Forcael, E, Alucema, E, Burkart, A, García-Alvarado, R, Sepúlveda-Morales, J, & Martínez, E. (2024). Visual Programming Routines Used in 3D Concrete Printing Processes for Different Geometries. In Herrera, RF, Salazar, LA, (Editors), *Proceedings of the IX Ibero-American Congress of Construction Management and Technology* (IX ELAGEC2024).

VISUAL PROGRAMMING ROUTINES USED IN 3D CONCRETE PRINTING PROCESSES FOR DIFFERENT GEOMETRIES

Eric Forcael [1] - eric.forcael@uss.cl

Elena Alucema [2] - elena.alucema.c@gmail.com

Adolfo Burkart [2] - aburkartmedina@gmail.com

Rodrigo García-Alvarado [2] - rgarcia@ubiobio.cl

Javier Sepúlveda-Morales [2] - luisepul@egresados.ubiobio.cl

Eder Martínez [3] - eder.martinez@fhnw.ch

[1] *College of Engineering, Architecture, and Design, Universidad San Sebastián, Chile.*

[2] *College of Engineering, Universidad del Bío-Bío, Chile.*

[3] *College of Architecture, Civil Engineering and Geomatics, University of Applied Sciences and Arts Northwestern Switzerland, Switzerland.*

ABSTRACT

3D concrete printing has advantages over other techniques since it eliminates the need for formwork and allows the manufacture of complex designs. However, some challenges must be addressed, such as facilitating the computational design processes of the elements to be printed. Thus, this research focuses on designing programming routines used in 3D concrete printing processes for different geometries (sixteen shapes), measuring and performing analyses for the printing times of a robotic arm, which can be replicated in any 3D concrete printing process, regardless of their geometric complexity. Although some geometries may be printed faster than others, there were no significant differences between their printing times, which may provide designers, architects, engineers, and builders the freedom to build increasingly complex shapes without being constrained by their geometry.

KEYWORDS

Programming routines; 3D concrete printing; different geometries; robotic arm.

1. INTRODUCTION

In recent years, technological advancements have contributed to developing new additive manufacturing technologies, such as 3D concrete printing (Craveiro et al., 2019), which can reduce carbon dioxide emissions and decrease the raw materials required for construction (Adesina, 2020). Also, 3D concrete printing aims to reduce costs and construction times while providing more architectural and structural design freedom. Other advances include the implementation of communication protocols between BIM elements and 3D concrete printing (Forcael et al., 2021) and the development of sophisticated cyber-physical systems for the robot-controlled fabrication of concrete components (Vukorep et al., 2020).

Despite the advances in 3D concrete printing, several challenges still need to be addressed. One of those challenges concerns the accessible design of visual programming routines used in 3D concrete printing processes for different geometries. Therefore, the main objective of this research is to design visual programming routines for 3D concrete printing processes for different geometries, based on the following variables: the total height of the geometries analyzed, the height of each concrete extruded bead or layer, and the length of the 3D concrete printed bead, corresponding to the sum of perimeters of each geometry built from each concrete bead.

2. LITERATURE REVIEW

2.1. ADDITIVE MANUFACTURING IN THE CONSTRUCTION INDUSTRY

Additive manufacturing (AM) of concrete has been able to address some of the drawbacks of traditional methods for concrete construction, enabling the realization of new alternatives for design (Bos et al., 2016). Literature forecasts a favorable implementation of 3D printing in the year 2030, bringing a positive and transformative impact on Construction 4.0 as a particular application of Industry 4.0 (Forcael et al., 2020; Jiang et al., 2017), whose effects will be increased by the rapid development of affordable robotic systems (Gin et al., 2020).

AM, mainly 3D concrete printing, has garnered significant attention in the construction industry. The predominant method is extrusion-based 3D printing, wherein structural elements are meticulously crafted through the layer-by-layer deposition of printable concrete mixtures (Heidarnezhad & Zhang, 2022). 3D printing, combined with robotic arm handling concrete and specialized software, has produced precise geometric structures across various scales (Xiao et al., 2021). This approach offers several advantages over the conventional method of pouring concrete into the formwork, including cost and time efficiency, reduced environmental impact, and a notable decrease in accidents and fatalities on construction sites (Rollakanti & Prasad, 2022).

2.2. 3D CONCRETE PRINTING AND COMPLEX SHAPES

The 3D concrete printing process begins with creating a code that connects Building Information Modeling (BIM) elements and a robotic system. This connection facilitates the design of printing paths, ensuring the accurate 3D printing of the BIM-designed elements. Subsequently, the code is executed through a robotic arm to print the elements (Forcael et al., 2021), beginning with a 3D model in a CAD format that is later converted

to SLA format (Gardan, 2016). Specific software reads this format and cuts the part into "slices" to create a file containing the information for each layer.

On the other hand, the direct control of manufacturing tasks allows for adjusting design conditions and managing this process more efficiently due to the direct flow of information (Breseghello et al., 2021). This potential opens up greater possibilities in the construction industry to iterate more effective solutions between design and project construction, such as more efficient houses for different environments (Bazli et al., 2023) or more complex and optimized construction elements. Accordingly, developing visual programming routines used in 3D concrete printing processes for different geometries contributes to moving to more challenging 3D concrete printing applications.

Regarding printing complex shapes, several researchers in the field of 3D concrete printing seek to leverage parametric modeling to generate innovative and intricate designs (Ooms et al., 2021). These designs often encompass construction projects with complex geometries, such as domes featuring inclined layers, barrel vaults without external supports, and cantilevered structures with or without corbels (Carneau et al., 2020). Nevertheless, several practical challenges remain to materialize complex geometries efficiently (Prasittisopin et al., 2021). Thus, the present study considered developing unique shapes that allow experiencing new challenges in 3D-printed construction.

In addition, quantifying manufacturing precision is critical in defining process capability and quality control procedures (Xu et al., 2020). In this sense, a challenging ambit for 3D concrete processes is to deal with a large number of combinations of features (design, tool configurations, among others) to achieve a consistent and repeatable printing procedure (Xu et al., 2020). Thus, one of the main aspects to consider in a 3D concrete printing process is the programming of the routines needed to be sent to the robot that prints, which depends on a series of geometric variables.

3. METHODOLOGY

3.1. DESCRIPTION OF THE EQUIPMENT USED

The KUKA™ robot used for this research is the Quantec KR120 R2500 pro model shown in Figure 1, which runs some 3D concrete printing routines, according to the KUKA Software System Manual available at https://www.kuka.com/en-in/products/robotics-systems/software/system-software/kuka_systemsoftware.

Figure 1. Example of 3D concrete elements printed with the robot used in this research.

The way how the printing file visually programmed in this research is used by the robotic arm shown in Figure 1 is the following (Forcael et al., 2021): a) The KUKA robot is operated using a SmartPAD connected to the control unit, which, in turn, is connected to the robot via cables; b) A pendrive containing the printing file is inserted into the SmartPAD to send movement instructions for the robot to follow; c) Before running these instructions, the controller moves the robot to a starting point to begin the path; d) The movement speed can be controlled from the SmartPAD, and it is limited by the number of points the programmed element has on the tracking trajectory.

3.2. COORDINATE SYSTEM AND TYPES OF ROBOT MOVEMENTS

Four Cartesian coordinate systems are defined in the robot control unit: World, Robroot, Base, and Tool, as shown in Figure 2, where different movements can be programmed in a robot, such as 1) Specific movement of the axis: a PTP (Point to Point) type, where the robot moves the TCP (Tool Center Point) to the destination point along the fastest path, which is not necessarily the shortest path and therefore not a straight line, as shown in Figure 3a; 2) Trajectory movements: a LIN (linear) type, where the robot drives the TCP with a defined speed to the destination point along a straight line, as shown in Figure 3b, or of a CIRC (circular) type, where the robot drives the TCP to the destination point along the circular path (see Figure 3c); 3) SPLINE particular movement: a type of movement appropriate for complex curved trajectories, which can also be created with approximate PTP, LIN, and CIRC movements as shown in Figure 4. The robot control unit configures and executes the Spline block as a set of movements with approximate positioning for PTP, LIN, and CIRC (KUKA AG, 2023; KUKA Roboter GmbH, 2015).

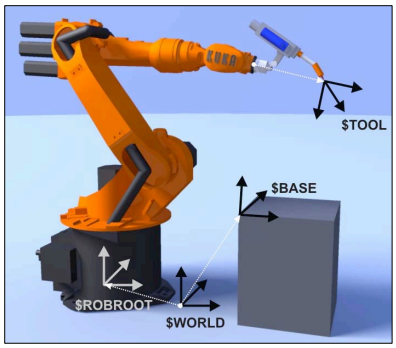


Figure 2. Four Cartesian coordinate systems in the robot (adapted from the KUKATM system software manual (KUKA AG, 2023)).

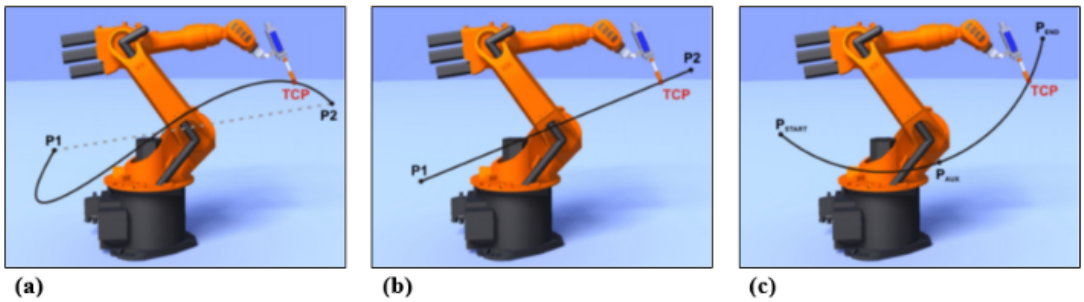


Figure 3. Movements (a) PTP, (b) LIN, and (c) CIRC (adapted from KUKATM system software manual (KUKA AG, 2023)).

Figure 4. Approximate PTP, LIN, and CIRC positionings (adapted from KUKATM system software manual (KUKA AG, 2023)).

3.3. VISUAL PROGRAMMING SOFTWARE FOR BIM

Visual programming tools such as Dynamo Studio (part of Autodesk software as Dynamo Sandbox and Dynamo Revit), Grasshopper, and others make it possible to generate computational designs accessible to all people, specialized or not, and allow automating tasks (Autodesk, 2021; Thabet et al., 2022). In this type of software, the elements are visually connected to define relationships and sequences of actions that make up custom algorithms used for various applications, from data processing to geometry generation.

On the other hand, KUKA|prc is a parametric control tool that makes robotics accessible to the creative industry (Braumann & Singline, 2021; Stumm et al., 2016). It is built on a core library defining specific classes and executes operations like collision verification, inverse and direct kinematics, and automated axes calculation. It works with applications such as Grasshopper and Rhinoceros (Braumann & Brell-cokcan, 2015).

4. PROFILES TO BE USED

In this research, sixteen different profiles were built using Dynamo Studio to obtain and compare the influence of printing time on the geometry of different shapes. Table 1 shows all the profiles modeled.

Table 1. Different shapes modeled.

Pyramids	Prisms	Random Geometry	Curved geometries	
			Curved Pyramids	Curved Prisms
Triangular	Triangular	Random 1	Triangular	Triangular
Square	Square	Random 2	Circular	Square
Hexagonal	Hexagonal	Random 3		
Circular	Circular	Random 4		

First, shapes are created as solid elements, which are later divided into layers, and from these, the measurements of each perimeter by section are obtained, where the sum of all the contours constitutes the length of the total bead to be printed (*Lc*). This dissociation process is the same for all geometries. Figure 5 shows the process of creating the solid element (a), which is later sectioned into layers (b) to obtain the perimeters (c).

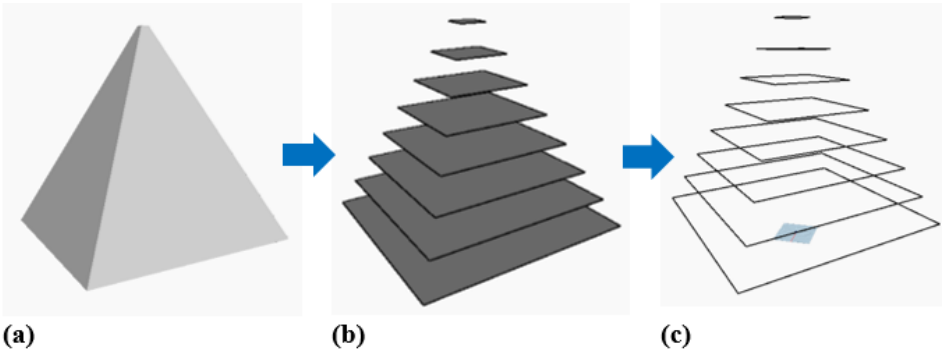


Figure 5. Model creation: (a) solid element, (b) layered element, and (c) extraction of perimeter curves.

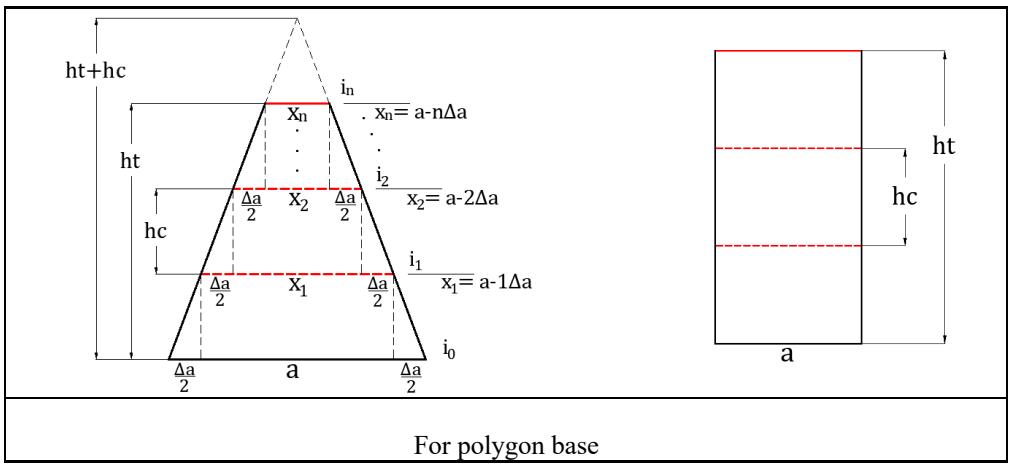
4.1. PARAMETERS TO USE IN THE MODELING

Figure 6 shows the different formulas used to model pyramids and prisms, where *Lc* is the sum of the perimeters of each layer. It is necessary to calculate the sum of the contours formed by the intersection of planes perpendicular to the base of the figure, which is equidistant from one another at a distance that is called *hc*, which corresponds to the height of the printing layer until reaching to the full height of the element called *ht*. These values are required to be entered to create the different shapes using equations 1 to 4.

Since the constructive elements used in building construction are perimeter printed, the geometries chosen for this research are hollow, where the objective is to develop printing routines for the shells that make up, layer by layer, the 3D concrete printing elements. It has to be noted that the thickness of the elements was considered constant since, during the printing procedures with robotic arms, the extruder nozzle was not changed. So then, what was intended to study was the behavior of the robot printing different trajectories of complex shapes, taking into account lines and curves, which finally allows for determining the printing time.

4.1.1. PYRAMIDS AND PRISMS

It has to be noted that the geometry shown in Figure 6, corresponding to the pyramid, is truncated at its apex, meaning there is a plane instead of a vertex. This condition occurs because it is impossible to print this point (physically, it is not feasible to print a point).



For polygon base

$$Lc = l * a\left[n - \frac{n(n+1)}{2}\left(\frac{hc}{ht+hc}\right)\right] \quad (1)$$

$$Lc = l \cdot a \cdot \frac{ht}{hc} \quad (2)$$

$$a = \frac{Lc}{l\left[n - \frac{n(n+1)}{2}\left(\frac{hc}{ht+hc}\right)\right]}$$

$$P = l \cdot a$$
$$a = \frac{Lc}{l \cdot n}$$

For circular base

$$Lc = 2\pi r\left[n - \frac{n(n+1)}{2}\left(\frac{hc}{ht+hc}\right)\right] \quad (3)$$

$$Lc = 2\pi r \cdot \frac{ht}{hc} \quad (4)$$

$$r = \frac{Lc}{2\pi\left[n - \frac{n(n+1)}{2}\left(\frac{hc}{ht+hc}\right)\right]}$$

$$P = 2\pi r$$
$$r = \frac{Lc}{2\pi \cdot n}$$

Figure 6. Geometry variables to print.

Where,

n: Total number of layers.

a: Length of the edge of the pyramid to be modeled.

x_1: Length of the basal edge of the pyramid to be modeled.

x_n: Length of the last edge of the pyramid to be modeled.

hc: Height of the layer.

ht: Total height of the element to be printed.

i: Number of a layer.

l: Number of sides of the polygon.

r: Radius.

P: Perimeter.

4.1.2. RANDOM GEOMETRY (DIFFERENCE OF SOLIDS)

A particular generated geometry comes from different solids whose perimeters are composed, as shown in Figure 7. Now, adding the lengths of the base of the composite figure, Equation 5 was used, where the length of the edge of the central square "*a*" is a function of *Lc*, *ht,* and *hc*.

$$a = \frac{Lc}{\left(\frac{\pi}{4} + 3\right) \cdot \frac{ht}{hc}} \quad (5)$$

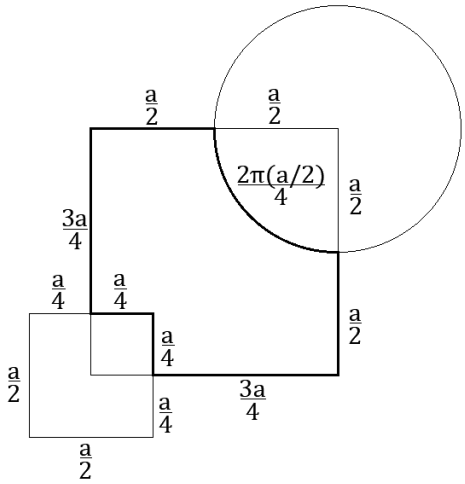


Figure 7. Configuration of a composed random geometry.

4.2. CODE FOR THE GENERIC STAGES

The developed code represents the digital model of the geometries. From this code, an SRC file is extracted, which contains the path of the extruder to be executed by the robot. Next, the code created for this research is detailed, which consists of five stages: 1) Geometry, 2) Trajectory curves, 3) Trajectory planes, 4) Coordinate displacement, and 5) Robot arm trajectory. It has to be noted that the developed code differs between figures only in stage one (Geometry). Thus, the process is explained for each profile, but its explanation is not repeated when stages share the same procedure.

4.2.1. STAGE 1 – GEOMETRY

The first step is the model design, where the result is the geometry as a solid, which varies for each geometry. Therefore, a different schematic view is needed for each of the sixteen geometries under study (due to space limitations, schematic views with the box diagrams and connectors were not included here but are available on request from the authors). The box diagrams and paths that come out from Stage 1 and then go to Stage 2 are shown in Figure 8, and from there, the boxes and paths of Stage 3 are shown in Figure 10, and so successively until reaching the final Stage 5 (Figure 14).

4.2.2. STAGE 2 – TRAJECTORY CURVES

Path curves are generated from the created geometry; the extruder printing path is defined from those path curves. These are developed by sectioning the geometry at a specific height, which depends on the extruder nozzle radius. This procedure translates into the height of each layer *hc*. A schematic view of this stage is shown in Figure 8.

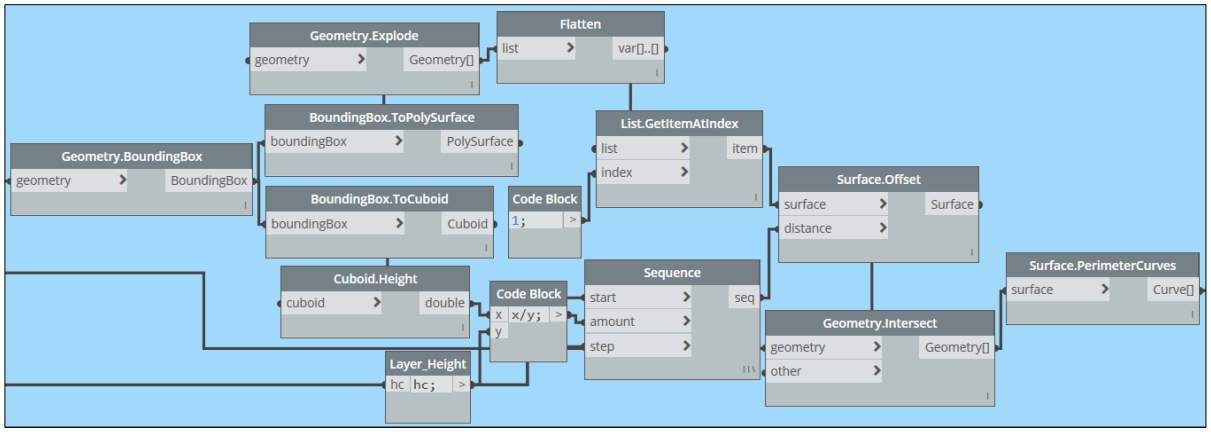


Figure 8. Schematic view of stage 2.

Next, the process is divided into substages, sequentially ordered as follows.

a) Bounding geometry through a box: The "BoundingBox" node makes a box bounded by the geometry it contains. This procedure is not shown at first, and to make it visible, the "BoundingBox.ToPolySurface" node is used, which allows the box to be presented as a collection of independent surfaces, as shown in Figure 9 (a).

b) Box decomposition into independent surfaces and selection of the base: The "Geometry.Explode" node enables the division of geometry into smaller components. The box faces that contain the pyramid are separated into independent surfaces. Then, the "Flatten" node allows delivery of the 1D list flattened from the input multidimensional list. Finally, the node "List.GetItemAtIndex" takes an input list and provides the list item corresponding to the specified input index, getting the bottom face of the box as shown in Figure 9 (b).

c) Box sectioning: A solid cuboid is created from a bounding box constructed by the "BoundingBox" node, and the total height is measured. This total height of the cuboid is then divided by the layer height to obtain the number of layers. The "Sequence" node builds a list of numbers, where the list starts with the number of the "start" entry and then grows by the "step" entry, while the "amount" entry determines the number of elements in the list. Finally, the "Surface.Offset" node allows offsetting a surface in its normal direction by the specified distance, as shown in Figure 9 (c).

d) Geometry division: Through the "Geometry.Intersect" node, it is possible to obtain the geometry of the intersection of two objects. In this case, the pyramid's geometry intersects with the surfaces resulting from sectioning the box. As a result, the pyramid's geometry is equally fractionated (see Figure 9 (d)).

e) Obtaining perimeter curves: Using the node "Surface.PerimeterCurves", the edge curves of a surface were created as a series of curves shown in Figure 9 (e). Thus, the trajectory curves originate from extracting the perimeter curves of the layers.

f) Visualization of trajectory curves: Finally, the visualization of the surfaces in each layer is deactivated. The substage results in the trajectory curves (Figure 9 (f)).

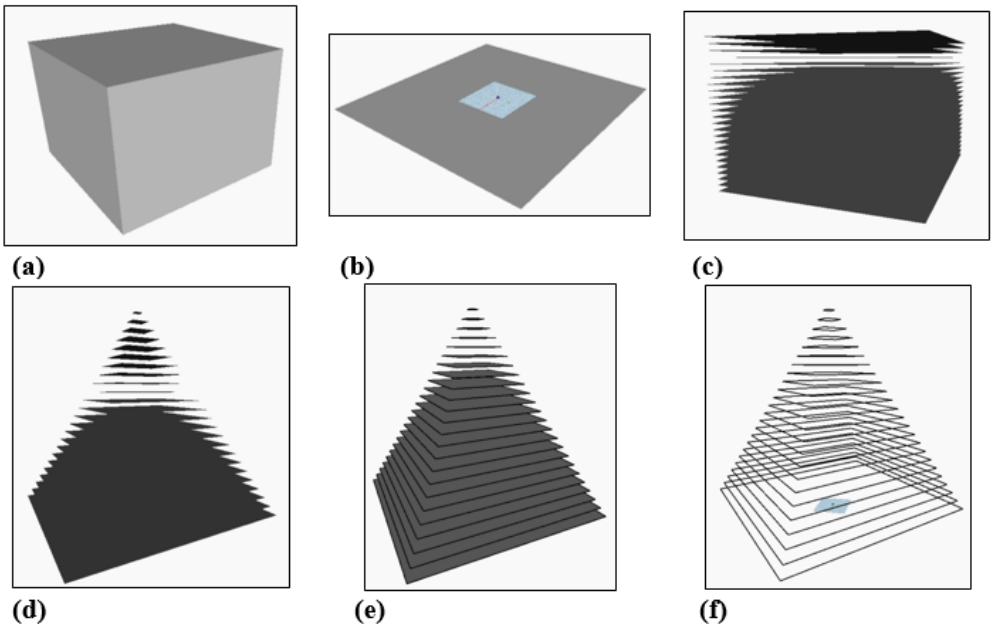


Figure 9. Steps to obtain the trajectory curves.

4.2.3. STAGE 3 – TRAJECTORY PLANES

The trajectory planes are generated from the trajectory curves and are schematically shown in Figure 10.

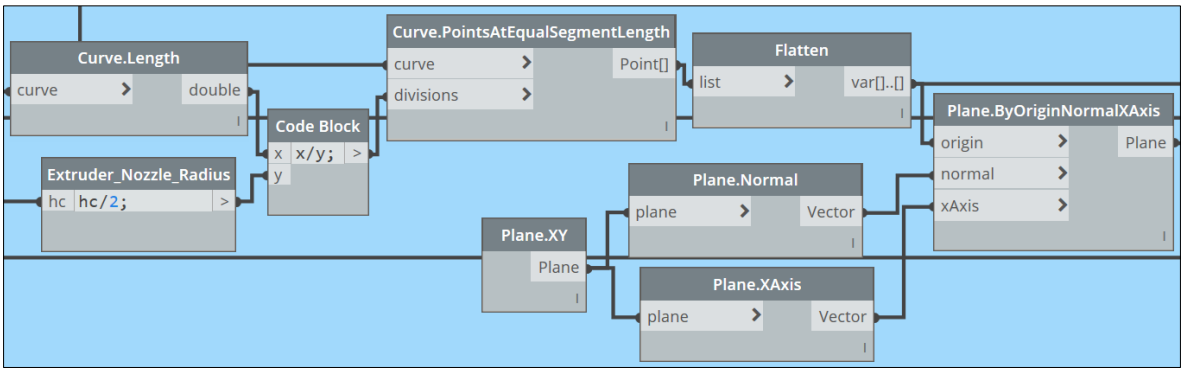


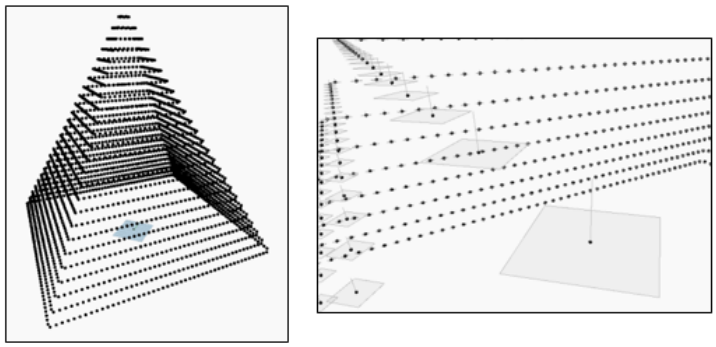
Figure 10. Schematic view of stage 3.

Subsequently, the process is then divided into two substages, which are ordered sequentially.

a) Generation of points on the curve: Through the "Curve.Length" node, the length of an input curve can be measured. Then, the length of the total curve of the geometry is divided by the radius of the extruder nozzle, which corresponds to the layer height divided in two. Then, the "Curve.PointsAtEqualSegmentLength" node returns a list of points along an input curve by dividing the curve into segments of equal length, as shown in Figure 11 (a).

b) Creation of plane with normal vector for each point: Using the "Flatten" node, the flattened 1D list of the multidimensional input list is obtained. Then, the "Plane.ByOriginNormalXAxis" node creates an "oriented" plane positioned at the origin of the normal vector point but with the specific orientation of the X-axis. In this

case, a normal vector plane was built for each generated point on the curve, as shown in Figure 11 (b).



(a) (b)

Figure 11. Result of substages (Trajectory planes)

4.2.4. STAGE 4 – TRAJECTORY PLANES

The created planes are derived from geometry and location independently of the robotic configuration. Thus, the path planes must be oriented toward the robot to position the extruder in the required location. Figure 12 shows a schematic view of this stage.

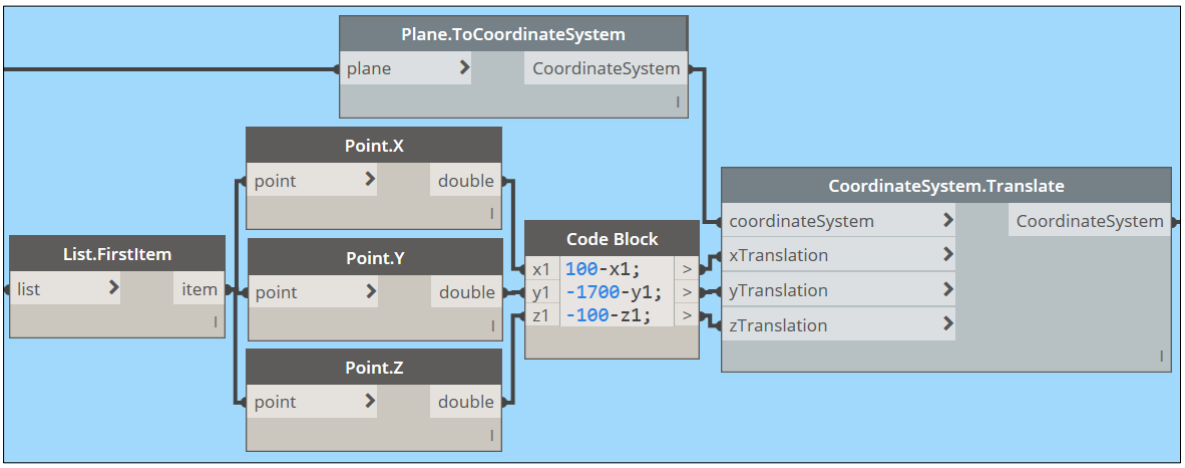


Figure 12. Visual programming of the displacement of coordinate planes.

Next, the process is divided into substages ordered sequentially for the coordinate displacement stage.

a) Creation of coordinate systems on each plane of points: The "Plane.ToCoordinateSystem" node allowed the creation of a coordinate system based on the input plane, using the plane origin (XAxis and YAxis) as shown in Figure 13 (a).

b) Coordinates of the first point of the geometry: the "List.FirstItem" node selects the first point of the point curve created in the previous stage (Figure 13 (b)), then the X, Y, and Z components are obtained separately to be able to be subtracted from the print space coordinates. This procedure will allow changing the parameters *Lc*, *ht,* and *hc* never to move the first point of the figure.

c) The coordinate system in print space: Using the "CoordinateSystem.Translate" node, an original coordinate system was transformed into a new coordinate system based on translation distances in the X, Y, and Z directions. This procedure intends to translate

all the geometry points concerning some point to the new coordinate system created, as shown in Figure 13 (c).

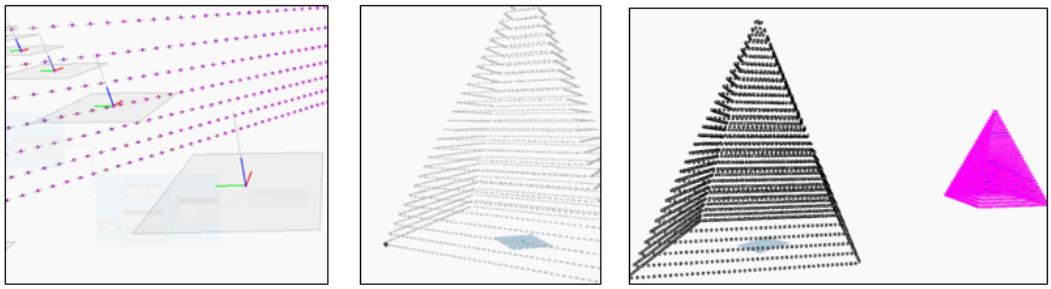


Figure 13. Result of the substages (Displacement of coordinate planes).

4.2.5. STAGE 5 – ROBOTIC ARM TRAJECTORY

The last section of the code corresponds to the development of the robot's path or the inverse kinematics (IK) so that the robot achieves the intended position, where KUKA|prc is used to solve the IK. The movement speed of the robot must be modified according to the flow speed of the material coming out from the extruder to obtain a stable and consistent print. Figure 14 shows a schematic view of this stage.

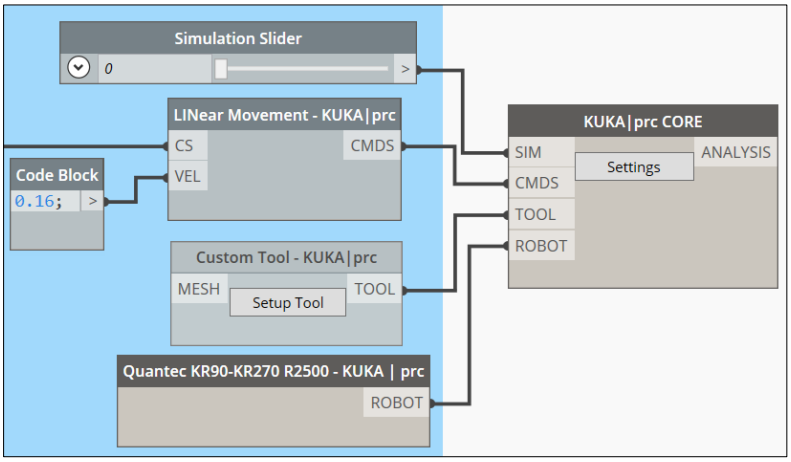


Figure 14. Stage 5: Robotic arm trajectory.

Next, the process is divided into sequentially ordered substages for the robot trajectory stage.

1. KUKA|prc engine: The "KUKA|prc CORE" oversees all the simulation and code generation. This procedure allows the visualization of the movement graph of the robot's six axes as a function of time.
2. Sliding control: The simulation could be controlled through its slider (by moving the slider, the robot moves through the program).
3. Linear movement: The robot allows multiple movements, the most important being LINear and PTP movements. LINear moves connect coordinate systems with a straight line (best when precision is needed). PTP moves to connect coordinate systems with the least amount of axis rotation (they are fast, but they do not follow a straight line). The speed value is an optimal value calculated so that there are no interruptions in the printing process.

4. Robot Tool: A personalized tool for the robot simulation was used with the data of the extruder coordinates with which the concrete is deposited. The data used were: X-axis: -193 mm, Z-axis: 255 mm, Y rotation: -90 degrees (setting used for all figures created).
5. KUKA™ robot model: The KUKA™ robotic arm model is selected. In this research, the model robot used was the Quantec KR90-KR270 R2500.

5. ANALYSIS OF RESULTS

This section is divided into two parts: 1) Time comparison between all geometries and 2) Analysis of simple geometries. First, the KUKA|prc software enabled the programming of the robotic arm in charge of the 3D concrete printing process, including a full kinematic simulation of the robot and considering the three variables under study (*ht*, *hc,* and *Lc*), where the printing process times for the sixteen studied geometries was recorded as shown in Table 2. Secondly, as a complementary analysis focusing solely on the variable *Lc* (2D analysis), the same procedure was conducted for simple geometries (triangular, square, hexagonal, and circular bases, i.e., no irregular geometries).

5.1. TIME COMPARISON BETWEEN ALL GEOMETRIES

For comparison, the time data of the sixteen geometries were arranged in Table 2, keeping the values of the variables constant (*Lc*=28800mm, *ht*=500mm, and *hc*=10mm), and the diameter of the nozzle is equal to *hc* and corresponds to the height of the printed layer. It has to be noted that despite the multiple sizes of nozzles available in the laboratory where the tests were conducted, the nozzle sizes used in this research were 10mm and 20mm. The results are graphically displayed in Figure 15.

Table 2. Printing times for each geometry.

Geometry	Printing times (s)
1. Triangular pyramid	178.84
2. Square pyramid	179.32
3. Hexagonal pyramid	180.26
4. Circular pyramid	182.63
5. Triangular prism	178.60
6. Square prism	180.10
7. Hexagonal prism	182.03
8. Circular prism	182.44
9. Curved triangular pyramid	178.96
10. Curved circular pyramid	182.64
11. Curved triangular prism	179.66
12. Curved square prism	180.18
13. Random 1	183.16
14. Random 2	181.75
15. Difference in solids	176.85
16. Polygon by points	178.96

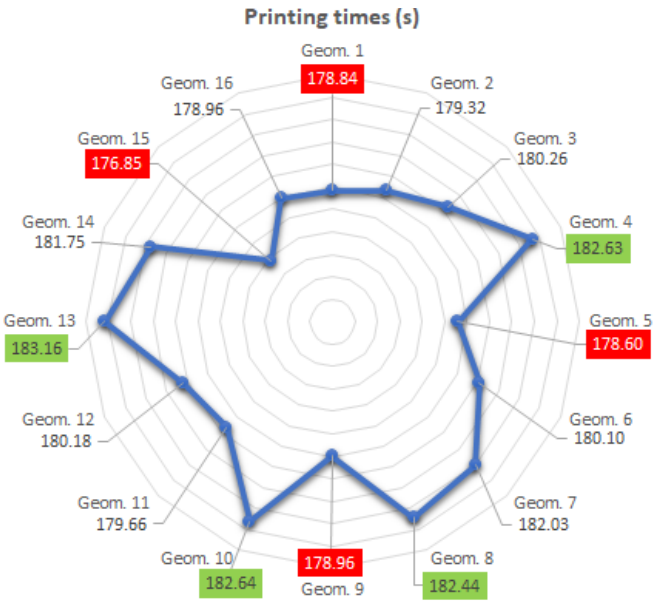


Figure 15. Chart with printing times for each geometry.

Based on the information shown in Table 2 and Figure 15, it can be observed that the geometries that take the longest to print are those in green (mainly with circular or curved bases; geometries 4, 8, 10, and 13) and the fastest are those in red (with more straight shapes or with a base with fewer edges; geometries 1, 5, 9 and 15). These results appear to be logical. That is, more complexity implies more time needed for printing. However, it is relevant to note that the differences between the printing times are not significant (3.57% between the highest and lowest values).

5.2. ANALYSIS OF SIMPLE GEOMETRIES

An additional analysis of simple geometries was performed to understand the time variations better. This analysis considered four bases: triangle, square, hexagon, and circumference, corresponding to the bases of prisms and considering the same values of perimeter and velocity. It is crucial to indicate that, to evaluate only the effect of the length of the extruded bead (*Lc*), the other two variables (*ht* and *hc*) were ignored in the estimation of the times so that the analysis carried out corresponds then to an analysis in two dimensions (2D analysis), where the point-to-point analysis was much more detailed and accurate as shown below.

To explain this analysis, the triangular prism geometry will be used as an example, where Figure 16 shows the different distances between points that make up the selected geometry. The distance d_1 (red lines) is the distance between points, which is the same for all the geometries since it depends on the diameter of the robot extruder. The distance d_2 (green lines) is the one that is generated in the vertices, and it is produced because the robot, when printing, goes through the points of the geometry without including the vertices; consequently, the robotic arm approximates this distance by joining the two equidistant points of the vertex of adjacent edges (for this reason d_2 depends on the angle of the vertex). Finally, d_3 (blue lines) is the distance generated when switching to the next layer, where this value depends on *hc* (layer height) and the distance d_2. Thus, what was done was to analyze only the first layer of a 3D printed figure (for this reason, the distance d_3 is considered) to understand the variation in the printing time of the complete figure.

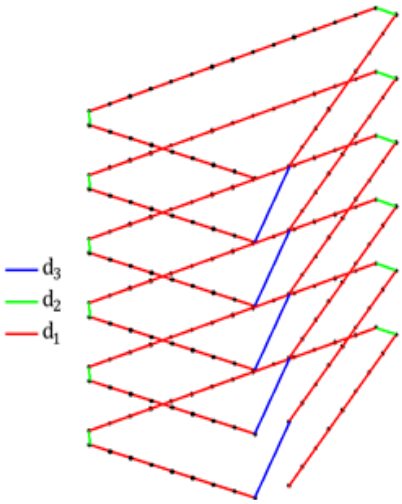


Figure 16. Point-to-point simulation for the triangular prism geometry.

Thus, this procedure for the triangular prism geometry is expanded to the other three regular bases considered in this comparative analysis (square, hexagon, and circumference). However, for comparison purposes, only the times to print the bases of the chosen geometries were considered, as shown in Figure 17.

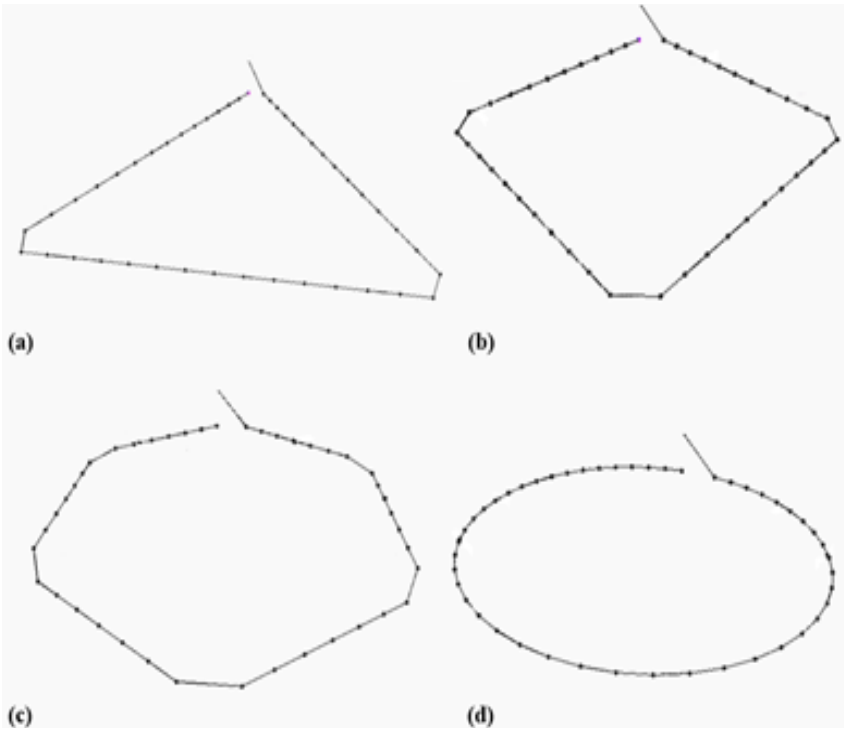


Figure 17. Bases considered in the comparative analysis: a) triangle, b) square, c) hexagon, and d) circumference.

The results are consolidated in Table 3, showing the following information: the vertices of the first layer for each prism, the number of points per layer, the spaces between them, the distances d_1, d_2, and d_3, the actual *Lc* per layer (which corresponds to the perimeter of the first layer plus d_3, that is, the "jump" to the next layer is included), and finally the printing time delivered by the simulation.

Table 3. Printing times for simple geometries.

Base	Vertices	Points per layer	Spaces between points	d_1	d_2	d_3	*Lc* real per layer	Printing Time (s)
(a)	3	45	48	12.5	12.5	27.95	1176.79	7.42
(b)	4	44	48	12.5	17.68	30.62	1182.38	7.45
(c)	6	42	48	12.5	21.65	33.07	1185.38	7.47
(d)	0	48	48	12.5	0	35.34	1208.66	7.62

Table 3 shows that, although the number of points varies for each base, the number of spaces remains constant since, from a calculation perspective, they have the same basal perimeter. It can also be seen that d_2 increases its value as the base polygon has more edges (for the circumference, this value is zero since it has no vertices). Therefore, the greater the number of edges in a regular polygon, the greater its interior angles, creating a cut at the vertices with a larger length. All this affects the calculation of *Lc* because, although strictly speaking, all the geometries should have the same perimeter, these slight variations in measurement generate differences in terms of the perimeter, which also impact printing times delivered by the simulation. However, as previously observed in the analysis of the sixteen geometries, the differences were not significant here either (barely milliseconds).

The two analyses performed (first for all the geometries and then for the simple geometries analyzed in detail for a specific layer) show that the shape of the geometries does not significantly influence the printing time. Furthermore, the second analysis (which did not consider the variables *hc* and *ht*) highlights the relevance of the variable *Lc*. Eventually, *Lc* does not depend on other factors, such as the number of geometry vertices, the different distances considered (d_1, d_2, and d_3), or the number of printing points followed by the robotic arm. Thus, despite the previous analyses showing the role that the variables *hc* and *ht* play in the visual programming of different geometries — along with other parameters such as the distance between printing points or the number of vertices of a geometry, the most relevant variable is the length of the extruded bead *Lc* in the 3D concrete printing process.

6. DISCUSSION

First, it is relevant to emphasize that advanced visual programming algorithms are becoming increasingly important in solving challenging designing problems that involve large data sets or complex shapes to be built (Forcael et al., 2021). On the other hand, traditional algorithms may not handle such problems efficiently, which is where visual programming routines come into play.

As part of this research, the computing routines proposed considered three main variables: the total height of the geometries (*ht*), the height of each concrete extruded bead or layer (*hc*), and the length of the 3D concrete printed bead (*Lc*). Although all the variables participate in the programming methodology necessary for the 3D concrete printing process, the only variable that influences the printing time is *Lc* since the total height of the geometry (*ht*) is known in advance, at the same as the height of the extruded bead (*hc*). However, in the case of the height of the extruded bead (*hc*), although it is

technically possible to manufacture a nozzle that can modify its opening to change the printing height for the same element consequently, there are no benefits associated with this modification, at least, again, for the exact printed component (a wall for example).

Despite the previous findings, regarding limitations and future research, it must be noted that singularities commonly found in construction elements (electric boxes, windows, and others) were not considered in the present study and could be an essential way for future research. In this sense, developing easy-to-use programming tools that are more flexible and adaptable to different types of geometries printed in construction projects, such as the methodology developed here, will require considering shortly other complexities and challenges present in 3D concrete printing processes.

7. CONCLUSIONS

This research focused on developing visual programming routines used in 3D concrete printing processes for different geometries. A sample of sixteen geometries was analyzed along with the process variables (extruded bead length, perimeters of each layer, total height, and layer height). The procedure allowed the creation of visual computational routines that can be replicated in any 3D concrete printing process, expanding the frontiers of printed construction to all construction projects, regardless of their geometric complexity.

On the other hand, although the printing speed depends on the number of axis changes (XYZ) in diverse geometries, the differences found were not significant, giving designers, architects, engineers, and builders the freedom to build increasingly complex shapes without being constrained by their geometry.

Finally, this study's limitations are related to not considering other variables that are also linked to geometric aspects, such as the existence of singularities in the geometry (electric boxes in a wall, for example) or the need for reinforcing steel in printed elements for seismic zones, which can also influence the printing process and should be part of future research.

REFERENCES

Adesina, A. (2020). Recent advances in the concrete industry to reduce its carbon dioxide emissions. *Environmental Challenges*, *1*, 100004. https://doi.org/10.1016/j.envc.2020.100004

Autodesk. (2021). *Dynamo Developer*. Dynamo. https://dynamobim.org/#developer

Bazli, M., Ashrafi, H., Rajabipour, A., & Kutay, C. (2023). 3D printing for remote housing: Benefits and challenges. *Automation in Construction*, *148*, 104772. https://doi.org/10.1016/j.autcon.2023.104772

Bos, F., Wolfs, R., Ahmed, Z., & Salet, T. (2016). Additive manufacturing of concrete in construction: potentials and challenges of 3D concrete printing. *Virtual and Physical Prototyping*, *11*(3), 209–225. https://doi.org/10.1080/17452759.2016.1209867

Braumann, J., & Brell-cokcan, S. (2015). Adaptive Robot Control - New Parametric Workflows Directly from Design to KUKA Robots. *Proceedings of the 33rd ECAADe Conference*, *2*, 243–250.

Braumann, J., & Singline, K. (2021). Towards Real-Time Interaction with Industrial Robots in the Creative Industries. *2021 IEEE International Conference on*

Robotics and Automation (ICRA), 9453–9459. https://doi.org/10.1109/ICRA48506.2021.9561024

Breseghello, L., Sanin, S., & Naboni, R. (2021). Toolpath Simulation, Design and Manipulation in Robotic 3D Concrete Printing. *The 26th Annual Conference of the Association for Computer-Aided Architectural Design Research in Asia, CAADRIA 2021*, 623–632. https://doi.org/10.52842/conf.caadria.2021.1.623

Carneau, P., Mesnil, R., Roussel, N., & Baverel, O. (2020). Additive manufacturing of cantilever - From masonry to concrete 3D printing. *Automation in Construction*, *116*, 103184. https://doi.org/10.1016/j.autcon.2020.103184

Craveiro, F., Duarte, J. P., Bartolo, H., & Bartolo, P. J. (2019). Additive manufacturing as an enabling technology for digital construction: A perspective on Construction 4.0. *Automation in Construction*, *103*, 251–267. https://doi.org/10.1016/j.autcon.2019.03.011

Forcael, E., Ferrari, I., Opazo-Vega, A., & Pulido-Arcas, J. A. (2020). Construction 4.0: A Literature Review. *Sustainability*, *12*(22), 9755. https://doi.org/10.3390/su12229755

Forcael, E., Pérez, J., Vásquez, Á., García-Alvarado, R., Orozco, F., & Sepúlveda, J. (2021). Development of communication protocols between bim elements and 3D concrete printing. *Applied Sciences (Switzerland)*, *11*(16). https://doi.org/10.3390/app11167226

Gardan, J. (2016). Additive manufacturing technologies: State of the art and trends. *International Journal of Production Research*, *54*(10), 3118–3132. https://doi.org/10.1080/00207543.2015.1115909

Gin, Y., Saner, B. B., & Ramage, M. H. (2020). Robotic 3D printing with earthen materials as a novel sustainable construction method. *Proceedings of IASS Annual Symposia, IASS 2020/21 Surrey Symposium: Advanced Manufacturing Techniques*, 1–10.

Heidarnezhad, F., & Zhang, Q. (2022). Shotcrete based 3D concrete printing: State of art, challenges, and opportunities. *Construction and Building Materials*, *323*, 126545. https://doi.org/10.1016/j.conbuildmat.2022.126545

Jiang, R., Kleer, R., & Piller, F. T. (2017). Predicting the future of additive manufacturing: A Delphi study on economic and societal implications of 3D printing for 2030. *Technological Forecasting and Social Change*, *117*, 84–97. https://doi.org/10.1016/j.techfore.2017.01.006

KUKA AG. (2023). *KUKA System Software 8.6 Manual*. https://www.kuka.com/en-de/products/robot-systems/software/system-software/kuka_systemsoftware

KUKA Roboter GmbH. (2015). *KUKA System Software 8.3*.

Ooms, T., Vantyghem, G., Van Coile, R., & De Corte, W. (2021). A parametric modelling strategy for the numerical simulation of 3D concrete printing with complex geometries. *Additive Manufacturing*, *38*, 101743. https://doi.org/10.1016/j.addma.2020.101743

Prasittisopin, L., Sakdanaraseth, T., & Horayangkura, V. (2021). Design and Construction Method of a 3D Concrete Printing Self-Supporting Curvilinear Pavilion. *Journal of Architectural Engineering*, *27*(3), 05021006. https://doi.org/10.1061/(ASCE)AE.1943-5568.0000485

Rollakanti, C. R., & Prasad, C. V. S. R. (2022). Applications, performance, challenges and current progress of 3D concrete printing technologies as the future of sustainable construction – A state of the art review. *Materials Today: Proceedings*, *65*, 995–1000. https://doi.org/10.1016/j.matpr.2022.03.619

Stumm, S., Braumann, J., & Brell-Cokcan, S. (2016). Human-Machine Interaction for

Intuitive Programming of Assembly Tasks in Construction. *Procedia CIRP*, *44*, 269–274. https://doi.org/10.1016/j.procir.2016.02.108

Thabet, W., Lucas, J., & Srinivasan, S. (2022). Linking life cycle BIM data to a facility management system using Revit Dynamo. *Organization, Technology and Management in Construction: An International Journal*, *14*(1), 2539–2558. https://doi.org/10.2478/otmcj-2022-0001

Vukorep, I., Zimmermann, G., & Sablotny, T. (2020). Robot-Controlled Fabrication of Sprayed Concrete Elements as a Cyber-Physical-System. In *Second RILEM International Conference on Concrete and Digital Fabrication* (pp. 967–977). https://doi.org/10.1007/978-3-030-49916-7_94

Xiao, J., Ji, G., Zhang, Y., Ma, G., Mechtcherine, V., Pan, J., Wang, L., Ding, T., Duan, Z., & Du, S. (2021). Large-scale 3D printing concrete technology: Current status and future opportunities. *Cement and Concrete Composites*, *122*, 104115. https://doi.org/10.1016/j.cemconcomp.2021.104115

Xu, J., Buswell, R. A., Kinnell, P., Biro, I., Hodgson, J., Konstantinidis, N., & Ding, L. (2020). Inspecting manufacturing precision of 3D printed concrete parts based on geometric dimensioning and tolerancing. *Automation in Construction*, *117*(June), 103233. https://doi.org/10.1016/j.autcon.2020.103233