inventions

MDPI

Article

Real-Time Precision in 3D Concrete Printing: Controlling Layer Morphology via Machine Vision and Learning Algorithms

João M. Silva [1,*], Gabriel Wagner [2], Rafael Silva [3], António Morais [4], João Ribeiro [4], Sacha Mould [2], Bruno Figueiredo [4], João M. Nóbrega [2,*] and Paulo J. S. Cruz [4]

1 BUILT CoLAB—Collaborative Laboratory for the Future Built Environment, 4150-171 Porto, Portugal
2 Institute for Polymers and Composites—IPC, University of Minho, 4800-058 Guimarães, Portugal; gmpwagner@gmail.com (G.W.); d4224@dep.uminho.pt (S.M.)
3 Digital Transformation Colab—DTx, 4800-058 Guimarães, Portugal; rafael.silva@dtx-colab.pt
4 Lab2PT, IN2PAST, School of Architecture, Art and Design, University of Minho, 4800-058 Guimarães, Portugal; antonio.morais@eaad.uminho.pt (A.M.); joao.ribeiro@eaad.uminho.pt (J.R.); bfigueiredo@eaad.uminho.pt (B.F.); pcruz@eaad.uminho.pt (P.J.S.C.)
* Correspondence: joao-mig-silva@hotmail.com (J.M.S.); mnobrega@dep.uminho.pt (J.M.N.)

Abstract: 3D concrete printing (3DCP) requires precise adjustments to parameters to ensure accurate and high-quality prints. However, despite technological advancements, manual intervention still plays a prominent role in this process, leading to errors and inconsistencies in the final printed part. To address this issue, machine learning vision models have been developed and utilized to analyze captured images and videos of the printing process, detecting defects and deviations. The data collected enable automatic adjustments to print settings, improving quality without the need for human intervention. This work first examines various techniques for real-time and offline corrections. It then introduces a specialized computer vision setup designed for real-time control in robotic 3DCP. Our main focus is on a specific aspect of machine learning (ML) within this system, called speed control, which regulates layer width by adjusting the robot motion speed or material flow rate. The proposed framework consists of three main elements: (1) a data acquisition and processing pipeline for extracting printing parameters and constructing a synthetic training dataset, (2) a real-time ML model for parameter optimization, and (3) a depth camera installed on a customized 3D-printed rotary mechanism for close-range monitoring of the printed layer.

Keywords: 3DCP; additive manufacturing; computational modeling; automation; machine learning; computer vision

Citation: Silva, J.M.; Wagner, G.; Silva, R.; Morais, A.; Ribeiro, J.; Mould, S.; Figueiredo, B.; Nóbrega, J.M.; Cruz, P.J.S. Real-Time Precision in 3D Concrete Printing: Controlling Layer Morphology via Machine Vision and Learning Algorithms. *Inventions* **2024**, *9*, 80. https://doi.org/10.3390/inventions9040080

Academic Editor: Joshua M. Pearce

Received: 24 June 2024
Revised: 6 July 2024
Accepted: 9 July 2024
Published: 16 July 2024

1. Introduction

Driven by environmental concerns and the growing need to identify sustainable solutions and cost-effective construction methods, the construction industry is confronted by unprecedent challenges to revolutionizing this sector with innovative solutions. With a considerable environmental impact worldwide, the construction industry accounts for 40% of solid waste generation, 40% of energy consumption, and 38% of greenhouse gas emissions [1]. This framework is leading the industry to pursue a revolutionary milestone by implementing 3D printing technologies to accomplish their activities, innovating not only the construction method but also redefining how structures are built, offering design freedom that allows for the construction of innovative and complex concrete geometries while significantly reducing material waste and costs [2–4].

Different from traditional methods, which usually require hand-made efforts and intensive resource practices, 3D printing technologies take advantage of machines and computer tools to build structures in an automatic layer-by-layer manner. This allows for the accurate deposition of material where it is indeed needed, avoiding undesired deposition and material waste. Furthermore, combined with the environmental benefits it

brings, the use of 3D printing technologies in this area provides opportunities to conceive of structures never seen before, leveraging the design imagination of architects and engineers to improve structure efficiency. Thus, as this technology promises to revolutionize the construction industry, it becomes imperative to investigate and optimize the entire process, understanding the influence of different printing parameters to reach the desired printed design.

Throughout the 3D printing processes, the volatile material characteristics of concrete often require frequent adjustments to printing parameters due to its very particular challenges, which can also pose health concerns due to volatile organic compound (VOC) emissions [5,6]. Factors such as environmental conditions, variations in concrete batches, mixing duration, and even the geometric definition of printing paths can significantly influence the final quality of printed pieces. Nevertheless, as depicted in Figure 1, in the context of 3D concrete printing (3DCP) at ARENA—the robotic fabrication research lab of the School of Design and Architecture of the University of Minho—two primary variables crucial to layer morphology quality stand out: material flow and motion speed.

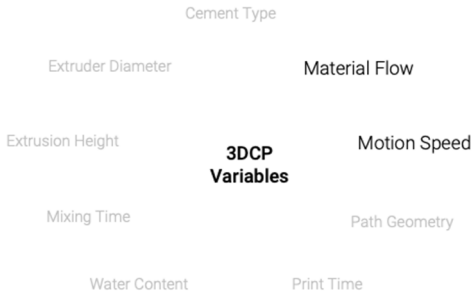
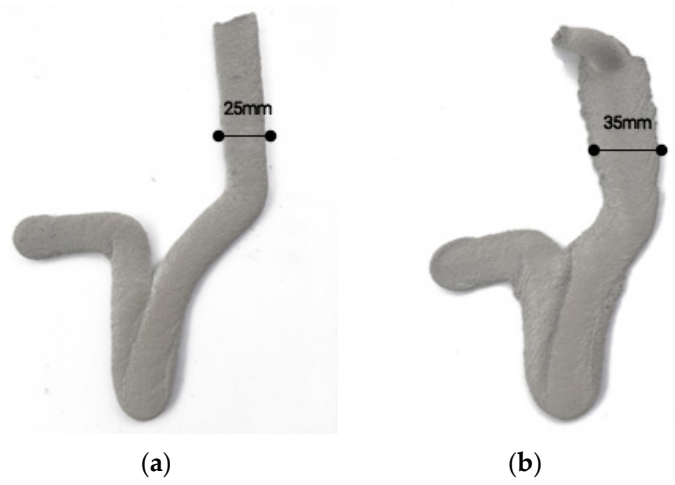


Figure 1. A survey of the most important variables involved in the printing process in the context of 3D concrete printing at ARENA.

Based on our experience with robotic 3D concrete printing (3DCP), it has become evident that frequent adjustments to both the motion speed of the extruder and the rate of material deposition are essential to ensure high-quality results (Figure 2). Elevated motion speed values lead to a decrease in layer width, as do high extrusion flow rates. However, altering either of these variables can also have additional effects.



(**a**) (**b**)

Figure 2. Effects of material flow and motion speed on layer width: (**a**) constant flow and speed; (**b**) varying both parameters.

A low motion speed prolongs the printing duration, which is detrimental to the concrete-printing process. Abrupt changes to the extrusion flow rate alter the material behavior due to the varying kinetic energy imparted to the mix by the spindle rotation.

Therefore, we have found that maintaining a consistent flow rate while adjusting the motion speed produces superior results within our setup.

The 3D concrete-printing process requires different mechanisms to be controlled, tuned, calibrated, and work synchronously in order to produce a quality print. It is characterized by a pipeline where a lot of data can be captured and processed to learn hidden correlations between variables and machine setups that can help improve the quality of fabrication. A data engine capable of capturing these data can be vital to construct datasets that represent the history of the fabrication process, something extremely valuable to track down where things went wrong, why certain artefacts may have happened, or even to construct further studies like a life cycle assessment.

Automating the control of motion speed offers several advantages, including reducing human-induced errors and response delay, providing fine control over the desired layer shape according to the design, and freeing operators from frequent adjustments. However, accurately predicting the need for changes in motion speed is challenging. To enable this automation effectively, two conditions must be met: the precise measurement of the printed layer width and understanding the degree of change in motion speed required to achieve the desired width under specific conditions.

In this study, an extreme gradient boosting (XGB) algorithm is informed by a depth camera (Intel Realsense L515) mounted on a custom rotary mechanism, leveraging computer vision (CV) frameworks like OpenCV [7]. These frameworks facilitate real-time measurement, while the machine learning model accurately predicts printing parameter values based on current conditions. Supporting our machine learning (ML) implementation is the generation of a dataset from a numerical solver simulation utilizing OpenFOAM. This approach ensures material-specific neural network training. The generation of synthetic datasets, as proposed previously using encoded images [8], offers an efficient alternative to the resource-intensive process of conducting numerous printing runs to collect the requisite data [9].

In the realm of 3D printing optimization, the integration of machine learning (ML) represents a vast and burgeoning area of research [10]. This application spans various techniques, including fused deposition modeling, stereolithography, and selective laser melting, and encompasses activities from material preparation to predicting the mechanical properties of printed components [11]. Vision-based systems, which extract information from captured images or videos, constitute a notable subset of these techniques. Significant advancements have been made across various applications due to the ability of machine vision to automatically identify and measure events with great accuracy in real time. A notable example of this can be found in the world of sports, where it is used to automatically identify and measure key moments during matches [12]. In the context of automation in manufacturing processes, computer vision can increase efficiency and productivity of industries as it can facilitate the identification and manipulation of objects in assembly lines [13]. Further in the manufacturing environment, these systems facilitate tasks such as deviation calculation from digital models [14], automatic path correction [15], defect detection [16], surface quality control [17], and process monitoring [18]. Our proposal aligns with this trend, aiming to integrate a camera-based system with an ML model. However, the approach proposed in this work stands out by focusing on autonomously controlling the motion speed of a six-axis robot throughout the printing process, with the unique objective of maintaining a desired layer width.

2. Materials and Methods

2.1. Fabrication Framework

This study aims to achieve four primary objectives: (i) generate a comprehensive dataset of printing parameters for 3D concrete-printing (3DCP) fabrication, (ii) facilitate real-time monitoring of printed layer width, (iii) assess the accuracy of XGB in predicting the correct values for printing parameters according to a pre-intended outcome, and (iv) evaluate the implications of XGB-based control on the design process.

Figure 3 outlines the steps required for implementing XGB and its integration within a fabrication framework. The material properties of the concrete mix are integrated into a simulation of the 3D concrete-printing (3DCP) process, enabling the extraction of values used in different printing parameters (e.g., the movement speed of the extruder) to achieve a certain printed layer shape, as well as other measurable characteristics of the printed layer (e.g., width and height) to generate a synthetic dataset. This dataset is subsequently utilized to train an XGB model, which is deployed in real time with support from a computer vision system. A data-capturing pipeline is employed to refine the model using experimental data. The framework serves two main purposes: firstly, to more accurately estimate the time and material required to complete the print, considering unpredictable variations in motion speed and material flow rate throughout the process; and secondly, to autonomously control the printing parameters.

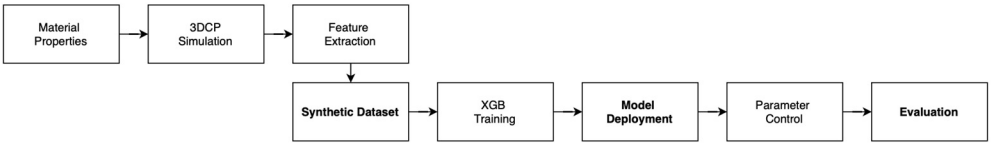


Figure 3. In a 3DCP fabrication setting, the process involves generating data, training a machine learning algorithm, and then deploying and evaluating the model.

2.2. Machine Learning Framework

2.2.1. Data Pipeline

In order to understand how layer morphology is impacted by the different variables involved in the printing process, it is essential to capture multiple states—i.e., the many combinations of printing parameters and the resulting layer widths, as demonstrated in Figure 4. The analysis of states looks to answer questions such as "how does layer width relate to the other parameters?", and "is there a way to predict how thick or thin the extruded layer will be given the values for all the other variables?". By capturing and evaluating thousands of these states, we can start modeling the relationship between the multiple variables and the target variable—layer width.

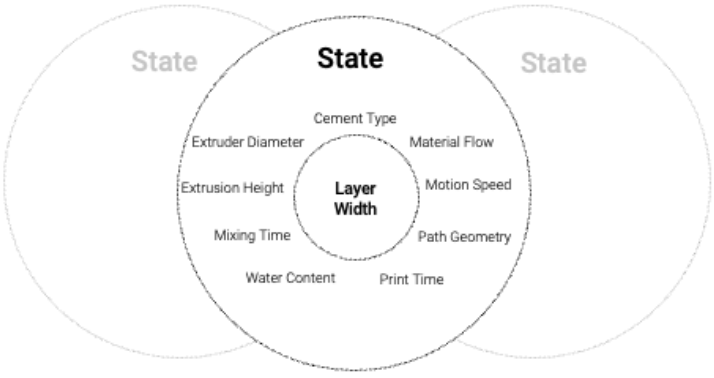


Figure 4. Understanding the impacts of 3DCP variables on layer width in different states. The figure highlights how different states result in varying printed layer morphologies (shown in grey is a reference to other possible states for different printing runs).

However, datasets for machine learning applications require a very large number of samples to be effective. This is particularly problematic in the context of digital fabrication, as producing hundreds to thousands of prototypes can be extremely costly both in time and material resources. For this reason, our approach was to build a synthetic dataset from a numerical simulation of 3DCP, tailored to the material properties of the concrete mix used. These simulations, as illustrated in Figure 5, can be computed at marginal costs when compared to the printing process, as well as be produced with the exact variable specifications that can capture a wide and representative corpus of data. Thus,

it must be taken into account that numerical simulations are performed in a discretized domain (mesh) and governed by discretized equations that represent an approximation of the real-world solution, which then requires a special attention to correctly setting up the simulation to provide accurate results and avoid improper/incorrect forecasts of the experimental printings.

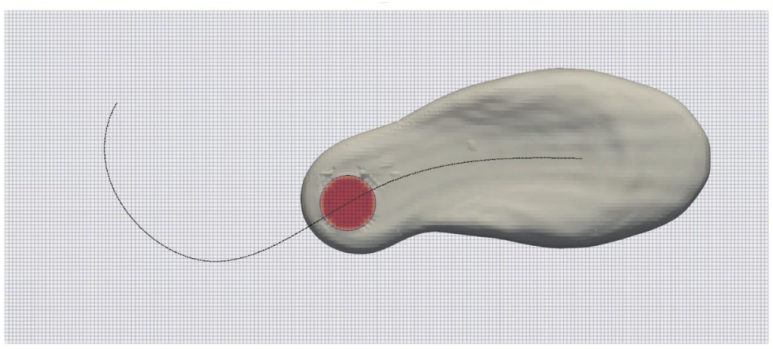


Figure 5. Simulation of the 3DCP process to create a dataset for machine learning.

To accurately represent the concrete material used in printing at the lab, a process of fine-tuning and calibration was undertaken to ensure an adequate behavior between the digital and the physical material, as shown in Figure 6. This process entailed identifying the material properties (density, yield stress, kinematic viscosity, and consistency coefficient), taking measurements of printed specimens and studying how the simulation would react to changes to the printing parameters (material flow rate, motion speed, extrusion height, etc.). Thus, motivated by inherent numerical constraints that may slightly deviate the simulated result from the experimental one, different simulations were initially performed with different printing parameters (as observed in Figure 6c) to study how the simulation reacts to changes in printing parameters, providing insights for crucial adjustments in simulation conditions (such as mesh refinement) to obtain the same behavior as the experimental printed material while balancing computational time.

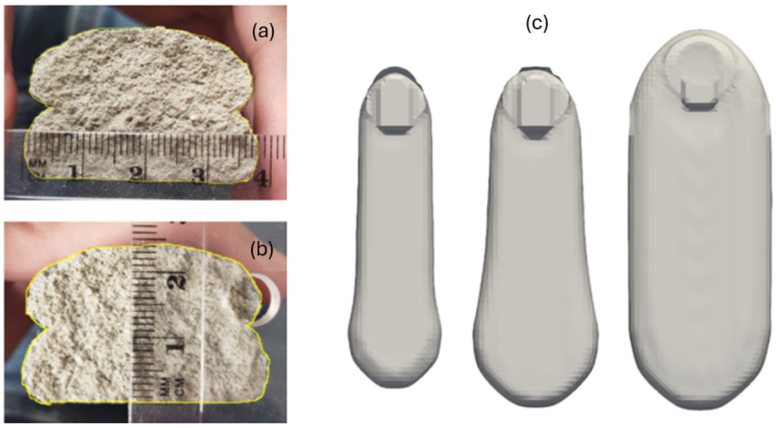


Figure 6. Survey of printed specimens (**a**,**b**), and first simulation experiments on variations of material flow (**c**).

2.2.2. Numerical Simulation

To accurately simulate the rheological behavior of the concrete mixture, which is classified as a generalized Newtonian fluid (GNF) with yield stress, the Herschel–Bulkley model was employed. This model describes a non-linear relation between the shear stress and the shear rate (variable viscosity) above yielding, where the flow of this fluid only

occurs once the shear stress is greater than the material yield stress [19,20]. This behavior is described by the following Equation (1):

$$\tau_{ij} = \tau_0 + 2KD_{ij}^n \quad (1)$$

This model relates the shear stress tensor components (τ_{ij}) and the rate of deformation tensor components (D_{ij}^n), which is equivalent to the shear rate tensor, according to three material parameters: consistency index (K), yield stress (τ_0), and viscosity exponent (n) [19,21]. The viscosity exponent describes the viscosity non-linearity above yielding, which states that when $n < 1$ the viscosity decreases with increasing shear rates (shear-thinning behavior); $n > 1$ corresponds to a shear-thickening behavior, and when $n = 1$ the model matches the Bingham model, corresponding to a constant viscosity (Newtonian fluid) [19,22].

In this work, the material had a shear-thickening behavior ($n > 1$), and the input parameters used in the simulation are stated in Table 1.

Table 1. Input parameters used to describe the behavior of the concrete mixture [23].

Variable	Value
Density [ρ]	2070 kg/m^3
Yield Stress [τ_0]	8.28 kPa
Consistency Index [K]	20.7 Pa.s$^{1.56}$
Viscosity Exponent [n]	1.56

The numerical simulations executed in this work were performed using the "overInterDyMFoam" solver from OpenFOAM [24] v2112, which is an open-source computational library for several engineering, academic, and scientific applications. This solver uses the VOF (Volume of Fluid) method to model the flow of two immiscible, incompressible, and isothermal fluids: concrete mixture and air. The two-phase flow is integrated with an overset mesh strategy [25], which combines a background mesh with an overset mesh. This approach allows the extrusion nozzle to move within the domain and deposit the concrete mixture onto the background mesh. In the overset approach used by "overInterDyMFoam", two meshes are used: the backgroundMesh, which is static and represents the printing area, and the oversetMesh, which is dynamic and represents the extrusion nozzle region. In this approach, the oversetMesh can move inside the backgroundMesh along a predefined path. This enables the exchange of information and fluid flow results between the two meshes, mimicking the behavior of the real system. For that purpose, these meshes must be generated independently and later merged into one unique overlapped mesh, as illustrated in Figure 7.

In order to obtain the correct interaction between meshes and fluid structure (concrete mix and nozzle geometry), this solver uses the concept of "cellTypes" to distinguish three different types of cells that compose this computational domain, labeling them as 0, 1, and 2. Cells with values equal to 0 are defined as the cells that will be calculated in the simulation, representing all cells of the static mesh and some of the dynamic mesh; those with values equal to 1 are the interpolating cells, which represent the interface between overlapping meshes (oversetMesh and backgroundMesh) as well as between structure and mesh (extrusion nozzle and oversetMesh); and cells with values equal to 2 are the ones that will not be calculated in the simulation, acting as inactive cells that represent the geometry of the structure (extrusion nozzle) in the oversetMesh. The representation of the cell types is shown in Figure 8, which illustrates a sliced view of the computational domain.

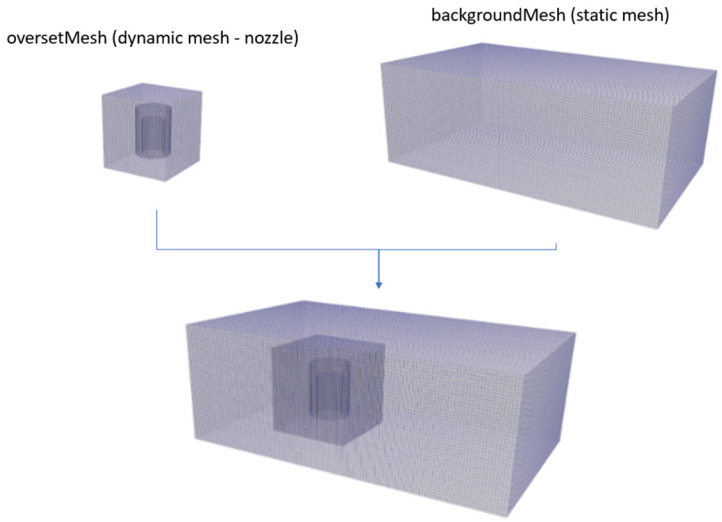


Figure 7. Representation of a computational domain (**bottom**) composed by backgroundMesh and oversetMesh (**top**).

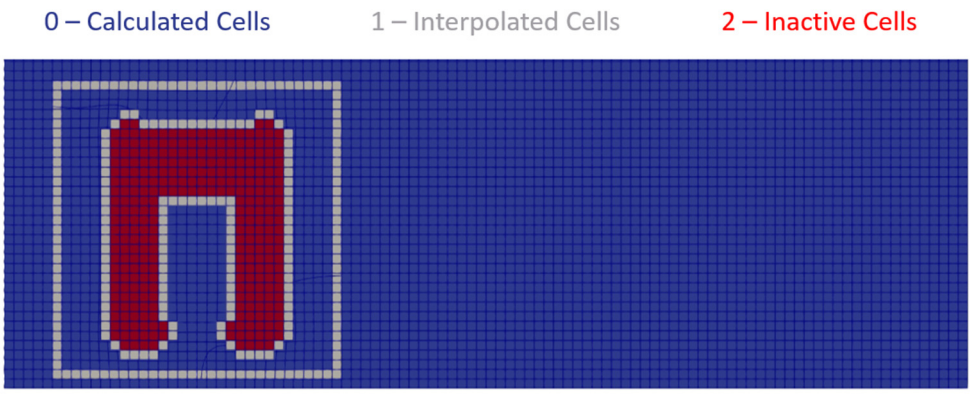


Figure 8. Representation of cellTypes in a lateral sliced view of computational domain.

To precisely replicate the dynamics and attributes of the simulated flow, it is imperative to organize the external faces of the mesh into coherent groups, as depicted in Figure 9 by varied colors. These grouped entities are designated by patches, which are crucial to assigning appropriate boundary conditions, since they define how the system interacts with its surroundings.

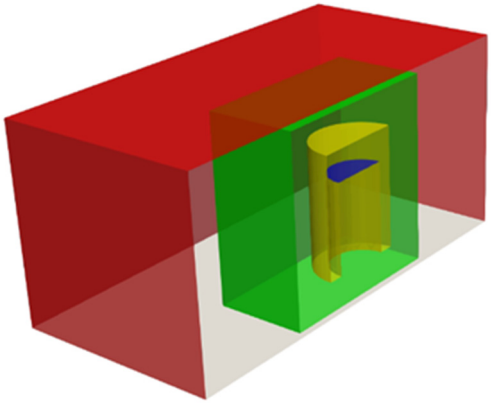


Figure 9. Representation of patch definitions in a clipped view of the computational domain. The patches data is provided in Table 2.

Table 2 shows the boundary conditions applied to the different solution fields, where the outlet and floor patches are related to the backgroundMesh, and the inlet, overset, and nozzleWalls patches are related to the oversetMesh.

Table 2. Employed boundary conditions for numerical simulations in the patches illustrated in Figure 9.

Patch	Pressure, p	Velocity, U
inlet	fixed flux	fixed value Uin
overset	overset	overset
nozzle Walls	fixed flux	0
outlet	constant atmospheric pressure	null normal gradient
floor	fixed flux	0

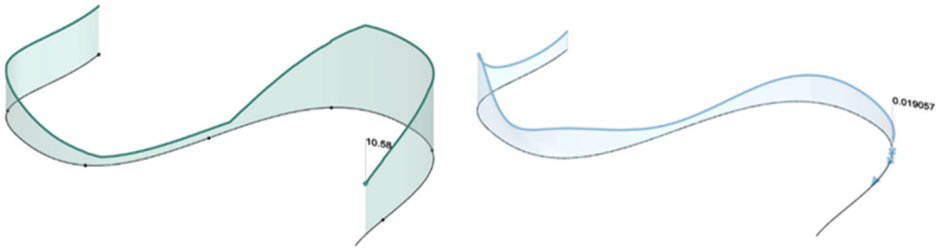
2.2.3. Dataset Generation

Firstly, a Python script was developed to generate unique geometrical configurations of print paths as shown in Figure 10. The objective of this script was to create a vast and diverse collection of geometries that would empower the trained model to be applied to any type of object, regardless of its geometrical definition.

Figure 10. Sample of printing paths generated by the Python script.

The script works by populating 7–13 control points that construct a spline. Additional parameters, such as the maximum radius at which the control points are placed (and therefore the distance between points), or the edginess—a parameter that controls the smoothness of the curve—can be fine-tuned to create additional variations.

For each chosen print path geometry, each of the other variables (material flow, motion speed, extrusion height, etc.) is matched in different combinations along that path, as shown in Figure 11. This is essential to model not just how individual variations to one parameter impact the width, but also how changes in many of them will. In the modeling code, the printing path is inserted by specifying the various locations of the extrusion nozzle (oversetMesh) and the velocity between them.



(**a**) (**b**)

Figure 11. Visualization of the variation of (**a**) motion speed and (**b**) material flow for a portion of the printing path.

A total of 6 simulations were collected using this method. The output for each of these simulations is an STL type file containing a triangulated 3D surface mesh, which allows for quantifying how the changes to the variables throughout a print impact the resulting layer shape. Figure 12 shows the output of 3D surface meshes from simulation.

Figure 12. Output of 3D meshes from the simulation process.

Once produced, these mesh files can be post-processed to create the dataset that will be used to train the machine learning model, as shown in Figure 13. The post-processing stage is about creating the necessary features—that is, things that can be measured about and/or related to the extruded layer at any point in time during the print and can help explain the problem to a machine learning model. Features will include not just the variables that we are in control of during a print—motion speed, material flow, extruder diameter, etc.—but also new aspects that can help explain what is unique for each point (such as the overlap with portions already extruded, or the evolution of the direction of the extruder by taking its angle in relation to a previous moment in the print, etc.). A more detailed explanation of features is provided below:

- Time (s)—The time elapsed since the start of the print until reaching that position;
- Speed (cm/s)—The motion speed of the extruder at that location;
- Flow (m/s)—The material flow at that location;
- Distance (mm)—The distance between that location and the previous location sampled in the dataset;
- Angle (degrees)—The difference in angle between the tangent vector at that location and the previous datapoint (important to distinguish between the type of motion being executed—straight, curved, etc.);
- Height (cm)—The distance of the extruder tip to the deposition plane (printing base or previous layer);
- Diameter (cm)—The diameter of the extruder;
- Width (cm)—The width of the layer at that location;
- Overlay—The number of intersections between a perpendicular line to the printing path drawn at that location and previous locations.

The extraction of features was conducted in Grasshopper by importing the mesh and computing the different measurements. Once extracted, the combination of locations and features constructed the final dataset used to train the machine learning model. The employed dataset was composed of 2245 entries and 9 features, taken from all the different simulations produced before. A sample of this dataset is provided in Table 3, in which each row represents a sampled location in the print path and each column represents one of those features.

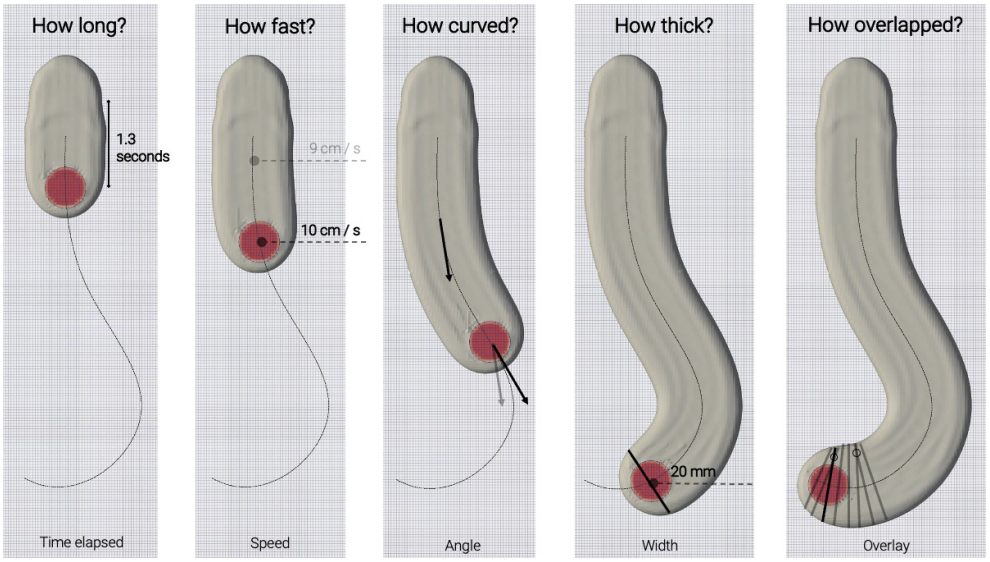


Figure 13. Post-processing of simulation meshes to extract dataset features.

Table 3. Sample of the dataset structure.

Time [s]	Speed [cm/s]	Flow [m/s]	Distance [mm]	Angle [degree]	Height [cm]	Diameter [cm]	Width [cm]	Overlay
4.26344	11.3	0.233	30	0.036599	12	20	57.427403	0
7.22211	10.0	0.264	40	0.035724	12	20	57.337229	2
1.24866	12.7	0.264	60	0.033314	12	40	56.499766	0
3.27521	8.4	0.264	30	0.031016	8	20	55.649535	0
2.30175	8.9	0.264	30	0.028557	12	40	55.088962	3
15.3283	9.1	0.264	12	0.026146	10	20	57.243402	0
33.35485	7.7	0.292	30	0.023822	12	40	58.402407	0
64.3814	11.3	0.292	14	0.021642	12	20	63.236045	11
114.40795	11.0	0.292	30	0.01961	8	10	63.371653	0
26.43449	8.3	0.292	15	0.017804	10	20	63.022733	5
88.46104	12.3	0.307	15	0.015641	12	25	60.956338	0
179.48759	10.0	0.307	45	0.013088	14	20	61.726337	3
57.51414	10.0	0.307	76	0.010921	6	25	66.683418	1
61.54069	9.6	0.331	10	0.009159	12	20	63.430119	0

3. Results and Discussion

3.1. Algorithm: Training and Inference

3.1.1. Model Architecture

The choice of machine learning algorithm is an essential part of the process to attain good results during inference. It is therefore essential to accurately structure and clean the data with the final application in mind, as well as suiting it to the type of algorithm used. An evaluation of our dataset reveals two clear distinct applications: (1) a model that will help designers achieve variable layer shape by predicting the speed at which the extruder needs to move in different parts of the geometry; and (2) the automated correction of layer shape during fabrication. The first application is related to the design stages, before manufacturing, by understanding the impacts of speed in the outcome. The second application is related to the fabrication process itself, automating the control of speed in

relation to live measured values to maintain the target layer widths even if other conditions change. The dataset structure is shown in Figure 14.

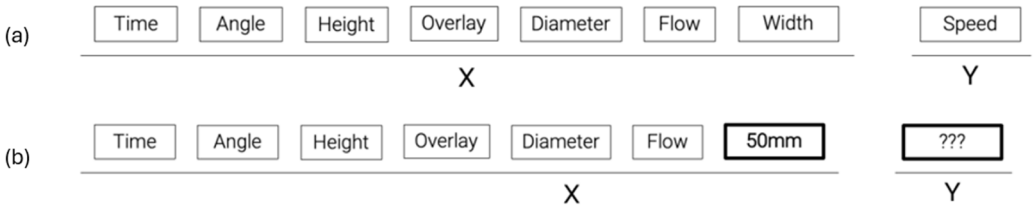


Figure 14. Dataset structure for (**a**) model training and (**b**) inference.

Even if there are differences between model architectures for each application, both require the same data structure. It is therefore important to identify the target feature (Y) used to control layer width in relation to each state (X). After training, inference is achieved by locking a target value for the layer width and predicting the value of speed that will result in that fixed layer width. Another important step is scaling the values for the different features between 0 and 1, an important technique to be able to use the model with different domains of measurements. Finally, the dataset is split between training (70%) and validation data (30%).

Different model architectures were evaluated to understand what type of algorithm would provide the best performance, as shown in Figure 15. A popular machine learning package, ScikitLearn 1.2.2, was used to gain access to these architectures and fit them to our dataset.

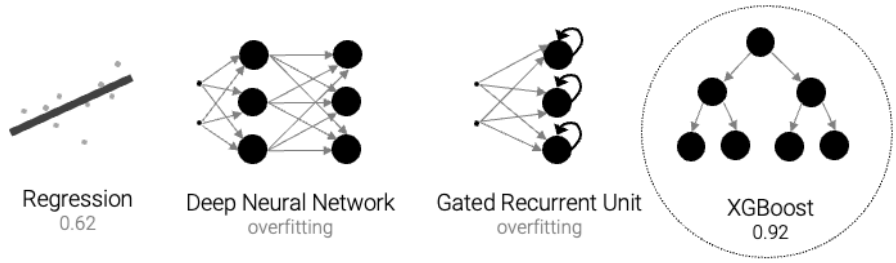


Figure 15. Model architectures and their performance, evaluated with the synthetic dataset.

The first model evaluated, a linear regression, showed very low accuracy (0.62). This result reveals the complexity of the relationships between state variables, too high for such a simple model. This was an expected result, but nonetheless important to validate the need for a more advanced machine learning technique.

A deep neural network consisting of different combinations of dense layers was also tried. This model showed good results for the portion of data it was trained on but failed to yield good predictions for unseen data—a behavior also known as overfitting. This shows that, although this type of architecture could eventually provide good results, the amount of data it required was greater than the corpus established so far. For that reason, we looked at a different approach.

A gated recurrent unit, an algorithm belonging to the family of recurrent neural networks (models that take entire sequences of data points at a time during training), was also evaluated. We believed this model would have good performance due to its ability to track the evolution throughout the printing path by modeling large sequences of data points (both current and past ones) when inferring motion speed. This could be an important aspect in the context of 3DCP, as each state is not independent from past ones. Structuring the dataset as a sequence with a time dimension allowed us to encode acceleration, as well as characterize the print path geometry through the angle to p-1 (deviation in vector direction to the previous point). The points were spaced out at irregular intervals (distance to p-1) to accommodate any limitations in reading accuracy of the layer width using the camera setup. Due to the lack of the necessity for a long sequence length, we opted to test

a gated recurrent unit (GRU) beyond the more commonly used long short-term memory (LSTM) architecture. This network requires less memory and is faster [26], both important factors to achieve real-time inference and potentially deploy it in a minimal setup like a Raspberry Pi board. However, the problem of overfitting was again encountered. The sequenced nature of this model requires structured data, meaning that each sequence had to be extracted from a simulation and kept in its original order. This is problematic because it greatly reduces the amount of training data available.

Finally, an XGBoost (extreme gradient boosting) architecture achieved the greatest results, with an accuracy of 0.92 when predicting motion speed. XGBoost is a variation of decision tree algorithm that is known for very good results with tabular data, such as in our case. This model has the advantage of being able to intake unstructured data points, which makes it possible to combine all the simulations into one data set.

3.1.2. Model Application

The XGBoost model was then trained in two variations. Variation 1 predicted layer width, given a state that included motion speed. Variation 2 predicted motion speed, given a state that included the layer width. Figure 16 shows a comparison between the layer width values extracted from a simulation and the predicted values of Variation 1 of the XGBoost model. The results showed excellent performance throughout the entire print path. A closer look revealed that the predictions were worse for the curved portions of the print path, indicating that an expansion in the dataset in relation to the print path geometry could provide even better performance in these areas.

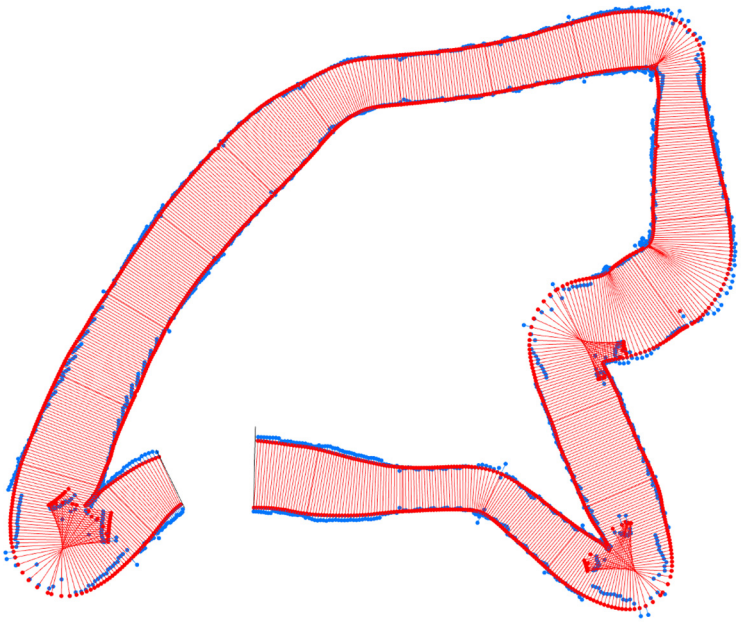


Figure 16. Visualization of layer width during a print: a comparison between simulation (red) and prediction (blue) of unseen data.

Particularly important for the application of this model in the context of the design was to establish a live connection between commonly used modeling software and the Python environment where inference could be computed. For this reason, we developed a solution built on top of Hops—a component for Grasshopper that adds external functionality to the software by connecting it with Python environments running on local or remote machines, as demonstrated in Figure 17. This integration allows designers to apply the model to their own geometries, simply providing the layer widths they are interested in achieving. The model will return the correct values for motion speed, which can then be used in normal 3D printing workflows where the code that is provided to the robot or printing machines needs to already have information on traveling speeds at each location.

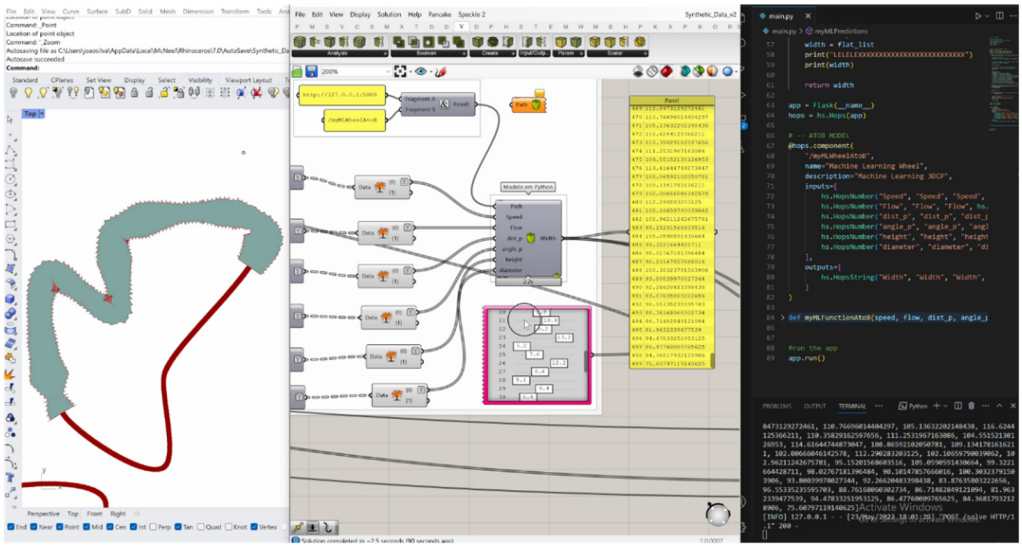


Figure 17. Implementation of the model with the Grasshopper environment.

For the live monitoring application, the dataset had to be expanded in order to create a relationship between the current state (the state that encapsulates the current motion speed and layer width) and the target state (a state that would encapsulate the correct measurement of layer width and the motion speed value necessary to achieve it). Instead of predicting a value for motion speed, this model should predict how much the current motion speed needs to change in order to attain a provided target width. To create this relationship between states, the model previously trained was used. It can already predict motion speed given a value for layer width, and vice versa. Therefore, it can expand the original dataset with new data for target values. The new dataset structure is shown in Figure 18.

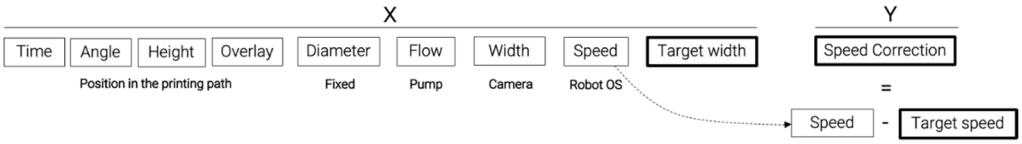


Figure 18. New dataset structure for training a model capable of adjusting parameters in real time during fabrication.

Another important aspect for this model is that inference will happen during fabrication. Because of this requirement, all the variables need to be provided in real time. The variables time, angle, height, and overlay are constructed by reading the position of the extruder in the print path. Diameter refers to the geometry of the extruder and is therefore known a priori. Flow is given by reading the pump machine with an Arduino. However, the layer width needs to be measured in real time, not by post-processing a simulation like before. For that reason, we had to develop a computer vision algorithm that could provide the necessary information during the print for inference.

The evaluating accuracy for both the XGBoost model for designing applications—predicting speed for a given width (Figure 19)—and the XGBoost model for monitoring applications—predicting speed correction given a target width with camera readings (Figure 20)—presented good results. Accuracies above 80% are generally considered high performance. However, we believe an increase in the dataset size could push these accuracy values to better results. Furthermore, it is important to notice that these results are still to be systematically evaluated under real print scenarios, where noise in data readings can possibly have a negative impact on performance.

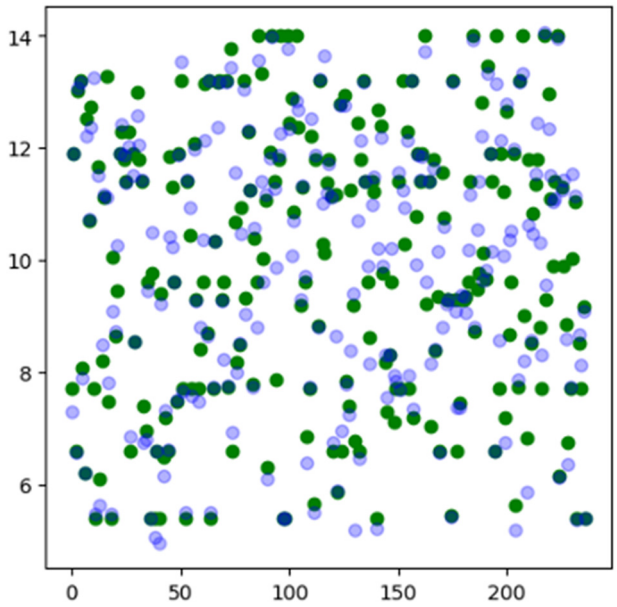


Figure 19. Accuracy of XGBoost for designing applications: 0.87. Ground truth (green) vs. predictions (blue) for motion speed in unseen data.

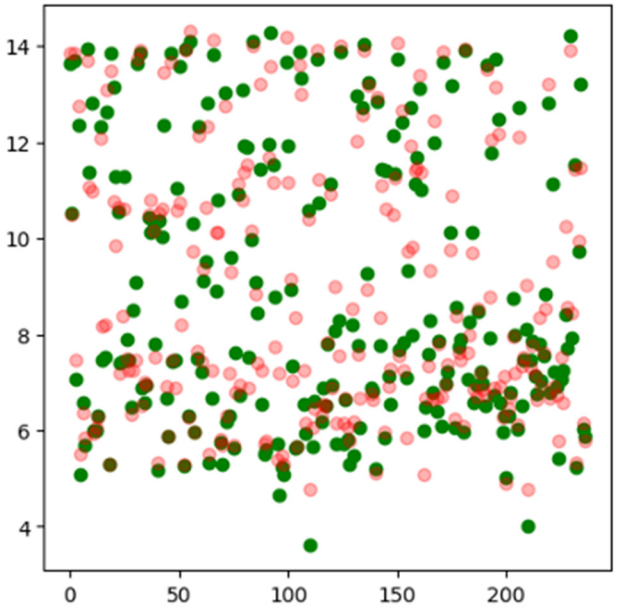


Figure 20. Accuracy of XGBoost for live monitoring application: 0.91. Ground truth (green) vs. predictions (red) for motion speed correction in unseen data.

3.2. Computer Vision System

To ensure accurate monitoring of layer width and precise motion control in a 3DCP fabrication setup, a computer vision system is indispensable. To achieve this, a specialized system (refer to Figure 21) was devised, capable of accommodating a camera and allowing it to rotate 360 degrees for alignment with the printing direction. This system comprised a pair of gears: one linked to a stepper motor regulating rotation and the other affixed to a 3D-printed support holding the camera parallel to the ground, with the extruded layer positioned at the image's center. The rotation of the stepper motor, and consequently the camera, was orchestrated by an Arduino unit external to the illustrated system. Conversely, image-processing operations were executed on a Raspberry Pi, powered by a separate

power bank. The Raspberry Pi, camera, and power bank rotated synchronously, ensuring a self-contained system free from constraints posed by tangled wires.

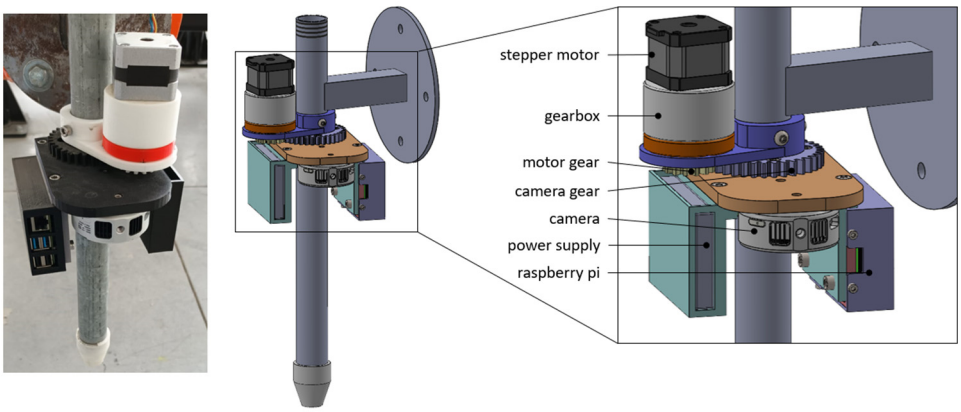


Figure 21. Rotary mechanism for camera tracking.

The real-time control of the rotary mechanism was facilitated by linking the Grasshopper environment with an Arduino through the Funken library [27]. Given the frequent fluctuations in motion speed during fabrication, it was imperative to provide motor angle adjustments promptly. The procedure involved exposing the live position of the extruder via the JOpenShowVar API v0.2.2 [28], computing the vector difference from the initial vector at the origin, and correlating it to the required number of motor steps for aligning the camera with the print path.

3.3. Real-Time Deployment

At the heart of the proposed framework, depicted in Figure 22, was real-time communication among various computational processes and fabrication machines. The Grasshopper environment served as the central hub, receiving live extruder location data and controlling both the concrete pump and rotary mechanism via Arduino serial communication. Concurrently, the Python v3.9 environment handled the processing of the depth camera's video stream, extracting information regarding layer width. These data were subsequently relayed to the machine learning model. Utilizing a sequence of data capturing movement type and printing parameters, the model determined the optimal extruder speed required to sustain or achieve a predefined target layer width.

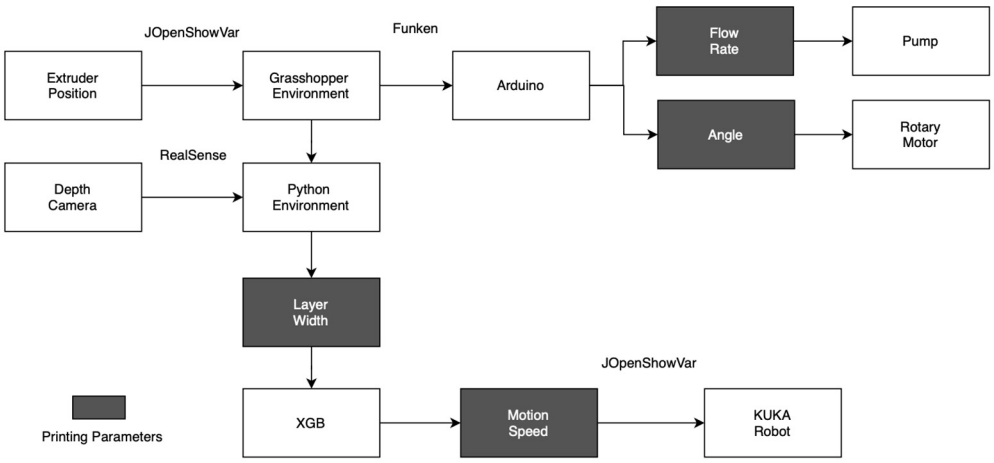


Figure 22. Diagram showing the data pipeline between software and hardware.

Figure 23 demonstrates the self-alignment of the camera with the extrusion by the rotary system.

Figure 23. Automatic self-alignment of the camera with the extrusion by the rotary system.

Figure 24 demonstrates the camera video stream as it performs real-time measurements of the printed layer.

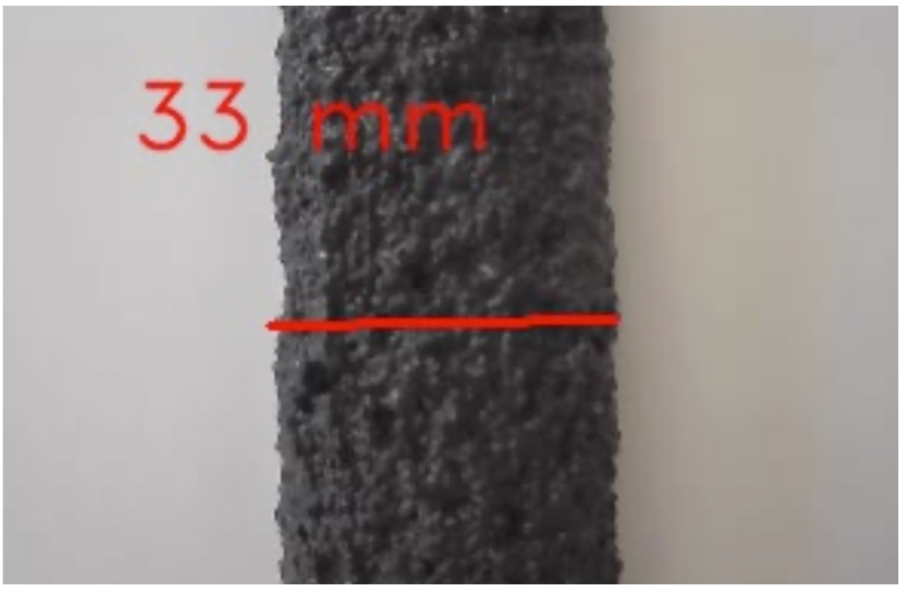


Figure 24. Print screen of the camera video stream as it performs real-time measurement of the printed layer width.

4. Conclusions

The work described involved applying various technologies to create a smart system for additive manufacturing. The main goal was to automate control of the printed layer width, specifically by controlling the moving speed of the robot. The desired outcome was to predict the robot's motion speed along the printing path to maintain a predetermined layer width or achieve a new target width.

The process began by using an algorithm to generate a collection of geometries that simulated different printing patterns. These geometries were then used in a numerical simulation to create a dataset consisting of various combinations of variables. This dataset was later used to train a machine learning model. Different model architectures were tested until a viable solution for real-time application was found—an XGBoost model. To monitor the layer width during printing, a physical system capable of tracking the printing path and measuring the printed layer width in real time using a depth camera was developed.

While further experiments are needed to fully evaluate the fabrication framework, the work undertaken has resulted in a machine learning algorithm that can accurately predict the robot's speed with 87% precision for any target width and a wide range of printing setups. However, it should be noted that these results still need to be validated in real-world printing scenarios. The work has also led to the development of a smart physical system that can monitor the print in real time and adjust the motion speed, which is crucial for controlling the layer width and may have other potential applications.

Concerning challenges encountered, the first one involved the rotary mechanism, which affected the frequency of the camera's readings. Depth streaming typically requires a USB cable connection, which limited the camera's rotation due to the risk of the wire becoming tangled in the extruder. To address this, we integrated a Raspberry Pi directly into the mechanism, eliminating the need for a direct connection to a computer. However, the Raspberry Pi's limited processing capacity occasionally caused failures in the video stream readings.

Additionally, the most significant challenge to the application of our system was that the simulations and synthetic dataset were specifically tailored to the exact concrete mix used in our lab. This means that the system cannot be directly applied to other variations of concrete mix without creating a new dataset and retraining the model. In our case, the material mix was generally fixed, so this was not an issue. However, we recognize that expanding the dataset to include other concrete mixes could enhance the system's versatility and broaden its application opportunities, opening doors for future work.

Author Contributions: Conceptualization, J.M.S., G.W., J.M.N., R.S., S.M., A.M., J.R., B.F. and P.J.S.C.; methodology, J.M.S., G.W., J.M.N., S.M., B.F. and P.J.S.C.; software, J.M.S., G.W. and J.M.N.; validation, J.M.S., A.M., J.R. and P.J.S.C.; investigation, J.M.S., G.W., J.M.N., R.S., S.M., A.M., J.R., B.F. and P.J.S.C.; data curation, J.M.S. and G.W.; writing—original draft preparation, J.M.S. and G.W.; writing—review and editing, J.M.S., G.W., J.M.N., R.S., S.M., A.M., J.R., B.F. and P.J.S.C.; supervision, J.M.N., S.M., B.F. and P.J.S.C.; project administration, J.M.N., S.M., B.F. and P.J.S.C.; funding acquisition, J.M.N., S.M., B.F. and P.J.S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is co-funded by: the European Regional Development Fund (ERDF) through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) of the Portugal 2020 Program Project No. 47108, "SIFA"; Funding Reference: POCI-01-0247-FEDER-047108; FCT—Fundação para a Ciência e a Tecnologia through the projects Lab2PT—Landscapes, Heritage and Territory Laboratory [UIDB/04509/2020], IN2PAST—Associate Laboratory for Research and Innovation in Heritage, Arts, Sustainability and Territory [LA/P/0132/2020], IPC-Institute for Polymers and Composites [UID/CTM/50025/2019 and UIDB/04436/2020] and by FCT Doctoral SFRH/BD/145832/2019. This work was also supported under the base funding project of the DTx CoLAB—Collaborative Laboratory, under the Missão Interface of the Recovery and Resilience Plan (PRR), integrated in the notice 01/C05-i02/2022.

Data Availability Statement: The data that support the findings of this study are available from the corresponding authors upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. De Schutter, G.; Lesage, K.; Mechtcherine, V.; Nerella, V.N.; Habert, G.; Agusti-Juan, I. Vision of 3D printing with concrete—Technical, economic and environmental potentials. *Cem. Concr. Res.* **2018**, *112*, 25–36. [CrossRef]
2. Khan, M.S.; Sanchez, F.; Zhou, H. 3-D printing of concrete: Beyond horizons. *Cem. Concr. Res.* **2020**, *133*, 106070. [CrossRef]
3. Paul, S.C.; van Zijl, G.P.A.G.; Gibson, I. A review of 3D concrete printing systems and materials properties: Current status and future research prospects. *Rapid Prototyp. J.* **2018**, *24*, 784–798. [CrossRef]
4. Lakhdar, Y.; Tuck, C.; Binner, J.; Terry, A.; Goodridge, R. Additive manufacturing of advanced ceramic materials. *Prog. Mater. Sci.* **2021**, *116*, 100736. [CrossRef]
5. Baduge, S.K.; Navaratnam, S.; Abu-Zidan, Y.; McCormack, T.; Nguyen, K.; Mendis, P.; Zhang, G.; Aye, L. Improving performance of additive manufactured (3D printed) concrete: A review on material mix design, processing, interlayer bonding, and reinforcing methods. *Structures* **2021**, *29*, 1597–1609. [CrossRef]

6. Minář, J.; Pilnaj, D.; Uřičář, J.; Veselý, P.; Dušek, K. Application of solid-phase microextraction arrows for characterizing volatile organic compounds from 3D printing of acrylonitrile-styrene-acrylate filament. *J. Chromatogr. A* **2023**, *1705*, 464180. [CrossRef]
7. Bradski, G. The OpenCV Library. *Dr. Dobb's J. Softw. Tools* **2000**, *25*, 122–125.
8. Rossi, G.; Nicholas, P. Encoded Images: Representational protocols for integrating cGANs in iterative computational design processes. In *ACADIA 2020 Distributed Proximities: Proceedings of the 40th Annual Conference of the Association for Computer Aided Design in Architecture, Online, 24–30 October 2020*; Elsevier: Amsterdam, The Netherlands, 2021; Volume 1, pp. 218–227.
9. Thomsen, M.R.; Nicholas, P.; Tamke, M.; Gatz, S.; Sinke, Y.; Rossi, G. Towards machine learning for architectural fabrication in the age of industry 4.0. *Int. J. Archit. Comput.* **2020**, *18*, 335–352. [CrossRef]
10. Mahmood, M.A.; Visan, A.I.; Ristoscu, C.; Mihailescu, I.N. Artificial neural network algorithms for 3D printing. *Materials* **2021**, *14*, 163. [CrossRef]
11. Goh, G.D.; Sing, S.L.; Yeong, W.Y. A review on machine learning in 3D printing: Applications, potential, and challenges. *Artif. Intell. Rev.* **2021**, *54*, 63–94. [CrossRef]
12. Goh, G.L.; Goh, G.D.; Pan, J.W.; Teng, P.S.P.; Kong, P.W. Automated Service Height Fault Detection Using Computer Vision and Machine Learning for Badminton Matches. *Sensors* **2023**, *23*, 9759. [CrossRef]
13. Golnabi, H.; Asadpour, A. Design and application of industrial machine vision systems. *Robot. Comput. Integr. Manuf.* **2007**, *23*, 630–637. [CrossRef]
14. Langeland, S.A. Automatic Error Detection in 3D Pritning Using Computer Vision. Master's Thesis, The University of Bergen, Bergen, Norway, 2020.
15. Sutjipto, S.; Tish, D.; Paul, G.; Vidal-Calleja, T.; Schork, T. Towards Visual Feedback Loops for Robot-Controlled Additive Manufacturing. In *Robotic Fabrication in Architecture, Art and Design 2018*; Springer International Publishing: Cham, Switzerland, 2019; pp. 85–97. [CrossRef]
16. Kong, L.; Peng, X.; Chen, Y. Fast and Accurate Defects Detection for Additive Manufactured Parts by Multispectrum and Machine Learning. *3D Print. Addit. Manuf.* **2023**, *10*, 393–405. [CrossRef] [PubMed]
17. Garfo, S.; Muktadir, M.A.; Yi, S. Defect Detection on 3D Print Products and in Concrete Structures Using Image Processing and Convolution Neural Network. *J. Mechatron. Robot.* **2020**, *4*, 74–84. [CrossRef]
18. Delli, U.; Chang, S. Automated Process Monitoring in 3D Printing Using Supervised Machine Learning. *Procedia Manuf.* **2018**, *26*, 865–870. [CrossRef]
19. Nguyen, V.H.; Remond, S.; Gallias, J.L. Influence of cement grouts composition on the rheological behaviour. *Cem. Concr. Res.* **2011**, *41*, 292–300. [CrossRef]
20. Macosko, C.W. *Rheology Principles, Measurements, and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 1994.
21. Güneyisi, E.; Gesoglu, M.; Naji, N.; Ipek, S. Evaluation of the rheological behavior of fresh self-compacting rubberized concrete by using the Herschel-Bulkley and modified Bingham models. *Arch. Civ. Mech. Eng.* **2016**, *16*, 9–19. [CrossRef]
22. Konan, N.A.; Rosenbaum, E.; Massoudi, M. On the Response of a Herschel–Bulkley Fluid Due to a Moving Plate. *Polymers* **2022**, *14*, 3890. [CrossRef]
23. Feys, D.; Verhoeven, R.; De Schutter, G. Evaluation of time independent rheological models applicable to fresh self-compacting concrete. *Appl. Rheol.* **2007**, *17*, 56244. [CrossRef]
24. OpenFOAM, The Open Source CFD Toolbox. Available online: https://www.openfoam.com/ (accessed on 11 April 2023).
25. OpenFOAM: User Guide: Overset. Available online: https://www.openfoam.com/documentation/guides/v2112/doc/guide-overset.html (accessed on 17 May 2023).
26. FeltusChristophe. Learning Algorithm Recommendation Framework for IS and CPS Security. *Int. J. Syst. Softw. Secur. Prot.* **2022**, *13*, 1–23. [CrossRef]
27. Stefas, A.; Rossi, A.; Tessmann, O. Funken Serial Protocol Toolkit for Interactive Prototyping. In Proceedings of the 36th eCAADe Conference, Łódź, Poland, 17–21 September 2018; Volume 2.
28. Sanfilippo, F.; Hatledal, L.I.; Zhang, H.; Fago, M.; Pettersen, K.Y. Controlling Kuka industrial robots. *IEEE Robot. Autom. Mag.* **2015**, *22*, 96–109. [CrossRef]