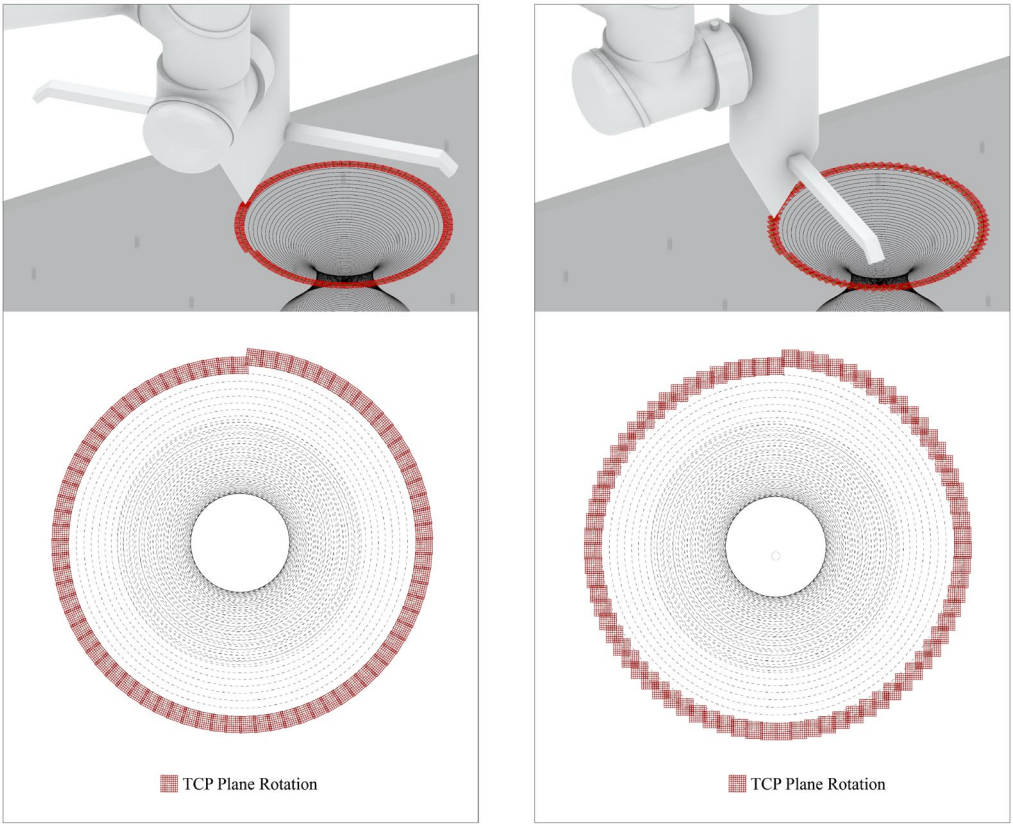


Figure 2.28: Option (2) Demonstration Left, Option (1) Demonstration Right

(1) **Nozzle Movement Without TCP Rotation**

In this strategy, the nozzle follows the toolpath while maintaining a fixed orientation. Since the camera is rigidly mounted along the nozzle axis, its viewing direction shifts relative to the print as the nozzle progresses along curved paths, especially in circular or helical toolpaths. This results in inconsistent top-view angles, which may affect ML model performance (Figure 2.28 right).

(2) **Nozzle Movement With TCP Rotation**

This strategy involves synchronizing the nozzle orientation with the toolpath curvature (Figure 2.28 left). At each waypoint, the UR5 robot is programmed to rotate the nozzle such that its axis remains tangential to the circular path. This ensures that the nozzle continuously points toward the center of curvature. Consequently, the camera maintains a consistent angle relative to the printed geometry throughout the motion (Figure 2.29).

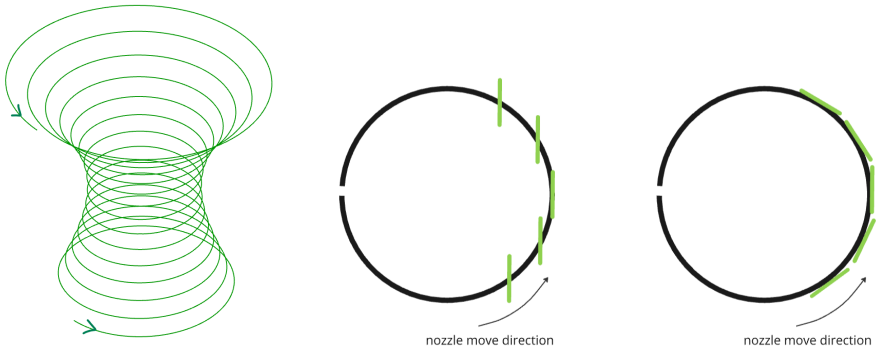


Figure 2.29: Demonstration of the sliced 3D printing path in Grasshopper (left), and comparison of the TCP with and without rotation (right)

Benefits:

The consistent tangential orientation results in a stable camera viewing angle throughout the printing process. As a result, this strategy may enhance the performance of ML models used for calibration, as it preserves a consistent spatial relationship between the camera and the printed geometry.

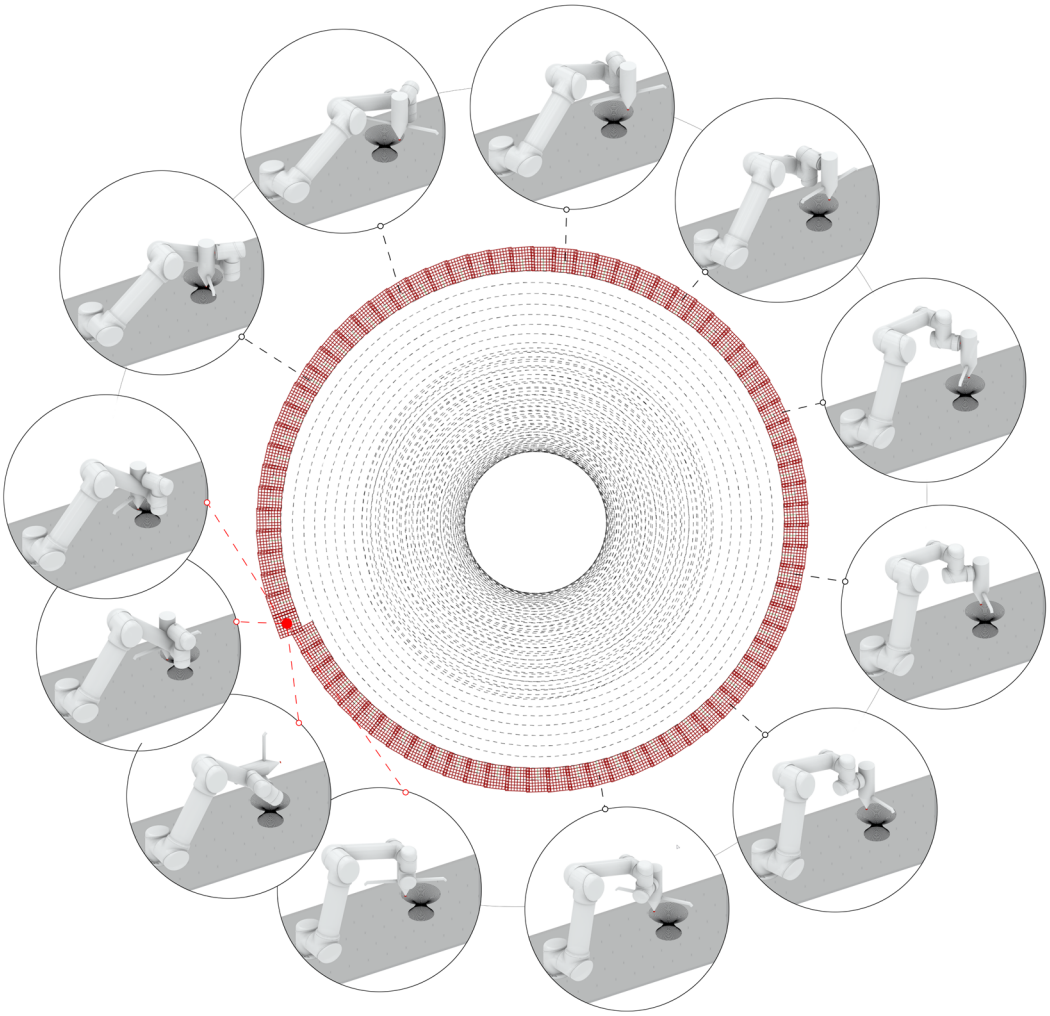


Figure 2.30: Explanation of sudden rotations between layers

Challenges:

In practice, the robot encounters a significant discontinuity in orientation when transitioning between layers. Specifically, a 180° rotation (Figure 2.30) is observed at the start of each new layer, despite identical end-effector poses being provided at the end of the previous layer. This leads to abrupt posture changes that can disrupt the printing process and introduce surface defects at layer junctions.

This is not a limitation of URScript itself, but rather a result of the inverse kinematics (IK) solver and internal motion planning algorithm used by the UR5 controller. The robot selects joint configurations based on optimization heuristics, and without explicit control over pose continuity, may choose mirrored solutions that cause sudden rotations between layers.

To diagnose the source of this issue, I verified that the end-effector poses (position and orientation) supplied to the robot were identical at the end of one layer and the start of the next. Additionally, all

waypoint planes extracted in Grasshopper were confirmed to have consistent upward-pointing normal vectors, ruling out errors in the input geometry. These findings point to the robot's internal IK solver as the source of the issue. Due to time constraints, a full resolution of this issue was not implemented.

2.2 Investigation of overhang print

An initial investigation was conducted on the current robotic 3DP setup **with Self-made Motor** to evaluate its performance and make preparation for dataset creation. The first step involved tuning several key printing parameters to match and sync with each other, aiming to ensure consistent and smooth clay extrusion: tank pressure was set to 1 bar, the extruder motor speed was adjusted to 27–28 revolutions per minute (RPM), the robotic arm movement speed was tested within the range of 6–13 mm/s, and the clay moisture content was optimized accordingly. Subsequently, the printing level height was calibrated within the Grasshopper script before initiating any test prints.

Through initial experimentation with the self-made motor setup, RAMS was systematically varied from 6 mm/s to 13 mm/s to identify the optimal range that synchronizes material extrusion rate and robotic motion, as shown in the Figure 2.31 below. At lower speeds (6–7 mm/s), the extrusion process exhibited slight over-extrusion, where the clay material was deposited faster than the robot's movement, resulting in thicker shell layers than intended. In contrast, higher speeds (11–13 mm/s) resulted in under-extrusion and discontinuities in the printed paths, indicating that the extrusion rate could not keep pace with the arm movement.

The investigation pinpointed the optimal arm movement speed range between 8 mm/s and 10 mm/s, where extrusion quality was most consistent. In this range, the extrusion rate closely matched the RAMS, resulting in stable shell thickness that aligned with the nozzle diameter and produced continuous, high-fidelity prints.

This finding highlights the delicate interplay between mechanical motion control and material flow speed in robotic 3DCP. Maintaining this synchronization is fundamental to producing structurally sound and dimensionally accurate prints, especially when printing overhangs and complex geometries.

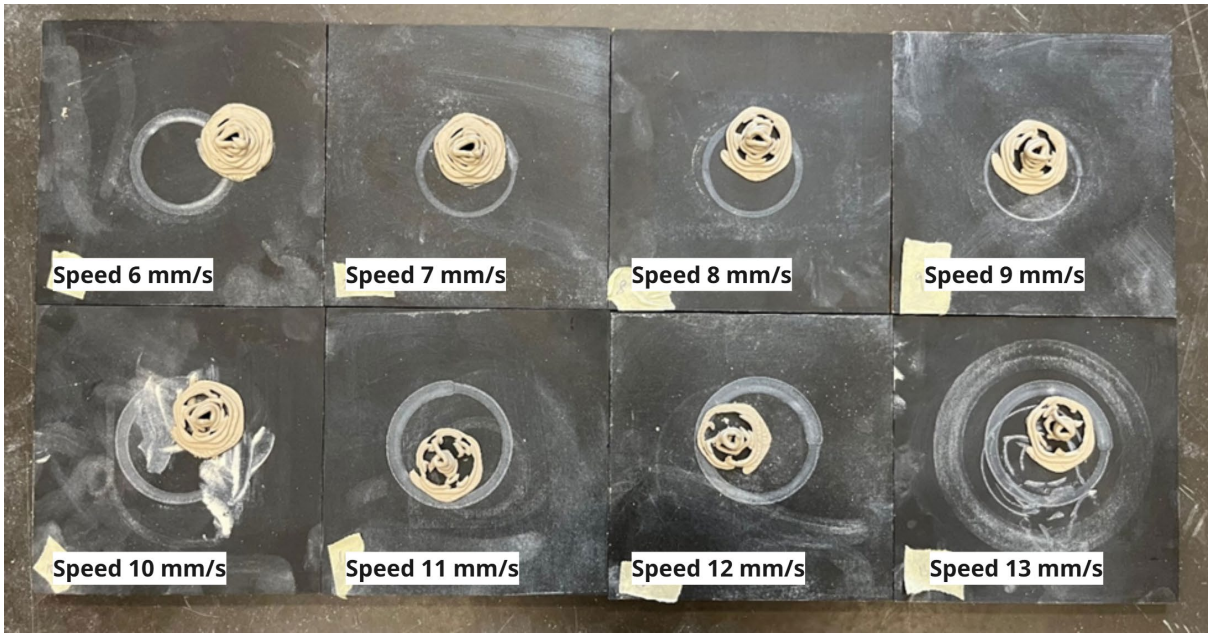


Figure 2.31: Image of the initial test prints produced at RAMS values ranging from 6 mm/s to 13 mm/s.

2.3 Machine learning

As introduced in the methodology section, the current dataset collection primarily focuses on two key printing quality parameters: extrusion quality and overhang success. Both parameters are critical for assessing the structural integrity and visual fidelity of 3DCP prototypes.

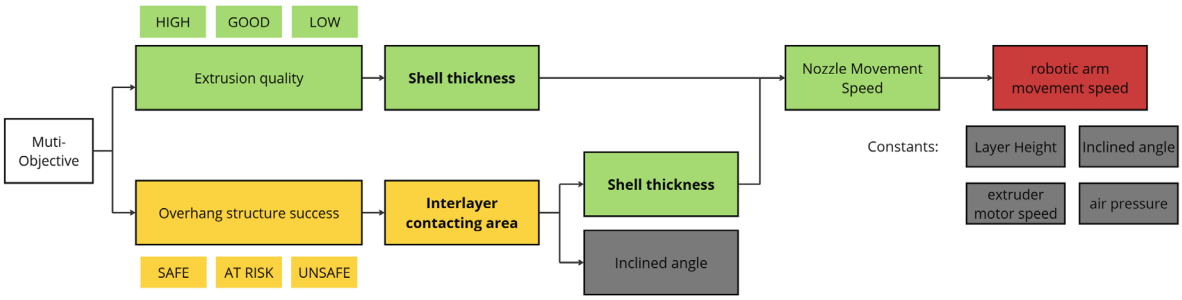


Figure 2.32: Influencing factors of ML two parameters

Extrusion Quality is directly related to the shell thickness, which depends mainly on the nozzle movement speed and is indirectly influenced by the robotic arm movement speed. Achieving the optimal shell thickness ensures sufficient material deposition for robust layer adhesion without causing excessive deformation from over-extrusion.

Overhang Success evaluates the stability and accuracy of printed overhang structures, and it is affected by both the shell thickness and the interlayer contacting area, specifically the horizontal

overlap between adjacent layers. The inclined angle of the overhang plays a crucial role here; larger angles reduce the contacting area, making the structure more prone to sagging or collapse if the shell thickness or adhesion is insufficient.

In this study, several parameters including inclined angle, layer height, air pressure, and motor speed, are maintained as constants during real-time calibration and printing correction. By controlling these variables as constants, the study can focus on investigating the multi-objective relationship between extrusion quality and overhang success, optimizing the printing parameters within a constrained time frame.

The ML model is expected to leverage visual cues related to shell thickness and interlayer adhesion, reflected in extrusion quality and overhang success labels, to predict and optimize these multi-dimensional printing outcomes.

2.3.1 Dataset generation

A real dataset was generated using the robotic 3DP setup equipped with a WASP motor, which produces less vibration and thus improves image clarity. Key printing parameters were again tuned for consistent and smooth clay extrusion: tank pressure was set to 0.24 MPa, the extruder motor speed was set at an unknown constant value (extruder cleaning mode), and robotic arm movement speed was tested across a range from 1 % to 10 % of base speed.

Printing issues during experiments that may influence the data consistency:

- Due to the small 1L clay tank, printing 2–3 prototypes required refilling the clay material, which involved reassembling the tank to the frame. This process could slightly change the nozzle and camera locations, necessitating camera calibration before each print.
- The 1L clay tank lacked perfect airtightness, sometimes causing air leaks and extrusion failures. Air pressure had to be increased and adjusted to approximate normal extrusion conditions, though slight deviations persisted, possibly affecting data accuracy.
- An unexpected issue occurred when one of the Raspberry Pi camera cables broke during printing. A shorter spare cable was used, requiring replacement of the Raspberry Pi and potentially altering the camera position.

2.3.1.1 Dataset collection

An extensive series of experiments (Figure 2.33) was performed to evaluate the robotic 3DCP system's performance over four overhang angles (130°, 140°, 150°, and 160°) and a range of RAMS from 1 % to 10 %.

For each condition, prints were evaluated based on extrusion quality and overhang success. Extrusion was categorized into three classes (under, good, and over) based on observed shell thickness.

Overhang success was classified as safe, at risk, or unsafe, according to form accuracy and structural stability (refer to Section 1.1 for evaluation standards).

A total of 31,153 image pairs were initially collected.

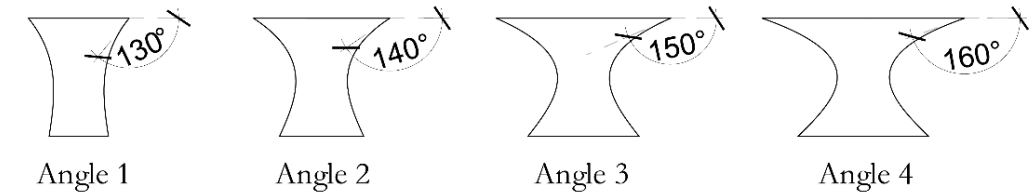
Figure 2.33: Full dataset collection

2.3.1.2 Data filtering

Of the 31,153 image pairs, 12 prototypes printed using the self-made motor produced 12,633 strongly blurred images caused by vibration; these were removed first. From the remaining 18,520 pairs, approximately 470 images taken before printing start and after printing end (without fresh print information) were discarded. In the remaining 18,050 pairs, 517 additional images with slight blurring from WASP motor vibration were also excluded.

Ultimately, 17,553 image pairs were retained for machine learning model training.

2.3.1.3 Data rating (labelling standard)



Speed / Angle	S1 0.1	S2 0.2	S3 0.3	S4 0.4	S5 0.5	S6 0.6	S7 0.7	S8 0.8	S9 0.9	S10 1
A1 130°	√	√	√	√	√	√	√	√	×	×
A2 140°	√	√	√	√	√	√	×	×	×	×
A3 150°	√	√	√	×	×	×	×	×	×	×
A4 160°	√	√	√	×	×	×	×	×	×	×

Legend: Parameter1: Extrusion
- Under Extrusion, label: [0]
- Good Extrusion, label: [1]
- Over Extrusion, label: [2]

Parameter2: Overhang success
- Safe, label: [0]
- At risk, label: [1]
- Unsafe, label: [2]
- Failure (no data)

Figure 2.34: Results of dataset labeling for two parameters: extrusion quality and overhang printability

In this study, each image pair was labeled using a two-dimensional classification scheme, where the first digit indicates extrusion quality, categorized as under extrusion (0), good extrusion (1), or over extrusion (2). The second digit represents overhang success, classified as unsafe (0), at risk (1), or safe (2).

The labeling criteria were derived from dataset collection conducted across four overhang inclined angles (130°, 140°, 150°, and 160°) and varying RAMS ranging from 1% to 10% of base speed. Table 1 below presents example data from the Angle 1 prototype print collection. Data for Angles 2, 3, and 4 are provided in Appendix 10.2.

Angle 1										
	Speed	Motor	Extrusion	Extrusion class	Overhang	Overhang class	label	Note	image amount	pair amount
1	0.06	WASP	Good	1	safe	2	1,2	successful print	662	331
2	0.1	WASP	Under(bottom)	0	unsafe	0	0,0	fall down after successfully print	430	215
3	0.09	WASP	Under(bottom)	0	unsafe	0	0,0	fall down after successfully print	668	334
4	0.08	WASP	Under(top overhang)	0	safe/at risk	1	0,1	successful print(top overhang has holes)	848	424
5	0.07	WASP	thin thickness	0	safe	2	0,2	successful print	892	446
6	0.01	WASP	under(unsuccessful)	0	unsafe	0	0,0	can't stick on the previous layer	24	12
7	0.01	WASP	under(unsuccessful)	0	unsafe	0	0,0	can't stick on the base	60	30
8	0.04	WASP	Good	1	safe	2	1,2	successful print(layer thickness varies)	772	386
9	0.07	WASP	Good	1	safe	2	1,2	successful print	1322	661
10	0.01	WASP	Extremely Over	2	at risk	1	2,1	fat print, successful	916	458
11	0.05	WASP	Good in general	1	safe	2	1,2	successful print(layer thickness varies)	2092	1046
the second day printing										
24	0.03	WASP	Over(good)	2	at risk	1	2,1	fat print, successful, unfinish for top	262	131
25	0.03	WASP	good-over	2	safe	2	2,2	unfinished, a bit fat	122	61
26	0.03	WASP	Good(over a little)	2	safe	2	2,2	a little bit fat, successful	1214	607
27	0.04	WASP	Good(varies)	1	safe	2	1,2	unfinished	226	113
28	0.08	WASP	under	0	unsafe	0	0,0	can't well stick, uncontinous	222	111
29	0.01	WASP	Over	2	at risk	1	2,1	fat, unfinish printing	220	110
30	0.1	WASP	UNDER	0	unsafe	0	0,0	can't stick on the base	44	22
31	0.02	WASP	Over	2	at risk	1	2,1	fat, unfinish printing	90	45
Total										5543

Table 1: Prototype data collection and labelling example for Angle 1. More data are shown in the appendix

By analyzing the print collection, distinct extrusion behaviors emerged, with over extrusion occurring at speeds between 1% and 3%, characterized by shell thickness significantly exceeding the nozzle diameter (3 mm), resulting in excessive material deposition, geometric deformation, and spreading, although these prints generally remained structurally sound.

Good extrusion, observed at speeds between 4% and 6%, produced shell thickness approximately equal to the nozzle diameter, yielding prints with strong structural integrity and accurate form, particularly at lower inclined angles.

Under extrusion, prevalent at speeds from 7% to 10%, was associated with shell thickness thinner than the nozzle diameter, insufficient material deposition, poor layer adhesion, structural weakness, and frequent print failures, especially at higher overhang angles.

Overhang success was assessed based on printing completeness and structural stability, where safe prints were completed successfully with minimal deformation and strong adhesion. At-risk prints showed noticeable sagging or distortion, and unsafe prints failed during or shortly after printing due to collapse or adhesion failure.

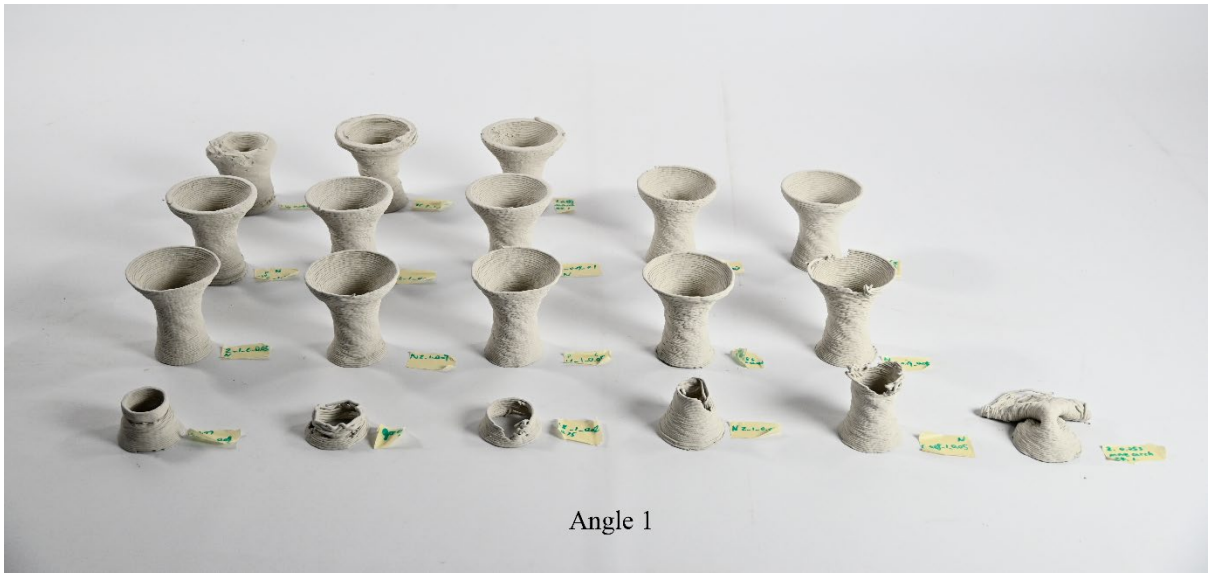


Figure 2.35: Example of Angle 1 print collection including under-, good-, over extrusion with safe, at risk and unsafe situations

The distribution of label combinations (Table 2) demonstrated that the 130° angle produced the broadest spectrum of successful prints, frequently achieving the optimal combination of good extrusion with safe overhang, whereas higher angles (140°, 150°, and 160°) exhibited progressively more at-risk and unsafe classifications, reflecting the increased geometric challenges and reduced interlayer contact. Notably, the combination of over extrusion with unsafe overhang (label 2,0) was absent, indicating that while over extrusion induces geometric deformation, it effectively prevents adhesion-related failures.

Label	Description
0,0	Under extrusion, overhang failure: prints easily collapse or have poor adhesion to the base or previous layer.
0,1	Under extrusion, overhang at risk: very few successes, but thin shell overhang shows holes (mainly at low angles). (Figure 2.36)
0,2	Under extrusion, overhang success (rare): occasional success but mostly failure.
1,0	Good extrusion, large-angle overhang failure: failures mainly caused by excessive overhang angle.
1,1	Good extrusion, large-angle overhang at risk: risk mainly due to insufficient contact area.
1,2	Good extrusion, overhang success: typical successful combination, stable especially at smaller angles.

2,0	Over extrusion, overhang failure (none): almost nonexistent because over extrusion ensures adhesion.
2,1	Over extrusion, overhang at risk: over-extrusion causes deformation and sagging of upper layers. (Figure 2.37)
2,2	Over extrusion, overhang success: slightly over-extruded but overall successful with good form.

Table 2: The distribution of label combinations analysis

Figure 2.36: Example of overhang part under-extrusion

Figure 2.37: Examples of overhang sagging without collapse

A detailed examination of the label combinations revealed that failure modes involving under extrusion (0,0; 0,1; 0,2) are primarily driven by insufficient material deposition, resulting in thin shell thickness, weak structural bonding, and subsequent collapse or print instability, with limited influence from overhang angle. Conversely, prints exhibiting good extrusion but failure or risk (1,0; 1,1; 1,2) predominantly failed due to geometric constraints posed by larger overhang angles, where reduced layer contact area compromised structural stability. Over extrusion cases (2,0; 2,1; 2,2) rarely resulted in outright failure. Instead, they displayed excessive material buildup and deformation, with extrusion quantity exerting a stronger influence on print quality than the overhang angle.

Overall, the majority of successful prints were categorized as (1,2), representing optimal conditions of good extrusion coupled with safe overhang performance. Failures clustered mainly around under extrusion or high overhang angles with compromised layer contact.

Calculation of Dominant Factor Proportion

Following a general analysis of label combinations, it is insightful to examine the dominant factor between the two parameters across the print collection.

The dominant factor proportion was calculated by analyzing the dataset labels that describe the combined effects of shell thickness decided by extrusion amount and overhang inclined angle (OIA) on print success. For each printing speed (the direct factor affects the shell thickness in current experiment set up) and OIA pair, the dataset samples were classified according to whether extrusion quality or OIA was the primary influence on the printing outcome. This classification was based on a detailed review of the label combinations (e.g., 0,0; 1,2; 2,1, etc.) and their corresponding print quality notes. The proportion of samples dominated by extrusion or by OIA was then computed by dividing the number of samples primarily influenced by each factor by the total samples within that speed-angle group. This yielded two proportion values per speed and angle combination, representing the relative dominance of extrusion and overhang in determining print success. A demonstration of the calculation formula is shown below:

$$\text{Dominant Factor Proportion} = \frac{\text{Number of samples dominated by a factor}}{\text{Total samples at given speed and angle}}$$

Analysis of Dominant Factors Affecting Print Success

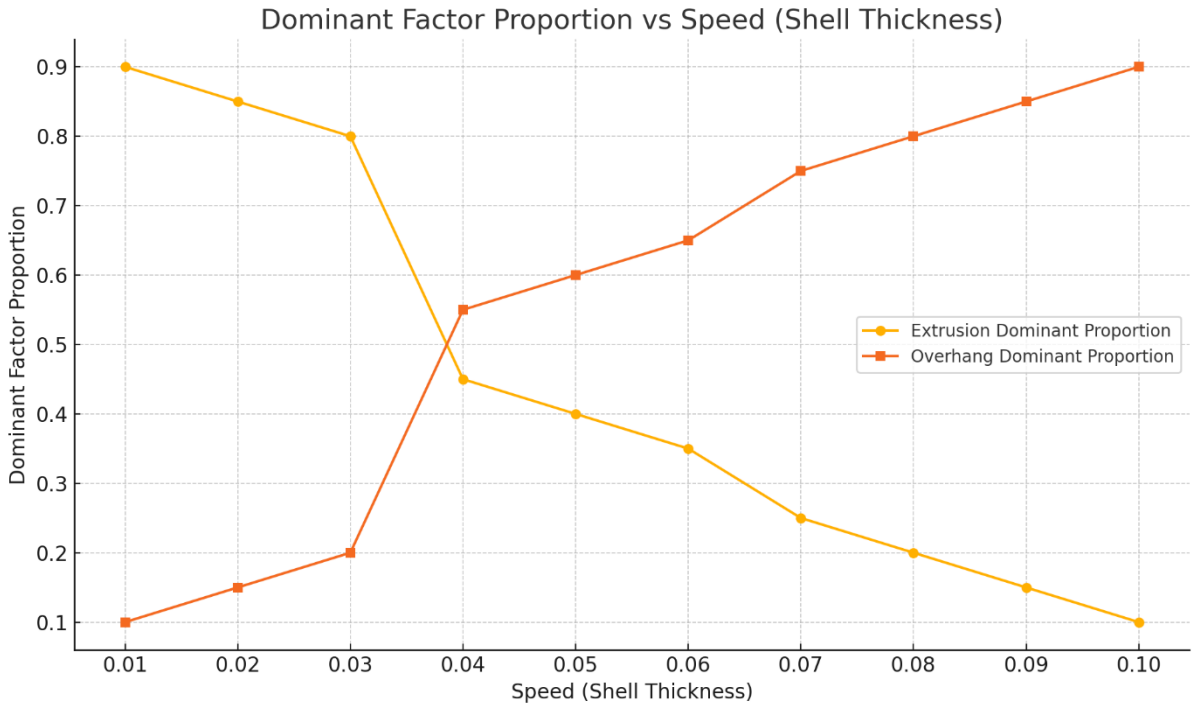


Figure 2.38: Dominant Factor Proportion Analysis for printing speed

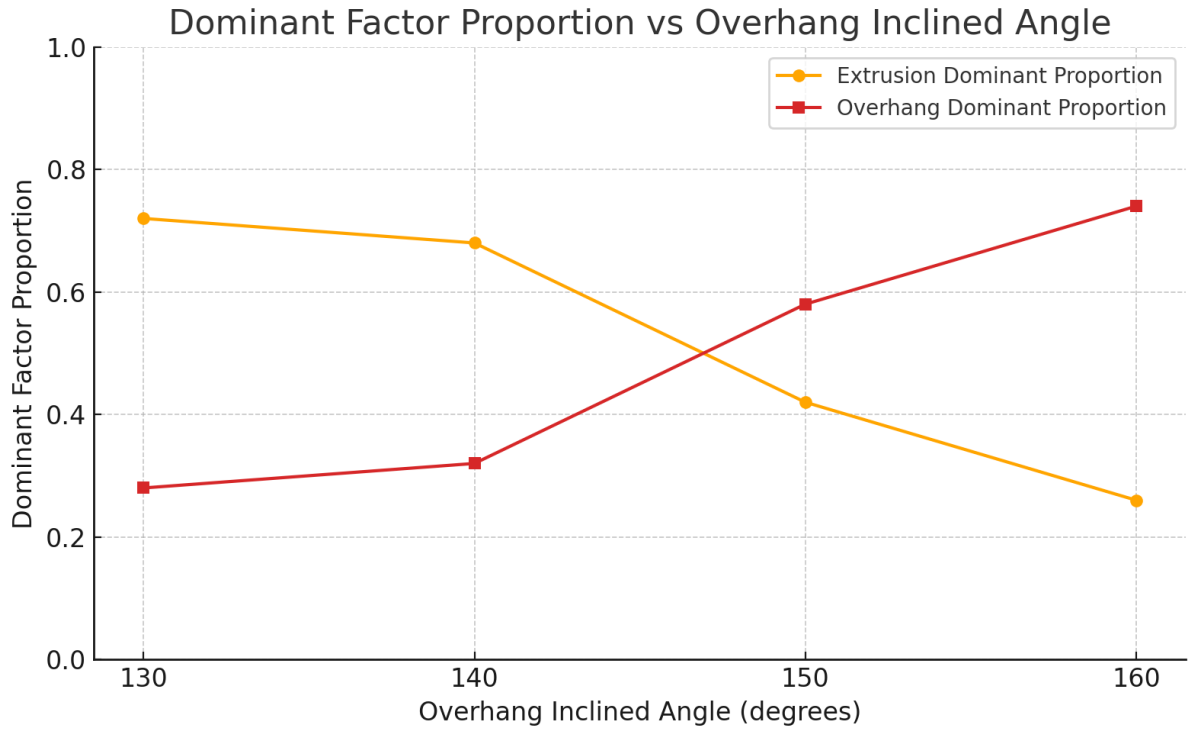


Figure 2.39: Dominant Factor Proportion Analysis for Overhang inclined angle

An interesting finding from the dataset is the dynamic shift in dominant factors influencing print success, observed both as a function of printing speed (the direct influence factor of shell thickness) and overhang inclined angle (OIA). The first analysis (Figure 2.38) illustrates how the dominance

between extrusion quality and OIA changes with printing speed. At lower speeds (0.01 to 0.03), shell thickness, controlled by extrusion quality, overwhelmingly governs print success. This aligns with observations that excessive extrusion at slow speeds enhances layer adhesion and structural stability, even if some deformation in shape occurs.

As printing speed increases into the mid-range (0.04 to 0.06), the influence between extrusion and OIA balances out. This transitional regime reflects a complex interplay where both material flow and geometric constraints critically determine print quality. Beyond 0.07 speed, OIA emerges as the primary limiting factor, with under-extrusion causing weaker interlayer adhesion and the geometric challenges of steeper overhangs leading to frequent print failures.

Complementing this, the second analysis (Figure 2.39) shows the variation of dominant factor proportion across different OIA. It reveals that at lower inclined angles (130°, 140°), extrusion quality plays a more significant role in print success, whereas at steeper angles (150°, 160°), the geometric challenges posed by the OIA increasingly dominate. This trend highlights that printability is not solely governed by material deposition but is highly sensitive to the geometry of the printed feature.

From these findings, it is concluded that extrusion amount is the fundamental factor controlling print success, as insufficient extrusion causes the majority of failures, while overhang inclined angle acts as a secondary but critical limiting factor, particularly under good extrusion conditions where steep angles reduce interlayer adhesion and increase defect risk. Over extrusion generally mitigates adhesion failures but compromises geometric accuracy and surface quality through deformation. This nuanced understanding informs parameter optimization and highlights the interplay between material deposition and geometric constraints in achieving reliable robotic 3DCP.

2.3.1.4 Data preprocessing

The raw images, initially captured at a resolution of 3500 × 4608 pixels, are first paired and undergo a series of preprocessing steps including warping, cropping, and merging based on the established calibration parameters (Details described in Section 2.1.2.1). After merging, the images are further cropped to a size of 224 × 224 pixels, which corresponds to the input dimensions required by the machine learning model used in subsequent analysis.

To account for variations in lighting conditions during dataset collection, brightness adjustment factors are applied to each sample's set of images. This normalization step helps reduce the impact of lighting inconsistencies, which, as demonstrated in preliminary research, can significantly affect prediction accuracy due to the inherent sensitivity of computer vision-based models to illumination changes.

Further image augmentation techniques are applied to the cropped images to improve dataset robustness. These augmentations include rotation, scaling, mirroring, brightness modifications, and normalization. Such data augmentation enhances the model's generalization capability by simulating

diverse imaging conditions and perspectives.

Finally, for each processed image, a corresponding CSV file is generated to serve as input labels for the machine learning training. The CSV files contain the updated image paths, following the format 'dataset_filtered/{renamed_image}.jpg', alongside the associated parameter labels, facilitating organized and consistent dataset management.

2.3.1.5 Imbalanced Distribution and Underlying Reasons

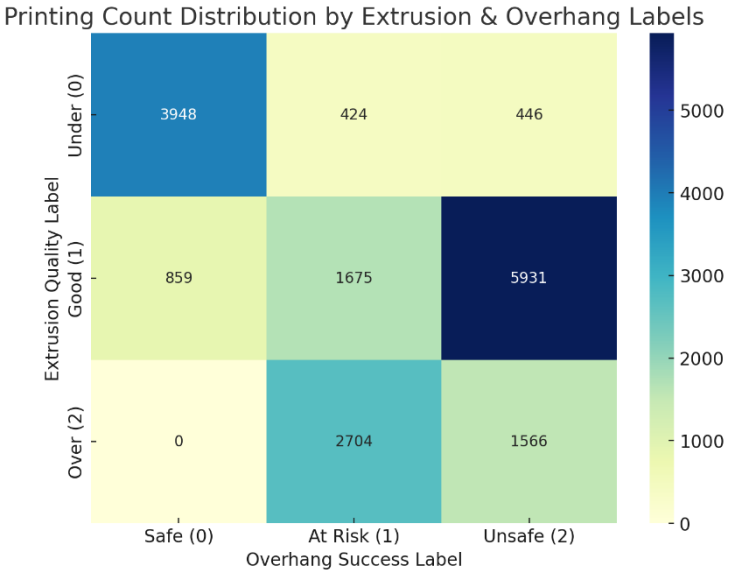


Figure 2.40: Printing count distribution by label combinations

The dataset exhibits noticeable imbalance, which stems primarily from the complex interaction between printing parameters and geometric factors as mentioned before.

This uneven label distribution presents challenges during machine learning model training, as models may become biased towards majority classes, reducing sensitivity to minority but important cases. To address this, training on a balanced subset of the dataset, constructed around the minority label (2,2), is implemented to improve model robustness and fairness.

Figure 2.41 illustrates the label distribution of the full dataset, highlighting the imbalance, while Figure 2.42 shows a balanced subset distribution where samples are evenly represented across label classes.

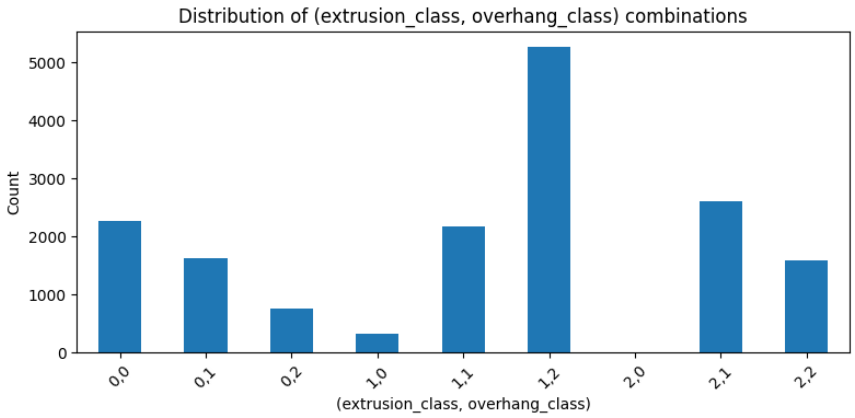


Figure 2.41: label Distribution of full dataset

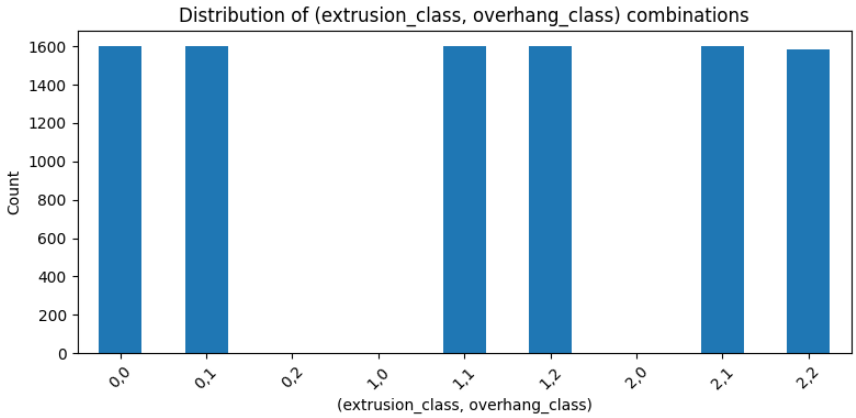


Figure 2.42: label distribution of balanced dataset

For future work, it is proposed to compare model performance between training on this balanced subset and training on the full dataset with advanced techniques such as class weighting, upsampling, and downsampling. These strategies aim to mitigate class imbalance effects by either assigning higher importance to underrepresented classes during training or adjusting sample counts to create a more uniform label distribution.

2.3.2 Model architecture, training and performance

2.3.2.1 Model architecture

(1) Resnet 56

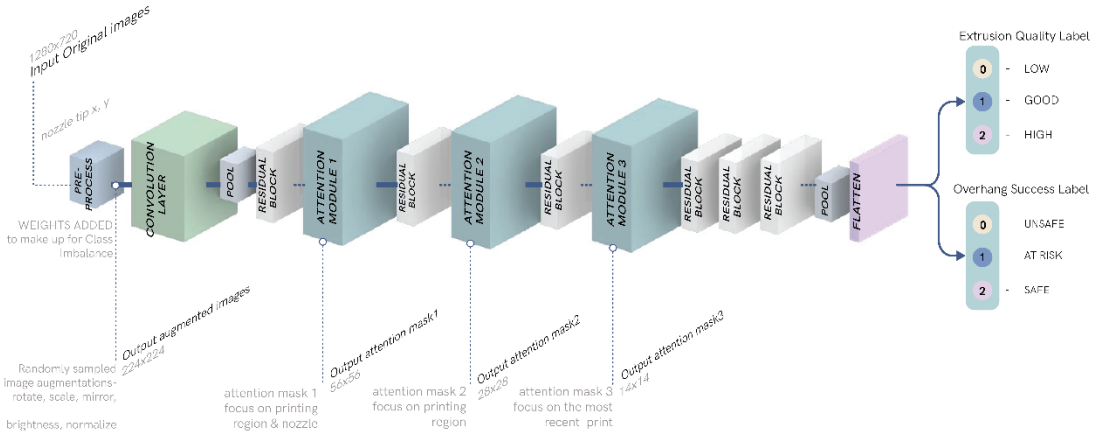


Figure 2.43: Resnet 56 architecture

The defect detection model used in this project is based on the Residual Attention Network (ResNet-56) architecture, originally proposed by Wang et al. (2017) and later applied to anomaly detection in PLA AM by Brion and Pattinson (2022). This architecture combines deep residual learning with attention mechanisms to enhance its capacity for distinguishing different focus areas of extrusion features in complex visual data. Each attention module in the network comprises a trunk branch for feature propagation and a mask branch that adaptively generates attention maps. These maps modulate the trunk features, allowing the network to emphasize salient regions while suppressing noise and irrelevant background information. By integrating this attention structure within a residual framework, the model maintains efficient gradient flow and prevents degradation across layers, even in deeper networks. During training, the model learns to identify deviations in print quality by capturing spatial and contextual relationships between extrusion parameters and visual cues. To support model interpretability, Gradient-weighted Class Activation Mapping (Grad-CAM) is utilized. Grad-CAM produces class-specific localization maps that emphasize the regions most influential to the model's output, thereby providing visual insight into the network's reasoning. This transparency facilitates the identification of which image regions the model attends to after each residual block, enabling evaluation of its attention focus and refinement of the dataset collection strategy accordingly.

Preliminary Findings from ResNet-56 Training on Pre-Study Dataset

Initial experiments using the ResNet-56 architecture on a pre-research dataset have yielded several important observations. First, the network successfully extracts and highlights key features related to

extrusion variability and layer height, demonstrating notable potential for enhancing real-time anomaly detection in AM processes. However, prediction accuracy remains inconsistent across different conditions, indicating the need for further refinement.

Moreover, residual attention mechanisms embedded within the architecture prove effective in directing the model's focus to relevant spatial regions, thereby improving interpretability and responsiveness to subtle changes in material deposition. Multi-head learning structures, where the model simultaneously predicts multiple interrelated parameters, have shown superior performance compared to single-head configurations. This suggests that the full representational capacity of the ResNet-56 model is better leveraged when handling multi-task outputs.

The consistency of environmental conditions during data collection, such as lighting, background, and substrate color, has emerged as a critical factor. Uniform experimental settings help minimize domain shifts that may otherwise compromise the model's generalizability. In situations where such consistency cannot be maintained for real-world deployment, a substantially larger and more diverse dataset would be required to ensure robust performance.

Additionally, clear and systematic labeling of the dataset is essential. Ambiguities or inconsistencies in manual labeling can introduce noise, leading to increased prediction errors and diminished model reliability. Lastly, maintaining uniform image dimensions between training and calibration datasets has proven to significantly enhance prediction accuracy. Variations in image size can distort the model's attention, particularly when cameras are reinstalled or repositioned, making spatial consistency a key consideration for ongoing model deployment and validation.

(2) DINOv2-Based Hybrid Network Architecture

In the previous section, I discussed the design and performance of the ResAttNet-56 architecture applied to dual-task classification of 3D-printed clay object. Although the model exhibited reasonable convergence on the training set, its performance on validation process showed limited generalization. Despite several rounds of tuning, the learned weights from the ResNet-56-based model did not yield satisfactory prediction accuracy, particularly for more subtle variations in print quality.

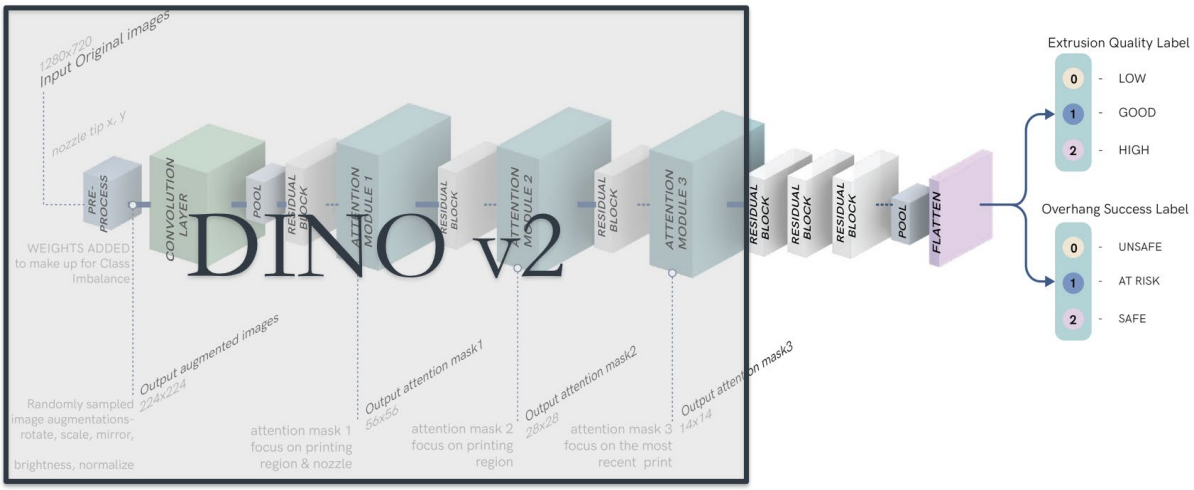


Figure 2.44: demonstration of DINOv2-Based Hybrid Network architecture

To address this, I hypothesized that the earlier layers in ResAttNet-56, especially the convolutional and shallow attention stages, might not be learning sufficiently discriminative or abstract features from the input images. As an alternative, I propose replacing the front-end feature extractor of ResAttNet-56 with the DINOv2 ViT-S/14 model, a state-of-the-art self-supervised vision transformer.

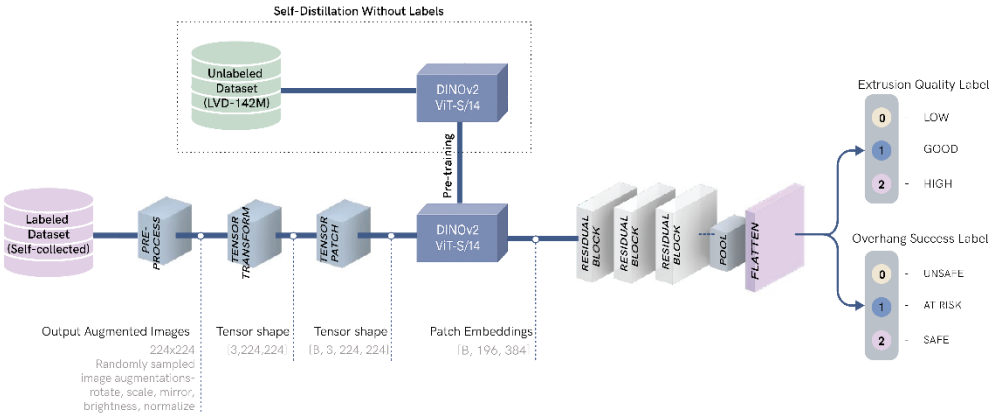


Figure 2.45: demonstration of DINOv2-Based Hybrid Network architecture

DINOv2 is a self-supervised vision transformer pretrained on large-scale unlabeled image datasets using the DINOSAUR pipeline developed by Meta AI (Maxime Oquab, 2024). It is known for capturing semantic-rich visual representations across scales. Its output patch tokens are spatially attentive, making them particularly well-suited for downstream tasks requiring fine-grained classification.

Motivations for replacing image feature extractor with DINOv2

Improved feature abstraction: DINOv2 uses transformer-based attention mechanisms across patch tokens, enabling it to capture more global and semantically aligned image features than shallow

convolutional layers.

Self-supervised pretraining advantage: Unlike supervised models trained on limited labels, DINOv2 learns general-purpose visual features from diverse image corpora without human annotation, increasing its transferability.

Architectural Integration

Since the reference DINOv2 codebase is optimized for single-head binary classification tasks and does not directly support multi-task or multi-class learning, I constructed a new model module, DINO2ResAttClassifier, which integrates DINOv2 as a fixed front-end feature extractor and connects its outputs to the backbone of a two-head ResNet56.

To build the hybrid model DINO2ResAttClassifier, the model modification involves the following key stages:

1. Token Extraction and Projection: The input image is passed through DINOv2, and patch-level tokens (excluding the class token) are extracted from the last attention layer (x_norm_patchtokens). These are reshaped into a spatial feature map and projected from 384 to 1024 channels using a 1×1 convolution to match the expected input dimensions of ResNet56.

2. Backbone Integration: Instead of using the full ResNet56, I selectively retain only its later residual blocks (res4, res5, res6) and its classification structure. This segment performs deeper reasoning over the feature maps provided by DINOv2, enabling hierarchical refinement.

3. Dual Prediction Heads: To accommodate the multi-label classification task, predicting both layer_height and extrusion classes, the architecture is equipped with two separate linear heads, each producing a 3-class output.

4. Training Strategy: The model is trained via a dual cross-entropy loss function, one for each label. Optimization is conducted using AdamW with learning rate scheduling via ReduceLROnPlateau. To ensure modularity and clarity, we implemented the model in PyTorch Lightning with clear separation of training, validation, and test steps.

This integration leverages the pretrained attention-rich representations of DINOv2 while preserving the effective residual learning of ResNet56. By replacing the original ResNet56's early convolutional, pooling, and attention layers with DINOv2's transformer-based extractor, the new hybrid model is expected to improve generalization and expressiveness in downstream multi-class prediction tasks. Importantly, it is supposed to enable more efficient learning with limited labeled data, benefiting from DINOv2’s large-scale pretraining.

In conclusion, this hybrid architecture combines the best of both paradigms: DINOv2’s rich pretrained

semantics and ResNet56's effective spatial encoding, leading to improved multi-head classification performance and a structured basis for future architectural comparisons.

Differences between ResNet-56 and DINOv2-based hybrid model

ResNet-56 is a convolutional network that excels at extracting local features through hierarchical filters and attention modules but relies on supervised learning with labeled data. It may lack the ability to capture high-level semantic information without large datasets. In contrast, DINOv2 uses a transformer architecture with self-attention over image patches, enabling it to capture global context and more abstract features. Pretrained in a self-supervised manner on vast unlabeled data, DINOv2 learns rich, generalizable visual representations that improve robustness to subtle variations. By combining DINOv2's powerful feature abstraction with ResNet-56's spatial reasoning and classification layers, the hybrid model leverages the strengths of both approaches to enhance accuracy and generalization in multi-label classification tasks.

2.3.2.2 Training ML models

This study evaluates two neural network architectures, ResNet-56 and a DINOv2-based hybrid model, for defect detection and quality prediction in robotic 3DCP with the two datasets trained and tested separately: one pre-research dataset and one newly generated dataset.

The pre-research side-view old dataset consisted of 3,000 training images with a three-class classification scheme for both layer height and extrusion quality. These 3,000 images were randomly selected from the full old dataset by sampling equally from each label combination to obtain a balanced dataset. To evaluate the impact of incorporating DINOv2 as a feature extractor, two training pipelines were developed using the same dataset and label standards. The training time for the DINOv2-Based Hybrid Network was approximately 2 hours and 34 minutes, compared to 1 hour and 11 minutes for ResNet-56.

The new top-view dataset for current research comprised 9,584 images with balanced label combinations. Training on this larger dataset took 11 hours and 34 minutes for the DINOv2 model and 1 hour and 55 minutes for ResNet-56.

Both models were trained and tested on balanced datasets with consistent data preprocessing and training parameters, enabling a fair comparison of performance and robustness.

ResNet-56 Model Hyperparameters

Parameter	Value
Number of Hidden Layers	Residual blocks: res4, res5, res6 (backbone later stages)

Activation Functions	ReLU (with BatchNorm + ReLU in residual blocks)
Nodes / Channels	Input channels: 1024 (after 1×1 conv projection), Output features: 2048
Epochs	50
Batch Size	32
Optimizer	AdamW
Learning Rate	0.001
Learning Rate Scheduler	ReduceLROnPlateau (monitor: val_loss, factor=0.1, patience=3)
Loss Function	CrossEntropyLoss (dual-head classification: layer_height and extrusion losses summed)

The ResNet-56 model was configured with a preprocessing pipeline that included resizing input images to 224×224 pixels, tensor conversion, and normalization with mean and standard deviation values calculated from the dataset. Training employed a batch size of 32 and a learning rate of 0.001 with a maximum of 50 epochs. The dataset used for training was a balanced labeled set, with corresponding CSV metadata for supervision. The model leverages deep residual connections and attention mechanisms to focus on spatial features relevant to extrusion defects and layer height variations. Training was accelerated on GPU hardware to ensure efficient convergence.

DINOv2-Based Hybrid Model Hyperparameters

Parameter	Value
Number of Hidden Layers	Residual blocks: res4, res5, res6 (backbone later stages) + frozen DINOv2 frontend
Activation Functions	ReLU + BatchNorm + AdaptiveAvgPool
Nodes / Channels	DINOv2 output: 384-dim projected to 1024 channels; final ResNet output features: 2048
Epochs	50
Batch Size	32
Optimizer	AdamW

Learning Rate	0.001
Learning Rate Scheduler	ReduceLROnPlateau (monitor: val_loss, factor=0.1, patience=3)
Loss Function	CrossEntropyLoss (dual-head classification: extrusion and overhang success losses summed)

The DINOv2-based hybrid model uses a similar preprocessing pipeline for consistency. It incorporates the self-supervised pretrained DINOv2 vision transformer as a fixed feature extractor, outputting 384-dimensional embeddings, which are then reshaped and passed through residual blocks and multi-head linear classifiers. The same training hyperparameters—batch size 32, learning rate 0.001, and up to 50 epochs—were applied. The balanced labeled dataset with identical normalization parameters ensured comparability with the ResNet-56 results. This model architecture aims to capture semantically rich and global visual features from the transformer backbone, complemented by the residual network's hierarchical reasoning capabilities.

2.3.3 Model Performance Comparison

2.3.3.1 Grad-CAM – Result Comparison

To better understand how each model learns to extract visual features during training, I employed Gradient-weighted Class Activation Mapping (Grad-CAM) and Grad-CAM++ to visualize the attention focus of two distinct image classification models: a ResNet-56-based Residual Attention Network and a DINOv2-based vision transformer.

I initially applied visualization techniques to the pre-research old dataset images. After evaluating different methods, Grad-CAM++ was chosen for its superior localization and clarity. Using this method, I subsequently visualized samples from the new top-view dataset to analyze model attention behavior.

ResNet-56 Attention Analysis

I first applied Grad-CAM to the ResNet-56-based model at different residual blocks (res1 to res4). The generated heatmaps reveal how the model's spatial attention progressively shifts across the network depth, from focusing on low-level geometric edges, layer texture and nozzle tip to capturing higher-level contextual patterns such as the backgrounds. At each stage, Grad-CAM operates by extracting the forward feature maps (activations) and the corresponding gradients during backpropagation:

- For res1, the tensor shape was torch.Size([1, 256, 56, 56]).
- For res2, torch.Size([1, 512, 28, 28]).

- For res3, torch.Size([1, 1024, 14, 14]).
- For res4, torch.Size([1, 2048, 7, 7]).

Grad-CAM computes the importance of each channel using the average of its gradients and then linearly combines the feature maps using these weights. This yields a 2D attention map that is resized to 224×224 and overlaid onto the input image. The resulting heatmaps clearly highlight nozzle regions, deposition paths, and other discriminative areas relevant for prediction (Figure 2.46).

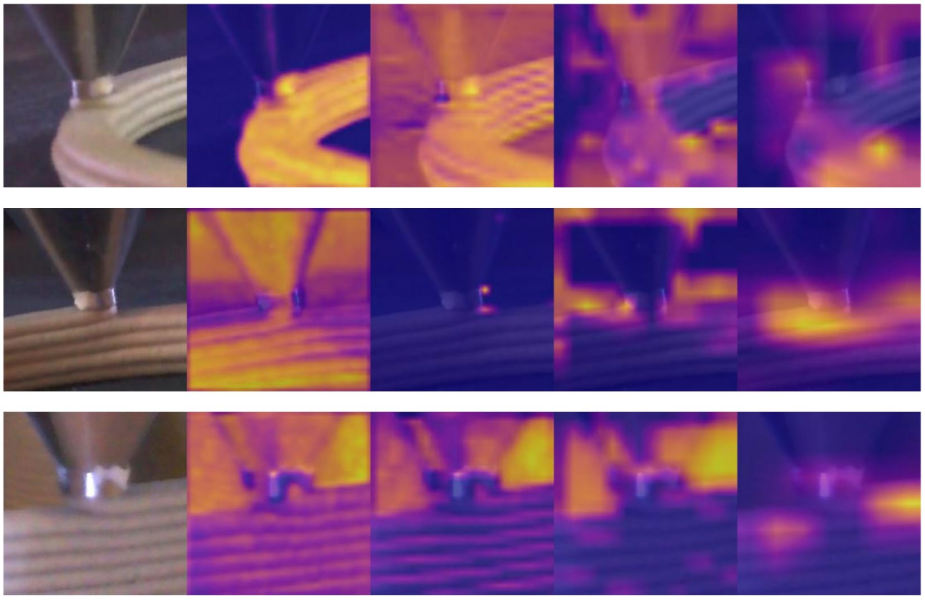


Figure 2.46: original image for old dataset from pre-research (side view), heatmap for Resnet 56 after res1, res2, res3, res4 (from left to right)

To improve spatial precision, I further adopted Grad-CAM++ (Figure 2.47), which introduces higher-order derivatives to better capture pixel-level importance in overlapping object regions. This technique yielded more focused heatmaps, especially in deeper blocks such as res4 and res6, reinforcing the idea that deeper features encode abstract representations.

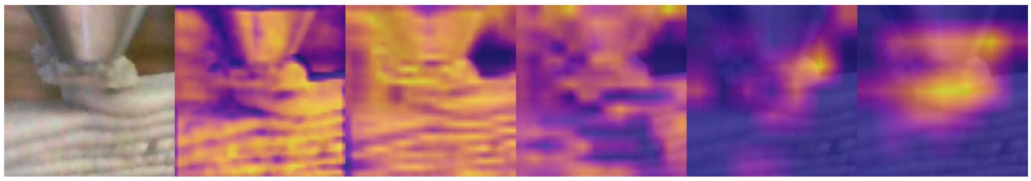
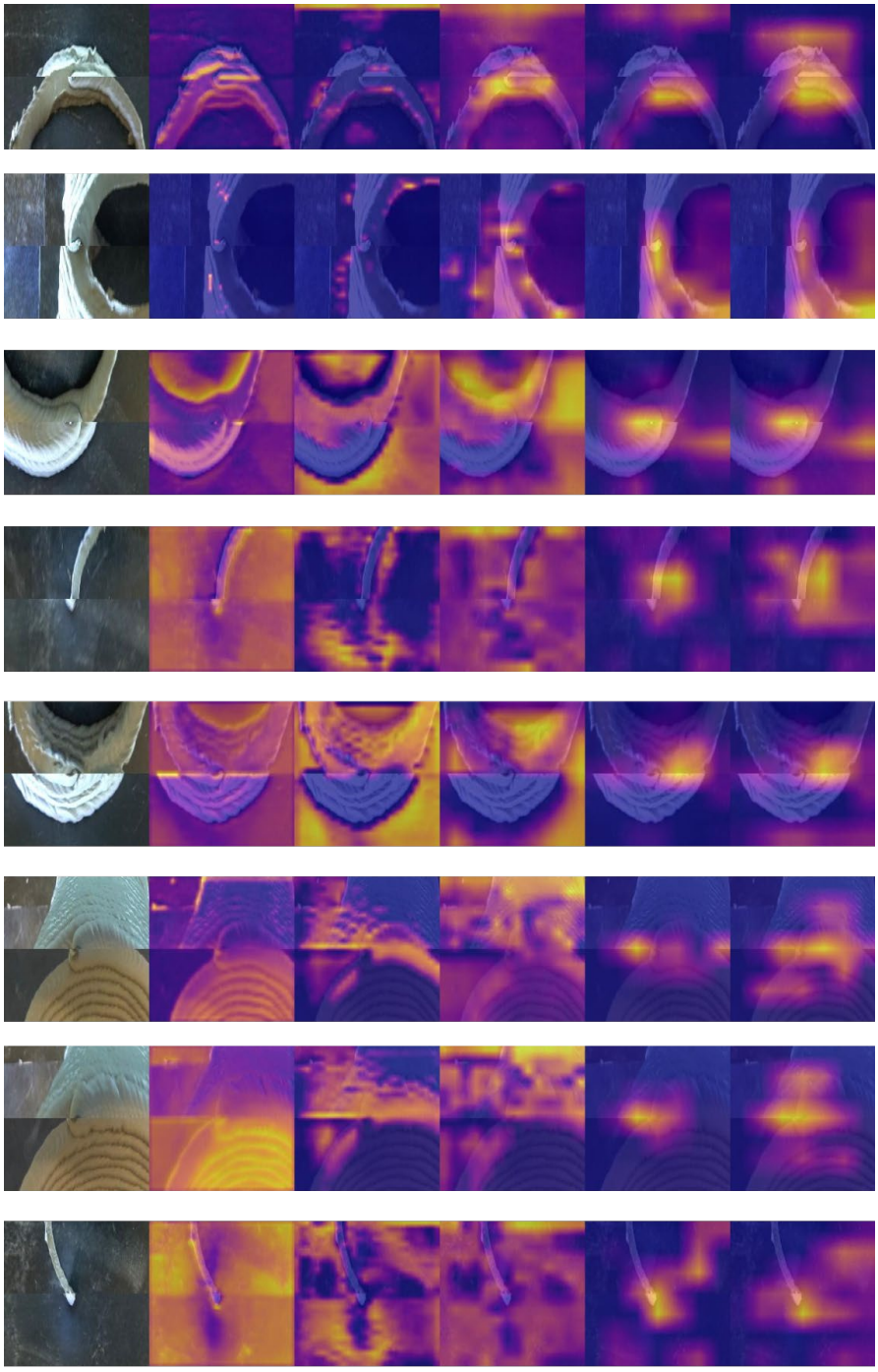


Figure 2.47: original image for old dataset from pre-research (side view), heatmap for Resnet 56 after res1, res2, res3, res4, res6 (from left to right)

After training with the new dataset, a random selection of 10 images was visualized using Grad-CAM++ to inspect how spatial attention evolves through the network. The results (Figure 2.48) demonstrate that as convolutional layers deepen and the attention mechanism refines, the model

increasingly concentrates its focus on regions surrounding the nozzle tip. This focus remains consistent and robust across different samples, highlighting the nozzle area and deposition paths as the primary discriminative features for classification. This localization is crucial because it reflects the model's ability to identify fine-grained geometric and textural cues associated with extrusion quality and overhang success. The heatmaps also reveal subtle distinctions in layer textures and structural edges, underpinning the model's interpretability and targeted attention on physically relevant print



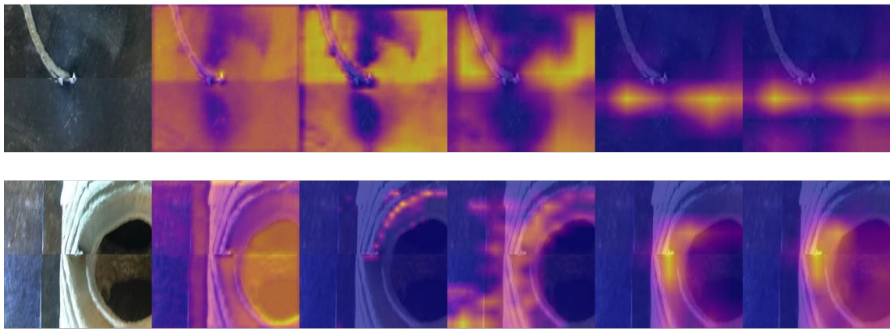


Figure 2.48: original image for new dataset (top view), heatmap for Resnet 56 after res1, res2, res3, res4, res6 (from left to right)

DINOv2 Transformer Attention Analysis

Unlike ResNet, the DINOv2-based model is not trained end-to-end in our pipeline. Instead, it is used solely as a frozen feature extractor to compute 384-dimensional embeddings, which are then fed into a small multi-head MLP classifier predicting two discrete labels: layer_height_class and extrusion_class.

Due to this architectural separation, Grad-CAM cannot be directly applied to the MLP classifier: the embedding vectors lack spatial dimensions and activations, making them unsuitable for gradient-based spatial attribution.

Therefore, to inspect DINOv2's internal attention, I applied Grad-CAM to the transformer backbone itself during the embedding extraction stage. Attention was visualized by targeting the last norm layer of the final transformer block. Since Vision Transformers do not naturally generate convolutional feature maps, I used the reshape_transform utility to reshape the flattened patch tokens into a 2D spatial format.

The generated Grad-CAM heatmaps (Figure 2.49) illustrate that DINOv2 tends to attend more broadly across the nozzle region and printing material, yet the attention is diffused compared to ResNet's strongly localized response. While these activations still reveal semantically meaningful regions, they lack the crisp boundaries observed in residual blocks of convolutional networks. The choice of colormap also affects visual interpretation—jet or plasma can be misleadingly saturated; I adopt viridis or custom RGB mapping to better highlight local attention patterns.

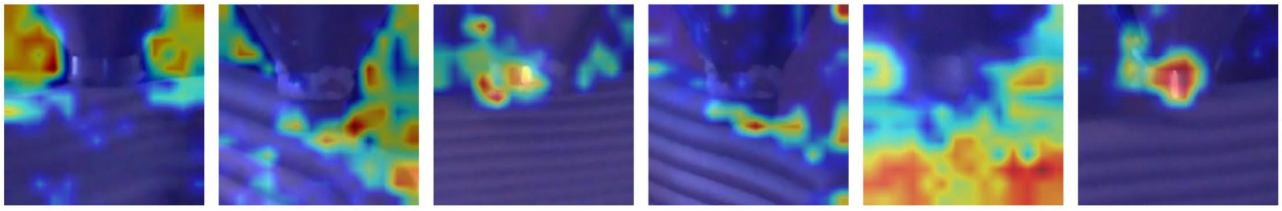


Figure 2.49: Heatmap visualization of different samples for the DINO v2-based model for old dataset

The Grad-CAM visualization results from the new dataset for the DINOv2-based model resemble those obtained with the old dataset. The generated heatmaps (Figure 2.50) illustrate that DINOv2 attends more broadly across the nozzle region and printing material. However, compared to the highly localized and focused attention maps of ResNet-56, DINOv2's attention is more diffused and spread out over larger areas. This is consistent with the transformer's global attention mechanism, which captures more contextual and semantic information but with less spatial precision. The diffuse activations still correspond to semantically meaningful regions relevant to the task but lack the sharp boundaries and crisp localization seen in convolutional networks.

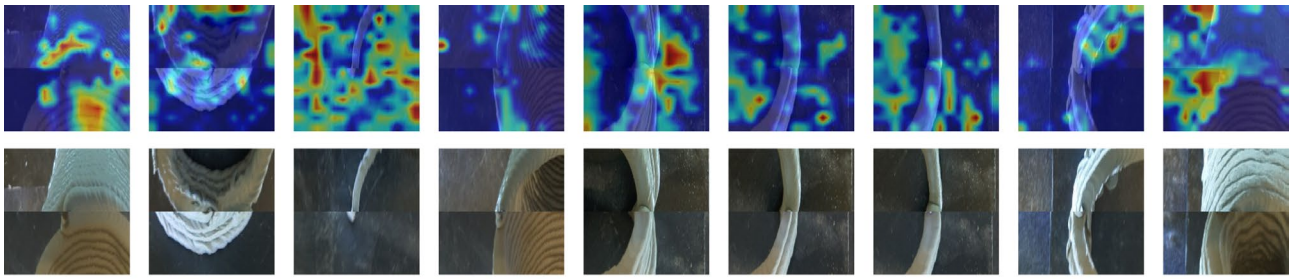


Figure 2.50: Heatmap visualization of different samples for the DINO v2-based model for new dataset

Conclusion

Grad-CAM results demonstrate that the ResNet-56 model progressively learns localized, hierarchical attention from res1 to res4, making it well-suited for tasks involving spatially structured patterns such as 3DP layer analysis. In contrast, DINOv2's pre-trained transformer backbone offers broader contextual attention but lacks fine-grained spatial specificity when used purely as a feature encoder.

This analysis highlights a key trade-off in model design: convolutional networks provide explicit spatial bias that enhances attention localization, while transformers excel in global context modeling at the cost of spatial interpretability when used without fine-tuning.

2.3.3.2 Learning Curves Comparison

As illustrated in Figures 2.51 and 2.52 (old dataset), and Figures 2.53 and 2.54 (new dataset), both models demonstrated steady improvements in training and validation accuracy. However, the DINOv2-based model consistently exhibited faster convergence and superior combined accuracy across both datasets.

- **Training Accuracy:** On the old dataset, the DINOv2 hybrid model achieved over 96% combined training accuracy after approximately 1,500 steps, with individual parameter accuracies exceeding 97%. In comparison, ResNet-56 required more iterations to reach similar accuracy levels but tended to plateau with slightly lower extrusion accuracy. On the new dataset, both models reached a final training accuracy of approximately 97%, with the DINOv2 model maintaining a slight edge in convergence speed.

- **Validation Accuracy:** For the old dataset, the hybrid model's validation accuracy exceeded 94% for each prediction head, stabilizing around 88% combined accuracy. ResNet-56 reached comparable layer height accuracy (~95%) but exhibited lower extrusion accuracy (often below 90%), leading to reduced combined accuracy (~81%). On the new dataset, validation accuracy stabilized around 95% for DINOv2 and 96% for ResNet-56, with the DINOv2 model showing more consistent stability.
- **Loss:** Across both datasets, the DINOv2-based model maintained lower training and validation losses, reflecting improved generalization and more stable optimization compared to ResNet-56.

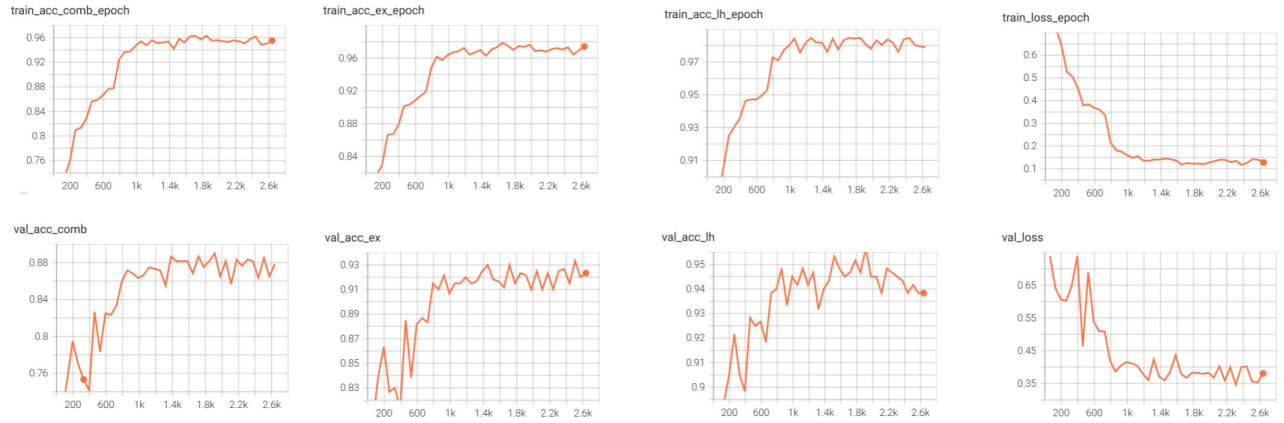


Figure 2.51: Learning curves from DINOv2-Based Hybrid Network Architecture for old dataset

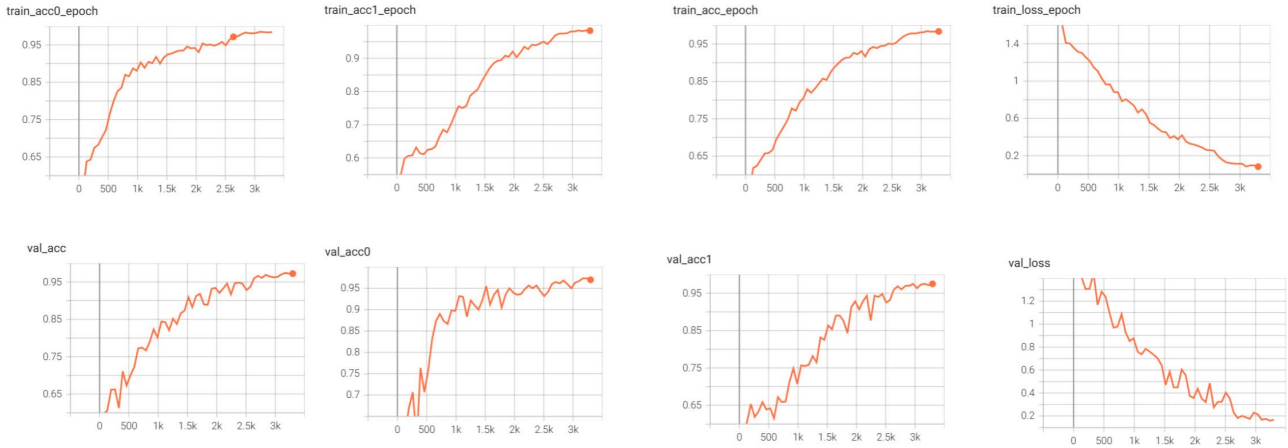


Figure 2.52: Learning curves from Resnet 56 Architecture for old dataset

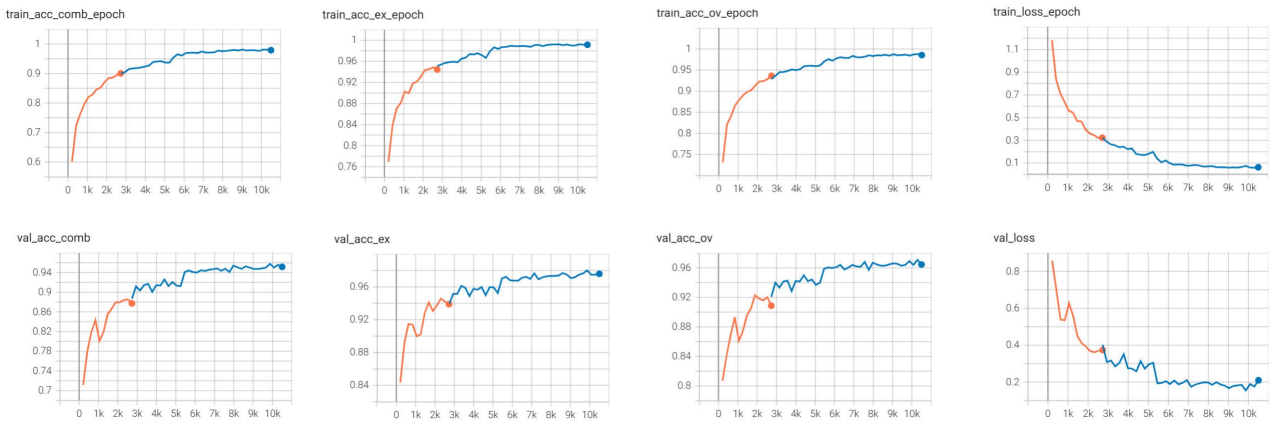


Figure 2.53: Learning curves from DINOv2-Based Hybrid Network Architecture for new dataset

(**Note:** The training process was unexpectedly interrupted at epoch 11 and later resumed, which is reflected in the learning curves as two distinct lines.)

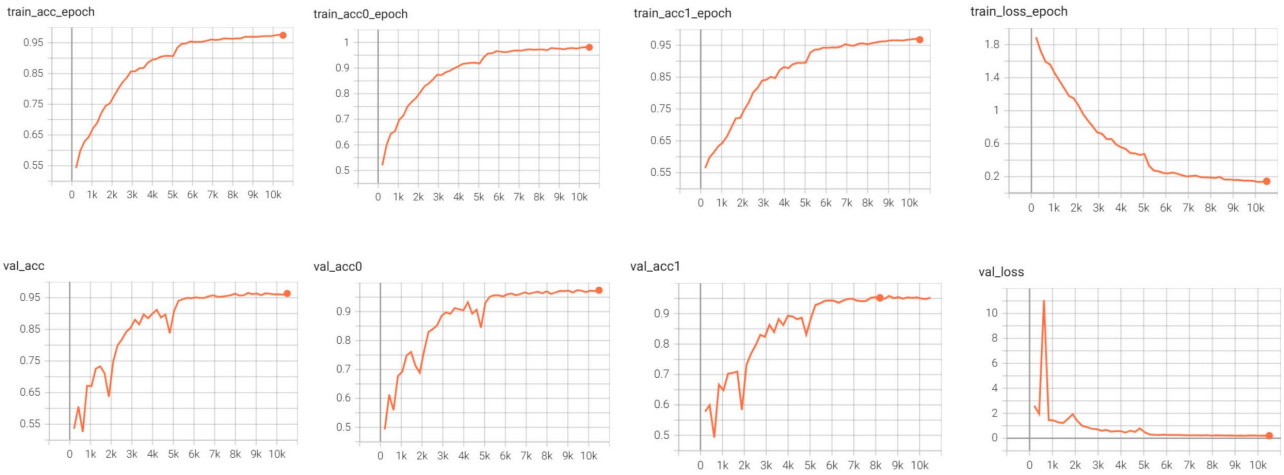


Figure 2.54: Learning curves from Resnet 56 Architecture for new dataset

2.3.3.3 Prediction Accuracy Matrices

Figures 2.55 and 2.56 (old dataset), alongside Figures 2.57 and 2.58 (new dataset), present the normalized confusion matrices for the two architectures.

Old Dataset:

- Parameter 1 (Layer Height):
  The DINOv2-based hybrid network achieved nearly perfect classification, with almost all predictions aligned on the diagonal, indicating excellent distinction among "Low," "Good," and "High" classes. The ResNet-56 model showed more confusion between "Good" and "High" classes, with several misclassifications.
- Parameter 2 (Extrusion Quality):

The DINOv2 model showed better performance, producing fewer misclassifications and more consistent predictions across classes. ResNet-56 exhibited significant confusion, often misclassifying "Good" and "High" labels as "Low," highlighting difficulty distinguishing extrusion quality levels.

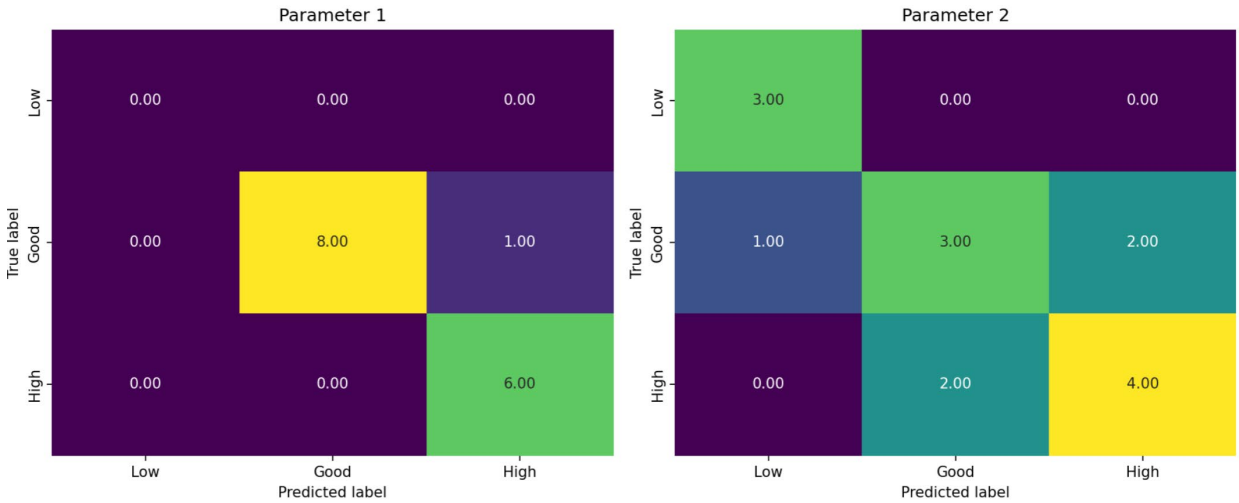


Figure 2.55: Prediction accuracy matrix from DINOv2-Based Hybrid Network Architecture for old dataset

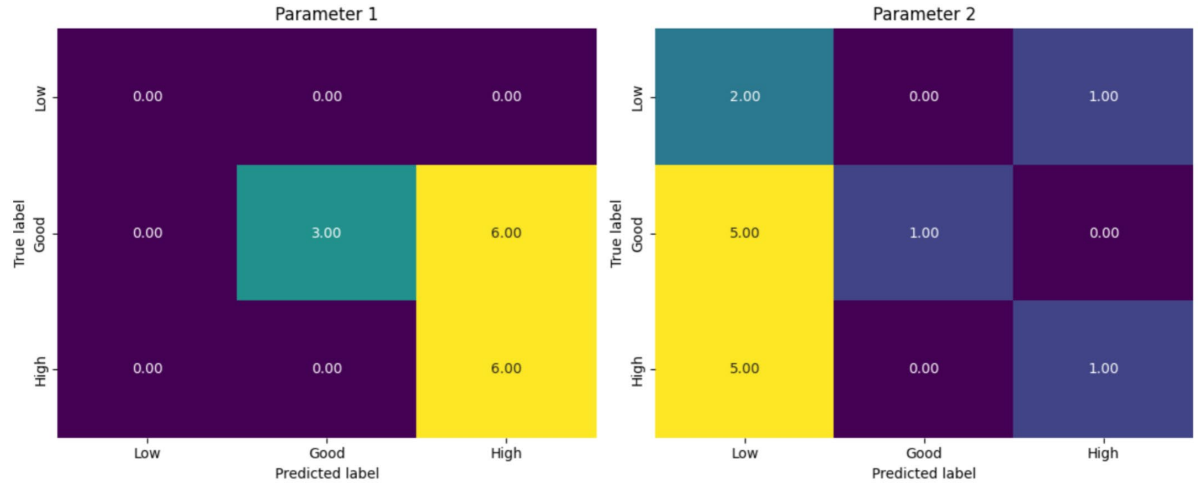


Figure 2.56: Prediction accuracy matrix from Resnet 56 Architecture for old dataset

New Dataset:

- Parameter 1 (Extrusion Quality):
  The DINOv2 model maintained strong classification consistency, with fewer off-diagonal errors compared to ResNet-56. The ResNet-56 model again showed more frequent misclassifications, especially between "Good" and "High" extrusion classes.

- Parameter 2 (Overhang Success):
  DINOv2 achieved better discrimination of "Low," "Good," and "High" overhang success

levels, with limited confusion primarily between adjacent classes. ResNet-56's predictions were less reliable, with notable confusion particularly between "Good" and "Low" classes.

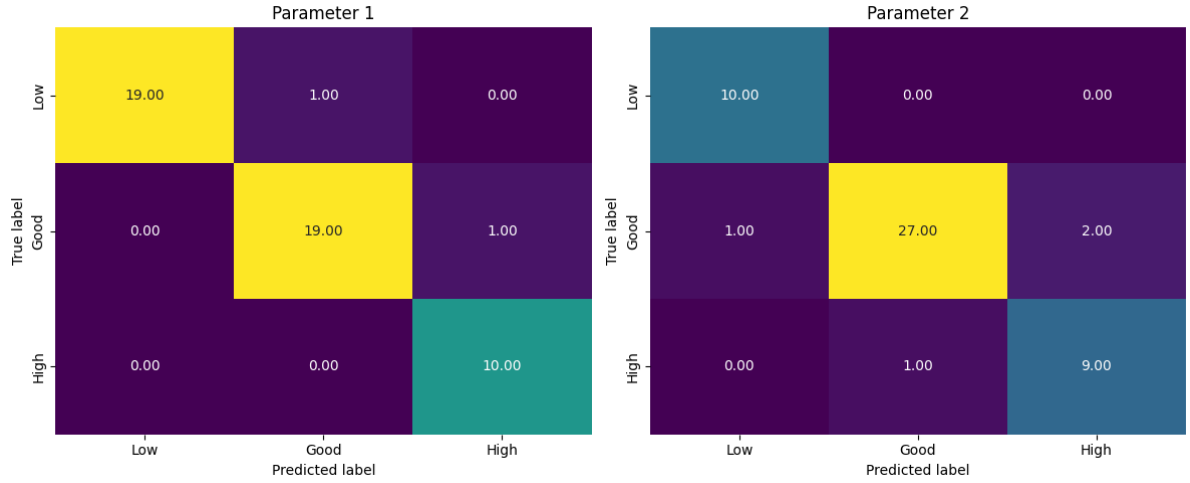


Figure 2.57: Prediction accuracy matrix from DINOv2-Based Hybrid Network Architecture for new dataset

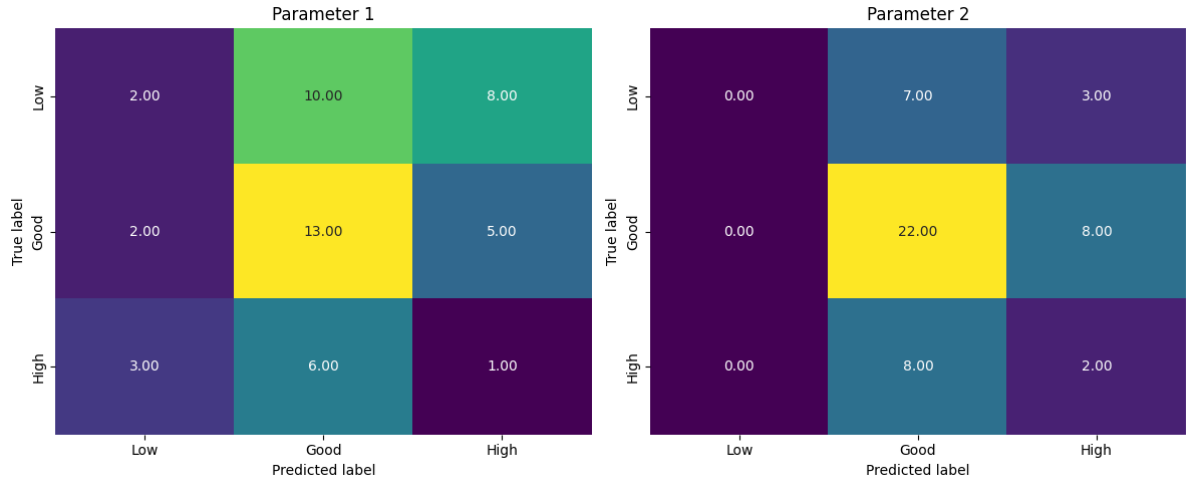


Figure 2.58: Prediction accuracy matrix from Resnet 56 Architecture for new dataset

Overall, the confusion matrices confirm that the DINOv2-Based Hybrid Network surpasses ResNet-56 in prediction accuracy and class separability for both datasets and parameters, demonstrating stronger feature extraction and classification capabilities.

2.3.3.4 Conclusion

Integrating DINOv2 as a feature extractor in the hybrid network architecture substantially improves training efficiency and final prediction accuracy compared to the ResNet-56 baseline. The powerful self-supervised patch token representations from DINOv2 enhance early-stage feature extraction, while retaining the spatial aggregation and dual-head classification design of ResNet-56. This design

achieves faster convergence, superior generalization, and more stable optimization on both the smaller old dataset (layer height and extrusion) and the larger new dataset (extrusion and overhang success).

Although the DINOv2 model requires longer training times (approximately double that of ResNet-56), the gains in accuracy and robustness, especially in distinguishing subtle class differences in extrusion and overhang success, justify the increased computational cost. These results indicate that hybrid architectures combining pretrained vision transformers with customized CNN heads provide a promising framework for complex multi-label image classification tasks in quality assessment applications.

3. Validation

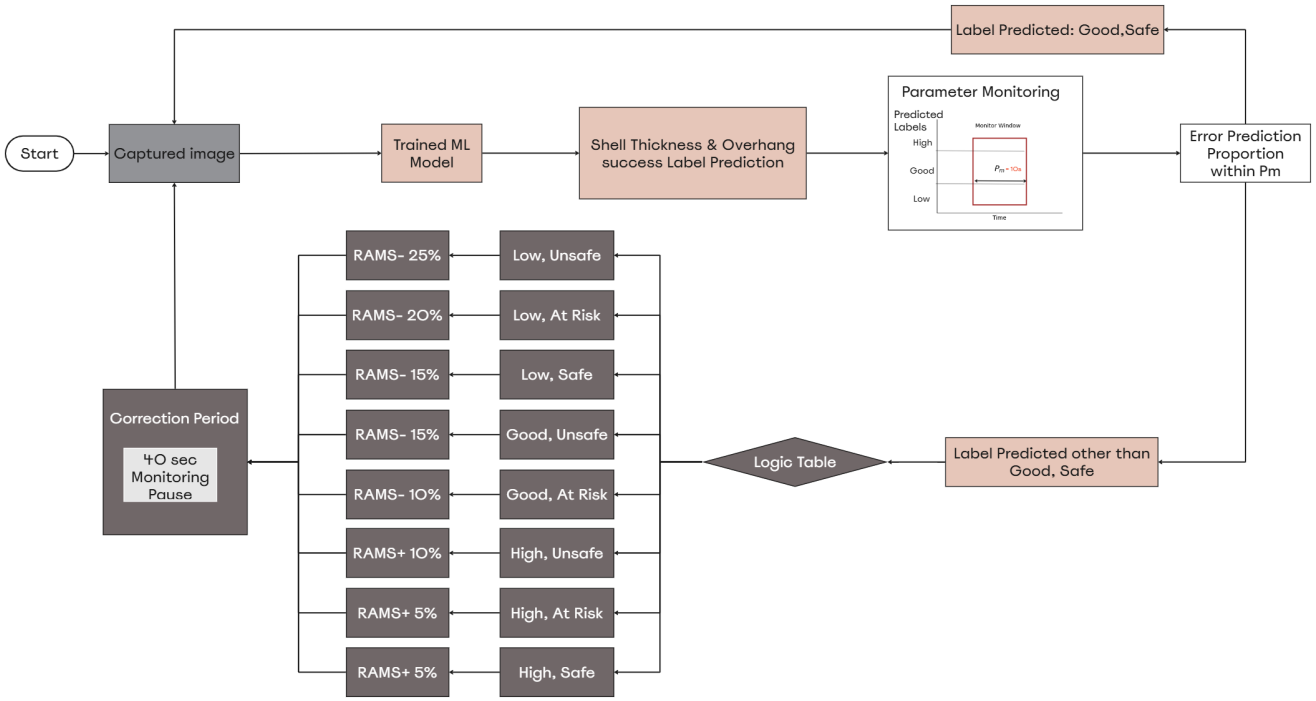


Figure 3.1: the workflow of real-time correction

As illustrated in Figure 3.1, the close loop begins by processing the captured images through the designated ML model (either Resnet 56 or DINO v2 based model) to identify deviations in shell thickness (resulting from low or high extrusion) as well as overhang failures. These predictions are continuously monitored within a specified time window (Pm). If a deviation from the desired shell thickness and/or a reduction in the interlayer contact area below the safe threshold is detected for more than 10 seconds, corrective action is executed.

In such cases, the system sends a command to the UR5 controller to adjust the speed slider, modifying the remaining robot program by overwriting the RAMS parameter. This controls the printing speed by either increasing or decreasing its value based on the observed conditions.

Control Logic for Real-Time Speed Adjustment Based on ML Predictions

To implement real-time correction during the robotic 3DCP process, a heuristic control strategy was developed to dynamically adjust RAMS based on ML predictions of two key quality indicators: extrusion quality and overhang success.

Extrusion quality is classified into three levels: low, good, and high, while overhang success is evaluated as safe, at risk, or unsafe. The control logic prioritizes extrusion quality as the dominant factor

influencing structural integrity, with overhang condition serving as a secondary modifier to fine-tune the speed adjustment, according to conclusion of the label combination and dominant factor analysis in Section 2.3.1.3.

The decision rules are summarized as follows:

- When extrusion quality is low, the system consistently reduces the printing speed, as insufficient material flow compromises interlayer bonding and structural formation:
    - 25% reduction if overhang is unsafe – a significant drop is necessary to prevent print failure caused by both material insufficiency and structural instability.
    - 20% reduction if overhang is at risk – this moderate reduction seeks to increase deposition without sudden changes that might cause instability.
    - 15% reduction if overhang is safe – even if overhang is stable, under-extrusion needs correction to restore intended wall thickness.
- When extrusion quality is good, the system applies corrections only in response to overhang issues:
    - 15% reduction if overhang is unsafe – while extrusion is adequate, overhang failure likely results from poor bonding at curvature, so a moderate reduction is used.
    - 10% reduction if overhang is at risk – early signs of overhang issues are corrected with a slight speed decrease.
    - No adjustment if overhang is safe – both indicators are within acceptable ranges.
- When extrusion quality is high, over-deposition may lead to excessive wall thickness or bulging:
    - 10% increase if overhang is unsafe – a faster movement reduces deposition and helps regain shape fidelity.
    - 5% increase if overhang is at risk – a moderate correction prevents further material buildup.
    - 5% increase if overhang is safe – a minor adjustment is used to slowly return to optimal extrusion levels.

Justification of Adjustment Magnitudes

The selected speed adjustment percentages (ranging from 5% to 25%) are based on empirical testing and observed system sensitivity. Larger changes (e.g., ±25%) are applied only in critical scenarios to rapidly correct severe deviations, while smaller adjustments (e.g., ±5% or ±10%) are used in more stable or borderline conditions to avoid overcompensation or sudden transitions that may cause mechanical jitter or deposition errors. These values were chosen to strike a balance between responsiveness and

system stability, especially considering the inherent lag in material flow response and mechanical inertia in clay-based 3DP.

Logic table			
Extrusion Quality	Overhang Success	Action on RAMS	Rationale
Low	Unsafe	Decrease by 25%	Critical failure in both metrics. Speed must be substantially reduced to compensate for poor material flow and overhang instability.
Low	At Risk	Decrease by 20%	Low extrusion requires correction; overhang is nearing failure—moderate-to-high reduction helps stabilize.
Low	Safe	Decrease by 15%	Extrusion insufficient, but structure is holding. A moderate reduction improves deposition without overcorrecting.
Good	Unsafe	Decrease by 15%	Adequate extrusion but overhang is failing—slightly stronger correction to improve bonding at curvature.
Good	At Risk	Decrease by 10%	Minor instability in overhang—moderate reduction may help improve bonding while maintaining print flow.
Good	Safe	No change	Optimal condition—no adjustment necessary.
High	Unsafe	Increase by 10%	Over-deposition is likely the cause of overhang sagging or collapse—moderate speed increase reduces material input.
High	At Risk	Increase by 5%	Slight over-extrusion and approaching instability—small correction can help avoid future defects.
High	Safe	Increase by 5%	Excess extrusion but stable—small increase helps restore ideal wall thickness gradually.

Table 5: Logic table for speed adjustment strategy

After each execution of adjustment, the system enters a 40-second monitoring pause to allow printing for stabilization before resuming real-time monitoring and potential further correction.

Comparison & evaluation

Comparison and Evaluation of Default Printing versus Real-Time Closed-Loop Printing with

Curved Shell

To evaluate the effectiveness of the real-time closed-loop control system, three prototypes were printed using the same geometry and initial RAMS but under different printing conditions, as shown in the Figure 3.2 below.

Figure 3.2: Comparison of Default Printing (1,2) and Real-Time calibration printing (3)

The first two prototypes on the left (Prototype 1 and Prototype 2) were printed using the default process without any intervention or parameter adjustment during printing. All prints began at 3% of the base speed, corresponding to a slight under-extrusion condition. Prototype 1 exhibited significant under-extrusion, resulting in poor structural integrity. The base collapsed, and weak layer adhesion in the midsection caused fresh layers to fail to bond with previous layers, leading to premature termination of the print. Prototype 2 managed to complete printing, but once the printing reached the region with a large inclined overhang angle, substantial sagging and layer shifts occurred. These defects altered the intended curvature of the overhang and compromised the print quality.

In contrast, Prototype 3 was printed using the same geometry and initial speed but with the real-time closed-loop correction system active. This system dynamically adjusted the RAMS at the large inclined

overhang region to optimize layer adhesion. Specifically, the speed was reduced at the steep overhang to increase the interlayer contacting area. While this resulted in slight over-extrusion locally, enhancing shell thickness beyond the nominal target, it effectively prevented sagging and maintained the designed outer contour of overhang curvature by strengthening the structural stability. Subsequently, the system increased the speed to reduce extrusion and optimize print quality.

However, a limitation was observed near the final layers. The system continued increasing speed adjustments, causing a transition into an under-extrusion state, which is undesirable for print quality and structural integrity. This was attributed to instability in the trained ML model guiding the control adjustments. Despite this drawback, the real-time closed-loop system demonstrated clear advantages in maintaining geometric fidelity and structural performance compared to the default printing approach. The main improvement needed is to expand the dataset and train a more stable and robust ML model.

4. Prototype and Final Product Design

4.1 Prototype Design for Dataset Collection

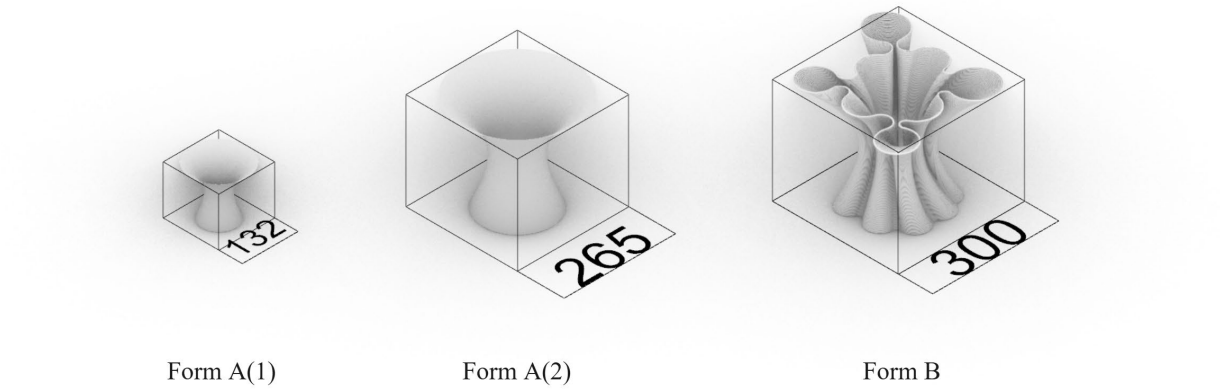


Figure 4.1: the design for validation prototypes

Figure 4.2: the dataset intended to be printed with different inclined angle of overhang

To construct a ML model that can detect detection and process correction in real time, a series of overhang prototypes (shown in Figure 4.2) were designed to serve as the training dataset. However, due to time constraints, only a selection of overhang angles (130°, 140°, 150°, and 160°) were chosen for printing as the dataset (Figure 4.3). The aim of this selection was to include both shallow overhang angles that print can successfully with good extrusion, as well as high-risk steep angles to evaluate the effect of reduced layer contact area.

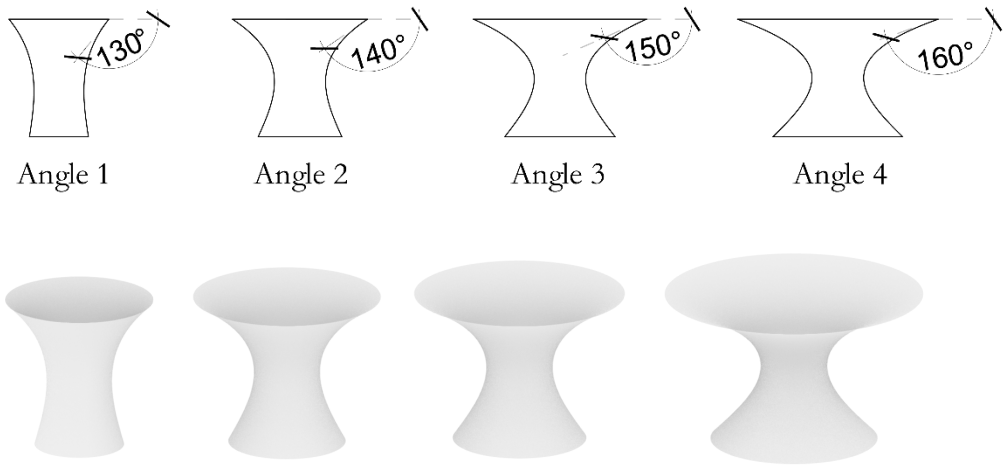


Figure 4.3: the selected overhang geometry to print

The initial geometry (referred to as Form A(1) in Figure 4.1) was deliberately kept simple in form of overhang tower. This choice was motivated by the need to isolate and observe two key visual features during the printing process: shell thickness and interlayer contact area. The assumption is that a simplified geometry, compared to a highly intricate overhang, may allow the model to more clearly learn the correlation between geometric conditions and printing quality, including successful overhang formation.

4.2 Prototype Scale and Printability Constraints

The ultimate goal of this research is to develop a robust real-time defect detection and correction system tailored for construction-scale AM using clay materials. In this context, construction-scale refers to directly building structures on-site, eliminating the need for transporting prefabricated components and reducing traditional complex construction processes.

However, it is noted that construction-scale AM is a long-term goal, not the immediate focus of this project. The current research focuses on developing a foundational framework, covering dataset collection, ML model training, and real-time closed-loop control at a smaller laboratory scale. This groundwork is essential for ensuring reliability and accuracy before advancing to full construction-scale printing, which will involve addressing larger workspace, material handling, and environmental challenges.

As such, the experimental prototypes were designed to be as large as possible within the constraints of the laboratory setup. These constraints include the working envelope of the UR5 robotic arm, the fixed volume of the clay tank (1 L), and the desire to complete a full print cycle without mid-process refilling.

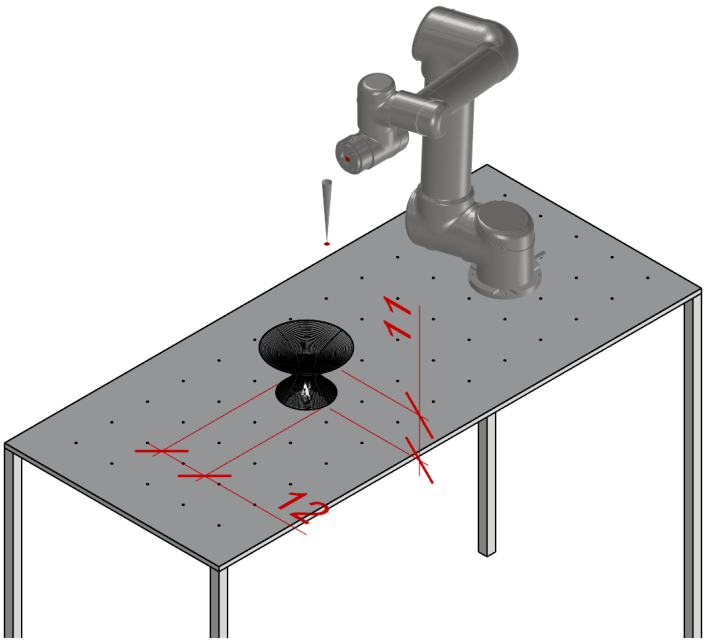


Figure 4.4: the largest prototype can be printed within the constraints of the laboratory setup

Given these limitations, the chosen dimensions for Form A(1) with a height of 110 mm represent the maximum viable print size that avoids the need to refill the clay tank during printing, even under conditions of over-extrusion. This size ensures uninterrupted printing while still offering enough geometric complexity to observe meaningful deviations in overhang performance. A layer height of 1.5 mm was adopted throughout the experiments to balance print resolution with time efficiency and to maintain adequate interlayer adhesion.

4.3 Final Product Design and Validation Strategy

During the validation phase, two types of test forms will be printed to assess the scalability and generalizability of the trained ML model.

- **Validation with Known Geometry (Form A):**
  The first step involves printing the same basic form used during dataset collection (Form A(1)). This allows direct evaluation of the model's predictive accuracy under familiar geometric and scale conditions.
- **Validation with Upscaled Geometry (Form A(2)):**
  Next, an enlarged version of the same form (Form A(2), height 265 mm) will be printed to examine the model's **scalability**. This prototype occupies the full vertical reach of the UR5 arm and represents the **maximum print size currently achievable**. Successful predictions in this case would indicate the model's capacity to maintain accuracy across different scales of the same geometry.
- **Validation with Complex Geometry (Form B):**
  Finally, a more intricate design (Form B), featuring multiple curvatures and overhangs, will be printed to test the model's generalization ability. This form was not included in the training dataset, so it provides a critical benchmark for evaluating whether the ML system can adapt to previously unseen overhang configurations. Positive results here would validate the robustness of the system for diverse clay-based architectural components.

Note on Validation with Upscaled and Complex Geometries

Due to time constraints within the scope of the current project, the implementation and testing of validation using Upscaled Geometry (Form A(2)) and Complex Geometry (Form B) have not yet been completed. These steps are planned for future work and are essential for fully assessing the scalability and generalizability of the trained ML model beyond the initial prototype scale and simplified geometries.

4.4 Slicing and Toolpath Generation

As discussed in previous section 2.1.1.2, all prototype geometries were sliced using a customized Grasshopper script that divides the volume into discrete layers and generates continuous helical toolpaths. The script outputs machine-readable URScript files containing *Move j* commands aligned with the robot's kinematics. Each toolpath is carefully generated to maintain a consistent nozzle-substrate distance and orientation, particularly in regions of overhang, where maintaining tangency and shell thickness is critical to structural success.

This strategy, combining procedural form generation, real-time monitoring, and data-driven calibration, is designed to ensure that both the methodology (ML-based correction) and the material system (clay-based LDM printing) are applied meaningfully, with consideration for scale, fabrication feasibility, and architectural relevance.

5. Conclusion

The experimental setup utilized a UR5 robotic arm, which was successfully configured to establish a foundational platform for implementing real-time speed adjustments. This research developed and validated a ML-based closed-loop control system for robotic 3DCP, enabling real-time detection and correction of extrusion defects, particularly in overhang structures. By integrating visual data and a multi-objective neural network model, the system dynamically adjusts printing speed to maintain optimal shell thickness and interlayer adhesion, significantly improving print quality and structural integrity. Among the evaluated models, DINOv2-Based Hybrid Network demonstrated strong capability in capturing spatial features relevant to printing defects.

The closed-loop feedback system showed promising robustness at the laboratory scale, laying groundwork for scaling up to construction-scale 3DP. Future research should emphasize expanding dataset diversity, improving model stability, and extending applicability to other materials such as concrete. Additionally, refining robotic arm motion control remains crucial to achieve more reliable and stable printing performance.

Overall, this study represents an important step towards automated, intelligent 3DCP, with potential to support complex architectural geometries and more efficient fabrication workflows.

6. Discussion

Follow-Up Questions and Future Research Directions

Building on the foundation established in this research, several critical questions remain to be addressed to further advance real-time defect detection and correction in robotic 3DCP.

One key challenge is how to balance the competing influences of layer self-weight and extrusion quality to ensure successful overhang printing. As layers accumulate, their own weight can cause deformation or sagging, especially at steep overhang angles. Future research could explore predictive models that estimate the evolving stress and load during printing, enabling preemptive adjustments in extrusion rate or print speed to mitigate structural instability.

Another important area is the integration of predictive stress and load analysis into the printing process. Current real-time control relies primarily on visual feedback; however, incorporating physical simulations or ML-based prediction models could allow for anticipatory corrections, improving print fidelity and reducing the need for reactive interventions. This approach might involve combining pre-print optimization with real-time adjustments to balance print quality and structural integrity dynamically.

The correction methodology that simultaneously optimizes multiple objectives, such as maximizing both extrusion quality and overhang success, can be further improved. Since these objectives often conflict, there is a set of optimal trade-off solutions rather than a single best solution. Techniques like Pareto optimization can be applied to explore this solution space, enabling designers to select configurations based on specific performance priorities. Alternatively, weighted sum methods offer a simpler approach by combining multiple objectives into a single scalar score, though proper normalization is essential to prevent bias due to differing scales. More advanced strategies, including multi-objective Bayesian optimization and reinforcement learning, can be possible solutions for intelligently navigating complex control scenarios and discovering effective parameter settings.

Finally, an essential consideration is how to balance necessary parameter adjustments during printing with the overall consistency and appearance of the final product. Excessive or frequent corrections may cause visual artifacts or unintended deformation. Future work should investigate strategies that optimize correction frequency and magnitude.

Future Application:

(1) Potential For Material Change: Concrete

Previously, I engaged in discussions with the concrete 3DP technicians at BouwLab (a leading Dutch company specializing in digitalization and industrialization across the entire construction

chain). During these exchanges, we studied the requirements and feasibility of applying the current real-time closed-loop correction system to concrete 3DP.

Concrete shares many similarities with clay as a cementitious material in AM. However, concrete as a construction material, presents greater challenges due to its complex rheology, faster setting, curing behavior, and higher sensitivity to environmental factors, increasing the need for real-time monitoring and correction in 3DP.

As noted by concrete 3DP technicians, continuous adjustments are crucial in concrete 3DP to handle fluctuations in moisture, mixture consistency, and temperature. These adjustments often involve modifying pumping speed, rotor-stator settings, or applying heat to accelerate curing. Multi-head machine learning models that integrate sensor data (e.g., water flow, temperature, pressure) with real-time imaging shows potential to predictively optimize printing quality and reduce defects without manual intervention

However, this requires extensive data collection and tailored training. It remains to be seen if the current system can evolve into a robust, scalable solution managing multiple variables simultaneously for high-quality, construction-scale concrete additive manufacturing.

(2) Nozzle Size Change to Test Scale-Up Possibility of ML

Scaling the 3DP process by varying nozzle size can be an essential step to assess the adaptability and robustness of the developed ML model before applying it to larger scale 3DP.

Investigating the model's performance across multiple nozzle sizes will help determine the ML model's scalability and whether transfer learning strategies are needed. This line of inquiry will pave the way for a more flexible correction system, which is crucial for practical deployment in industrial or construction contexts where nozzle sizes increase and extrusion shapes may vary.

(3) Scaling Up to Construction-Scale AM Using a 6-Axis Gantry System

As mentioned in previous sections, the broader aim of this research is to create a closed-loop control system capable of supporting construction-scale AM with clay and potentially other materials. Moving from the laboratory-scale UR5 robotic arm setup to a full-scale 6-axis gantry system introduces significant challenges such as managing larger workspaces, delivering materials at higher volumes, and ensuring stable environmental conditions throughout extended printing processes. Construction-scale AM involves on-site fabrication of architectural or structural elements, which can reduce reliance on prefabrication, lower manual labor, decrease construction waste, and enhance overall sustainability. This paradigm shift has the potential to transform traditional building methods by enabling more efficient workflows and complex design possibilities.

7. Reflection

7.1 Introduction

This reflection provides a critical review of the graduation process and explores the potential real-world impact of my thesis in the broader context of the architectural and construction industries.

My thesis focuses on intelligent manufacturing, aiming to develop an AI-assisted calibration system for clay robotic 3DP. This interdisciplinary research integrates robotics, computer science, artificial intelligence, and AM. It combines robotic programming and computational design from the Design Informatics department in the Faculty of Architecture with experimental innovation from the Shaping Matter Lab in the Faculty of Aerospace Engineering, which focuses on bio-inspired, sustainable, and intelligent materials through AM. Ultimately, this research applies advanced AI-assisted robotic 3DP techniques to the field of architecture.

From my point of view, architecture, as a traditionally slow-to-adapt field, is hard to engage deeply with cutting-edge technologies such as artificial intelligence at the practical level. However, when I set foot in the realm of computational design and intelligent manufacturing, I realized that it is the invention of tools and fabrication methods, rather than certain design concepts, that has historically driven iteration and progress in this industry. Automation improves the productivity, precision, and sustainability of architectural projects while reducing dependency on manual labor, which further pushes the whole industry to evolve beyond its comfort zone. Intelligent manufacturing is therefore an intermediary for building technologists to rethink the boundaries of the architecture industry and embrace cross-disciplinary innovations that can propel the field forward.

Thus, the ambition of this thesis is to fully automate the AM of construction materials. Rather than relying on manual observation and adjustment, artificial intelligence takes over this redundant task with higher accuracy and less material waste. As a long-term goal, this research serves as a foundation for scaling the system, from small-scale robotic arms to 6-axis gantry systems for on-site, mold-free construction 3DP, with potential adaptation from clay to concrete, laying the foundation for fully automated, construction-scale AM.

7.2 Research Journey & Personal Development

7.2.1 Starting Point

This research journey has involved continuous learning, technical challenges, and a growing sense of resilience. It began with inspiration from the CORE electives. In the CORE project, our team developed an open-loop calibration system for clay 3DP using computer vision and ML. I implemented the Attention-56 deep learning network and real-time material flow control for adaptive pre-print calibration. However, the system showed limitations in automation and accuracy, and the

unpredictable behavior of clay revealed the need for a more robust solution.

Building on this foundation, my graduation thesis focused on a closed-loop calibration system, with a greater emphasis on architectural and structural design. Switching from a WASP clay printer to a 6-axis robotic arm equipped with a clay extruder allowed for more dynamic control. The prototype evolved into an overhang structure, allowing exploration of architectural expression. To improve ML performance, I trained and compared a second model, DINOv2, to evaluate its effectiveness against Attention-56.

7.2.2 Evolution of the Topic

While the research direction remained consistent, its scope narrowed significantly with guidance from my mentors. Initially (P1), the objectives were:

- Closed-loop calibration
- ML integration
- Construction-scale 3DP
- Structural stability monitoring
- Robotic arm integration

Due to time and resource limitations, P2 refined the focus to:

- Closed-loop calibration with a balance between local and global design coherence
- Multi-objective ML for extrusion quality and structural adhesion optimization
- Robotic 3DP
- Robotic arm integration

At this stage, the gantry system and structural stability validation were excluded, as both the extruder and feeding system for the gantry had to be developed from scratch, which was an unrealistic goal within the timeframe of a master's thesis.

By P3, the focus had narrowed to:

- Closed-loop calibration
- Multi-objective ML for extrusion quality optimization and overhang success
- Robotic 3DP

Although this narrowing process was lengthy, it allowed me to frame my project within a larger research context. The original idea from P1 was a complete workflow for automated on-site 3DP

calibration. However, to make the project feasible for a master's thesis, I had to select and implement the most critical parts.

7.2.3 Learning Process

My background in Building Technology helped me design the workflow and experiments from a designer's perspective. For example, I was able to design and fabricate the frame needed for the experiment using PLA 3DP. My prior experience with the UR5 robotic arm in the Design Informatics course also made it easier to generate robotic printing programs using Grasshopper.

However, my coding experience was limited. Aside from a two-week crash course during the CORE project, I had no background in Python. This project demanded extensive programming, requiring me to self-learn and debug continuously. Tasks like setting up Raspberry Pi cameras, writing scripts to take synchronized photos, logging data to CSV files, correcting image perspectives, merging images, training ML models, and generating a complete real-time correction workflow all required coding. Each step presented new, unexpected challenges and took considerable time to resolve.

A significant technical hurdle was establishing real-time control between my PC and the robotic arm. Despite two weeks of attempts, including scripting, consulting manuals, forums, and COMAU technicians, I discovered that the COMAU controller was outdated and incompatible. Budget limitations prevented upgrading the hardware, so I switched to the UR5 robotic arm in the end, which supports real-time connectivity.

Connecting the Raspberry Pi, UR5 robotic arm, and PC involved IP address reconfiguration, which took a long time to align the devices on the same frequency band. Adjusting the UR5 movement speed also required numerous troubleshooting attempts, as code that ran successfully on my PC did not always elicit a response from the robot. Eventually, through persistence, I resolved these issues.

7.2.4 Delays and Adaptation

The switch to UR5 brought additional delays. It was already scheduled for use in the Design Informatics course and by another graduate student, which meant I couldn't access it for a month. Compounding the issue, I experienced a sudden health problem and had to return to my home country for recovery. During this time, I adjusted my workflow, setting aside tasks that required the UR5 and focusing on other aspects: training DINOv2, generating Grad-CAM visualizations, and developing the Raspberry Pi camera system.

Meanwhile, preparing the clay extruder setup for both COMAU and UR5 took considerable time, as it needed to be designed, fabricated, and installed from scratch.

Hardware integration was particularly unpredictable. For instance, improper clay consistency often led to clogging in the extruder, stopping the internal blades and halting extrusion. Excess internal pressure caused similar blockages. Resolving these issues required manually remixing clay,

disassembling, and cleaning the extruder repeatedly. These failures were time-consuming and physically demanding.

Despite these setbacks, this intense problem-solving process allowed me to develop a wide range of skills: coding, robotic control, ML, system setup, and hardware integration. I learned to design experiments methodically, troubleshoot effectively, and construct a coherent, goal-oriented workflow. Additionally, these obstacles taught me the importance of adaptability, patience, and building in time buffers. Most importantly, I cultivated a persistent, solution-driven mindset and developed resilience that will benefit me in future research and practice.

7.3 Societal Impact

The completion of this thesis will not mark an end, but the opening chapter of a much broader and ongoing research journey. Future work could focus on scaling the system for construction-scale applications using a gantry setup and adapting the ML model for different materials, such as concrete. If successful, this would significantly reduce printing errors and human labor in real-world construction projects, while also minimizing material and time waste.

Once implemented in on-site 3DP, this system could promote the use of naturally sourced extrudable materials and reduce reliance on traditional, resource-heavy methods. With minimal or no human intervention, the printing process would become safer and more efficient, enhancing the feasibility of digitally fabricated architecture.

Another bold idea: a fully automated, error-free 3DP process could enable space-based construction. For example, using a mixture of moon soil and binders, the calibration system could autonomously 3D print habitats on the moon with minimal human input.

7.4 Future Direction

Motivated by the challenges and fulfillment of this research, I am highly motivated to pursue a PhD in intelligent construction. I am eager to deepen my knowledge of robotic operating systems (ROS), ML, and real-world applications of robotics in architecture. Discussions with professionals in the field confirmed that there is strong interest in applying such systems to large-scale concrete printing. The practical relevance of this research excites me, and I hope to contribute meaningfully to the development of intelligent robotic systems that advance sustainable construction and architectural innovation.

7.5 Conclusion

The most valuable takeaway from this thesis is not a specific technical skill but the ability to approach complex problems creatively and persistently. This journey has strengthened my confidence, expanded my interdisciplinary skill set, and inspired a long-term commitment to intelligent construction and sustainable architectural practices.

8. Acknowledgment

I would like to express my sincere gratitude to my mentors, Dr. Serdar Aşut, for his invaluable guidance on robotic control, as well as his comprehensive tutoring on the overall workflow and methodology throughout this project. My sincere thanks also go to Dr. Charalampos Andriotis for his insightful support and advice regarding machine learning inquiries, which greatly enriched this study.

I am grateful to Prof. mr. dr. M.N. Boeve, the delegate of the Board of Examiners, for her encouragement and support during my graduation process.

Special thanks are extended to Paul de Ruiter, Vera Laszlo, and Henry Kiksen for their assistance and contributions during the 3DCP experiments conducted in the LAMA lab. I also wish to acknowledge Shantha Kilambi and Gustavo Asai, PhD candidates from the Shaping Matter Lab at TU Delft, for their additional support and valuable discussions on ML methods and overall methodology.

Finally, I deeply appreciate the unwavering encouragement and support from my family and dear friends throughout this challenging journey.

9. Reference

Akhavan, J., Lyu, J., & Manoochehri, S. (2024). A deep learning solution for real-time quality assessment and control in additive manufacturing using point cloud data. *Journal of Intelligent Manufacturing*, *35*(3), 1389-1406. https://doi.org/10.1007/s10845-023-02121-4

Ansari, M. A., Crampton, A., & Parkinson, S. (2022). A layer-wise surface deformation defect detection by convolutional neural networks in laser powder-bed fusion images. *Materials*, *15*(20), 7166.

Aşut, S., Ding, X., Guha, S., Ryu, S., & Wei, W. (2025, 01-05 September). *Enhancing 3D Clay Printing with Computer Vision and Deep Learning* eCAADe Annual Conference 2025, Ankara, Türkiye. (accepted, in press)

Avro, S. S., Atikur Rahman, S. M., Tseng, T.-L., & Fashiar Rahman, M. (2024). A deep learning framework for automated anomaly detection and localization in fused filament fabrication. *Manufacturing Letters*, *41*, 1526-1534. https://doi.org/https://doi.org/10.1016/j.mfglet.2024.09.179

Bhooshan, S., Mele, T., & Block, P. (2018). Equilibrium-Aware Shape Design for Concrete Printing. In (pp. 493-508). https://doi.org/10.1007/978-981-10-6611-5_42

Brion, D. A., Shen, M., & Pattinson, S. W. (2022). Automated recognition and correction of warp deformation in extrusion additive manufacturing. *Additive Manufacturing*, *56*, 102838.

Brion, D. A. J., & Pattinson, S. W. (2022). Generalisable 3D printing error detection and correction via multi-head neural networks. *Nature Communications*, *13*(1), 4654. https://doi.org/10.1038/s41467-022-31985-y

Delgado Camacho, D., Clayton, P., O'Brien, W. J., Seepersad, C., Juenger, M., Ferron, R., & Salamone, S. (2018). Applications of additive manufacturing in the construction industry – A forward-looking review. *Automation in Construction*, *89*, 110-119. https://doi.org/https://doi.org/10.1016/j.autcon.2017.12.031

Ding, X., Aşut, S., & Andriotis, C. (2025). Closed-Loop Control of 3D Clay Printing Using Machine Learning. In *Digitalisation of the Built Environment: 4th 4TU/14UAS Research Day* (pp. 93-99). Hanze University of Applied Sciences.

Farahbakhsh, M., Kalantar, N., & Rybkowski, Z. (2021). *Impact of Robotic 3D Printing Process Parameters on Bond Strength: A Systematic Analysis Using Clay-Based Materials*.

Fastermann, P. (2016). *3D-Drucken* (2 ed.). Springer Berlin, Heidelberg. https://doi.org/https://doi.org/10.1007/978-3-662-49866-8

Fontúrbel, C., Cisnal, A., Fraile-Marinero, J. C., & Pérez-Turiel, J. (2023). Force-based control strategy for a collaborative robotic camera holder in laparoscopic surgery using pivoting motion [Original Research]. *Frontiers in Robotics and AI*, *Volume 10 - 2023*. https://doi.org/10.3389/frobt.2023.1145265

Froes, F., Boyer, R., & Rao, J. (2019). *Additive manufacturing for the aerospace industry*. https://doi.org/10.1016/C2017-0-00712-7

Fu, T.-H., Huang, T.-Y., & Li, D.-R. (2025). *Machine Learning-Enabled Process Monitoring and Error Detection in Material Extrusion-Based Additive Manufacturing*. https://doi.org/10.1115/IMECE2024-147155

Goh, G. D., Hamzah, N. M. B., & Yeong, W. Y. (2022). Anomaly Detection in Fused Filament Fabrication Using Machine Learning. *3D Printing and Additive Manufacturing*, *10*(3), 428-437. https://doi.org/10.1089/3dp.2021.0231

Grigoriadis, K., & Lee, G. (2024). *3D Printing and Material Extrusion in Architecture*. DOM publishers.

Gunasegaram, D. R., Murphy, A. B., Matthews, M., & DebRoy, T. (2021). The case for digital twins in metal additive manufacturing. *Journal of Physics: Materials*, *4*(4), 040401.

Gürsoy, B. (2018). From Control to Uncertainty in 3D Printing with Clay. In A. Kepczynska-Walczak & S. Bialkowski (Eds.), *36th International Conference on Education and Research in Computer Aided Architectural Design in Europe, eCAADe 2018* (pp. 21-30): Education and research in Computer Aided Architectural Design in Europe.

Haghiashtiani, G. A.-O., Qiu, K. A.-O., Zhingre Sanchez, J. D., Fuenning, Z. J., Nair, P. A.-O., Ahlberg, S. E., Iaizzo, P. A., & McAlpine, M. A.-O. (2020). 3D printed patient-specific aortic root models with internal sensors for minimally invasive applications. (2375-2548 (Electronic)).

Jang, S., Park, S., & Bae, C. J. (2020). Development of ceramic additive manufacturing: process and materials technology. (2093-985X (Electronic)).

Jiang, J., Xiong, Y., Zhang, Z., & Rosen, D. W. (2022). Machine learning integrated design for additive manufacturing. *Journal of Intelligent Manufacturing*, *33*(4), 1073-1086.

Jin, Z., Zhang, Z., Ott, J., & Gu, G. X. (2021). Precise localization and semantic segmentation detection of printing conditions in fused filament fabrication technologies using machine learning. *Additive Manufacturing*, *37*, 101696. https://doi.org/https://doi.org/10.1016/j.addma.2020.101696

Jin, Z., Zhang, Z., Shao, X., & Gu, G. X. (2023). Monitoring Anomalies in 3D Bioprinting with Deep Neural Networks. *ACS Biomaterials Science & Engineering*, *9*(7), 3945-3952. https://doi.org/10.1021/acsbiomaterials.0c01761

Khan, M. F., Alam, A., Siddiqui, M. A., Alam, M. S., Rafat, Y., Salik, N., & Al-Saidan, I. (2021). Real-time defect detection in 3D printing using machine learning. *Materials Today: Proceedings*, *42*, 521-528.

Kim, Y., & Park, S. (2023). Highly Productive 3D Printing Process to Transcend Intractability in Materials and Geometries via Interactive Machine-Learning-Based Technique. *Advanced Intelligent Systems*, *5*, 2200462. https://doi.org/10.1002/aisy.202200462

Klug, C., Herzog, S., Kaletsch, A., Broeckmann, C., & Schmitz, T. H. (2022). Forming of Additively Manufactured Ceramics by Magnetic Fields. *Ceramics*, *5*(4), 947-960.

Kontovourkis, O., & Tryfonos, G. (2020). Robotic 3D clay printing of prefabricated non-conventional wall components based on a parametric-integrated design. *Automation in Construction*, *110*, 103005. https://doi.org/https://doi.org/10.1016/j.autcon.2019.103005

Lee, S.-M., & Park, S.-H. (2025). Autonomous in-situ defect detection and correction in additive-lathe 3D printing process using variational autoencoder model. *Additive Manufacturing*, *98*, 104635. https://doi.org/https://doi.org/10.1016/j.addma.2024.104635

Li, Z., Zhang, Z., Shi, J., & Wu, D. (2019). Prediction of surface roughness in extrusion-based additive manufacturing with machine learning. *Robotics and Computer-Integrated Manufacturing*, *57*, 488-495.

López-Valdeolivas, M., Liu, D., Broer, D. A.-O., & Sánchez-Somolinos, C. A.-O. (2018). 4D Printed Actuators with Soft-Robotic Functions. LID - 10.1002/marc.201700710 [doi]. (1521-3927 (Electronic)).

Lu, L., Hou, J., Yuan, S., Yao, X., Li, Y., & Zhu, J. (2023). Deep learning-assisted real-time defect detection and closed-loop adjustment for additive manufacturing of continuous fiber-reinforced polymer composites. *Robotics and Computer-Integrated Manufacturing*, *79*, 102431.

Maxime Oquab, T. D., Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, Piotr Bojanowski. (2024). DINOv2: Learning Robust Visual Features without Supervision. *Transactions on Machine Learning Research*.

Muñoz, V. F., Garcia-Morales, I., Fraile-Marinero, J. C., Perez-Turiel, J., Muñoz-Garcia, A., Bauzano, E., Rivas-Blanco, I., Sabater-Navarro, J. M., & Fuente, E. d. l. (2021). Collaborative Robotic Assistant Platform for Endonasal Surgery: Preliminary In-Vitro Trials. *Sensors*, *21*(7), 2320. https://www.mdpi.com/1424-8220/21/7/2320

Nematollahi, B., Xia, M., & Sanjayan, J. (2017). Current progress of 3D concrete printing technologies. ISARC. Proceedings of the international symposium on automation and robotics in construction,

Ngo, T. D., Kashani, A., Imbalzano, G., Nguyen, K. T. Q., & Hui, D. (2018). Additive manufacturing (3D printing): A review of materials, methods, applications and challenges. *Composites Part B: Engineering*, *143*, 172-196. https://doi.org/https://doi.org/10.1016/j.compositesb.2018.02.012

OpenCV. Retrieved May 22 from https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html

Oti, J. E. (2010). *The development of unfired clay building materials for sustainable building construction*. University of South Wales (United Kingdom).

Paraskevoudis, K., Karayannis, P., & Koumoulos, E. P. (2020). Real-time 3D printing remote defect detection (stringing) with computer vision and artificial intelligence. *Processes*, *8*(11), 1464.

Parkes, J. (2021a). *Joris Laarman's 3D-printed stainless steel bridge finally opens in Amsterdam*. dezeen. Retrieved 09/06/2025 from https://www.dezeen.com/2021/07/19/mx3d-3d-printed-bridge-stainless-steel-amsterdam/

Parkes, J. (2021b). *Tecla house 3D-printed from locally sourced clay*. Retrieved 08/06/2025 from https://www.dezeen.com/2021/04/23/mario-cucinella-architects-wasp-3d-printed-housing/

Pérez-Ubeda, R., Zotovic-Stanisic, R., & Gutiérrez, S. C. (2020). Force Control Improvement in Collaborative Robots through Theory Analysis and Experimental Endorsement. *Applied Sciences*, *10*(12), 4329. https://www.mdpi.com/2076-3417/10/12/4329

Placone, J. K., & Engler, A. J. (2018). Recent Advances in Extrusion-Based 3D Printing for Biomedical Applications. (2192-2659 (Electronic)).

Ramiah, K., & Pandian, P. (2023). Effect of process parameters on the strength of ABS based FDM prototypes: novel machine learning based hybrid optimization technique. *Materials Research Express*, *10*(2), 025305.

Rosenthal, M., Henneberger, C., Gutkes, A., & Bues, C.-T. (2018). Liquid Deposition Modeling: a

promising approach for 3D printing of wood. *European Journal of Wood and Wood Products*, *76*(2), 797-799. https://doi.org/10.1007/s00107-017-1274-8

Singh, T., Kumar, S., & Sehgal, S. (2020). 3D printing of engineering materials: A state of the art review. *Materials Today: Proceedings*, *28*, 1927-1931. https://doi.org/https://doi.org/10.1016/j.matpr.2020.05.334

Teizer, J., Blickle, A., King, T., Leitzbach, O., & Guenther, D. (2016). Large scale 3D printing of complex geometric shapes in construction. ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction,

Toorandaz, S., Taherkhani, K., Liravi, F., & Toyserkani, E. (2024). A novel machine learning-based approach for in-situ surface roughness prediction in laser powder-bed fusion. *Additive Manufacturing*, *91*, 104354. https://doi.org/https://doi.org/10.1016/j.addma.2024.104354

Van Oosterwyck, N. (2018). *Real Time Human Robot Interactions and Speed Control of a Robotic Arm for Collaborative Operations*

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., & Tang, X. (2017, 21-26 July 2017). Residual Attention Network for Image Classification. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),

Witte, D. (2022). *Clay Printing: The Fourth Generation Brickwork*. https://doi.org/10.1007/978-3-658-37161-6

Wolf, A., Rosendahl, P. L., & Knaack, U. (2022). Additive manufacturing of clay and ceramic building components. *Automation in Construction*, *133*, 103956. https://doi.org/https://doi.org/10.1016/j.autcon.2021.103956

Wu, P., Zhao, X., Baller, J. H., & Wang, X. (2018). Developing a conceptual framework to improve the implementation of 3D printing technology in the construction industry. *Architectural Science Review*, *61*(3), 133-142.

Yang, H.-Q., Klug, C., & Schmitz, T. H. (2023). Fiber-Reinforced Clay: An Exploratory Study on Automated Thread Insertion for Enhanced Structural Integrity in LDM. *Ceramics*, *6*(3), 1365-1383.

Zhang, J., Wang, P., & Gao, R. X. (2019). Deep learning-based tensile strength prediction in fused deposition modeling. *Computers in industry*, *107*, 11-21.

Zubayer, M. H., Xiong, Y., Wang, Y., & Imdadul, H. M. (2024). Enhancing additive manufacturing precision: Intelligent inspection and optimization for defect-free continuous carbon fiber-reinforced polymer. *Composites Part C: Open Access*, *14*, 100451. https://doi.org/https://doi.org/10.1016/j.jcomc.2024.100451

10. Appendix

10.1 Potential research questions

This report investigates the use of ML models within a closed-loop system to detect, interpret, and correct errors during construction-scale 3DP. Based on a synthesis of existing literature, experimental findings, and theoretical analysis, the following potential research directions have been identified:

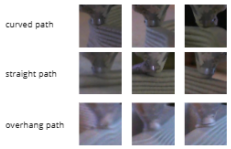
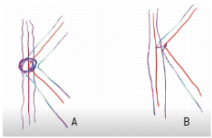
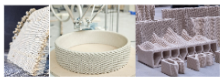
1. Structural Stability During Printing
   How can the system ensure that partially completed structures remain stable during fabrication, particularly when the toolpath interacts with unsupported or unfinished regions?
2. Geometry-Aware Prediction
   Can ML models be trained to distinguish between geometric contexts—such as corners, straight paths, and freeform curves—from top or sectional views, and associate them with printing outcomes? For example, how do image-based features differ between long, linear paths and curvilinear shapes, and how does this affect deposition performance?
3. Ridges from Toolpath Overlaps
   How can ridges caused by infill and outer layer intersections be detected and minimized? Could ML dynamically adjust toolpaths or extrusion rates to either reduce or aesthetically integrate these features?
4. Balancing Aesthetics and Function
   Can ML be optimized to distinguish between aesthetic and structural needs, especially when retaining or modifying surface ridges is contextually important?
5. Layer Weight Distribution
   How can the system ensure that lower layers are printed with higher material density for stability, while upper layers remain lighter? This may involve adjusting extrusion rates, speed, or path curvature dynamically.
6. Layer Adhesion and Overhang Performance
   What is the minimum required interlayer contact area for overhang stability? Can ML models be trained to optimize print paths to maintain adhesion even under minimal overlap conditions?
7. Contact Area vs. Stability
   How should the system balance contact area and print stability, and which parameters are most effective for controlling this relationship?
8. Discontinuous Toolpaths
   How can discontinuities from multi-directional printing be avoided—particularly at layer junctions—through optimized parameter transitions and path planning?
9. Outward vs. Inward Overhangs

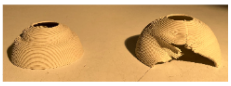
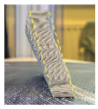
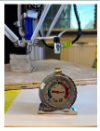
Why are outward overhangs typically more stable than inward ones, and how should printing parameters be adjusted accordingly?

10. Support Material Strategy
    Under what conditions is the use of support material (e.g., soft clay molds or adaptive scaffolds) necessary, and how should it be integrated with the base material?
11. Real-Time Shape Deviation Control
    Can ML respond to early-stage geometric deviations in unsupported spans by adjusting printing parameters before failure propagates?
12. Consistency Despite Identical Conditions
    Why do prints with identical paths, clay, and parameters yield different results? What environmental or system-level factors contribute, and how can real-time feedback be used to improve consistency?
13. Environmental Adaptation
    Can the system detect environmental changes (e.g., rising temperature or humidity) and adapt accordingly—such as slowing extrusion or modifying paths—to preserve print quality?
14. Anticipating Environmental Effects
    How quickly do environmental shifts affect clay behavior, and can the system predict and preemptively adjust before the next layer is printed?
15. Moisture Management in Large Prints
    How can uneven drying be mitigated in large-scale, multi-layer prints to ensure proper adhesion and structural performance?
16. Self-Correction via Material Properties
    Can the natural viscosity and malleability of clay be leveraged to self-correct uneven surfaces or layer inconsistencies during printing?
17. Critical Parameter Identification for Overhang Success
    Which parameters most directly influence overhang stability, and how can the system adjust them during fabrication?
18. Defect Detection and Compensation
    How can ML detect defects such as under-extrusion or layer shifting in real time and implement corrective actions without interrupting the print?
19. Inclination-Based Parameter Adjustment
    What parameter changes are necessary for overhangs of varying inclinations, and can this be mapped as a function of angle?
20. Post-Detection Recovery
    How should the system respond after a defect is detected to continue printing successfully while minimizing material waste?
21. Cross-Scale Anomaly Management

Can ML segment localized defects without compromising global form coherence? For example, if a ridge or gap is detected in a curved shell structure, how can the model compensate locally while preserving the overall geometry and stability?

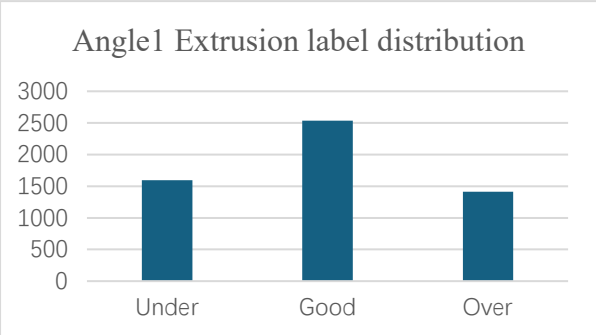
Given the complexity of integrating all these variables simultaneously, the current research focuses primarily on:
(6) Layer Adhesion and Overhang Performance, and (18) Defect Detection and Compensation, as the most immediate and feasible directions for implementing a functional ML-driven closed-loop correction system.

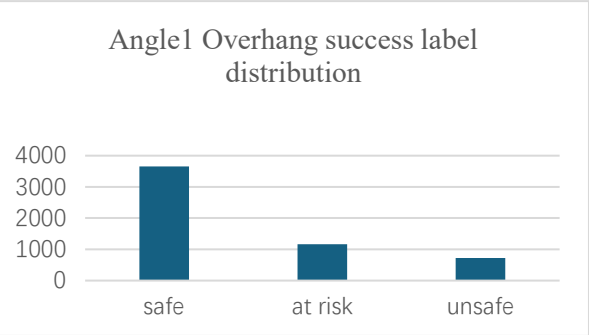
		Problem statement	Solution
1	**Structural Stability Assessment** 	Ensure that partially completed structures remain stable during printing, especially when interacting with unfinished parts.	Ensure that partially completed structures remain stable during printing, especially when interacting with unfinished parts.
2	**Geometry-Informed Predictions**  geometry features	How can ML interpret geometric differences by analyzing the specific characteristics of long straight lines versus freeform curves. This involves monitoring the overall shape while accounting for how images of these geometries may differ due to their distinct features. How does it correlate to the printing outcome?	 Develop ML models capable of recognizing the geometrical context (e.g., corners, straight lines, curves) using top-view or sectional images. These models should correlate image data with precise design locations to guide predictions.
3	**Ridges from Toolpath Overlaps**   B without ridges	Address ridges caused by intersections of infill and outer layers. To Avoid ridge: What is the optimal for this infill and outer layer? Can the ML model be trained so that it knows based on the nozzle diameter, how much it should come close to the shell?	It depends on material, extrusion, rate, nozzle diameter ML can optimize toolpath adjustments based on material, extrusion rate, and nozzle diameter to either minimize or aesthetically integrate these features. when the shape changes, when it gets more narrow, the situation changes, so maybe the toolpath can be adjusted 
4	**Integration of Aesthetic and Functional Needs**  A with ridges	Leave ridge as it is: • it adds to structure • it adds to aesthetic quaities Where is the optimal?	Optimize ML models to balance aesthetic and structural considerations in ridge retention or modification.
5	**Layer Weight Distribution** 	Enable ML models to understand structural requirements such as heavier lower layers for stability and lighter upper layers. In the construction, in the cross-section, the bottom part of the printing can be printed to be heavier, while printing to the upper layer, the structure can be printed lighter. What does it mean for the printing process? Can ML model understand the top layer should be lighter than bottom layer so it can make adjustment according to this. So from the place it starts overhanging, something else maybe needed or decrease printing speed, or extrude less. Can the ML model learn this and make the prediction.	Adjustments could include changes in printing speed, material extrusion, or overhang supports.
6	**Layer Adhesion and Flow Speed** 	Overhangs with minimal contact areas can still be printed successfully, particularly when the print path is continuous. Explore the correlation between layer contact area and flow speed.	ML models can optimize the print path to enhance stability even with minimal layer contact in overhangs.

7	**Touching Area and Stability** 	Determine the balance between layer contact area and structural integrity, allowing for precise parameter adjustments. What is the limit balance state between touching area and structural stability?	ML can optimize the toolpath and printing parameters, ensuring a continuous print process while maintaining stability. By analyzing the geometry and material behavior, ML models can dynamically adjust printing speed, extrusion rates, and toolpath to balance stability with design requirements.
8	**Discontinuous Toolpaths** 	Prevent untidy ends or unadhered layers in structures printed from multiple sides through toolpath and parameter optimization. How to prevent it by adjusting the printing parameters and tool path.	ML can analyze the print design to predict and minimize areas requiring discontinuous toolpaths. For unavoidable interruptions, the model can recalibrate parameters such as extrusion rate and retraction settings to ensure strong bonding and avoid weak joints.
9	**Overhang Geometry Challenges** 	Investigate why outward overhangs are more stable than inward contractions and optimize parameters for both geometries.	ML models can analyze geometric differences, identifying optimal printing parameters such as speed and extrusion rates for outward and inward overhangs. Real-time adjustments can optimize material deposition for each geometry.
10	**Support Requirements** 	Assess the conditions under which additional support (molds of clay, flexible structures) become necessary, and the material compatibility for such supports. To what extent, the overhang structure needs extra support/mold, and in what kind of material? clay; 3d-printed; flexible mold	ML can predict when and where supports are necessary based on the geometry and stress points. It can also determine optimal support material and structure for construction-scale 3D printing.
11	**Real-Time Shape Deviation Control** 	Mitigate deviations in unsupported structures by dynamically adjusting toolpaths and printing speeds. How to avoid the structure without support deviated from the designed shape, by real-time controlling the printing speed and toolpath?	ML-enabled closed-loop control can identify shape deviations using sensors and correct them by modifying toolpath or extrusion rates in real time.
12	**Consistency in Results** 	Why do identical printing parameters, toolpaths, and clay yield varying results? Identify environmental and parameter-related factors causing variation despite identical initial conditions, leveraging real-time monitoring to enhance repeatability.	ML models can monitor environmental factors and material conditions, adjusting parameters dynamically to achieve consistency.
13	**Environmental Adaptation**  evironmental temperature	Simulate variable environmental conditions (e.g., temperature) and their impact on printing quality. Instead of keeping the environment factor the same, manipulating environmental factors, particularly temperature, can help simulate on-site conditions and their impact on printing quality How quickly do environmental changes impact the clay mixture, and how can a machine learning model predict and adjust printing parameters accordingly before the next layer begins?	Use ML to adaptively manage parameters based on detected environmental changes.  While precise mathematical relationships may not be available, machine learning can identify patterns, such as reducing extrusion rate or speed with rising temperatures, and adjust parameters accordingly
14	**Moisture Retention in Large-Scale Prints** the premise: - need to print a very large scale - need to be a complex geometry that may lead to unevenly drying Local Adjustment	Address challenges in uneven drying of complex geometries to ensure adhesion between layers through moisture monitoring and regulation. (In large-scale, complex printing, uneven drying can prevent adhesion of new layers. Monitoring and adjusting moisture levels in printed layers ensures proper bonding for subsequent layers.)	Monitor the just printed layer's moisure and adjust the dried part of the clay layer to be wet enough for the next layer adhesive on top
15	**Self-Correction of Uneven Surfaces** 	Utilize clay's inherent viscosity and malleability to self-correct height differences during printing, optimizing for uneven bases Clay printing's viscosity and malleability allow for self-correction, enabling successful printing even on uneven surfaces sometimes. New layers naturally press and smooth previous imperfections, making it well-suited for in-situ environments.	ML models can predict surface irregularities and adjust toolpaths or extrusion pressure to leverage clay's self-correcting properties.
16	**Parameter Influence on Overhang Success**  	Identify and adjust critical parameters impacting overhang stability during printing. What printing parameters will influence the overhang structure print success? And how ML can help to monitor them?	ML can analyze geometry-specific parameters like inclination angles, contact areas, and extrusion rates to optimize overhang stability.
17	**Defect Detection and Compensation**  	Equip the system to detect and classify defects such as under-extrusion or layer shifting and implement corrective actions to maintain print quality. what printing errors to solve?	ML can classify defects in real-time and trigger corrective actions, such as adjusting extrusion speed or recalibrating toolpaths.
18	**Inclination-Based Adjustments**	Correlate overhang inclination degrees with required parameter adjustments for successful printing. According to different inclined degree of the overhang, how should ML adjust the parameters to make it print successfully?	ML can predict optimal initial settings based on material type, geometry, and environmental factors to minimize errors during printing.
19	**Post-Detection Recovery**	Develop strategies to compensate for detected deformations or defects to enable printing continuity and prevent material waste. How to compensate for deformation or defect after the system detects to save the printing and let it continue to print successfully?	ML can determine compensation strategies, such as localized extrusion adjustments, to correct defects and continue printing seamlessly.
20	**Cross-Scale Anomaly Analysis**	How can ML address both local anomalies (e.g., ridges, gaps) and global structural coherence?	Use ML for detailed segmentation of local anomalies while maintaining global design coherence. (For example, if a local anomaly like a ridge or gap is detected in a curved shell structure, the ML model should ensure that the corrective actions taken (e.g., additional material extrusion or smoothing) do not disrupt the overall curvature or stability of the shell. This approach ensures that both micro- and macro-level qualities of the structure are maintained, leading to a high-quality final product.)

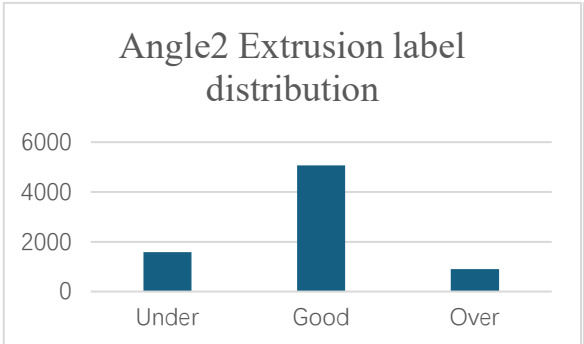
10.2 Dataset labelling summary

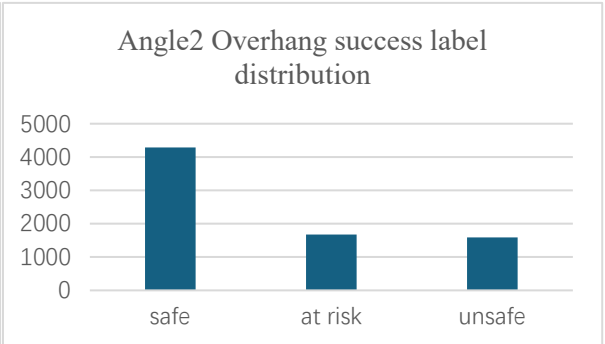
Angle 1										
	Speed	Motor	Extrusion	Extrusion class	Overhang	Overhang class	label	Note	image amount	pair amount
1	0.06	WASP	Good	1	safe	2	1,2	successful print	662	331
2	0.1	WASP	Under(bottom)	0	unsafe	0	0,0	fall down after successfully print	430	215
3	0.09	WASP	Under(bottom)	0	unsafe	0	0,0	fall down after successfully print	668	334
4	0.08	WASP	Under(top overhang)	0	safe/at risk	1	0,1	successful print(top overhang has holes)	848	424
5	0.07	WASP	thin thickness	0	safe	2	0,2	successful print	892	446
6	0.01	WASP	under(unsuccessful)	0	unsafe	0	0,0	can't stick on the previous layer	24	12
7	0.01	WASP	under(unsuccessful)	0	unsafe	0	0,0	can't stick on the base	60	30
8	0.04	WASP	Good	1	safe	2	1,2	successful print(layer thickness varies)	772	386
9	0.07	WASP	Good	1	safe	2	1,2	successful print	1322	661
10	0.01	WASP	Extremely Over	2	at risk	1	2,1	fat print, successful	916	458
11	0.05	WASP	Good in general	1	safe	2	1,2	successful print(layer thickness varies)	2092	1046
the second day printing										
24	0.03	WASP	Over(good)	2	at risk	1	2,1	fat print, successful, unfinish for top	262	131
25	0.03	WASP	good-over	2	safe	2	2,2	unfinished, a bit fat	122	61
26	0.03	WASP	Good(over a little)	2	safe	2	2,2	a little bit fat, successful	1214	607
27	0.04	WASP	Good(varies)	1	safe	2	1,2	unfinished	226	113
28	0.08	WASP	under	0	unsafe	0	0,0	can't well stick, uncontinous	222	111
29	0.01	WASP	Over	2	at risk	1	2,1	fat, unfinish printing	220	110
30	0.1	WASP	UNDER	0	unsafe	0	0,0	can't stick on the base	44	22
31	0.02	WASP	Over	2	at risk	1	2,1	fat, unfinish printing	90	45
Total										5543



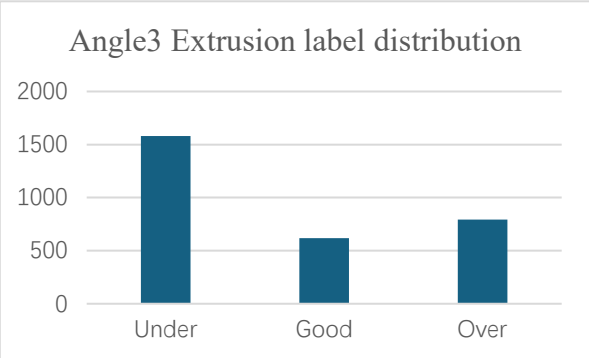


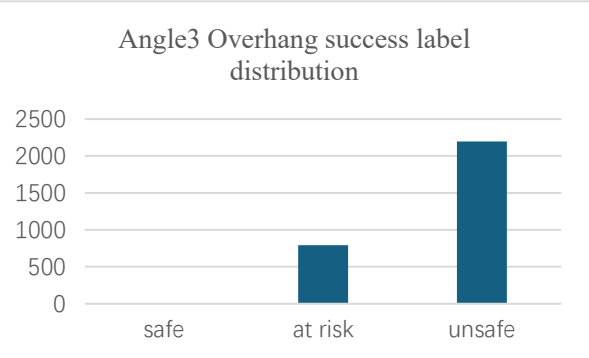
Angle 2										
	Speed	Motor	Extrusion	Extrusion class	Overhang	Overhang class		Note	image amount	pair amount
1	0.08	WASP	under(fail to print on top)	0	unsafe	0	0,0	fail to print top overhang,can't stick	1212	606
2	0.06	WASP	good(form deformation)	1	at risk	1	1,1	top overhang sagging because of self-weight	3350	1675
3	0.03/0.0	WASP	good-over-good	2	safe	2	2,2	clay change, the thickness varies	1796	898
4	0.08	WASP	under	0	unsafe	0	0,0	can't stick on the base & next layer	104	52
5	0.07	WASP	under(sagging&can't stick)	0	unsafe	0	0,0	sagging on bottom overhang & under E	978	489
6	0.09	WASP	under(fail to stick)	0	unsafe	0	0,0	too thin to stick on previous layer	880	440
7	0.05	WASP	good	1	safe	2	1,2	successful print	6788	3394
Total										7554



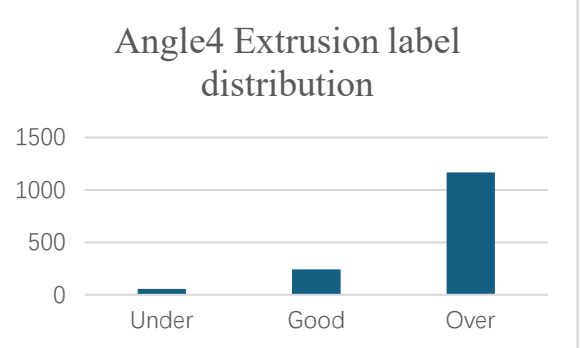


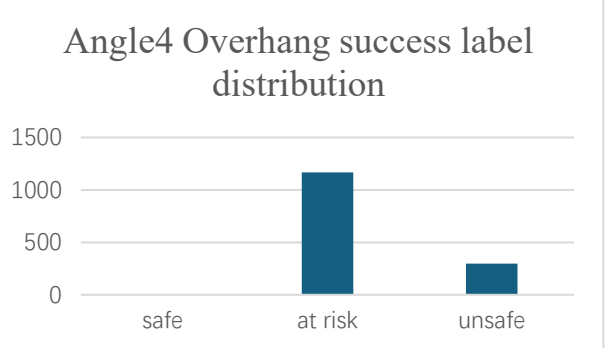
Angle 3										
	Speed	Motor	Extrusion	Extrusion class	Overhang	Overhang class		Note	image amount	pair amount
1	0.07	WASP	under	0	unsafe	0	0,0	can't stick & sagging	482	241
2	0.05	WASP	good	1	unsafe	0	1,0	good at bottom, can't stick in the middle	1234	617
3	0.08	WASP	under	0	unsafe	0	0,0	can't stick on base	228	114
4	0.06	WASP	good-thin	0	unsafe	0	0,0	good at bottom, can't stick in the middle	2452	1226
5	0.01	WASP	over	2	at risk	1	2,1	unfinished, fat	1588	794
Total										2992





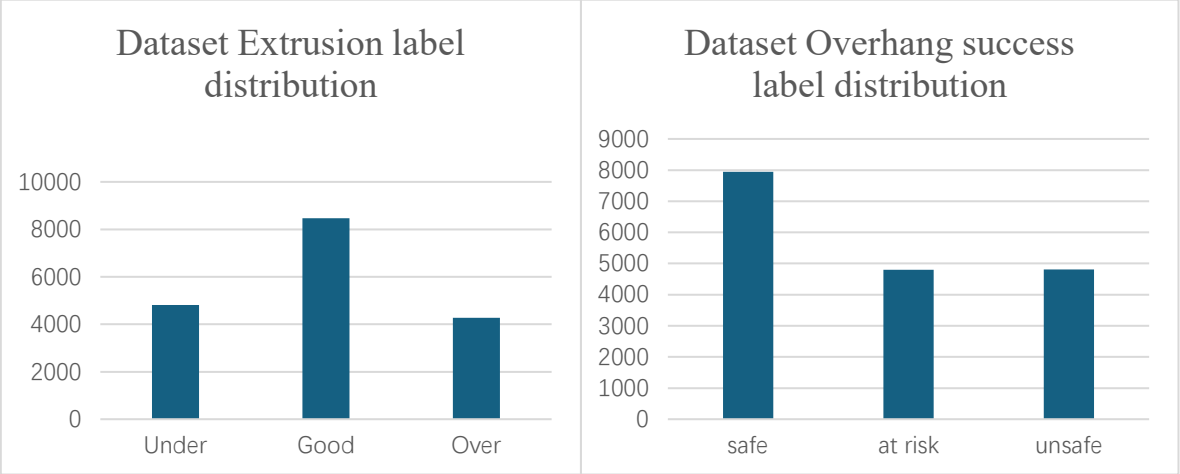
Angle 4										
	Speed	Motor	Extrusion	Extrusion class	Overhang	Overhang class		Note	image amount	pair amount
1	0.07	WASP	under	0	unsafe	0	0,0	can't stick on the base	112	56
2	0.05	WASP	good	1	unsafe	0	1,0	can't stick for bottom when the inclined angle bigger	484	242
3	0.01	WASP	over	2	at risk	1	2,1	clay use up, unfinished	320	160
4	0.01	WASP	over	2	at risk	1	2,1	new clay, fat, successful, sagging a lot	2012	1006
Total										1464



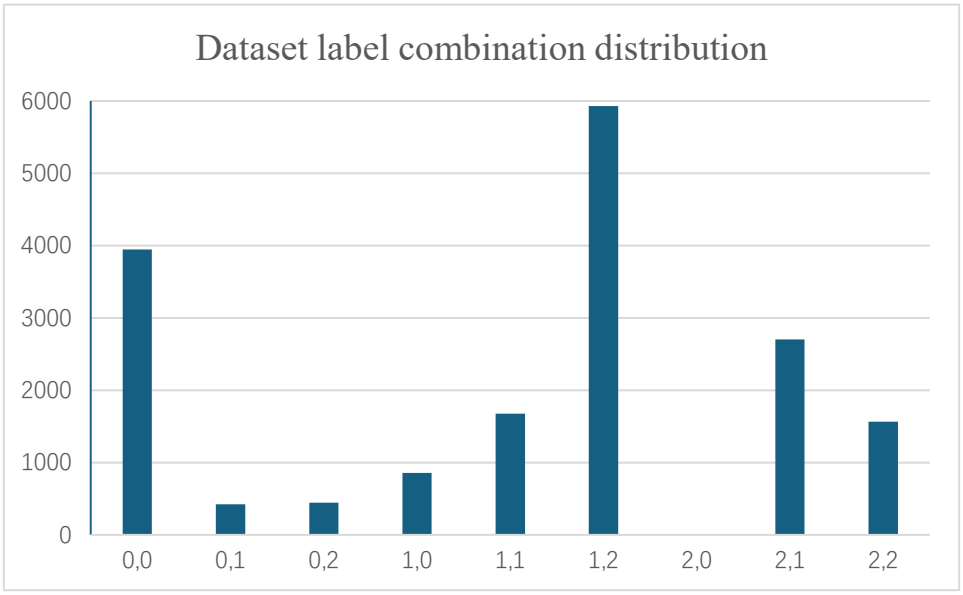


Dataset size in total	17553

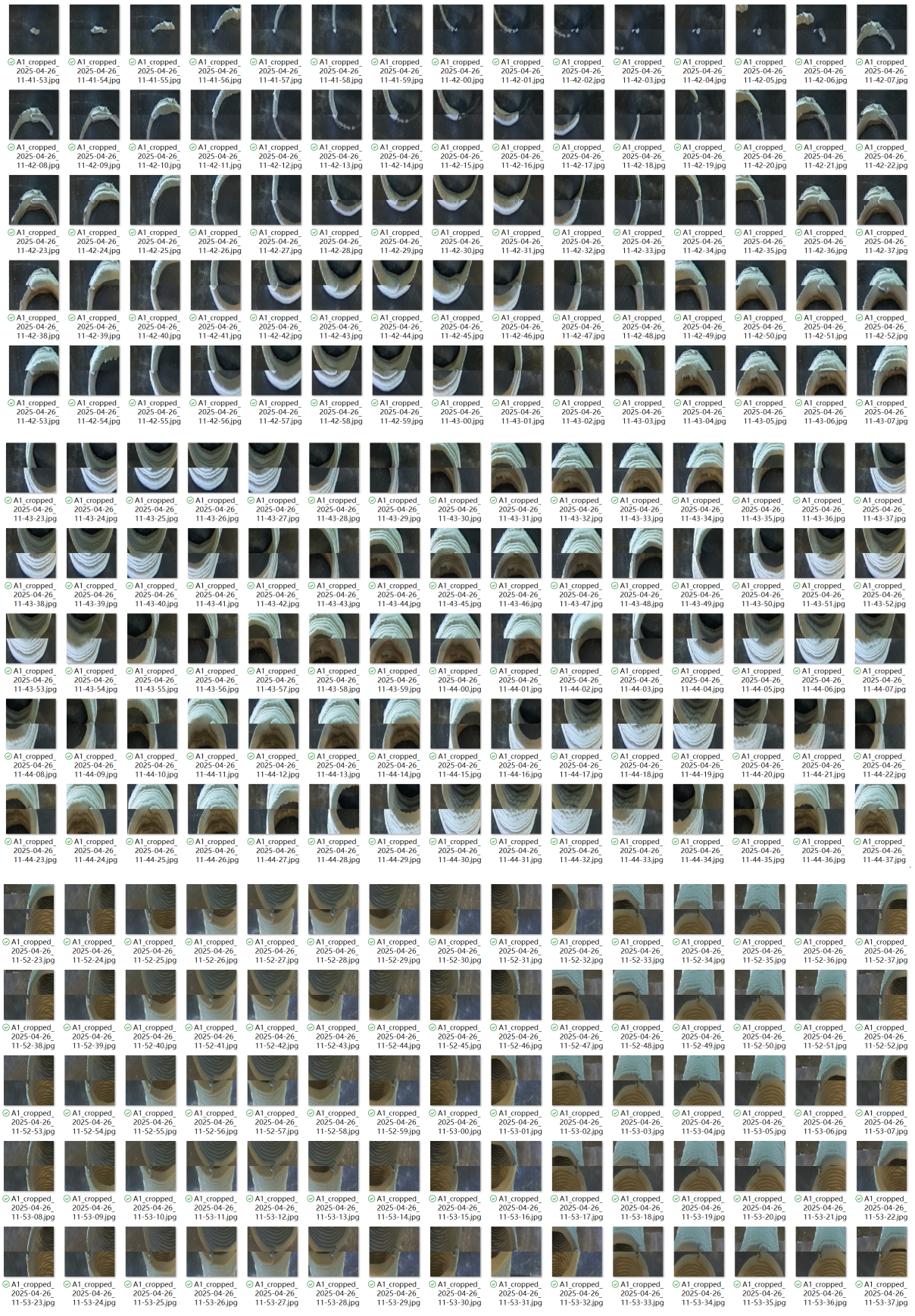
Dataset sample count by label						
parameter	class	Angle1	Angle2	Angle3	Angle4	total
extrusion	Under	1594	1587	1581	56	4818
	Good	2537	5069	617	242	8465
	Over	1412	898	794	1166	4270
overhang success	safe	3651	4292	0	0	7943
	at risk	1168	1675	794	1166	4803
	unsafe	724	1587	2198	298	4807

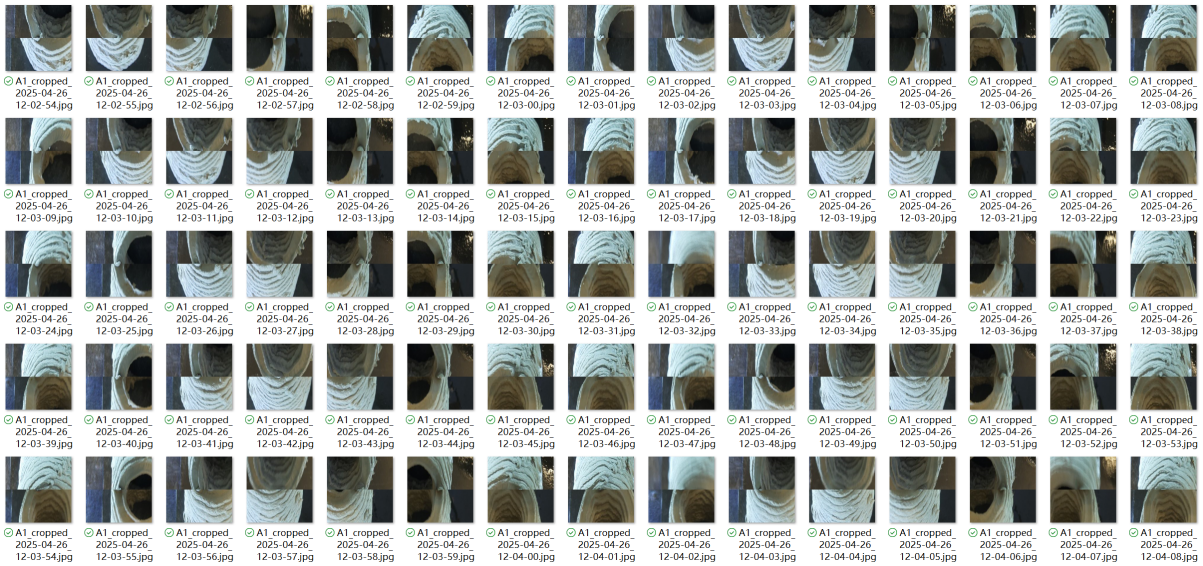


label combination	count
0,0	3948
0,1	424
0,2	446
1,0	859
1,1	1675
1,2	5931
2,0	0
2,1	2704
2,2	1566



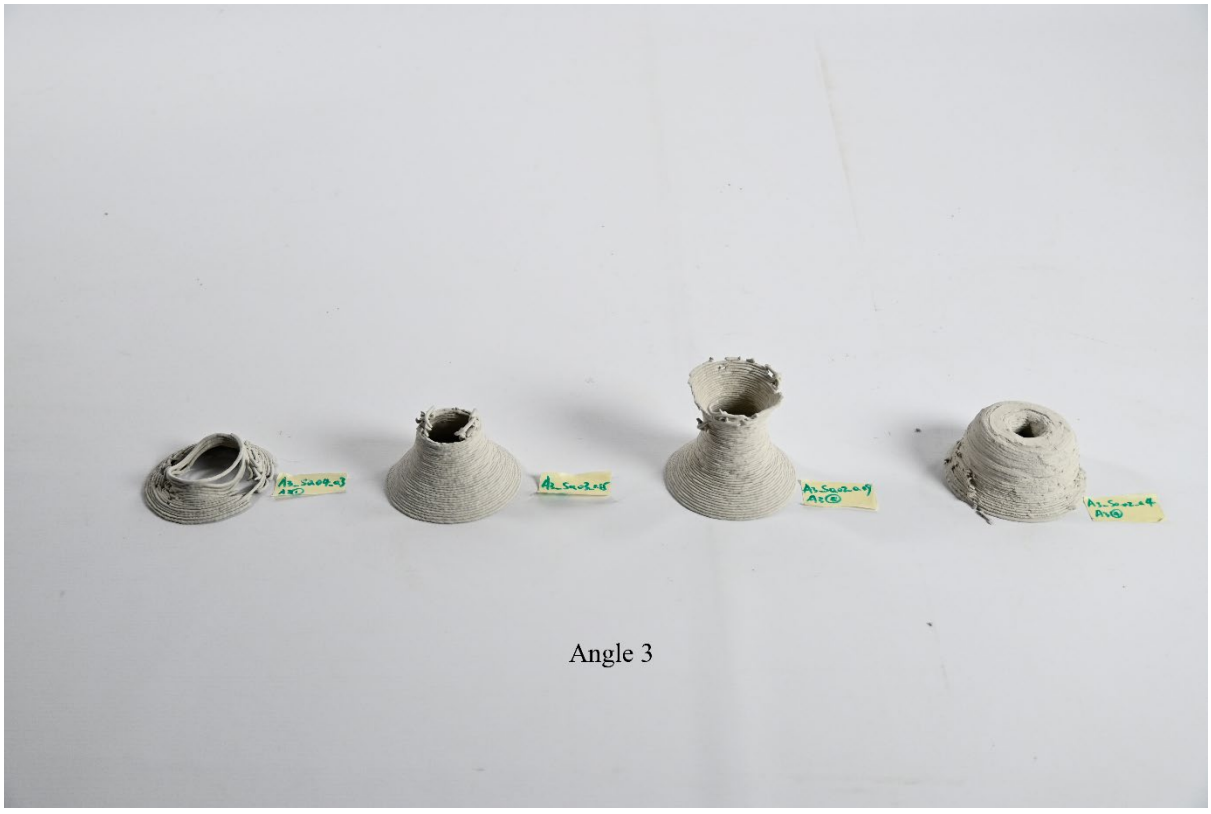
10.3 Dataset Example





10.4 Prototype Collection

Angle 2



Angle 3

Angle 4

10.5 Real-time Correction Workflow Codes

```
# ================================
# Real-time 3D Printing Calibration Loop
# Author: Xiaochen Ding
# Purpose: Closed-loop ML-based UR5 speed control for 3D clay printing
# ================================
import os
import time
import socket
import threading
import torch
import numpy as np
import pandas as pd
import cv2 as cv
from PIL import Image
from scipy import stats
from glob import glob
import paramiko
from scp import SCPClient
from model.network_module_DINOv2 import DINO2ResAttClassifier
from data.data_module_wholeworkflow import ParametersDataModule
from train_config import preprocess
```

```
from datetime import datetime
import matplotlib.pyplot as plt
import shutil

# UR Configuration
UR_IP = "192.168.1.100"
SCRIPT_PORT = 30003
DASH_PORT = 30002
INITIAL_SCALE = 0.5

# Paths
DATA_DIR = r"E:\OneDrive - Delft University of Technology\TUD Master\graduation
project\test_print_photo"
INPUT_FOLDER = os.path.join(DATA_DIR, "Image_detection")
OUTPUT_FOLDER = os.path.join(DATA_DIR, "Image_for_preprocess")
PREDICTION_FOLDER = os.path.join(DATA_DIR, "Image_for_prediction")
SAVE_FOLDER1 = os.path.join(DATA_DIR, "Image_for_save_raw")
SAVE_FOLDER2 = os.path.join(DATA_DIR, "Image_for_save_prediction")
DATA_CSV = os.path.join(DATA_DIR, "test_print.csv")
CHECKPOINT_PATH = r"E:\OneDrive - Delft University of Technology\TUD Master\graduation
project\ML\checkpoints\23042025\1234\DINO2ResAtt-model6.3_balanced_DINOv2-23042025-epoch=38-
val_loss=0.35-val_acc=0.00.ckpt"
WAYPOINTS_CSV = r"E:\OneDrive - Delft University of Technology\TUD Master\graduation
project\ML\UR5\movej_positions.csv"
DATASET_NAME = "closeloop_test_v1"

# Raspberry Pi Configuration
PI_IP = "192.168.1.185"
PI_USER = "user"
PI_PASS = "toi'sLAMA"
PI_IMAGE_DIR = "/home/user/Image_detection"
PI_TIMELAPSE_SCRIPT = "/home/user/camera_project/take_timelapse_xc.py"

# Constants
DATASET_MEAN = [0.2915257, 0.27048784, 0.14393276]
DATASET_STD = [0.066747, 0.06885352, 0.07679665]
BATCH_SIZE = 18
MONITOR_PAUSE = 20 #40

# SSH and SCP clients
ssh = None
scp = None
```

```
# Load model
model = DINO2ResAttClassifier.load_from_checkpoint(
    checkpoint_path=CHECKPOINT_PATH,
    num_classes=3,
    gpus=1,
)
model.eval()

def connect_pi():
    global ssh, scp
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(PI_IP, username=PI_USER, password=PI_PASS)
    scp = SCPClient(ssh.get_transport())

def start_timelapse():
    print("📸 Starting timelapse on Raspberry Pi...")
    ssh.exec_command(f"python3 {PI_TIMELAPSE_SCRIPT} &")

def stop_timelapse():
    print("🛑 Stopping timelapse on Raspberry Pi...")
    ssh.exec_command("pkill -f take_timelapse.py")

def sync_images_from_pi():
    print("📥 Transferring images from Raspberry Pi...")
    scp.get(PI_IMAGE_DIR, DATA_DIR, recursive=True)
    print("☑️ Images transferred to PC.")

def clear_remote_folder(remote_path):
    delete_cmd = f"rm -rf {remote_path}/*"
    stdin, stdout, stderr = ssh.exec_command(delete_cmd)
    exit_status = stdout.channel.recv_exit_status()
    if exit_status == 0:
        print(f"🧹 Cleared all contents from remote folder: {remote_path}")
    else:
        err = stderr.read().decode().strip()
        print(f"❌ Error deleting remote directory: {err}")

def backup_images_to_timestamped_folder(input_folder, save_root_folder):
    # 1. Check if input directory exists
    if not os.path.isdir(input_folder):
        raise ValueError(f"Input folder does not exist or is not a directory: {input_folder}")
```

```
    # 2. Create a timestamped subfolder under save_root_folder
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    destination_folder = os.path.join(save_root_folder, timestamp)
    os.makedirs(destination_folder, exist_ok=True)

    # 3. Iterate over all files in input_folder and copy images
    copied_count = 0
    for filename in os.listdir(input_folder):
        name_lower = filename.lower()
        _, ext = os.path.splitext(name_lower)

        src_path = os.path.join(input_folder, filename)
        dst_path = os.path.join(destination_folder, filename)
        shutil.copy2(src_path, dst_path)  # Copy preserving metadata
        copied_count += 1

    print(f"🗀 Created subfolder '{timestamp}' under '{save_root_folder}'")
    print(f"☑ Successfully copied {copied_count} images to: {destination_folder}")
    return destination_folder

def clear_folder(folder_path):
    if not os.path.isdir(folder_path):
        print(f"⚠ Path does not exist or is not a directory: {folder_path}")
        return

    for entry in os.listdir(folder_path):
        entry_path = os.path.join(folder_path, entry)
        try:
            if os.path.isfile(entry_path) or os.path.islink(entry_path):
                os.remove(entry_path)
        except Exception as e:
            print(f"Error deleting: {entry_path} -> {e}")

    print(f"🧹 Cleared all contents from: {folder_path}")

def set_speed_override(scale: float):
    msg = f"set speed {scale:.3f}\n".encode("ascii")
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((UR_IP, DASH_PORT))
        s.sendall(msg)

def load_waypoints(path):
    waypoints = []
```

```
    with open(path, 'r', encoding='utf-8-sig', newline='') as f:
        reader = pd.read_csv(f)
        for row in reader.itertuples(index=False):
            waypoints.append(list(row))
    return waypoints

def build_urscript_joint_arc(waypoints, acc=3.1416, vel=0.07, blend=0.01):
    lines = [
        "def Program():",
        "  Clay_extruderTcp  = p[0, 0.1765, 0.058, -1.5708, 0, 0]",
        "  Clay_extruderWeight = 1.78",
        "  Clay_extruderCog= [0, 0.1765, 0.058]",
        "  set_tcp(Clay_extruderTcp)",
        "  set_payload(Clay_extruderWeight, Clay_extruderCog)",
        f"  movej({waypoints[0]}, a={acc}, v={vel}, r=0)"
    ]
    for wp in waypoints[1:]:
        lines.append(f"  movej({wp}, a={acc}, v={vel}, r={blend})")
    lines.append("end")
    return "\n".join(lines)

def send_script(script: str, chunk_size=2084, delay=0.01):
    data = script + "\n"
    total = len(data)
    sent = 0

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((UR_IP, SCRIPT_PORT))
        # Send in chunks
        while sent < total:
            end = min(sent + chunk_size, total)
            block = data[sent:end].encode('utf8')
            s.sendall(block)
            sent = end
            time.sleep(delay)
    print("☑ URScript uploaded and executed.")

def suggest_speed_change(extrusion, overhang):
    if extrusion == 0:
        return -0.1 if overhang <= 1 else -0.2
    elif extrusion == 1:
        return +0.1 if overhang == 0 else (0 if overhang == 1 else -0.1)
    elif extrusion == 2:
```

```
        return +0.2 if overhang == 0 else +0.1
    return 0

def preprocess_images(input_folder, output_folder):
    param = np.load(r"E:\OneDrive - Delft University of Technology\TUD Master\graduation
project\dataset\cali_images\calibration_parameters0.npz")
    H_up = param['H_up']
    H_down = param['H_down']
    crop_up = param['crop_pts_up']
    crop_down = param['crop_pts_down']
    scale = param['scale_factor']
    warp_size_up = tuple(param['warp_size_up'])
    warp_size_down = tuple(param['warp_size_down'])
    output_size = tuple(param['output_size'])

    def crop_img(img, crop_pts, output_size):
        pts_dst = np.array([[0, 0], [output_size[0]-1, 0], [output_size[0]-1, output_size[1]-1], [0,
output_size[1]-1]], dtype=np.float32)
        H_crop = cv.getPerspectiveTransform(crop_pts, pts_dst)
        return cv.warpPerspective(img, H_crop, output_size)

    def crop_to_center(img, crop_size):
        h, w = img.shape[:2]
        center_x, center_y = w // 2, h // 2
        left = max(center_x - crop_size[0] // 2, 0)
        top = max(center_y - crop_size[1] // 2, 0)
        right = min(center_x + crop_size[0] // 2, w)
        bottom = min(center_y + crop_size[1] // 2, h)
        cropped = img[top:bottom, left:right]
        return cv.resize(cropped, crop_size, interpolation=cv.INTER_LINEAR)

    def add_black_border(img, border_size):
        return cv.copyMakeBorder(img, border_size[1], border_size[1], border_size[0], border_size[0],
cv.BORDER_CONSTANT, value=(0, 0, 0))

    os.makedirs(output_folder, exist_ok=True)
    img_list = glob(os.path.join(input_folder, "*.jpg"))
    cam0, cam1 = {}, {}
    for path in img_list:
        name = os.path.basename(path)
        if name.startswith("cam0_"):
            cam0[name[5:-4]] = path
        elif name.startswith("cam1_"):
```

```
            cam1[name[5:-4]] = path

    timestamps = sorted(set(cam0.keys()) & set(cam1.keys()))
    for ts in timestamps:
        img_up = cv.imread(cam0[ts])
        img_down = cv.imread(cam1[ts])
        if img_up is None or img_down is None:
            continue

        warp_up = cv.warpPerspective(img_up, H_up, warp_size_up)
        warp_down = cv.warpPerspective(img_down, H_down, warp_size_down)
        crop_u = crop_img(warp_up, crop_up, output_size)
        crop_d = crop_img(warp_down, crop_down, output_size)
        crop_u = cv.rotate(crop_u, cv.ROTATE_90_COUNTERCLOCKWISE)
        crop_d = cv.rotate(crop_d, cv.ROTATE_90_COUNTERCLOCKWISE)
        new_w = int(crop_u.shape[1] * scale)
        new_h = int(crop_u.shape[0] * scale)
        crop_u_scaled = cv.resize(crop_u, (new_w, new_h))
        x_up_center = new_w // 2
        x_down_center = crop_d.shape[1] // 2
        offset_x = x_down_center - x_up_center
        left = max(-offset_x, 0)
        x_up = left + max(offset_x, 0)
        x_down = left + max(-offset_x, 0)
        canvas_w = max(x_up + new_w, x_down + crop_d.shape[1])
        canvas_h = new_h + crop_d.shape[0]
        canvas = np.zeros((canvas_h, canvas_w, 3), dtype=np.uint8)
        canvas[0:new_h, x_up:x_up+new_w] = crop_u_scaled
        canvas[new_h:new_h+crop_d.shape[0], x_down:x_down+crop_d.shape[1]] = crop_d
        orig_h, orig_w = canvas.shape[:2]
        bordered = add_black_border(canvas, (1500, 1500))
        resized = cv.resize(bordered, (orig_w, orig_h))
        cropped_merge_img = crop_to_center(resized, (224, 224))
        output_path = os.path.join(output_folder, f"cropped_{ts}.jpg")
        cv.imwrite(output_path, cropped_merge_img)
        print(f"☑ Saved processed image to: {output_path}")

def update_csv_paths():
    df = pd.read_csv(DATA_CSV)
    # Find all cropped images in OUTPUT_FOLDER named cropped_{ts}.jpg
    cropped_files = sorted([
        fname for fname in os.listdir(OUTPUT_FOLDER)
        if fname.startswith("cropped_") and fname.endswith(".jpg")
```

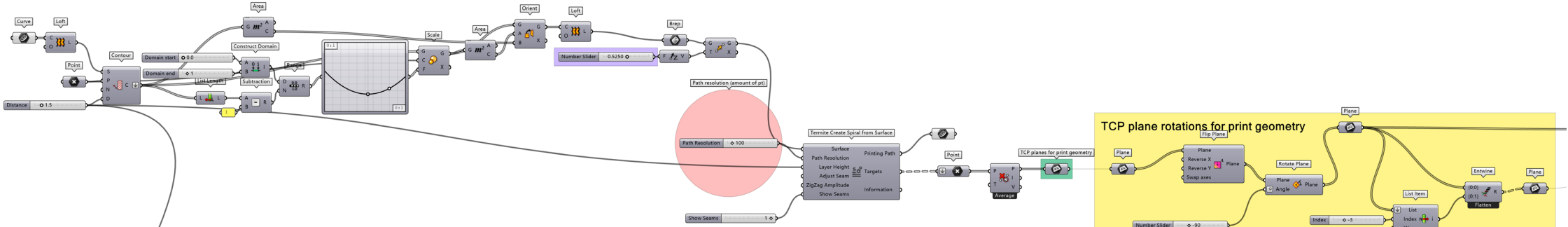
```
    ])
    new_paths = [os.path.join(OUTPUT_FOLDER, fname) for fname in cropped_files]

    if len(new_paths) != len(df):
        print(f"⚠ Warning: Number of cropped images ({len(new_paths)}) does not match CSV rows
({len(df)}), possible error.")

    min_len = min(len(new_paths), len(df))
    df.loc[:min_len-1, "img_path"] = new_paths[:min_len]
    df.to_csv(DATA_CSV, index=False)
    print(f"☑ Updated img_path in CSV file '{DATA_CSV}' with cropped image paths.")

def make_dirs(path):
    try:
        os.makedirs(path)
    except:
        pass
# Custom function to visualize and save images
def visualize_batch(batch, df, save_dir, dataset_std, dataset_mean):
    images, labels = batch
    batch_size = len(images)
    # Adjust image paths according to the total dataset, not just the batch
    image_filenames = df['img_path'].values[:len(images)]
    for i, (img, label) in enumerate(zip(images, labels)):
        print(f"Processing image {i+1}/{batch_size}")
        img = img.permute(1, 2, 0)  # Permute to (H, W, C) format for plotting
        img = img * torch.tensor(dataset_std) + torch.tensor(dataset_mean)  # Denormalize
        img = img.clamp(0, 1)
        # Convert to numpy for saving
        img_np = img.numpy()
        # Save each image individually
        output_filename = os.path.basename(image_filenames[i])
        output_path = os.path.join(save_dir, f"{output_filename}")
        plt.imsave(output_path, img_np)
        print(f"Saved image: {output_path}")
def Label_predict():
    data_module = ParametersDataModule(
        batch_size=BATCH_SIZE,
        data_dir=DATA_DIR,
        csv_file=DATA_CSV,
        dataset_name=DATASET_NAME,
        mean=DATASET_MEAN,
        std=DATASET_STD,
```

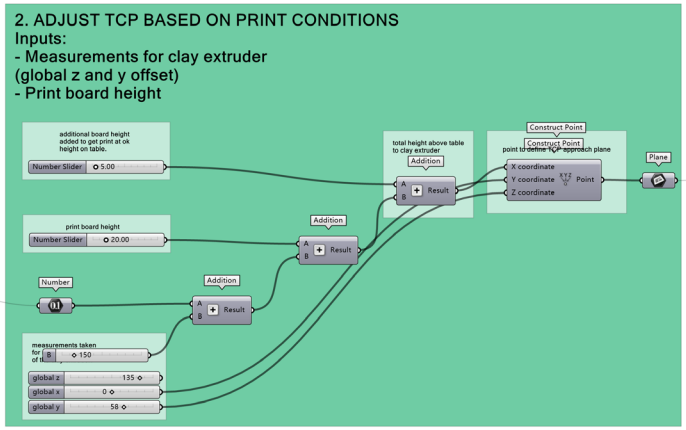
```
    load_saved=False,
    transform=True
)
# Load the CSV file and check paths
df = pd.read_csv(DATA_CSV)

# Setup data module, skip dataset split
data_module.setup(stage="test", save=False, test_all=True)

# Get dataloader (shuffle can be False or True as needed)
test_dataloader = data_module.test_dataloader()

# Process each batch to ensure all images are processed
for batch_idx, batch in enumerate(test_dataloader):
    print(f"Processing batch {batch_idx + 1}")
    visualize_batch(batch, df, PREDICTION_FOLDER, DATASET_STD, DATASET_MEAN)

print("All images processed.")

img_paths = [
    os.path.join(PREDICTION_FOLDER, img)
    for img in os.listdir(PREDICTION_FOLDER)
    if os.path.splitext(img)[1] == ".jpg"
]
# Step 3: Preprocess and predict labels
print("********* MudTracker3D sample predictions *********")
print("Layer_height | Extrusion")
print("***********************************************")
layer_height_preds = []
extrusion_preds = []

for img_path in img_paths:
    pil_img = Image.open(img_path)
    x = preprocess(pil_img).unsqueeze(0)
    y_hats = model(x)
    y_hat0, y_hat1 = y_hats
    _, preds0 = torch.max(y_hat0, 1)
    _, preds1 = torch.max(y_hat1, 1)
    preds = torch.stack((preds0, preds1)).squeeze()
    preds_str = str(preds.numpy())
    img_basename = os.path.basename(img_path)
    print("Input:", img_basename, "->", "Prediction:", preds_str)
    # Collect predictions
```

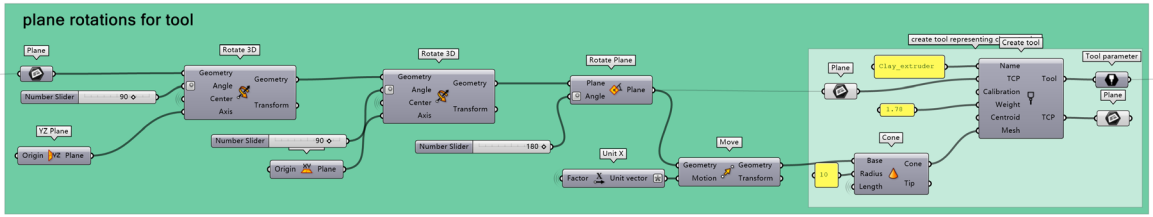
```
        layer_height_preds.extend(preds0.numpy())
        extrusion_preds.extend(preds1.numpy())

    mode_result0 = stats.mode(layer_height_preds)
    mode_result1 = stats.mode(extrusion_preds)

    final_layer_height_label = mode_result0.mode.item()
    final_extrusion_label    = mode_result1.mode.item()

    print(f"Layer Height: {final_layer_height_label}, Extrusion: {final_extrusion_label}")
    return final_layer_height_label, final_extrusion_label

def monitor_loop():
    scale = INITIAL_SCALE
    print(f"Initial speed: {scale}")
    connect_pi()
    clear_remote_folder(PI_IMAGE_DIR)
    while True:
        print("📸 Starting timelapse on Raspberry Pi...")
        start_timelapse()
        print("🕒 Waiting for 10 pairs of images on Raspberry Pi...")

        # Wait for Raspberry Pi to generate new photo files
        time.sleep(2)
        stdin, stdout, stderr = ssh.exec_command(f"ls -t {PI_IMAGE_DIR}")

        # Check if there are already 10 pairs of images
        while True:
            stdin, stdout, stderr = ssh.exec_command(f"ls {PI_IMAGE_DIR} | grep cam0_ | wc -l")
            count_cam0 = int(stdout.read().decode().strip())
            stdin, stdout, stderr = ssh.exec_command(f"ls {PI_IMAGE_DIR} | grep cam1_ | wc -l")
            count_cam1 = int(stdout.read().decode().strip())
            if min(count_cam0, count_cam1) >= 10:
                break
            time.sleep(2)

        stop_timelapse()
        sync_images_from_pi()
        clear_remote_folder(PI_IMAGE_DIR)
        preprocess_images(INPUT_FOLDER, OUTPUT_FOLDER)
        update_csv_paths()
        final_layer_height_label, final_extrusion_label = Label_predict()
```

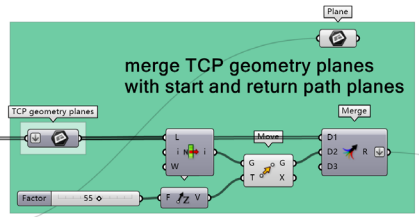
```
        if final_layer_height_label == 1 and final_extrusion_label == 1:
            break

        delta = suggest_speed_change(final_extrusion_label, final_layer_height_label)
        new_scale = min(1.0, max(0.1, scale + delta))
        set_speed_override(new_scale)
        scale = new_scale
        print(f"Adjusted new speed: {scale}")
        print(f"⏸ Pausing {MONITOR_PAUSE}s...")
        time.sleep(MONITOR_PAUSE)
        backup_images_to_timestamped_folder(INPUT_FOLDER, SAVE_FOLDER1)
        clear_folder(INPUT_FOLDER)
        clear_folder(OUTPUT_FOLDER)
        backup_images_to_timestamped_folder(PREDICTION_FOLDER, SAVE_FOLDER2)
        clear_folder(PREDICTION_FOLDER)
    stop_timelapse()
    scp.close()
    ssh.close()
    # Reset scale to 0.5 before exit
    set_speed_override(0.5)

def main():
    print("🚀 Uploading URScript and starting printing...")
    waypoints = load_waypoints(WAYPOINTS_CSV)
    script = build_urscript_joint_arc(waypoints)
    send_script(script)
    clear_folder(INPUT_FOLDER)
    clear_folder(OUTPUT_FOLDER)
    clear_folder(PREDICTION_FOLDER)
    print("⏳ Waiting 10s before monitoring...")
    time.sleep(10)
    monitor_loop()
    print("🎉 Print complete. Workflow finished.")

if __name__ == "__main__":
    main()
```

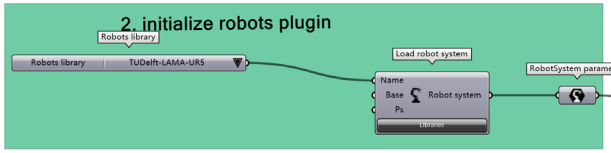
10.6 Slicing and URScript Generation Script

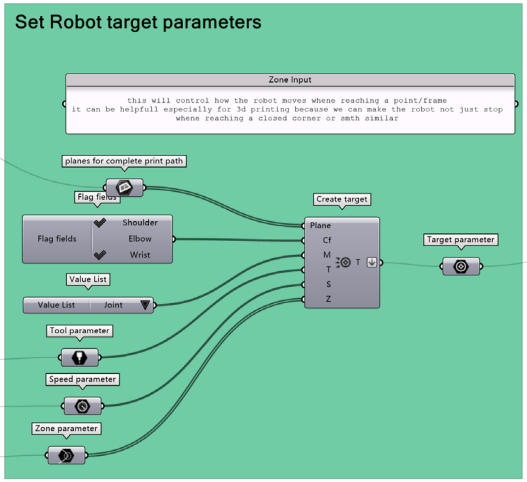

Curve
Loft
Point
Contour
Distance 1.5
Area
Domain start 0.0
Domain end 1
List Length
Construct Domain
Subtraction
Range
Scale
Area
Orient
Loft
Number Slider 0.5250
Brep
Path resolution (amount of pt)
Path Resolution 100
Termite Create Spiral from Surface
Surface
Path Resolution
Layer Height
Adjust Seam
ZigZag Amplitude
Show Seams
Printing Path
Targets
Information
Show Seams 1
Point
Average
TCP planes for print geometry
TCP plane rotations for print geometry
Plane
Flip Plane
Reverse X
Reverse Y
Swap axes
Rotate Plane
Angle
Number Slider -90
List Item
List
Index
Wrap
Index -3
Entwine
Flatten

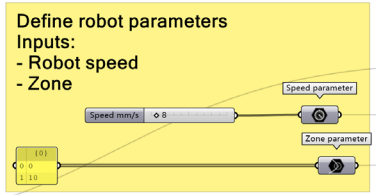

2. ADJUST TCP BASED ON PRINT CONDITIONS
Inputs:
- Measurements for clay extruder
(global z and y offset)
- Print board height
additional board height added to get print at ok height on table.
Number Slider 5.00
print board height
Number Slider 20.00
Number
measurements taken for
Number Slider 150
global z 135
global x 0
global y 58
Addition
Result
total height above table to clay extruder
Construct Point
point to define approach plane
X coordinate
Y coordinate
Z coordinate
Point
Plane

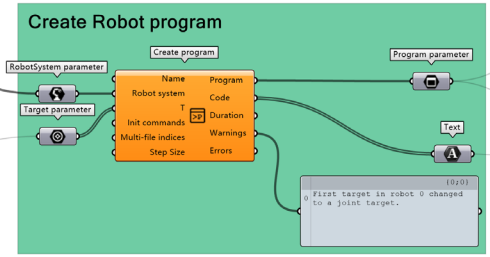

plane rotations for tool
Plane
Number Slider 90
YZ Plane
Origin
Rotate 3D
Geometry
Angle
Center
Axis
Transform
Number Slider 90
Number Slider 180
Rotate Plane
Unit X
Factor
Unit vector
Move
Motion
create tool representing
Create tool
Clay_extruder
1.78
Cone
Base
Radius
Length
Name
TCP
Calibration
Weight
Centroid
Mesh
Tool
Tip
Tool parameter

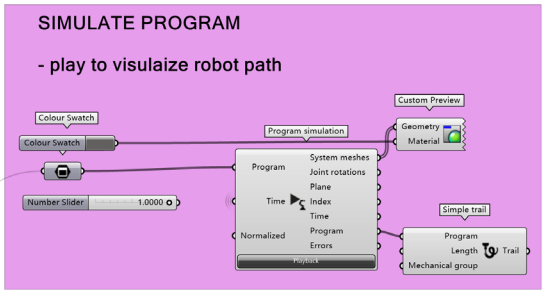

Plane
merge TCP geometry planes
with start and return path planes
TCP geometry planes
Merge
Move
Factor
55

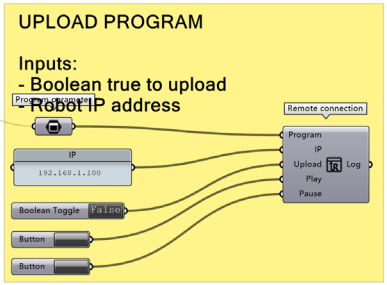

2. initialize robots plugin
Robots library
Robots library
TUDelft-LAMA-UR5
Load robot system
Name
Base
Ps
Robot system
RobotSystem parameter

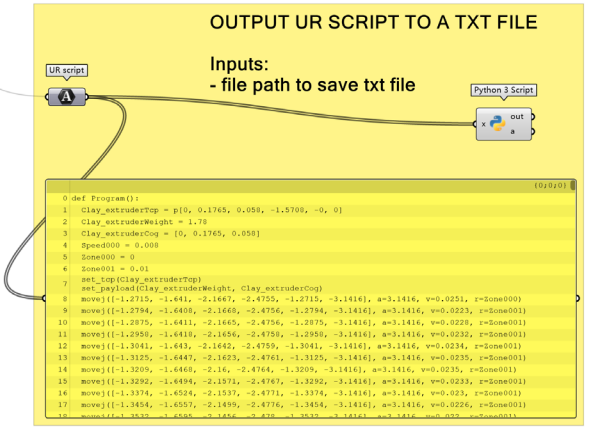

Set Robot target parameters
Zone Input
this will control how the robot moves whene reaching a point/frame
it can be helpfull especially for 3d printing because we can make the robot not just stop
whene reaching a closed corner or smth similar
planes for complete print path
Flag fields
Shoulder
Elbow
Wrist
Value List
Joint
Tool parameter
Speed parameter
Zone parameter
Create target
Plane
Cf
M
T
S
Z
Target parameter


Define robot parameters
Inputs:
- Robot speed
- Zone
Speed parameter
Speed mm/s
8
Zone parameter


Create Robot program
RobotSystem parameter
Target parameter
Create program
Name
Robot system
T
Init commands
Multi-file indices
Step Size
Program
Code
Duration
Warnings
Errors
Program parameter
Text
First target in robot 0 changed
to a joint target.


SIMULATE PROGRAM
- play to visulaize robot path
Colour Swatch
Program simulation
Custom Preview
Geometry
Material
Program
Time
Normalized
System meshes
Joint rotations
Plane
Index
Time
Program
Errors
Number Slider
1.0000
Simple trail
Program
Length
Trail
Mechanical group


UPLOAD PROGRAM
Inputs:
- Boolean true to upload
- Robot IP address
Program parameter
Remote connection
Program
IP
Upload
Play
Pause
Log
192.168.1.100
Boolean Toggle
False
Button
Button


OUTPUT UR SCRIPT TO A TXT FILE
Inputs:
- file path to save txt file
UR script
Python 3 Script
out
a
def Program():
Clay_extruderTcp = p[0, 0.1765, 0.058, -1.5708, -0, 0]
Clay_extruderWeight = 1.78
Clay_extruderCog = [0, 0.1765, 0.058]
Speed000 = 0.008
Zone000 = 0
Zone001 = 0.01
set_tcp(Clay_extruderTcp)
set_payload(Clay_extruderWeight, Clay_extruderCog)
movej([-1.2715, -1.641, -2.1667, -2.4755, -1.2715, -3.1416], a=3.1416, v=0.0251, r=Zone000)
movej([-1.2794, -1.6408, -2.1668, -2.4756, -1.2794, -3.1416], a=3.1416, v=0.0223, r=Zone001)
movej([-1.2875, -1.6411, -2.1665, -2.4756, -1.2875, -3.1416], a=3.1416, v=0.0228, r=Zone001)
movej([-1.2958, -1.6418, -2.1656, -2.4758, -1.2958, -3.1416], a=3.1416, v=0.0232, r=Zone001)
movej([-1.3041, -1.643, -2.1642, -2.4759, -1.3041, -3.1416], a=3.1416, v=0.0234, r=Zone001)
movej([-1.3125, -1.6447, -2.1623, -2.4761, -1.3125, -3.1416], a=3.1416, v=0.0235, r=Zone001)
movej([-1.3209, -1.6468, -2.16, -2.4764, -1.3209, -3.1416], a=3.1416, v=0.0235, r=Zone001)
movej([-1.3292, -1.6494, -2.1571, -2.4767, -1.3292, -3.1416], a=3.1416, v=0.0233, r=Zone001)
movej([-1.3374, -1.6524, -2.1537, -2.4771, -1.3374, -3.1416], a=3.1416, v=0.023, r=Zone001)
movej([-1.3454, -1.6557, -2.1499, -2.4776, -1.3454, -3.1416], a=3.1416, v=0.0226, r=Zone001)