www.ricuc.cl
LinkedIn Revista Ingeniería de Construcción

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

Comparative study of visual programming procedures for 3d concrete printing of different geometric shapes

Estudio comparativo de procedimientos de programación visual para la impresión 3D de hormigón de diferentes formas geométricas

Forcael, Eric*[1]; Alucema, Elena **; Burkart, Adolfo **; García-Alvarado, Rodrigo **; Sepúlveda-Morales, Javier **; Martínez, Eder ***

* Facultad de Ingeniería, Arquitectura y Diseño, Universidad San Sebastián, Santiago, Chile.
** College of Engineering, Universidad del Bío-Bío, Concepción, Chile.
*** School of Architecture, Civil Engineering and Geomatics, University of Applied Sciences and Arts Northwestern Switzerland (FHNW), Muttenz, Switzerland.

Fecha de Recepción: 14/10/2024
Fecha de Aceptación: 15/11/2024
Fecha de Publicación: 09/12/2024
PAG: 1-25

Abstract

3D concrete printing offers significant advantages over traditional construction methods, including eliminating formwork and facilitating the production of complex designs. However, challenges remain in optimizing the computational design processes for printing different geometries. This research presents a comparative analysis of visual programming routines used in 3D concrete printing for sixteen different geometries, focusing on the printing times of a robotic arm. The results show that although some geometries are printed faster than others, the differences in printing times are not significant. These findings suggest that architects, engineers, and constructors can confidently explore increasingly complex shapes without being constrained by geometric complexity, thereby expanding the creative potential of 3D concrete printing.

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

Resumen

La impresión de hormigón en 3D ofrece importantes ventajas sobre los métodos de construcción tradicionales, incluida la eliminación del encofrado y la facilitación de la producción de diseños complejos. Sin embargo, persisten desafíos a la hora de optimizar los procesos de diseño computacional para imprimir diferentes geometrías. Esta investigación presenta un análisis comparativo de rutinas de programación visual utilizadas en la impresión 3D de hormigón para dieciséis geometrías diferentes, centrándose en los tiempos de impresión de un brazo robótico. Los resultados muestran que, aunque algunas geometrías se imprimen más rápido que otras, las diferencias en los tiempos de impresión no son significativas. Estos hallazgos sugieren que los arquitectos, ingenieros y constructores pueden explorar con confianza formas cada vez más complejas, sin verse limitados por la complejidad geométrica, ampliando así el potencial creativo de la impresión de hormigón en 3D.

Palabras clave: Rutinas de programación; Impresión de hormigón en 3D; diferentes geometrías; brazo robótico.

Corresponding author: eric.forcael@uss.cl
Facultad de Ingeniería, Arquitectura y Diseño, Universidad San Sebastián, Santiago, Chile

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

1. Introduction

Recent technological advancements have led to new additive manufacturing technologies, such as 3D concrete printing (Craveiro et al., 2019), which can reduce carbon dioxide emissions and construction raw materials (Adesina, 2020). Specifically, the construction industry benefits from increasingly more sustainable processes, like 3D concrete printing, which eliminates formworks as waste. Along with automating the traditional construction process, 3D concrete printing aims to reduce costs and construction times while providing more architectural and structural design flexibility. In addition to the digital-construction advantages of 3D concrete printing, several efforts have been made to create a more sustainable and safer environment by developing a series of green solutions (Blanco et al., 2020); (Chen et al., 2022); (Dey et al., 2022); (Zou et al., 2021) and new systems to reach high precision and stable printing processes (Anton et al., 2021); (Tho and Thinh, 2021). Other advancements include communication protocols between BIM elements and 3D concrete printing (Forcael et al., 2021) and cyber-physical systems for robot-controlled fabrication of concrete components (Vukorep et al., 2020).

Despite advancements in 3D concrete printing, the accessible design of visual programming routines for various geometries remains challenging. Therefore, the goal of this research is to create 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 (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 addressed the drawbacks of traditional methods, enabling new design alternatives (Bos et al., 2016). It is anticipated that 3D printing will positively impact Construction 4.0 by 2030 (Forcael et al., 2020); (Jiang et al., 2017), amplified by the development of affordable robotic systems (Gin et al., 2020).

AM, predominantly 3D concrete printing, has gained attention in the construction industry. The primary method involves extrusion-based 3D printing, where structural elements are meticulously created through layer-by-layer deposition of printable concrete mixtures (Heidarnezhad and Zhang, 2022). 3D printing, combined with robotic arm handling concrete and specialized software, has produced accurate geometric structures across various scales (Xiao et al., 2021). This approach provides several advantages over traditional concreting methods, including cost and time efficiency, reduced environmental impact, and decreased accidents and fatalities on construction sites (Rollakanti and Prasad, 2022).

3D-printed construction also involves another great opportunity and challenge for building and infrastructure, along with the linkage of design, management, and project construction (Khajavi et al., 2021). In this sense, the construction industry is a sequential process, where the new digital technologies allow information to be related and additive construction to generate a continuous workflow. This relationship has great potential because it allows for an interactive optimization process, from concept to execution and vice versa (Khamis et al., 2022).

2.2 3D concrete printing and complex geometries

The 3D concrete printing process starts with creating a code that links Building Information Modeling (BIM) elements and a robotic system. This connection helps design paths for accurate 3D printing of the BIM-designed elements. The code is then 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.

Direct control of manufacturing activities 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 opportunities in the construction industry to develop effective solutions between design and project construction, such as more efficient houses for different environments (Bazli et al., 2023) or more complex construction elements. Thus, developing visual programming routines used in 3D concrete printing processes for different geometries contributes to more challenging 3D concrete printings.

Researchers in 3D concrete printing use parametric modeling to create complex designs for construction projects (Ooms et al., 2021). These designs often encompass complex construction geometries, such as domes, barrel vaults without external supports, and cantilevered structures

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

(Carneau et al., 2020). However, various practical challenges remain to materialize complex geometries efficiently (Prasittisopin et al., 2021). Thus, this study focuses on developing unique shapes to explore new challenges in 3D-printed construction.

Accurately measuring manufacturing precision is crucial to establishing process capability and quality control procedures (Xu et al., 2020). In 3D concrete printing, managing various feature combinations (e.g., design and tool configurations) is challenging to ensure consistent and replicable printing procedures (Xu et al., 2020). Therefore, a relevant aspect to consider in 3D concrete printing is the programming of the routines to be sent to the printing robot, which depends on a series of geometric variables.

3. Methodology

3.1 Description of the equipment used

The KUKATM robot used for this research is the Quantec KR120 R2500 pro model shown in (Figure 1). This equipment was selected for its sizeable geometric range of operation (2.5 meters of radius), load capacity (120 kilograms, which can easily support a concrete extrusion hose), the possibility of installation on a rigid, transferable base, and its versatility control on a computer with proprietary software or files in G code, carried out by other software such as KUKA|prc (Parametric Robot Control) that allows programming, simulating, and controlling the KUKA robotic arm from within the Grasshopper software. According to the KUKA Software System Manual, the robotic arm runs some 3D concrete printing routines.

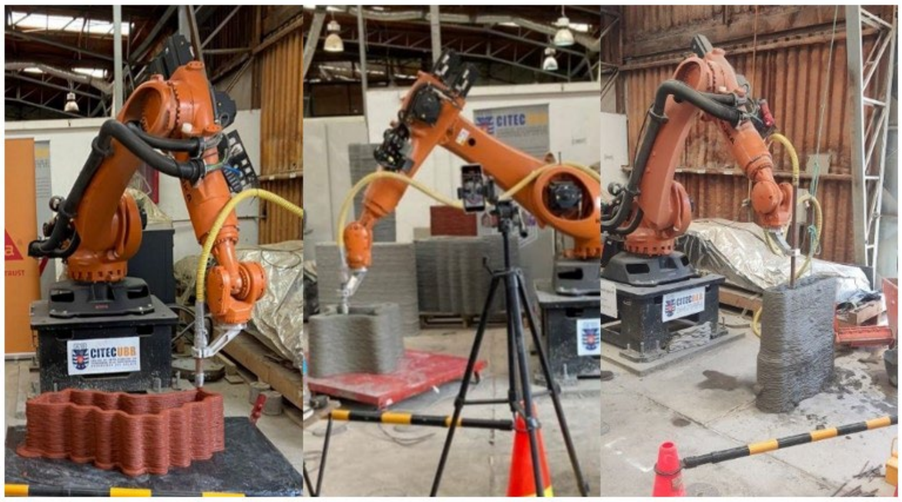


Figure 1. Example of 3D concrete elements printed with the robot used in this research.

Following is how the visually programmed printing file was used by the robotic arm shown in (Figure 1) (Forcael et al., 2021): a) The KUKA robot is operated using a SmartPAD connected to the control unit via cables; b) A pendrive containing the printing file is inserted into the SmartPAD to send movement instructions for the robot; c) Before running these instructions, the controller moves the robot to a starting point to begin the path; d) The movement speed is controlled from the SmartPAD.

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).

www.ricuc.cl
LinkedIn Revista Ingeniería de Construcción

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

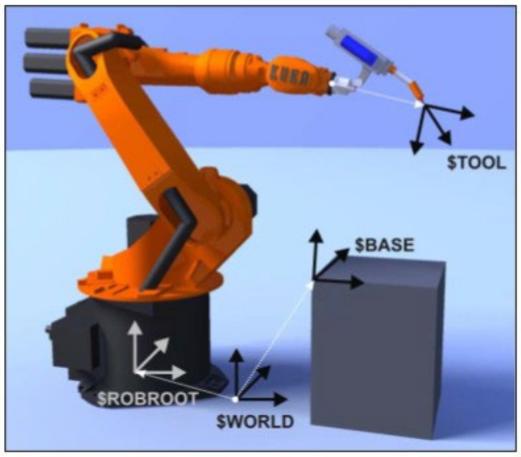


Figure 2. Four Cartesian coordinate systems in the robot (adapted from the KUKATM system software manual (KUKA AG, 2023)).

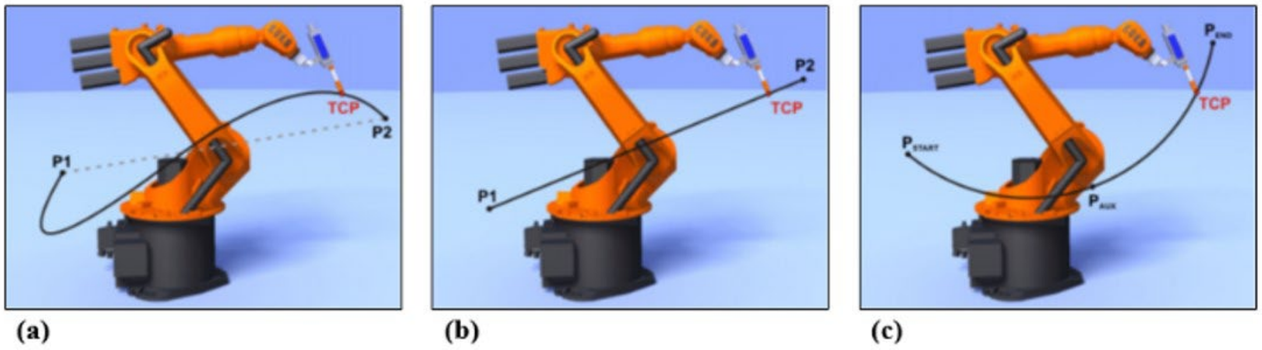


Figure 3. Movements (a) PTP, (b) LIN, and (c) CIRC (adapted from KUKATM system software manual (KUKA AG, 2023)).

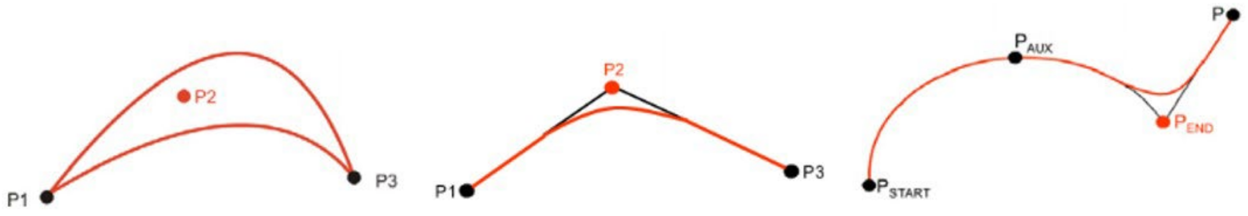


Figure 4. Approximate PTP, LIN, and CIRC positionings (adapted from KUKATM system software manual (KUKA AG, 2023)).

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 allow the creation of accessible computational designs and enable 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 create custom algorithms used for diverse applications, from data processing to geometry generation.

KUKA|prc is a parametric control tool that makes robotics accessible to the creative industry (Braumann and 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 and Brell-cokcan, 2015).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

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 diverse 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. Then, the measurements of each perimeter by section are obtained. This process is the same for all geometries. (Figure 5) shows the solid element (a), which is later sectioned into layers (b) to obtain the perimeters (c).

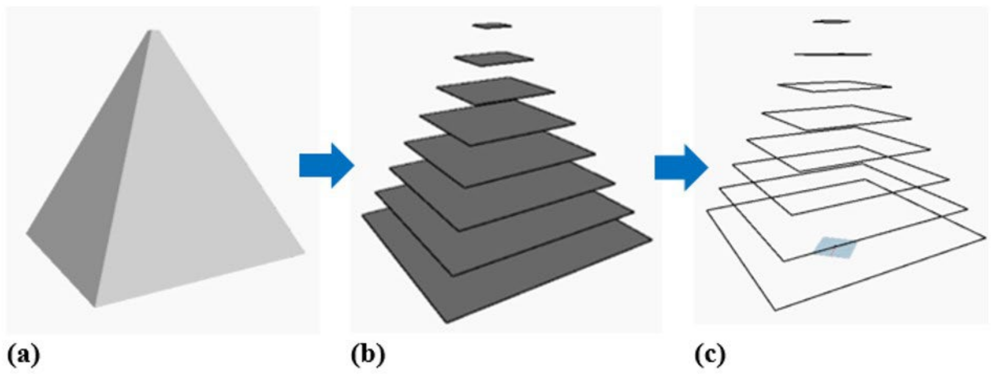


Figure 5. Model creation: (a) solid element, (b) layered element, and (c) extraction of perimeter curves.

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 are 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 entered to create the different shapes using equations 1 to 4, (Figure 6)

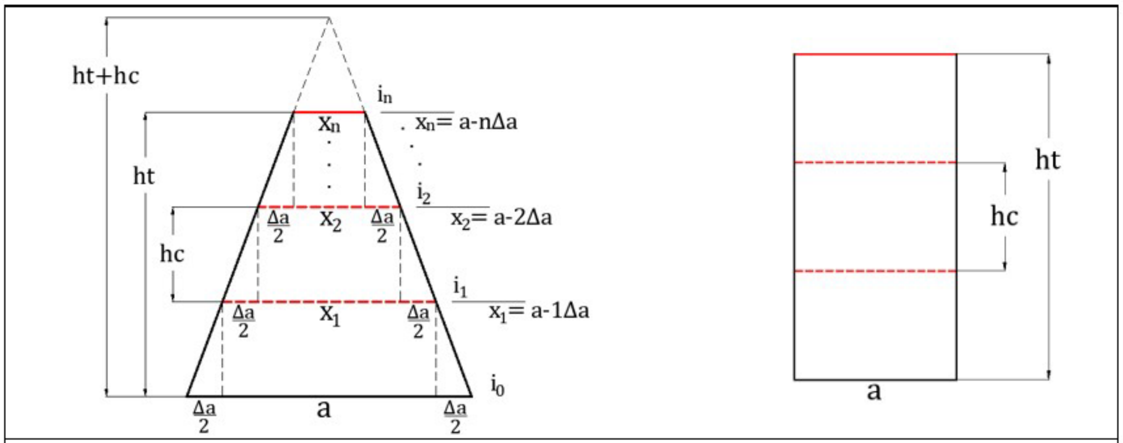
Since the constructive elements used in construction are perimeter printed, the geometries chosen for this research are hollow, where the objective is to develop printing routines for the shells that make up, layer by layer. The thickness of the elements was considered constant since, during the printing procedures with robotic arms, the extruder nozzle was not changed. Therefore, what was intended to be studied was the robot's behavior in printing different trajectories of complex shapes, considering lines and curves, allowing for determining the printing time.

4.1.1 Pyramids and prisms

(Figure 6) shows a pyramid 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).

ISSN: 0718-5073

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21



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)

Any 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*.

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

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

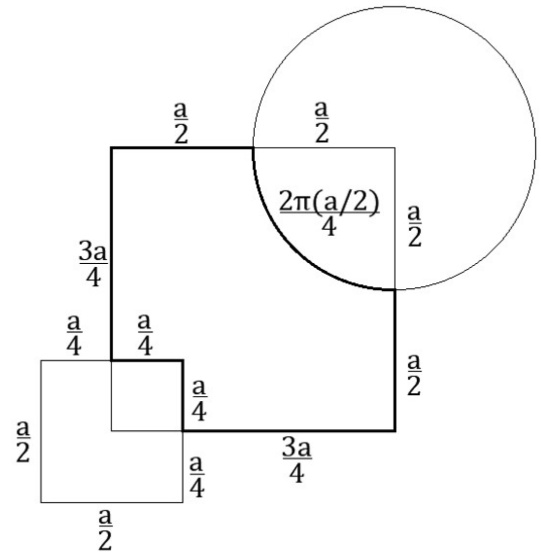


Figure 7. Configuration of a composed random geometry.

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. This 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, resulting in a shape as a solid, which varies for each geometry. Therefore, a different schematic view is needed for each of the sixteen geometries under study (all the 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).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

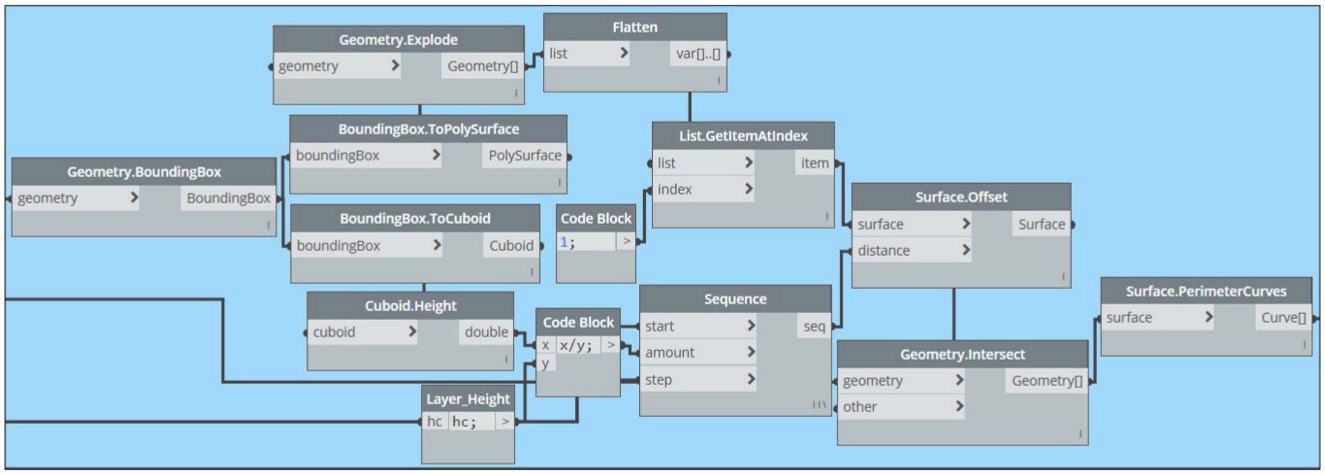


Figure 8. Schematic view of stage 2.

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. The pyramid’s geometry intersects with the surfaces, resulting from sectioning the box. Consequently, 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).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

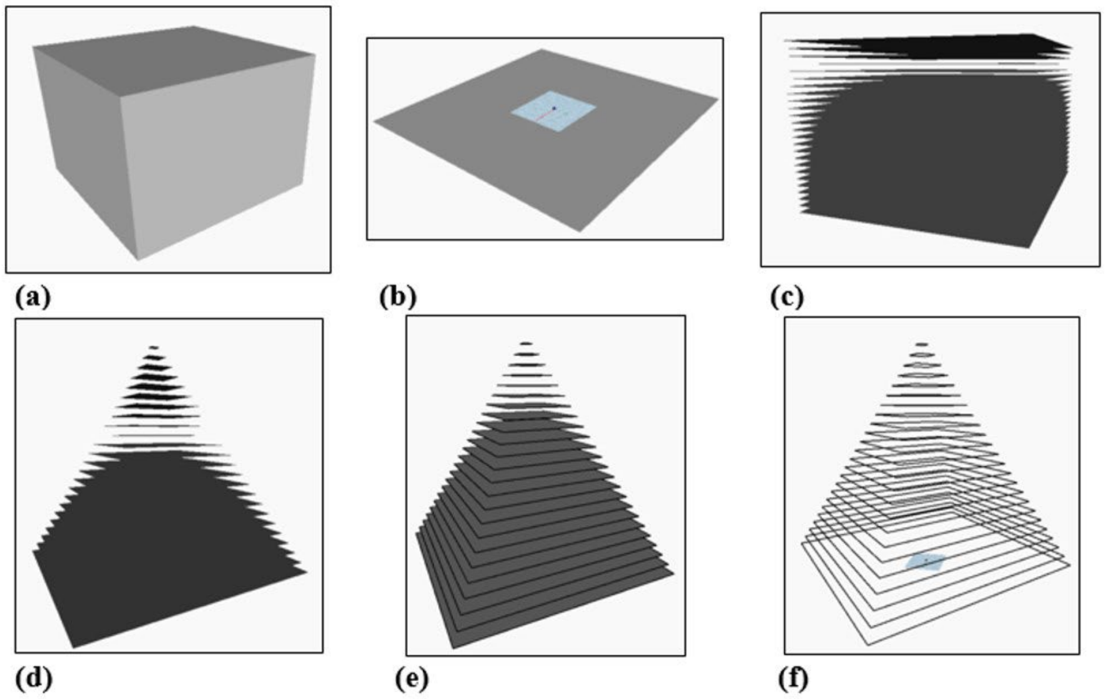


Figure 9. Steps to obtain the trajectory curves.

4.2.3 Stage 3 – trajectory planes

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

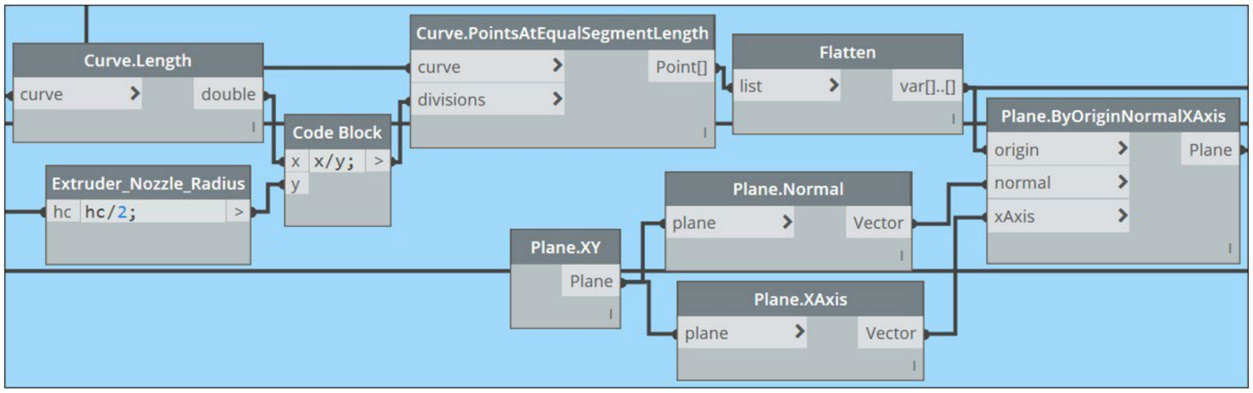


Figure 10. Schematic view of stage 3.

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 11a).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

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

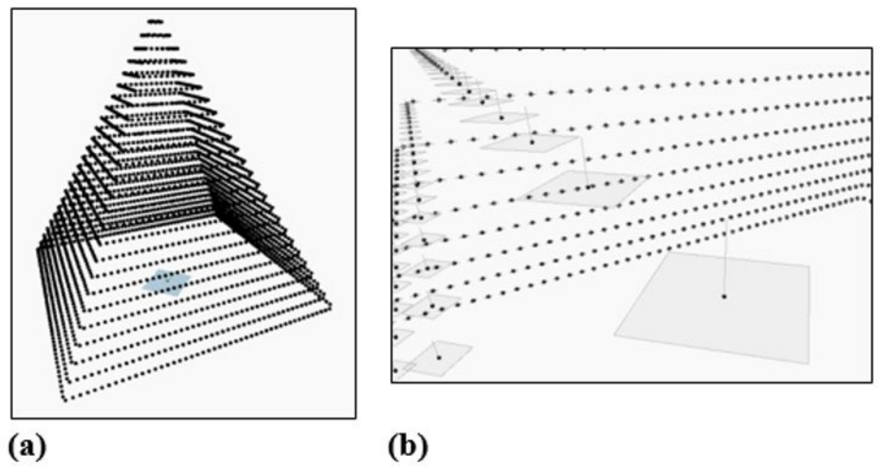
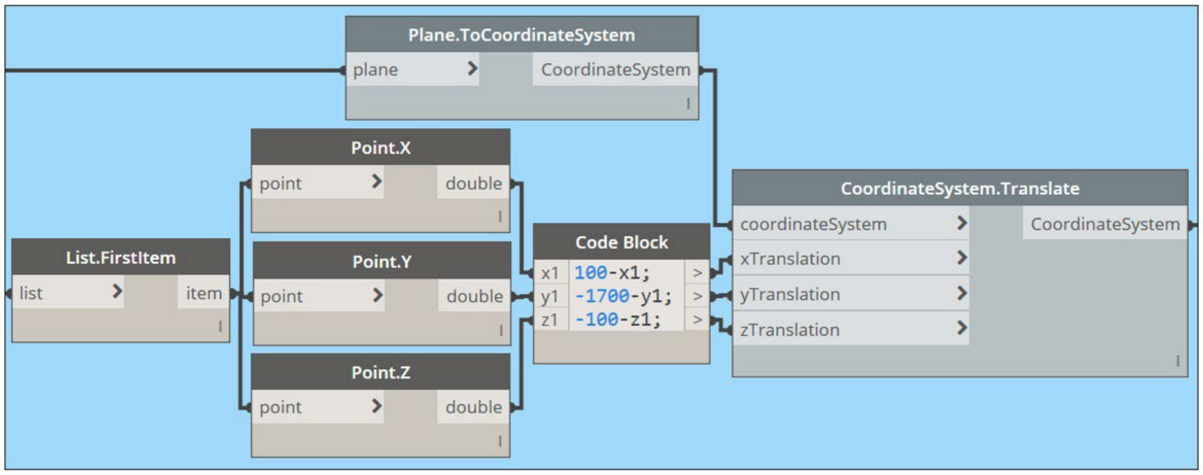


Figure 11. Result of substages (Trajectory planes).

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.



```
x1  100-x1;
y1  -1700-y1;
z1  -100-z1;
```

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 (X Axis and Y Axis) as shown in (Figure 13a).

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 13b), then the X, Y, and Z components are obtained separately to be able to be subtracted from the print space coordinates. This procedure allows changing the parameters Lc, ht, and hc never to move the first point of the figure.

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

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

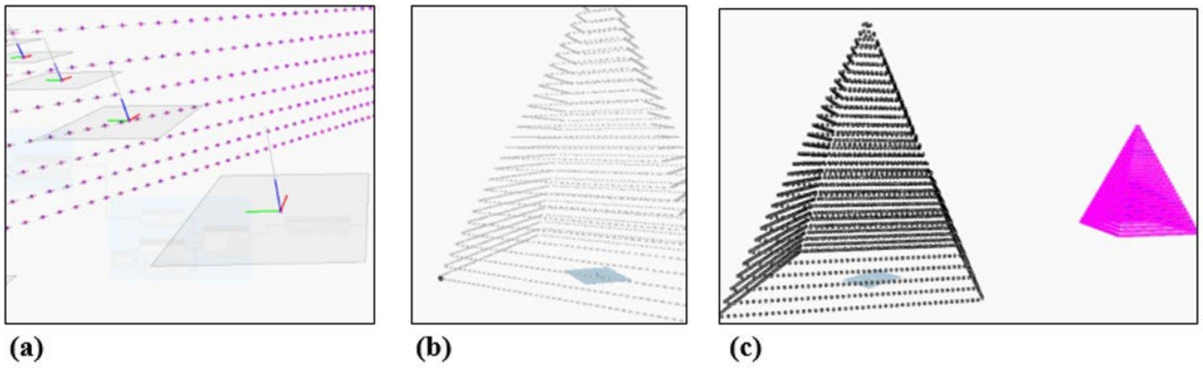


Figure 13. Result of the substages (Displacement of coordinate planes).

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.

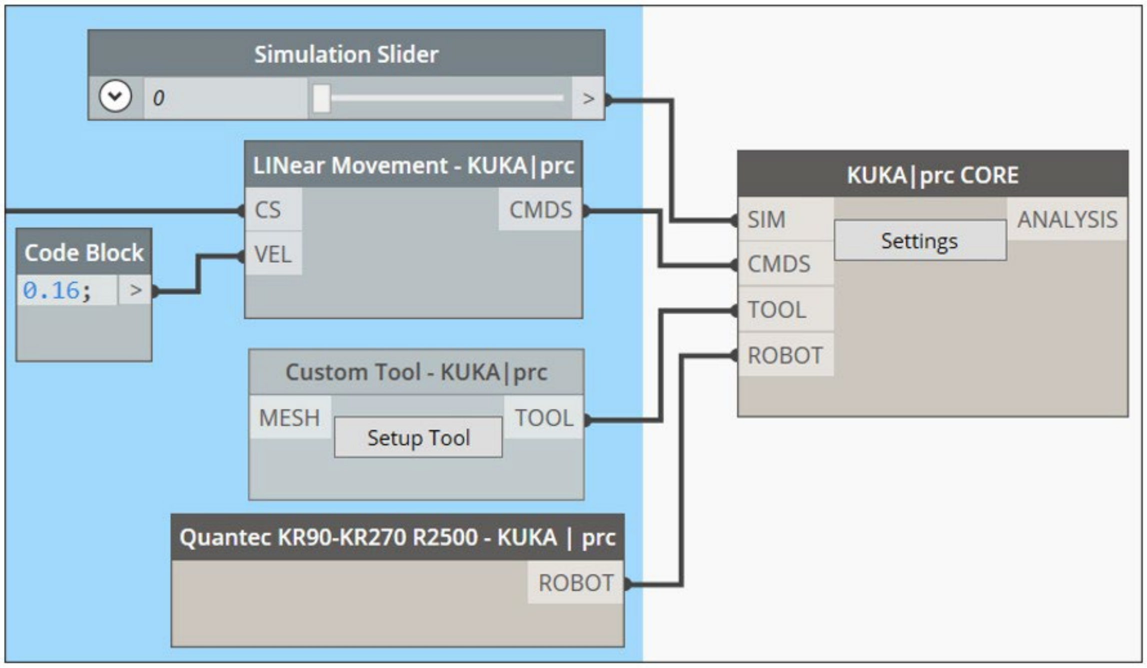


Figure 14. Stage 5: Robotic arm trajectory.

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).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

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. 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. KUKATM robot model: The KUKATM robotic arm model is selected (Quantec KR90-KR270 R2500).

4.3 Specific description of the geometry stage

The step-by-step explanation of this stage (Geometry) is divided into five substages since the code is similar in some figures as follows: 1) regular geometries: prisms and pyramids, 2) irregular geometries: curved pyramids and curved prisms, 3) irregular geometries: random 1 and 2; 4) irregular geometry: random 3 and 5) irregular geometry: random 4. In summary, (Figure 15) shows a flowchart for regular geometries and (Figure 21) for irregular geometries.

4.3.1 Regular Geometries: Pyramids and prisms

The procedure for regular geometries is the same until substage 3; the rest of the process is described separately for each model. (Figure 15) shows the procedure.

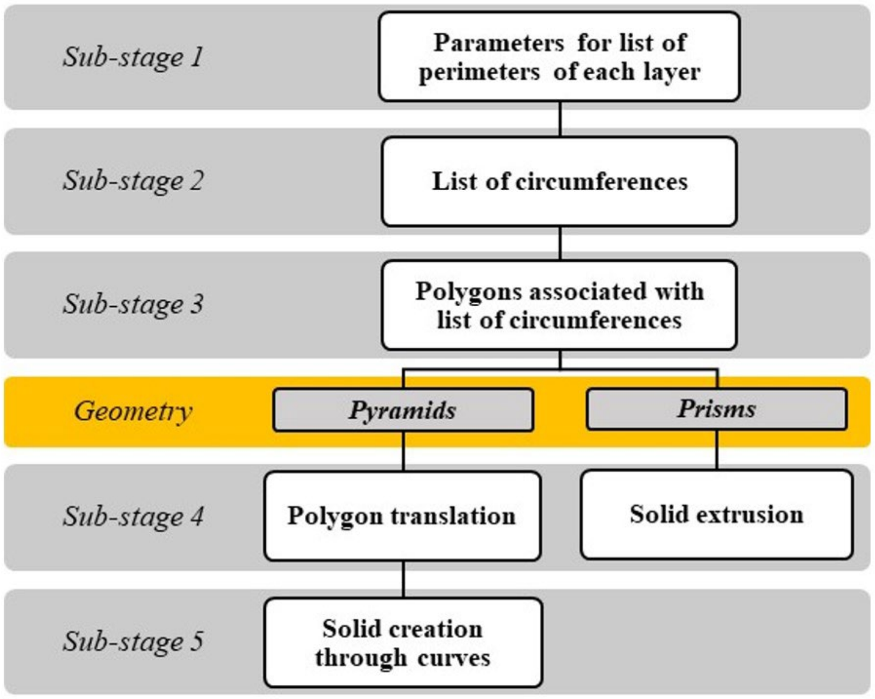


Figure 15. Geometry modeling procedure for regular geometries.

Consequently, the geometry substages for regular geometries are listed sequentially below:

1. Parameters for the list of perimeters of each layer: Using the “Code.Block” node, a sequence was created that begins with the length value of the basal edge called a (r, for geometries with a circular base) and ends with the last layer, xn. The edge formula varies depending on the geometry to be modeled.

2. List of circles: With the node “Circle.ByCenterPointRadius”, a circle with a center at an input point and a given radius is created. This step is created in the XY plane, with Z as normal. To execute the square root operation, it is necessary to use the “Math.Sqrt” node to draw a circle with a certain radius. (Figure 16d) shows the result.

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

3. Polygons inscribed in a list of circles: the "Polygon.RegularPolygon" node allows for inscribing a regular polygon inside a circle. In this case, the list of previously created circles was entered to inscribe a polygon. The result is a list of polygons located inside the list of circles, as shown in (Figure 16) (this step is unnecessary for the circumference).

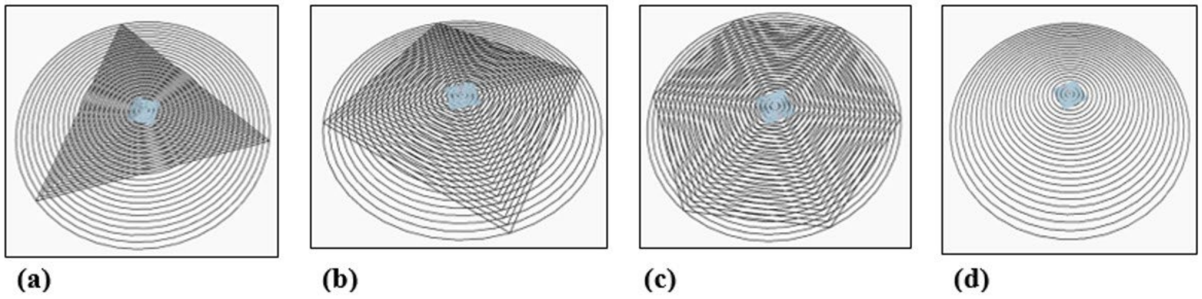


Figure 16. Polygons are inscribed in a list of circles: (a) triangle, (b) square, (c) hexagon. (d) List of circles.

Pyramids
1, 2, and 3. Steps are the same for Pyramids and prisms.

4. Polygon translation: The "Geometry.Translate" node translates the geometry along the Z axis (all polygons inscribed in the circles). The translation of the squares corresponds to a simulation of the perimeters of the layers required to print. This procedure is done by increasing the numerical sequence (from 0 to the total height ht), where the delta of the layer change is given by hc (layer height). (Figure 17b) shows the translated curves later used to create a solid.

5. Creating Solids Through Curves: Using the "ByLoft" node with "CrossSections" (taking a list of closed curves as input), a solid is created by hovering between the cross-section of the list of curves, as shown in (Figure 17), to each pyramid.

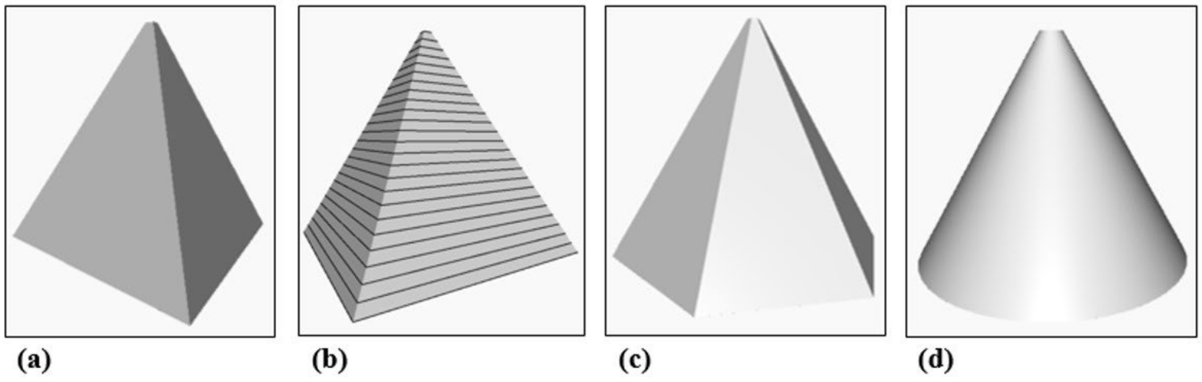


Figure 17. Geometry results for pyramids: (a) triangular, (b) square (shown with translated curves), (c) hexagonal, and (d) circular.

Prisms
1, 2, and 3. Steps are the same for Pyramids and prisms.

6. Solid Extrusion: The "Curve.ExtrudeAsSolid" node extrudes a closed curve, using an input number to determine the extrusion distance. The direction of this is determined by the normal vector of the plane in which the curve lies. This node covers the ends of the extrusion to create a solid, as shown in (Figure 18) for each prism.

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

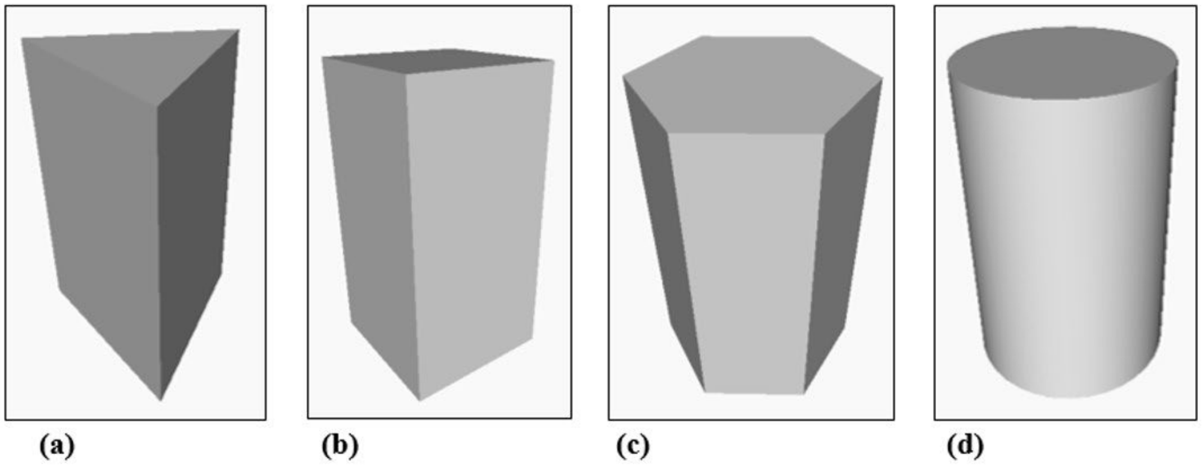


Figure 18. Geometry results for prisms: (a) triangular, (b) square, (c) hexagonal, and (d) circular.

4.3.2 Irregular Geometries: Curved pyramids – Curved prisms

The curved profiles are constructed by translating the curve of the basal geometry along the z-axis at every certain height given by hc. These curves are adjusted on the x-axis to offset one from the other. A sine function gives this delta; finally, the "ByLoft" node creates the solid by scrolling through the cross-section of the list of curves. (Figure 19) shows these geometries as trajectory curves to better visualize the curvatures.

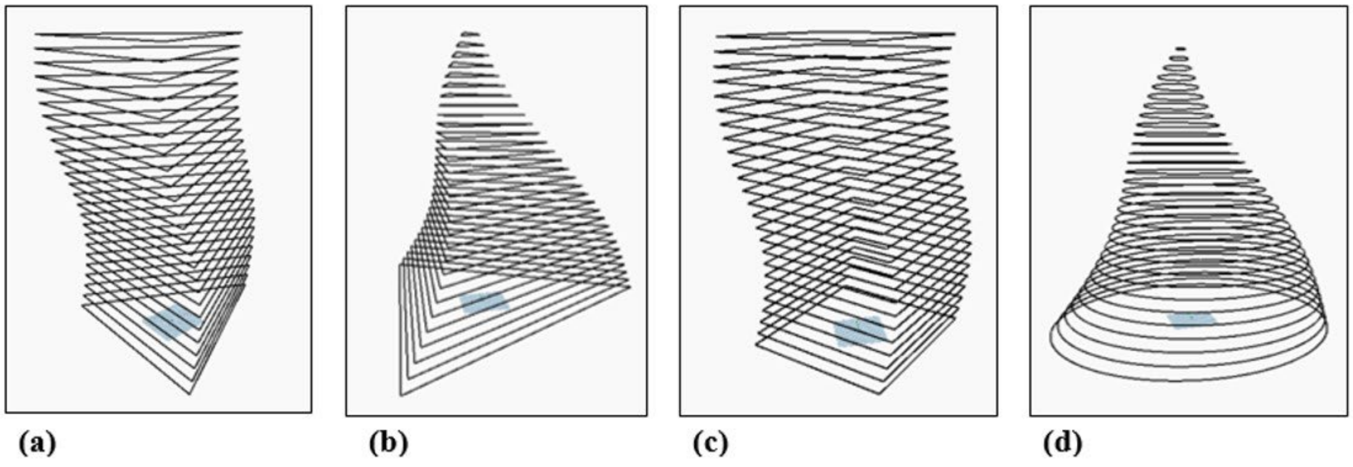


Figure 19. Curved geometries: (a) triangular prism, (b) triangular pyramid, (c) square prism, and (d) circular.

4.3.3 Irregular Geometries: Random geometries 1 and 2

These designs were built by generating random coordinates in the XY plane. Then, through a node, a curve was created connecting all the points until a closed curve was obtained, obtaining irregular solid geometries. (Figure 20) shows the procedure for modeling random geometries 1 and 2. These are very similar to substage 5, so they are explained together, showing the variations of some values, and then the process is divided and explained separately.

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

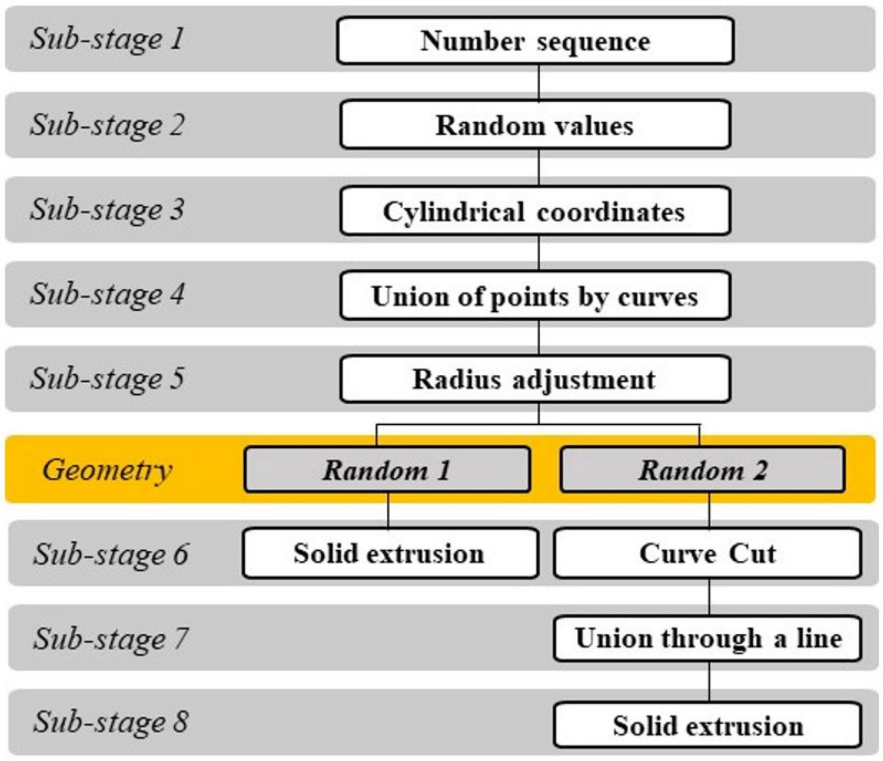


Figure 20. Process for modeling random geometries 1 and 2.

The geometry substages for irregular geometries (random 1 and 2) are listed sequentially below:

1. Number sequence: A number sequence was created from 0 to 360, with an increment of 360/(n-1).

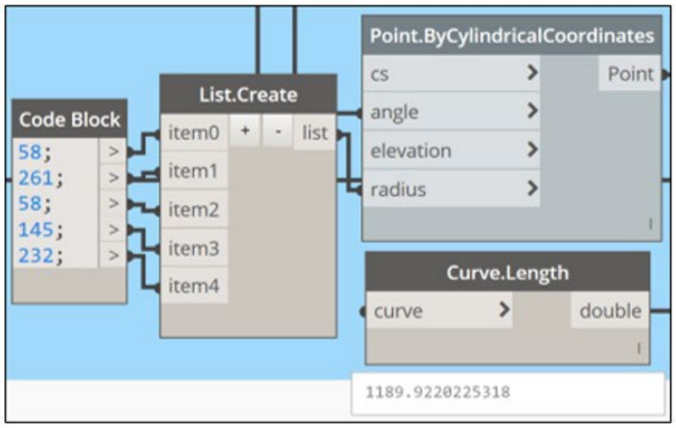
2. Random values: Five in the case of random 1, as shown in (Figure 21a), and nine in the case of random 2, as shown in (Figure 22a), were entered to indicate radius values, which are later used for input values in the node “Point.ByCylindricalCoordinates”.

3. Cylindrical coordinates: Using the node “Point.ByCylindricalCoordinates”, a point located within a cylindrical space is created. Points are then generated, each associated with an angle and radius value. These points are created in the XY plane without elevation values and are all referenced to the origin (0,0,0).

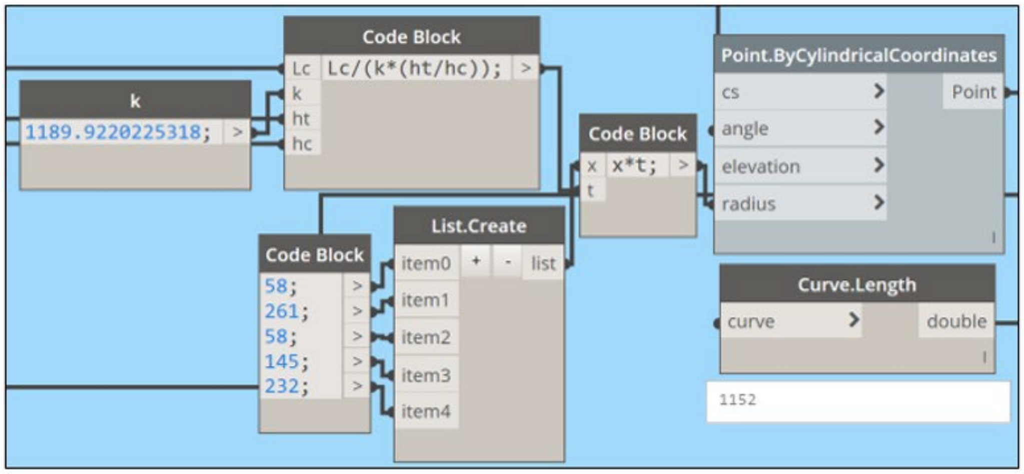
4. Union of points by curve: Through the node “NurbsCurve.ByPoints”, a list of points is introduced to draw a curve of Nurbs that allows closing the curve. In this case, the points generated in the XY plane are connected through a closed curve.

5. Radius adjustment: The values used for the radii must be adjusted to obtain the exact required curve length. For this, the length of the curve is considered as P(x) = x * t, where t is the correction or adjustment of the curve and x is the radii entered as shown in (Figure 21), (a) without radius adjustment and (b) with radius adjustment. Then, k values were obtained experimentally; when t takes the value of 1, the perimeters of the resulting geometries are equal to a constant (Random 1: k = 1189.922; Random 2: k = 1161.940). Thus, it is possible to arrive at the required curve length (Figure 19b)), which will finally deliver the intended extruded bead length (Lc).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21



(a)



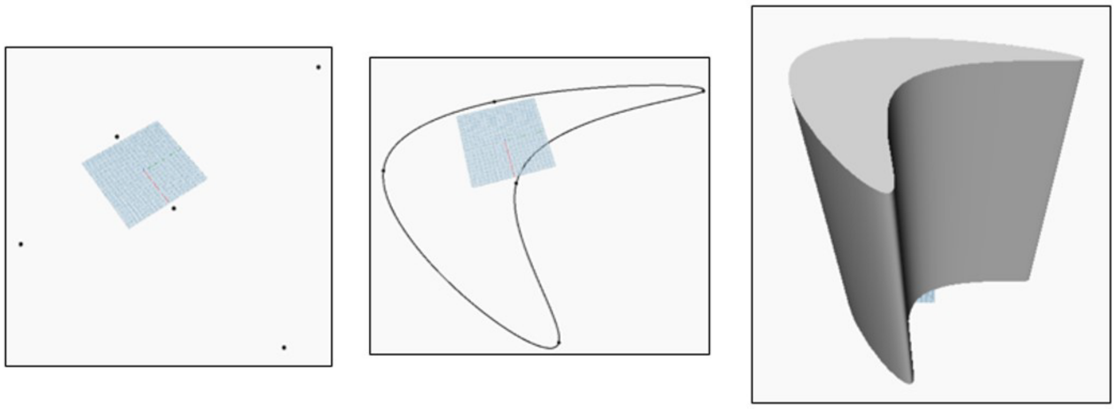
(b)

Figure 21. Substage for irregular geometries (a) without radius adjustment and (b) with radius adjustment.

Random 1

1, to 5. Steps are the same for Random 1 and 2.

6. Solid Extrusion: The "Curve.ExtrudeAsSolid" node extrudes a closed curve, using an input number to determine its distance, where the direction is determined by the normal vector of the plane in which the curve lies. This node will cover the ends of the extrusion to create a solid, completing the geometry procedure for this geometry as shown in (Figure 22).



(a) (b) (c)

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

Figure 22. Development of random geometry 1.

Random 2
1 to 5. Steps are the same for Random 1 and 2.

7. Trim Curve: The "Curve.TrimByParameter" node allows removing the start and end of an input curve by trimming it according to specified parameters to deliver the resulting center section of the curve.

8. Union through a straight line: Through the nodes "Curve.StartPoint" and "Curve.EndPoint", the start and end point, respectively, of an input curve are obtained as shown in (Figure 23c). Then, the node "Line.ByStartPointEndPoint" creates a line between two input points, as shown in (Figure 23d).

9. Solid Extrusion: The "Curve.Join" node joins two input curves in a new "PolyCurve" format, keeping the original curves strictly. Then, the node "Geometry.Translate" copies the base curve to a height of 600. Subsequently, a solid is created with "Solid.ByLoft", as shown in (Figure 23e).

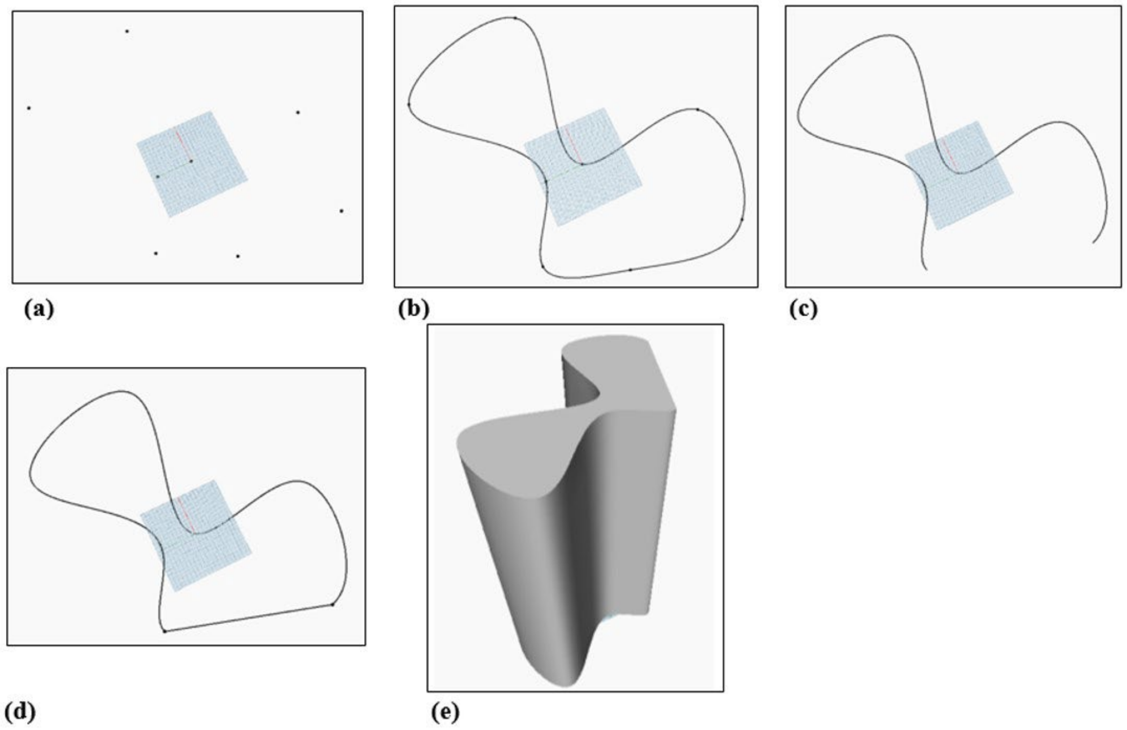


Figure 23. Development of random geometry 2.

4.3.4 Irregular Geometries: Composed random geometry (Difference of solids)

This geometry results from the difference in solids between the three elements, as shown previously in (Figure 7). First, a square was created, and then two opposite vertices were made: a circle and another smaller square. The substages for irregular geometries (composed of random geometries) are listed below sequentially.

1. Main geometry creation: With the "Rectangle.ByWidthLength" node, a square with an edge length equal to the previously calculated is created. The square is then extruded as a solid at a given height, as shown in (Figure 24a).

2. Base vertices of the main geometry: The node "Surface.ByPatch" allows the creation of a surface by filling the interior of a closed curve. The node "Topology.Vertices" produces a list of vertex locations for an input geometry, while the node "Vertex.PointGeometry" shows the vertex locations as points. A surface is created on the previously created square, then the vertex coordinate locations of this surface are obtained, and finally, the points of these vertices are obtained, as shown in (Figure 24b).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

3. Circumference in a vertex of the main geometry: Using the “List.GetItemAtIndex” node, point number one from the list of vertices of the created surface is obtained. A circle is then created with the “Circle.ByCenterPointRadius” node, with a radius of half the edge of the largest square and center point at the vertex point (Figure 24c) and extruded as a solid with the “Curve.ExtrudeAsSolid” node (Figure 25e).

4. Square in a vertex of the main geometry: Through the “List.GetItemAtIndex” node, point number three from the list of the vertices of the created surface is obtained. A square is then created with “Rectangle.ByWidthLength” of edge length equal to half the edge of the largest square and coordinates at the vertex point (Figure 24d) and extruded as a solid with “Curve.ExtrudeAsSolid” node (Figure 24e).

5. Solid Difference: Finally, the “Solid.DifferenceAll” node creates a new solid by subtracting a list of solids from the parent geometry, as shown in (Figure 24f).

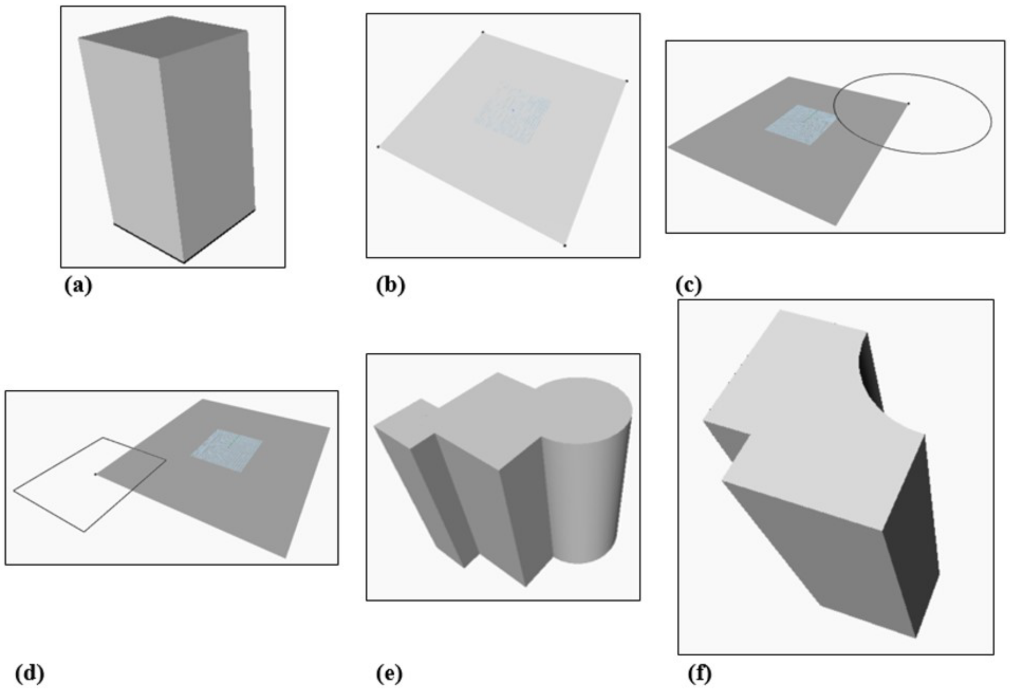


Figure 24. Development of composed random geometry.

4.3.5 Irregular Geometries: Random geometry as a Polygon by points

This design corresponds to an irregular polygon inscribed in a circle. First, a circle is created and divided into ten equally spaced points. Four of the ten points created are selected to be joined with a straight line to form an irregular polygon. Finally, the geometry in the XY plane is extruded as a solid. The substages for irregular geometries (polygon by points) are listed below sequentially.

1. Points selection on the circumference: First, a circumference is created, then using the node “Curve.PointsAtEqualSegmentLength” equally spaced points along the length of the curve are obtained based on the number of input divisions (10 by default) (Figure 25a). Finally, with the “List.GetItemAtIndex” node, points 1, 2, 4, and 8 from the ten created are selected (Figure 25b).

2. Perimeter adjustment: Through the node “Polygon.ByPoints”, the points of an input list are connected with straight lines to form a polygon (Figure 25 c). To get exactly the length of the required polygon, it is necessary to adjust the value used for the radius of the circumference. Let us call k an adjustment value of the radius to obtain the required perimeter (obtained experimentally), considering it as the resulting perimeter of the polygon when the radius is equal to 1. Thus, an adjustment to the radius is obtained to achieve precisely the required bend length. So, when r takes the value 1, the perimeter of the resulting geometry is equal to a constant k = 5.313.

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

3.Solid Extrusion: The extruded solid is obtained and shown in (Figure 25d).

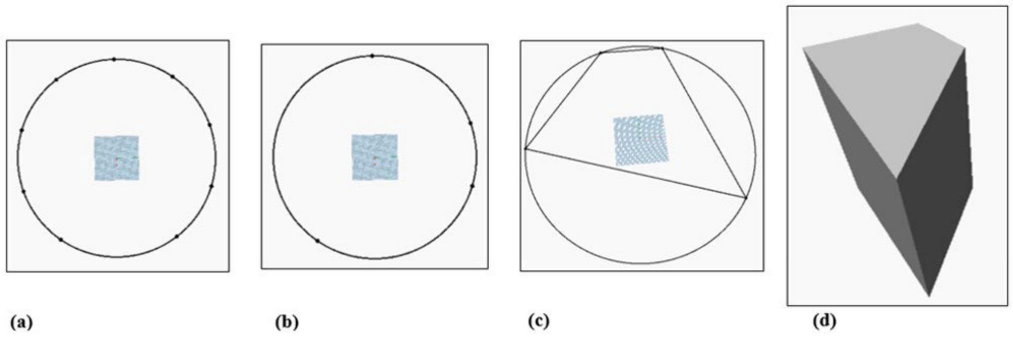


Figure 25. Development of random geometry 3.

5. Analysis of results

First, the methodology presented in this research is a valuable tool for efficiently developing computing codes used in 3D concrete printing processes for multiple shapes based on the sixteen geometries studied. Having tested different and complex geometries (prisms, pyramids, or polygons by points) opens up a wide range of opportunities for the construction sector since it is increasingly common to find more and more complex architectural geometries (Carneau et al., 2020); (Prasittisopin et al., 2021). In this sense, from a theoretical and methodological perspective, this research may contribute to the body of knowledge by providing a practical tool to compute program complex geometries to be printed in construction projects using visual routines that construction professionals may quickly implement without any previous programming experience.

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 that prints, 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 (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, (Table 2) and (Figure 26) show time data of the sixteen geometries, 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. 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.

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

Table 2. Printing times for each geometry.

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

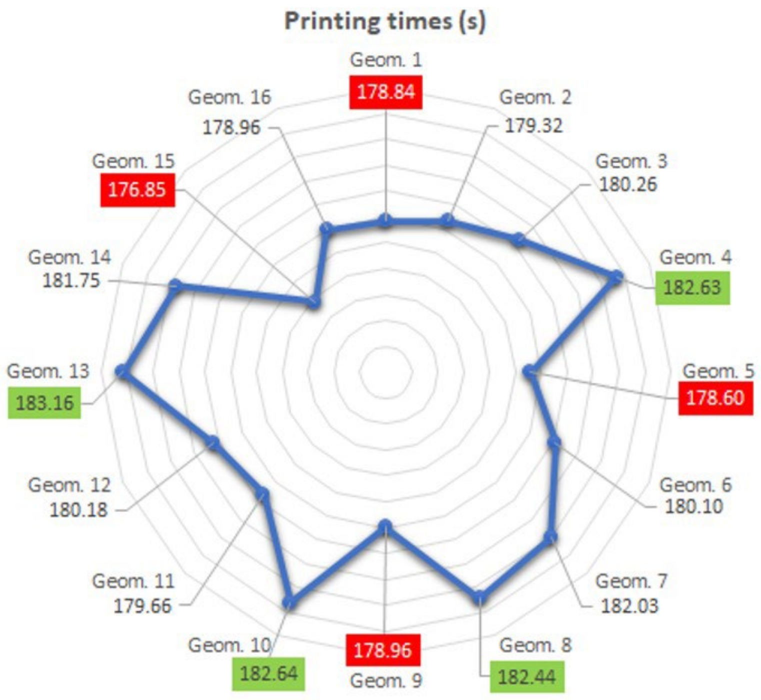


Figure 26. Chart with printing times for each geometry (labels of Geom. 1 to 16 correspond to those shown in Table 2).

(Figure 26) shows 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 evidence that more complexity implies more time 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).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

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. 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 time estimation 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, (Figure 27) shows the distances between points that make up the triangular prism geometry used as an example. The distance d1 (red lines) is the distance between points, which is the same for all the geometries since it depends on the robot extruder diameter. The distance d2 (green lines) is the one that is generated in the vertices, and 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 d2 depends on the angle of the vertex). Finally, d3 (blue lines) is the distance generated when switching to the next layer, where this value depends on hc (layer height) and the distance d2. Thus, what was done was to analyze only the first layer of a 3D printed figure (for this reason, the distance d3 is considered) to understand the variation in the printing time of the complete figure.

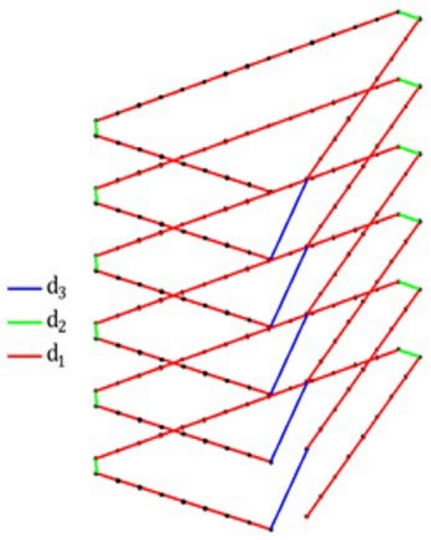


Figure 27. Point-to-point simulation for the triangular prism geometry (d_1: distance between points; d_2: distance generated in the vertices; and d_3: distance generated when switching to the next layer and depends on hc and d_2).

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 28).

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

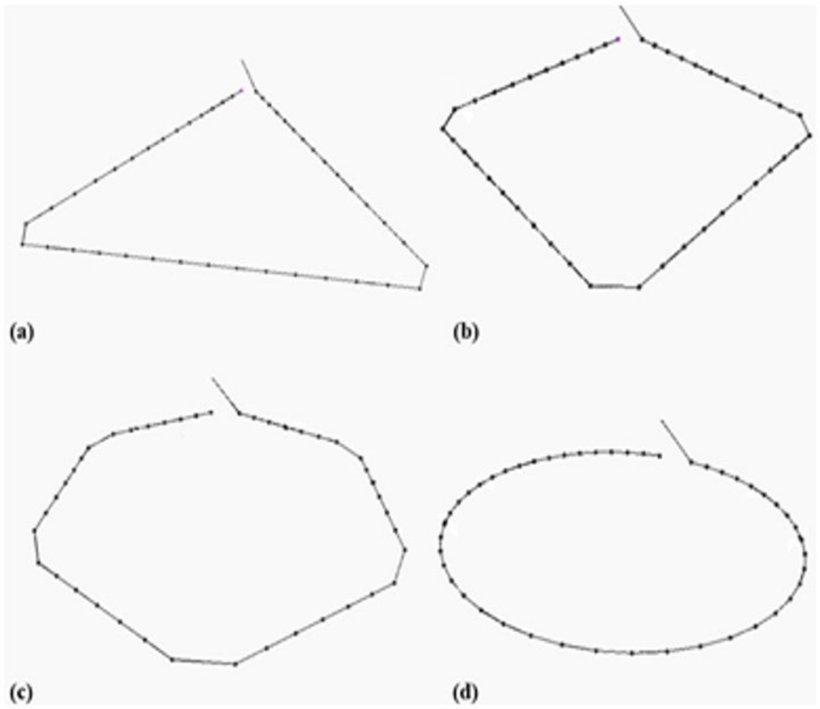


Figure 28. Bases considered in the comparative analysis: a) triangle, b) square, c) hexagon, and d) circumference.

The results are consolidated in (Table 3), which shows the following information: the vertices of the first layer for each prism, the number of points per layer, the spaces between them, the distances d1, d2, and d3, the actual Lc per layer (which corresponds to the perimeter of the first layer plus d3; 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 d2 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 show that the geometries' shape 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 (d1, d2, and d3), 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.

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

In summary, as part of this research, the computing routines work with 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, Lc is the only variable determining the printing time. However, in the case of the height of the extruded bead (hc), while it is technically possible to manufacture a nozzle that can modify its opening to change the printing height for the same element, no benefits are found with this modification for the same printed component. This finding is relevant as the printing height plays a crucial role in the 3D concrete printing process, being part of the control of geometric clashes with the design (Wolfs et al., 2018).

6. Discussion

Advanced visual programming algorithms are becoming increasingly important in solving challenging design problems involving large data sets or complex shapes (Forcael et al., 2021). Traditional algorithms may not efficiently handle such problems, which is where visual programming routines come into play.

The computing routines proposed in this research 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). From these variables, Lc is the one that influences the printing time the most. However, in the case of hc, although it is technically possible to manufacture a nozzle that can modify its opening to change the printing height for the same element, there are no apparent benefits associated with this modification, at least for an exact printed component (a wall for example).

Applying these results in the construction industry is relevant because it provides a working path based on a new additive manufacturing technology to produce construction elements faster and with more versatility. Building surroundings of structural parts, walls, or decorative elements, which can be integrated into the current construction, with facilities, connectors, and diverse finishing, providing a faster construction speed of these functions and direct control of their form and execution (García-Alvarado et al., 2020). This process reduces accessories, technical personnel, waste, transport, and construction accidents. It also contributes to the detailed management of the constructed geometry, allowing changes with more accuracy, along with the boost of structural optimization, improving designs, and reducing costs, materials, and construction times (Negrin et al., 2019).

Despite the previous findings, 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 relevant for future research. Thus, developing easy-to-use programming tools that are more flexible and adaptable to different types of geometries printed, 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 and the process variables (extruded bead length, perimeters, and heights) were analyzed. 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, regardless of their geometric complexity.
Although the printing speed depends on the number of axis changes (XYZ) in diverse geometries, the differences found were not significant. This allows architects, engineers, and constructors to build increasingly complex shapes, overlooking their geometry constraints.

Regarding potential applications of the research in 3D printing within construction and engineering, this research contributes to the body of knowledge by providing a practical tool to program complex geometries to be printed in construction projects, where the choice of geometry in 3D concrete printing has minimal consequences on overall printing durations, mitigating the environmental impact of construction processes. This finding is a fundamental benefit of 3D concrete printing for architectural sustainability. Regarding the study's limitations, they are related to not considering other variables that are also linked to geometric aspects, such as singularities in the geometry (electric boxes in walls, 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. Finally, another critical aspect to consider for future research is the role that the printing parameters studied in this research may play in structural failure during 3D concrete printing processes of complex construction geometries.

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

8. Notes on Contributors

Eric Forcael, Facultad de Ingeniería, Arquitectura y Diseño, Universidad San Sebastián, Santiago, Chile. **ORCID** https://orcid.org/0000-0002-3036-4329	**Elena Alucema,** College of Engineering, Universidad del Bío-Bío, Concepción, Chile. **ORCID** https://orcid.org/0009-0000-8676-4615
Adolfo Burkart, College of Engineering, Universidad del Bio-Bio, Concepción Chile. **ORCID** https://orcid.org/0009-0008-2526-7680	**Rodrigo García-Alvarado,** College of Architecture, Construction and Design, Universidad del Bío-Bío, Concepción, Chile. **ORCID** https://orcid.org/0000-0003-2216-2388
Javier Sepúlveda-Morales, College of Engineering, Universidad del Bío-Bío, Concepción, Chile. **ORCID** https://orcid.org/0009-0008-9885-806X	**Eder Martinez,** School of Architecture, Civil Engineering and Geomatics, University of Applied Sciences and Arts Northwestern Switzerland (FHNW), Muttenz, Switzerland. **ORCID** https://orcid.org/0000-0001-7918-9421

9. 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

Anton, A.; Reiter, L.; Wangler, T.; Frangez, V.; Flatt, R. J.; Dillenburger, B. (2021). A 3D concrete printing prefabrication platform for bespoke columns. Automation in Construction, 122, 103467. https://doi.org/10.1016/j.autcon.2020.103467

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

Blanco, I.; Ingrao, C.; Siracusa, V. (2020). Life-Cycle Assessment in the Polymeric Sector: A Comprehensive Review of Application Experiences on the Italian Scale. Polymers, 12(6), 1212. https://doi.org/10.3390/polym12061212

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

Chen, Y.; He, S.; Gan, Y.; Çopuroğlu, O.; Veer, F.; Schlangen, E. (2022). A review of printing strategies, sustainable cementitious materials and characterization methods in the context of extrusion-based 3D concrete printing. Journal of Building Engineering, 45, 103599. https://doi.org/10.1016/j.jobe.2021.103599

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

Dey, D.; Srinivas, D.; Panda, B.; Suraneni, P.; Sitharam, T. G. (2022). Use of industrial waste materials for 3D printing of sustainable concrete: A review. Journal of Cleaner Production, 340, 130749. https://doi.org/10.1016/j.jclepro.2022.130749

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

García-Alvarado, R.; Martínez, A.; González, L.; Auat, F. (2020). Projections of 3D-printed construction in Chile. Revista Ingeniería de Construcción, 35(1), 60–72. https://doi.org/10.4067/S0718-50732020000100060

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

Vol. 39
Nº3 2024
DOI: 10.7764/RIC.00114.21

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

Khajavi, S. H.; Tetik, M.; Mohite, A.; Peltokorpi, A.; Li, M.; Weng, Y.; Holmström, J. (2021). Additive Manufacturing in the Construction Industry: The Comparative Competitiveness of 3D Concrete Printing. Applied Sciences, 11(9), 3865. https://doi.org/10.3390/app11093865

Khamis, A. A.; Ibrahim, S. A.; Khateb, M. A.; Abdel-Fatah, H.; Barakat, M. A. (2022). Introducing the Architecture Parametric Design Procedure: From Concept to Execution. IOP Conference Series: Earth and Environmental Science, 1056(1), 012004. https://doi.org/10.1088/1755-1315/1056/1/012004

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.

Negrin, I.; Negrin, A.; Chagoyén, E. (2019). Metaheuristic optimization of structural sets of reinforced concrete. Revista Ingeniería de Construcción, 34(2), 181–192. https://doi.org/10.4067/S0718-50732019000200181

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

Tho, T. P.; Thinh, N. T. (2021). Using a Cable-Driven Parallel Robot with Applications in 3D Concrete Printing. Applied Sciences, 11(2), 563. https://doi.org/10.3390/app11020563

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

Wolfs, R. J. M.; Bos, F. P.; van Strien, E. C. F.; Salet, T. A. M. (2018). A Real-Time Height Measurement and Feedback System for 3D Concrete Printing. In High Tech Concrete: Where Technology and Engineering Meet (pp. 2474–2483). Springer International Publishing. https://doi.org/10.1007/978-3-319-59471-2_282

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

Zou, S.; Xiao, J.; Ding, T.; Duan, Z.; Zhang, Q. (2021). Printability and advantages of 3D printing mortar with 100% recycled sand. Construction and Building Materials, 273, 121699. https://doi.org/10.1016/j.conbuildmat.2020.121699