www.cad-journal.net
Computer-Aided Design AND APPLICATIONS

Design of a Lampshade Using Generative Design Methodology

Panagiota Ligka[1,a], Athanasios Manavis[2] and Panagiotis Kyratsis[1,b]

[1]University of Western Macedonia, [a]pligka@gmail.com, [b]pkyratsis@uowm.gr
[2]International Hellenic University, manavis.athanasios@gmail.com

Corresponding author: Panagiotis Kyratsis, pkyratsis@uowm.gr

Abstract. The present research focuses on the computational design and development of a generative floor lampshade produced through a parametric algorithm implemented in Grasshopper™ for Rhinoceros3D™. Generative design is understood here as the encoding of rules, constraints, and goals that allow form to emerge from stochastic and physical processes rather than from direct modeling. The design objectives were an organic, branching visual appearance and a porous wireframe surface that modulates light emission and casts intricate shadow patterns on surrounding surfaces. The core geometry is derived from a stochastically distributed network of points constrained to a lofted base surface, interconnected via a shortest path algorithm, and refined through a Kangaroo2™ physics solver applying anchor, mesh-constraint, and length-equalization goals. Variable pipe thickness, mapped to node height via a Graph Mapper™, differentiates the geometry toward the base for structural stability. Beyond the generative core, four extended modules are presented: a material and weight estimation module offering two methods (mesh-based and analytical frustum-based) with a parametric shell-fraction and infill model and a selectable material density from eight common FFF filaments; a part decomposition module that splits the geometry into independently printable segments using horizontal cutting planes, with automatic generation of interlocking cylindrical connectors at cut interfaces; a four-view technical drawing export module that produces orthographic and perspective projections as a .PDF layout; and a lightbulb picker that imports. OBJ models from a user-defined local library for accurate assembly visualization. The design was successfully fabricated as a scaled, single-part prototype on a Creality™ CR-M4.

Keywords: generative design; parametric modeling; Grasshopper™; shortest path algorithm; Kangaroo2™; lampshade; FFF; part decomposition; assembly connectors; organic geometry.
DOI: https://doi.org/10.14733/cadaps.2027.119-132

1 INTRODUCTION

Generative design is a computational design paradigm in which the designer does not model form directly but instead encodes the rules, constraints, and goals that allow form to emerge [2], [6]. It

© 2027 U-turn Press LLC, http://www.cad-journal.net

sits at the intersection of algorithmic design, the specification of a step-by-step generative procedure [12], and parametric design, the definition of relationships between variables that propagate automatically through the model [1]. Its defining characteristic is the introduction of stochastic elements and physical constraints into the generative process, so that the output cannot be fully anticipated by the designer [10].

The typical generative design process proceeds through six stages: definition of the design framework, definition of design goals, realization of the generative algorithm, production of candidate designs, evaluation of results, and iterative refinement [11]. This structure implies that the designer's primary creative act is the authoring of the algorithm rather than the shaping of the object. Parameter adjustment and constraint tuning replace conventional modeling operations, and the computer performs repetitive computation that would be impractical to perform manually.

This paper documents the application of this methodology to the design of floor lampshade, building upon established CAD product design frameworks [4]. The design objective was to produce a lampshade with an organic, growth-like appearance and a porous wireframe surface that modulates light emission and shadow projection, a geometry deeply tied to the principles of architectural computing [8]. The geometry is generated by a parametric algorithm developed in the Grasshopper™ visual programming environment [3], a plugin for Rhinoceros3D™ [9], and makes use of the Kangaroo2™ physics solver for constraint-based form-finding (Fig. 1) [7]. The paper describes the generative algorithm in detail and presents a set of extended modules for material estimation, part decomposition and assembly connector generation, technical drawing export, and lightbulb visualization to enable automated digital customization [5].

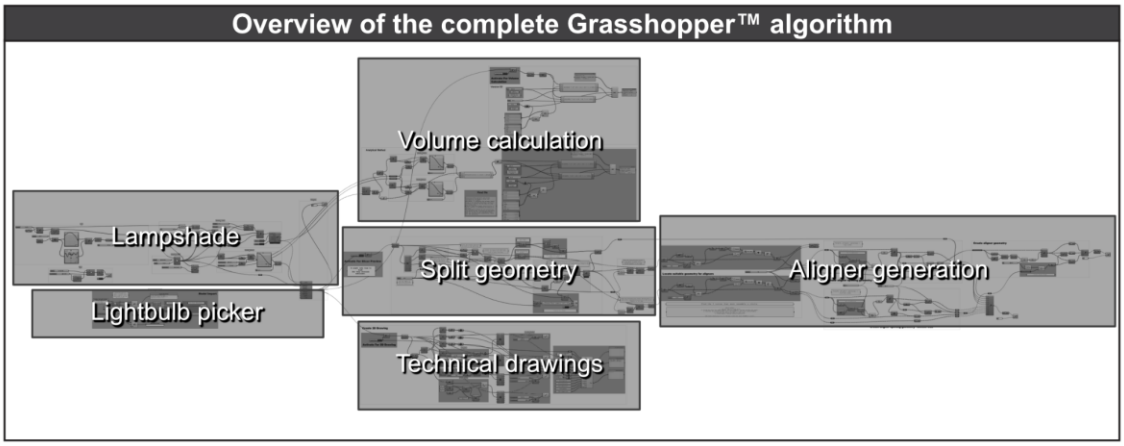

Figure 1: Overview of the complete Grasshopper™ algorithm.

2 DESIGN INTENT AND AESTHETIC FRAMEWORK

The ideation phase of the project began with a broad exploration of generative design strategies applicable to a lamp form, mapped across four conceptual clusters: biological growth (shortest path algorithms, L-systems, recursive branching), stochastic subdivision (Voronoi tessellation, Lloyd's algorithm, cracked-earth patterns), fields and forces (vector fields, isosurfaces, scalar fields), and environmental response (solar sculpting, kinetic louvres). Each cluster was associated with a set of formal references and computational logic. The exploration is documented in the ideation mind-map shown in Figure 2.

From this broader space, the biological growth cluster was selected as the primary generative logic, specifically the combination of a shortest path network with constraint-based relaxation. This direction was chosen for its capacity to produce branching, tree-like connectivity that reads as organic

© 2027 U-turn Press LLC, http://www.cad-journal.net

growth, and for its formal affinity with the natural precedents identified as aesthetic references: the venation patterns of dragonfly wings, the branching morphology of coral and root systems, and the porous microstructure of avian bone. These references share a common structural property, lightweight, non-repeating surfaces produced by simple iterative rules, which the generative approach could plausibly replicate.

The design intent was refined through a series of sketches exploring lamp forms derived from network and branching geometries. The sketches investigated a range of silhouette profiles, strut densities, and base configurations before converging on the key formal decisions: a rounded, organic profile produced by a lofted base surface; a wireframe-like surface structure described as a porous shell, allowing light emission and shadow play; and a fabrication method of Fused Filament Fabrication (FFF) 3D printing in PLA. These decisions were captured in the concept sketch and variants shown in Figure 2.

Two primary design objectives emerged from this ideation process. The first is an organic, growth-like visual appearance, achieved through the use of the shortest path algorithm that produces branching, non-repeating connections between stochastically distributed points, and through a rounded base surface that avoids geometric regularity. The second is a porous, wireframe-like surface that interacts dynamically with light, allowing it to escape through the interstices between struts and cast complex shadow patterns on surrounding surfaces. The parametric nature of the algorithm allows both objectives to be pursued simultaneously while maintaining extensive control over the resulting form through parameter adjustment.

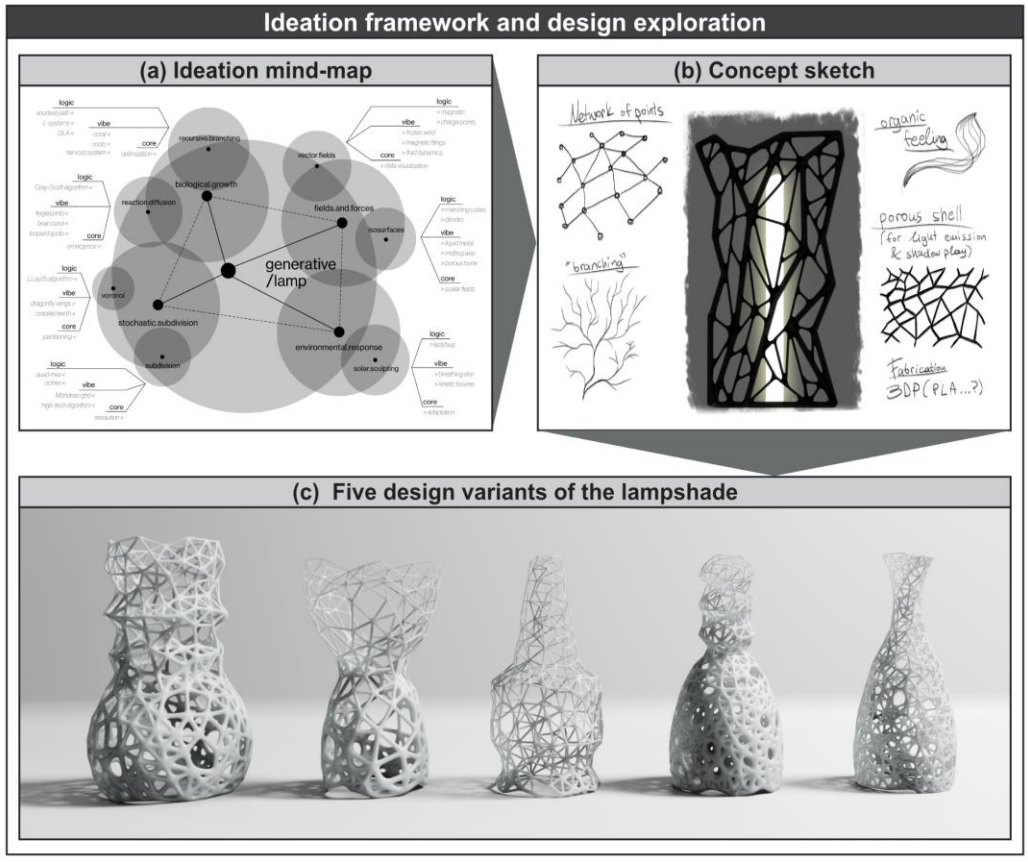

Figure 2: Ideation framework and design exploration.

© 2027 U-turn Press LLC, http://www.cad-journal.net

3 THE ALGORITHM

The Grasshopper™ algorithm is organized into five functional groups, each responsible for a distinct stage of the generative process. The groups are described in sequence below, and their Grasshopper™ implementations are illustrated in Figure 3.

3.1 Group 1: Base Geometry (Loft)

The script begins with a number slider that defines the overall height of the lampshade. A line of this length is divided into a Count of 20 equal segments, whose endpoints serve as the centers of a series of circles. The radii of these circles are modulated by two Graph Mapper™ components, one controlling the overall profile envelope and one introducing a secondary undulation, which together allow the designer to define the lampshade silhouette freely. The circles are lofted into a single smooth surface using the Loft component, forming the base geometry for subsequent operations.

3.2 Group 2: Network of Points (PopGeo™)

A stochastic distribution of points is generated on the base surface using a Populate Geometry component and a Seed parameter for repeatability. These points are connected using a Proximity 3D component with a Max Radius to limit the length of individual connections, producing a three-dimensional network of lines, whose density and topology vary across the surface.

3.3 Group 3: Morphogenesis via Constraint Solving (Kangaroo2™)

The raw network is passed through the Kangaroo2™ physics solver with four goal objects. A Length goal component drives segment equalization. An Anchor goal fixes points near the top and bottom edges. An On Mesh goal keeps all points constrained to the base surface. The solver is triggered manually via a Button component and can be paused via a Toggle.

The Anchor constraint fixes a subset of points near the top and bottom edges of the lampshade in their initial positions, preserving the overall form during relaxation. The On Mesh constraint ensures that all points remain on the surface of the base geometry throughout the solving process. The Length constraint drives the solver to equalize the lengths of the connecting segments toward a target value, producing a more uniform distribution of strut lengths across the network.

3.4 Group 4: Variable Thickness (NodeSize™)

The Z coordinate of each node in the solved network is extracted and remapped using a Remap Numbers component. The remapped value is passed through a Graph Mapper™ set to a descending curve, then multiplied by a scale factor B, producing a radius value per node that decreases from base to top. This radius list feeds directly into the MultiPipe component as the NodeSize™ input.

3.5 Group 5: Visualization (MultiPipe and Custom Preview)

The solved curve network and the NodeSize radii are passed to the MultiPipe component (labelled Materiality), which generates smooth branching tubular geometry. The MultiPipe inputs include Curves, NodeSize, SizePoints, EndOffset, StrutSize, Segment count, KinkAngle, CubeFit, and Caps. The output geometry is passed to a Custom Preview component for material visualization. A separate Base group generates the lamp base geometry using a cylinder.

4 MATERIAL AND WEIGHT ESTIMATION

A material estimation module was developed to compute the expected plastic volume and print weight of the lampshade for a given material and slicer configuration. The module is gated behind a Toggle Slicer Boolean switch because the underlying geometry processing is computationally intensive. Two independent calculation methods are provided and can be compared directly: a mesh-based method and an analytical method.

© 2027 U-turn Press LLC, http://www.cad-journal.net

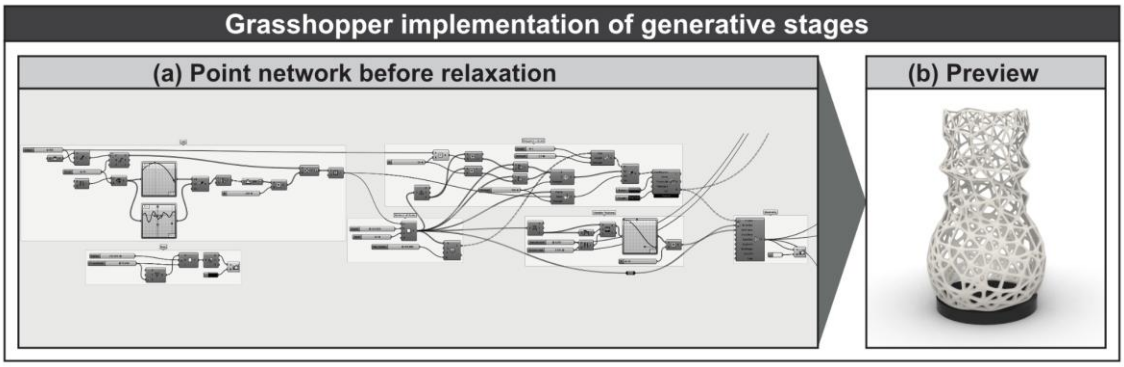

Figure 3: Grasshopper™ implementation of generative stages.

4.1 Mesh-Based Method

The mesh-based method computes volume by converting the MultiPipe Brep geometry to a mesh using a Mesh from SubD component and then applying the Volume component to the resulting mesh. This replaces an earlier Brep-based implementation that required approximately 15 seconds to evaluate; the mesh conversion reduces this to approximately 1 second. The accuracy trade-off was validated across ten models: the difference between the Brep and mesh results is consistently below 1%, which was considered an acceptable approximation. For applications requiring higher accuracy, the mesh resolution parameter D of the Mesh from SubD component can be increased, though this raises computation time significantly while yielding only marginal accuracy gains.

4.2 Analytical Method

The analytical method computes volume directly from the curve network produced by the Kangaroo2™ solver, without generating a solid mesh. For each curve in the network, the algorithm extracts the curve length L and evaluates the pipe radius at both endpoints, r_1 and r_2, using the same Graph Mapper™ remapping logic as the Variable Thickness group. The geometry of each strut is thus modelled as a truncated cone (frustum) with end radii r_1 and r_2 and axial length L, as illustrated in Figure 4. The volume of each strut is computed using formula 4.1.

$$V = (\pi \cdot L / 3) \times ({r_1}^2 + r_1 \cdot r_2 + {r_2}^2) \tag{4.1}$$

When validated against the mesh-based method across ten models, the analytical method consistently returns values approximately 70% higher. This discrepancy arises because at node junctions, where multiple struts converge, the frustum volumes of all meeting struts overlap spatially in the node region, causing it to be double- or triple-counted depending on node valence. A correction strategy is proposed: shorten each strut by its end radii ($\boldsymbol{L'} = \boldsymbol{L} - \boldsymbol{r}_1 - \boldsymbol{r}_2$) and add the volume of a sphere of the appropriate radius at each node exactly once, as illustrated in Figure 4. This would eliminate the double-counting, while preserving the contribution of the node material.

4.3 Material Parameters

Both methods feed their volume output V into the same weight estimation pipeline. Three user-configurable parameters are exposed: Shell Fraction, representing the proportion of the cross-section occupied by perimeter walls; Infill percentage, and a material selector offering eight common filament types with their densities: PLA (1.24 g/cm³), PETG (1.27 g/cm³), ABS (1.04 g/cm³), ASA (1.07 g/cm³), TPU 95A (1.21 g/cm³), Nylon (1.01 g/cm³), Polycarbonate (1.20 g/cm³), and Standard

© 2027 U-turn Press LLC, http://www.cad-journal.net

Resin (1.15 g/cm³). Two formulas are evaluated in parallel; one for print mass in grams (4.2), and another for effective plastic volume in cm³ (4.3). Results are assembled into a structured output panel showing material name, plastic volume, and estimated weight.

$$Weight = (V / 1000) \times D \times (S + (1 - S) \times i / 100) \quad (4.2)$$

$$Volume = (V / 1000) \times (S + (1 - S) \times i / 100) \quad (4.3)$$

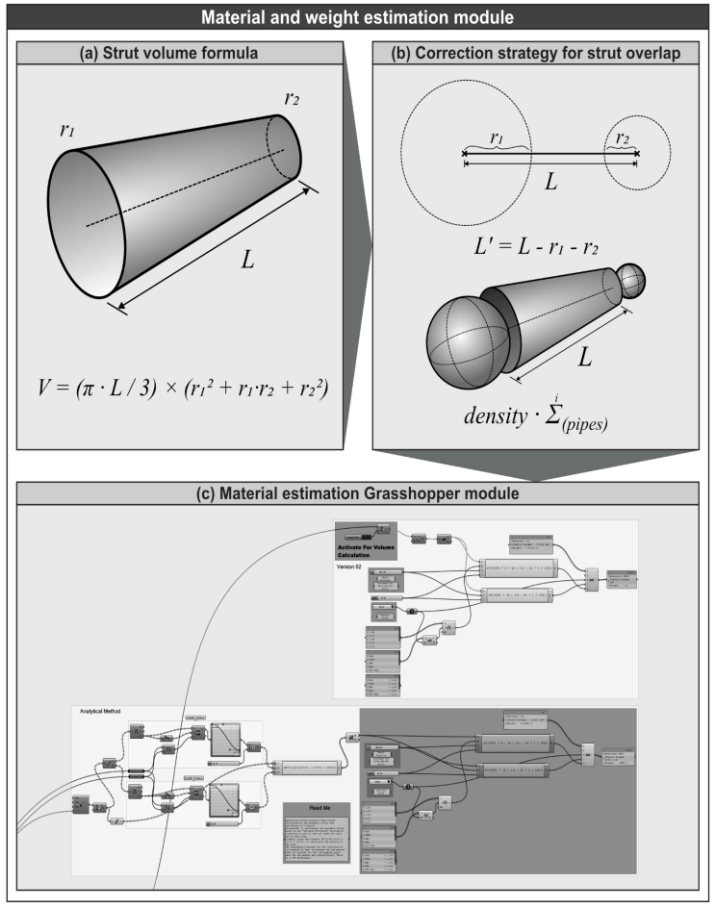

Figure 4: Material and weight estimation module.

5 PART DECOMPOSITION AND ASSEMBLY CONNECTORS

5.1 Geometry Splitting

A part decomposition module was developed to divide the lampshade geometry into a user-defined number of horizontally sliced segments for independent printing and subsequent assembly. The module is gated behind a Toggle Slicer Boolean switch. The current implementation operates on a

© 2027 U-turn Press LLC, http://www.cad-journal.net

mesh representation of the geometry rather than the original Brep, reducing computation time with no meaningful loss of geometric fidelity.

The number of segments is controlled by a Slicer integer slider, currently set to 4. The bounding box of the input mesh is used to construct a base cutting surface, which is arrayed vertically along a direction vector derived from the bounding box extents. The first surface in the array is removed to avoid a degenerate cut at the base. The remaining surfaces split the mesh sequentially into the target number of segments. A Find Intersecting Curves sub-component extracts the curve loops at each cut interface, used for both display and as input to the aligner generation system. Segments are sorted by Z centroid, labelled with a "Part number: N" Format string, translated along the Y axis for viewport legibility, and output to a Custom Preview (Fig. 5).

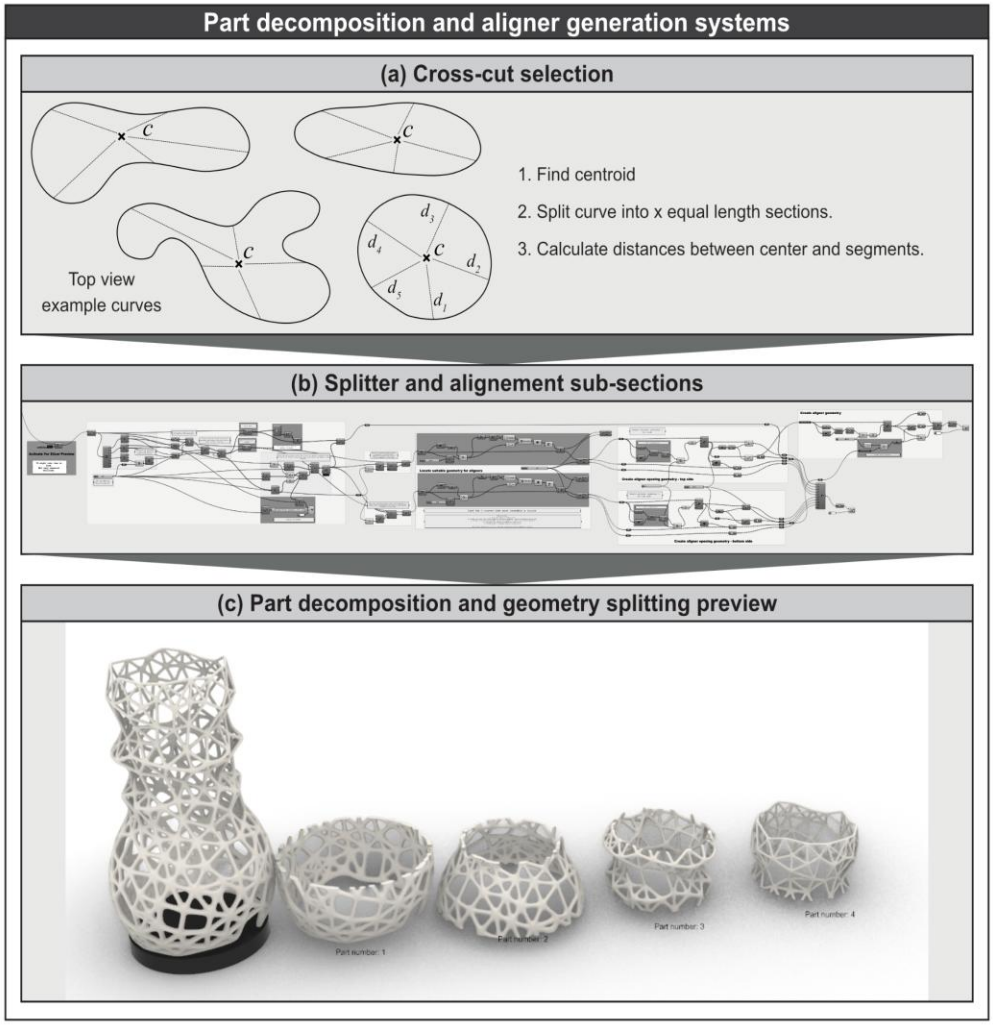

Figure 5: Part decomposition and aligner generation systems.

5.2 Aligner Generation

To enable physical assembly of the printed segments, a dedicated aligner generation sub-system automatically produces interlocking connector geometry at each cut interface. The rationale for the

© 2027 U-turn Press LLC, http://www.cad-journal.net

approach and the mathematical basis of the curve selection algorithm are illustrated in the diagrams in Figure 5.

The system selects candidate aligner locations based on a geometric property of the intersectional curves at each cut face. As illustrated in Figure 5, a cross-section that resembles a circle indicates that the underlying pipe strut is approximately vertical at that point. Vertical struts are preferable as aligner sites because they provide sufficient depth for the connector and produce a regular, manufacturable opening. Highly irregular or elongated cross-sections, produced by struts that cross the slicing plane at a steep angle, are poor candidates. One exception noted is when the slicing plane aligns perfectly with a sharp direction change at a junction of only two lines, which can also produce a near-circular profile despite the local geometry being less favorable.

The circularity of each intersectional curve is quantified using an error metric ε. Each curve is sampled at $N = 20$ uniformly arc-length-spaced points $P_i = (x_i, y_i)$. The centroid $P = (\bar{x}, \bar{y})$ is computed as the mean of these points. The radial distance from each sample to the centroid is $d_i = \|P_i - P\|$. The Wobble $W = d_{max} - d_{min}$ is the range of these distances. The mean radius $\bar{d} = \frac{1}{N} \cdot \sum_{i=1}^{N} d_i$ is then computed. The error value $\varepsilon = W / \bar{d}$ quantifies non-circularity: for a perfect circle $\varepsilon = 0$, and larger values indicate greater irregularity.

The three curves with the lowest ε at each cut interface are selected as aligner locations (the number is configurable via the Aligners slider). The selected curves are then moved to match their representative positions on the top and bottom faces of the adjacent sliced segments, handled separately for each side of the interface. Aligner opening geometry is created by offsetting each selected curve, translating it in the −Z direction for upper-face openings and in the +Z direction for lower-face openings, and extruding along the same vector. The boundary is closed with a Boundary Surface component. The aligner solid is created by applying a sliding-fit inward offset and extruding to the AlignerHoleDepth parameter. Boolean difference operations subtract the opening geometries from their respective segments, producing parts with integrated peg holes on one face and mating peg protrusions on the other.

6 TECHNICAL DRAWINGS

A technical drawing module was developed to produce a multi-view orthographic layout of the lampshade geometry and export it as a .PDF file directly from within the Grasshopper™ environment. Like the material estimation and part decomposition modules, it is gated behind a Toggle Slicer Boolean to prevent continuous recalculation. The full module is shown in Figure 6.

Four views are generated: Front, Right, Top, and Perspective. Each view is processed by a dedicated sub-group containing a Make2D component that projects the baked lampshade geometry into 2D linework. The geometry is first split Per Object using a Bake component so that Make2D receives individual objects rather than a merged mesh, which produces cleaner line separation. Three approaches to geometry preparation were evaluated for performance: the Brep method (17s), a simple mesh conversion (13s, with the top view taking disproportionately long for unknown reasons), and a decimated mesh (2s, but with all curves reduced to straight line segments). The current implementation uses the Brep method for documentation-quality output.

Each view's 2D linework is translated by a Factor offset along the X axis to position it within the layout: Front, Right, Top, Perspective. Each translated view is then moved to its final position on the drawing sheet using X and Y component sliders. The Perspective view sub-group additionally constructs a view plane from a 3-point definition and applies a Z-flip transform before projecting. All four positioned views are assembled in an In-canvas Visualize sub-group that merges their geometry for display.

The .PDF export is handled by a Python™ script component at the right end of the module. Its inputs are: the output file path, the four view geometry streams (Front, Right, Top, Persp), Width, Height, Unit (set to Millimeters), Sheet size (set to A3), and five title block text fields: Title, Designer, Engineer, Date, and Approved. A Run boolean input triggers the export; the panel note reads "Set

© 2027 U-turn Press LLC, http://www.cad-journal.net

Run = True to generate the PDF", making the export explicitly manual rather than automatic. The script writes the PDF to the specified path and returns the output file location.

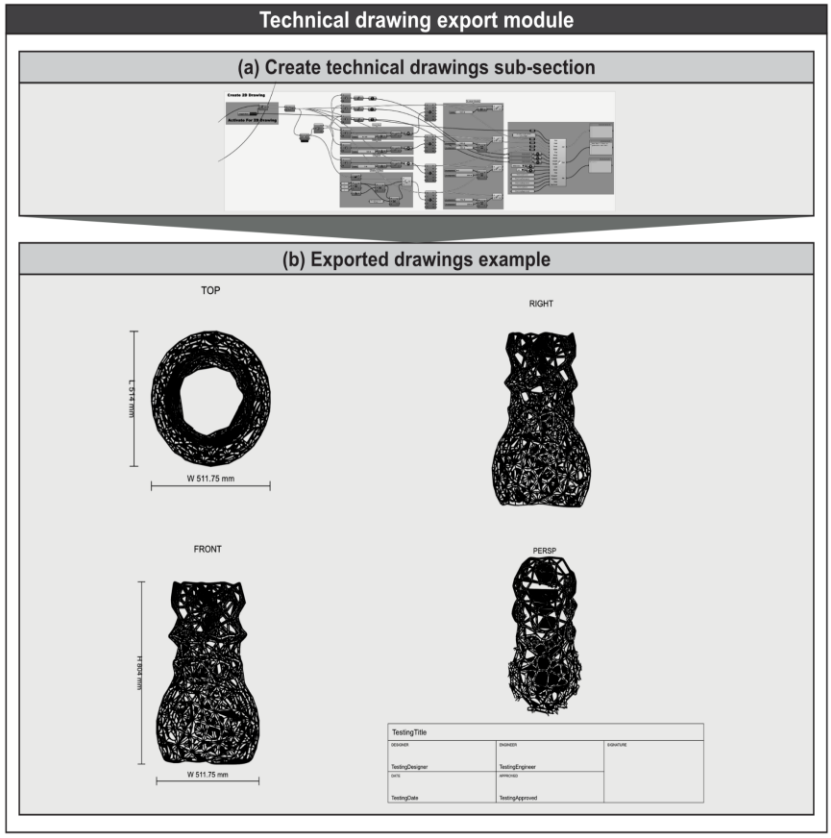

Figure 6: Technical drawing export module.

7 LIGHTBULB PICKER

7.1 Library Preparation

The lightbulb model library consists of a set of bulb types stored as individual .OBJ files inside a dedicated local folder. For the Grasshopper™ import pipeline to work correctly, each model must be origin-centered with its lowest point at the world origin. To prepare the library, a dedicated Rhino3D™ Python™ script was developed that automates this process in a single operation.

The script prompts the user to select the bulb objects and choose an output folder, then iterates over each. For each object it computes the "central lowest point", defined as the center of the bounding box in X and Y, and the minimum Z of the bounding box, and translates the object so that this point coincides with the world origin. The object is then exported as an individual .OBJ file named after its Rhino3D™ object name using Rhino's built-in export command called programmatically via rhinoscriptsyntax. After export the object is translated back to its original position, leaving the scene unchanged. The script thereby produces a folder of consistently oriented .OBJ files ready for use by the Grasshopper™ picker (Fig. 7).

© 2027 U-turn Press LLC, http://www.cad-journal.net

7.2 Grasshopper™ Picker

Within Grasshopper™, the picker module operates by reading the contents of the user-specified folder at runtime rather than relying on a hardcoded list of bulb names. The user configures the folder path once via a File Path component, which also filters for .OBJ and .STL file types. A List Item component paired with a number slider, labelled Model Selector, then selects a single file from the resulting list by index. Two display panels output the selected file's full path and its name for feedback, allowing the user to confirm which model is active without opening the file browser. The module is shown in Figure 7.

The selected file is imported into Grasshopper™ via an Import component, which reads the .OBJ geometry and outputs it into the canvas. Because OBJ files encode geometry in a coordinate system that does not necessarily match Rhino's world axes, the imported geometry passes through a rotation correction component (Fix Rotation) that reorients it from the YZ plane to the correct upright orientation. A subsequent Scale component (Fix Scale) applies a uniform scale factor, to bring the bulb geometry into the correct size relationship with the lampshade. The scaled and oriented geometry is then passed to the Custom Preview component alongside the rest of the lamp assembly.

This folder-driven approach means the library is fully extensible: adding a new bulb model requires only placing a correctly prepared .OBJ file in the folder, with no changes to the Grasshopper™ script itself. The module serves a representational purpose in the current version of the algorithm; future development could incorporate bulb-specific photometric data to enable performance-driven evaluation within the same environment.

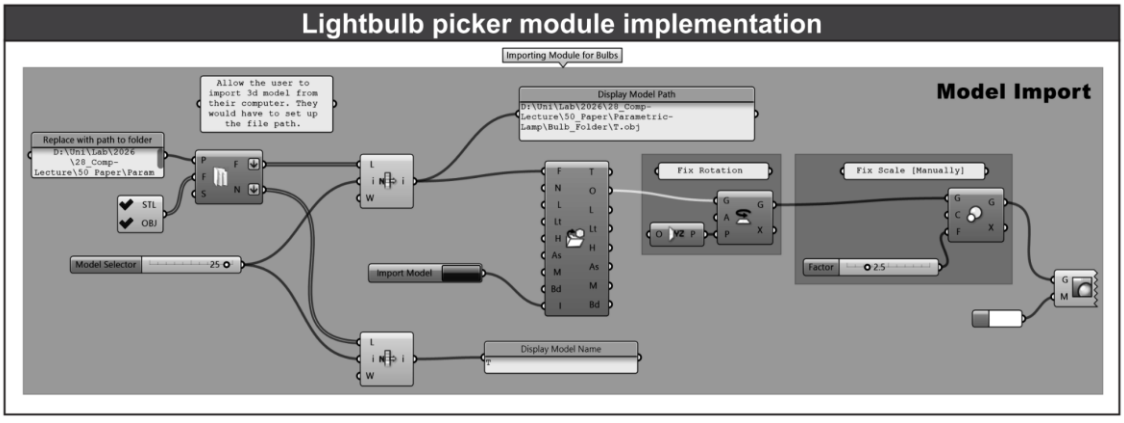

Figure 7: Lightbulb picker module implementation.

8 FABRICATION

8.1 Material and Equipment

The lampshade is intended for fabrication using FFF technology with white PLA filament. PLA was selected for its ease of printing, dimensional stability, and suitability for domestic interior applications. The open wireframe geometry of the lampshade is well-suited to FFF: the predominantly vertical and diagonal orientation of the struts minimizes the need for support structures, and the variable thickness gradient concentrates material at the structural attachment point while reducing mass toward the top.

© 2027 U-turn Press LLC, http://www.cad-journal.net

8.2 Prototyping and Build Volume

The proposed full-scale dimensions of the lampshade exceed the physical build envelope of standard desktop additive manufacturing systems. To evaluate the geometric fidelity, structural behavior, and aesthetic qualities of the algorithmically generated form, a scaled-down physical prototype was fabricated. This approach allowed the component to be manufactured as a single, continuous piece, eliminating the need for complex part decomposition, segment slicing, or post-assembly alignment features.

Fabrication of the scale prototype was performed on a Creality™ CR-M4 FFF 3D printer, utilizing its large build volume to accommodate the unified model in a single print run. By handling the scaled geometry as a single part, potential structural weaknesses associated with multi-segment interfaces were avoided, ensuring a direct and seamless transition from the digital Grasshopper™ model to the physical artifact (Fig. 8).

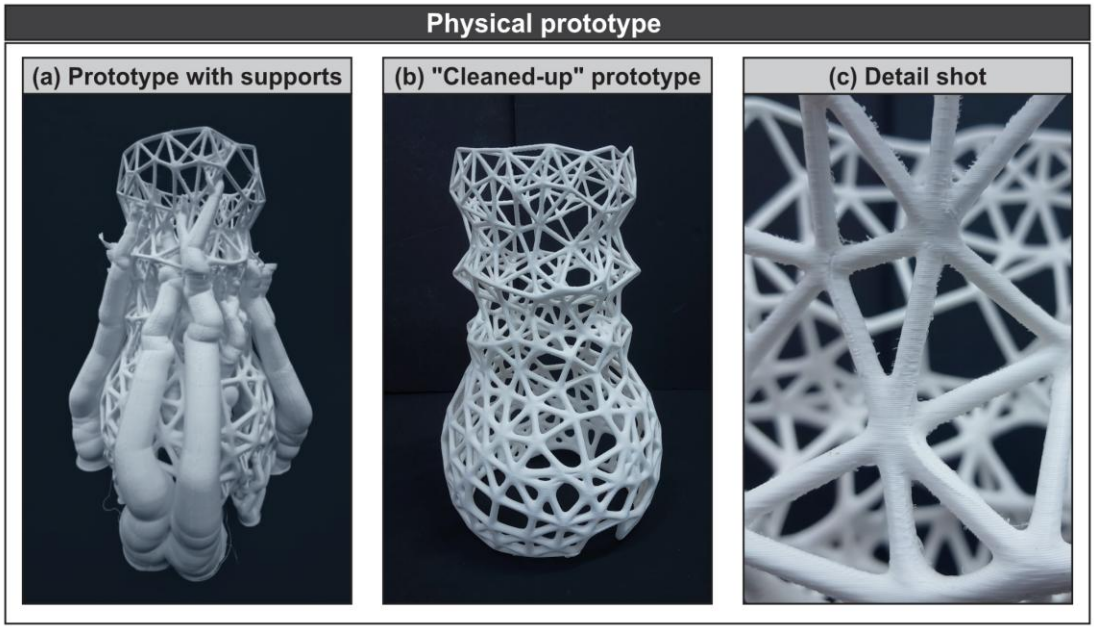

Figure 8: Physical prototype.

9 RESULTS AND DISCUSSION

The algorithm successfully produces lampshade geometries that satisfy the initial design objectives. Combining a shortest-path network with physics-based relaxation yields forms that are visually complex and organically coherent, without the rigid regularity of conventional parametric patterns. The branching, non-repeating topology of the strut network evokes the natural precedents identified in the ideation phase- coral growth, root systems, and dragonfly wing venation- while remaining clearly synthetic in character. The variable thickness gradient adds a perceptible hierarchy to the structure, reading as natural growth from a heavier, more robust base to a lighter, more open crown. The final rendered assembly is shown in Figures 9 and 10.

The parametric nature of the algorithm allows for extensive design exploration. Adjustments to the base profile curves, the number and density of distributed points, the maximum connectivity radius, and the target segment length each produce meaningfully different results. The material and weight estimation module enables these variants to be compared in terms of material consumption, supporting more informed design decision-making beyond purely visual evaluation.

© 2027 U-turn Press LLC, http://www.cad-journal.net

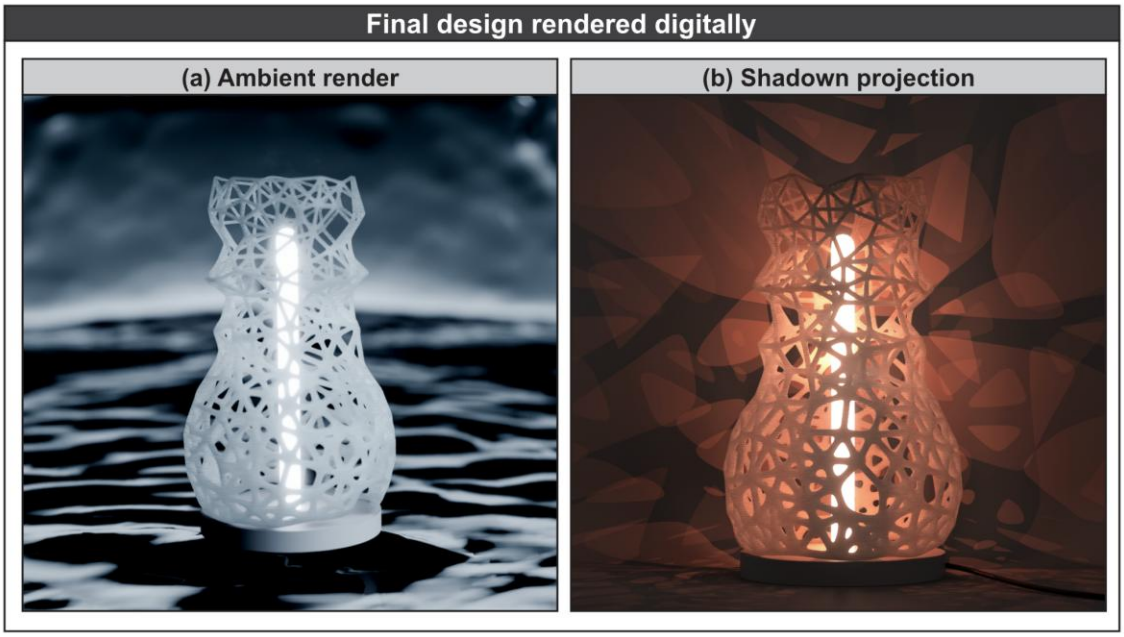

Figure 9: Final design rendered digitally.

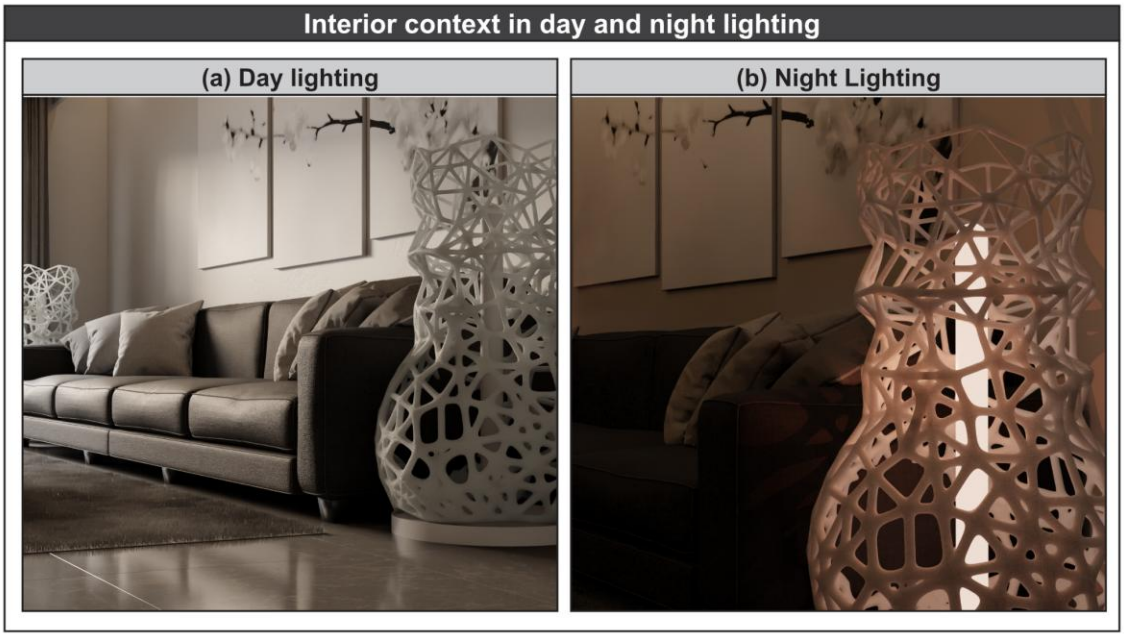

Figure 10: Interior context in day and night lighting.

The porous surface of the lampshade performs its intended optical function: light passes through the interstices between struts and projects an intricate pattern of shadows onto adjacent surfaces. The density of the network modulates the intensity and character of this shadow play; denser regions

© 2027 U-turn Press LLC, http://www.cad-journal.net

produce softer, more diffuse projections while sparser regions produce sharper, more defined ones. The render in Figure 9 shows this effect clearly against a dark background; the interior night render (Fig. 10) demonstrates the same behavior in a domestic context, with the shadow pattern extending across the surrounding wall and floor.

10 CONCLUSIONS

This paper has presented a generative design methodology for the development of a parametric floor lampshade, implemented as a multi-module Grasshopper™ algorithm. The approach embodies the generative design principle of encoding rules rather than modeling form: a stochastic point distribution, a shortest path connectivity algorithm, and a physics-based constraint solver combine to produce organic, non-repeating wireframe geometries whose precise configuration cannot be anticipated in advance but consistently satisfies the design objectives of branching visual character and porous surface quality. The algorithm demonstrates that complex, naturally-inspired forms can be derived from a small set of understandable computational rules, with extensive formal variation accessible through parameter adjustment alone.

Four extended modules complement the generative core. The material and weight estimation module offers both a fast mesh-based method and an analytical frustum-based method, providing practically useful fabrication data without leaving the Grasshopper™ environment. The part decomposition module addresses the scale of the design relative to available print volumes, generating interlocking cylindrical connectors at cut interfaces to enable future full-scale assembly. The technical drawing export module closes the documentation loop by producing four-view orthographic layouts directly from the algorithm. Finally, the lightbulb picker supports accurate design visualization by replacing the placeholder bulb with geometrically precise imported models.

Several directions for future development are identified. The analytical weight estimation method would benefit from the implementation of the proposed node-correction formula ($L' = L - r_1 - r_2$), plus a sphere at each junction), which would reduce the current ~70% overestimation. Additionally, while a scaled physical prototype was successfully fabricated as a single component on a Creality™ CR-M4 3D printer, future work will focus on fabricating the full 1:1 scale design using the developed part decomposition module. Longer-term directions include the incorporation of structural optimization criteria to minimize material usage, and a detailed photometric evaluation of the physical prototype's light-cast characteristics.

Panagiota Ligka, https://orcid.org/0009-0009-3356-1617
Athanasios Manavis, https://orcid.org/0000-0001-6627-0431
Panagiotis Kyratsis, https://orcid.org/0000-0001-6526-5622

REFERENCES

[1] Aish, R.; Woodbury, R.: Multi-level Interaction in Parametric Design, Smart Graphics, Lecture Notes in Computer Science, vol. 3638, Springer, Berlin, 2005, 151-162. http://doi.org/10.1007/11536482_13
[2] de Landa, M.: Philosophies of Design: The Case of Modelling Software, Verb Architecture Boogazine, Actar, Barcelona, 2001.
[3] Grasshopper3D: Grasshopper - Algorithmic Modelling for Rhino, https://www.grasshopper3d.com, Robert McNeel & Associates, 2007-2024.
[4] Kyratsis, P.; Gabis, E.; Tzotzis, A.; Tzetzis, D.; Kakoulis, K.: CAD Based Product Design: A Case Study, International Journal of Modern Manufacturing Technologies, 11(3), 2019, 110-115.
[5] Manavis, A.; Minaoglou, P.; Efkolidis, N.; Kyratsis, P.: Digital Customization for Product Design and Manufacturing: A Case Study within the Furniture Industry, Electronics, 13(13), 2024, 2483. http://doi.org/10.3390/electronics13132483

© 2027 U-turn Press LLC, http://www.cad-journal.net

[6] Menges, A.; Ahlquist, S.: Computational Design Thinking, John Wiley & Sons, Chichester, 2011.
[7] Piker, D.: Kangaroo2: Live Physics for Rhino and Grasshopper, https://www.food4rhino.com/en/app/kangaroo-physics, Robert McNeel & Associates, 2014-2024.
[8] Pottmann, H.; Asperl, A.; Hofer, M.; Kilian, A.: Architectural Geometry, Bentley Institute Press, Exton, PA, 2007.
[9] Rhinoceros 3D: Rhinoceros 3D NURBS Modelling, https://www.rhino3d.com, Robert McNeel & Associates, 1998-2024.
[10] Rutten, D.: Evolutionary Principles Applied to Problem Solving, Blog post, https://www.grasshopper3d.com/profiles/blogs/evolutionary-principles, 2010.
[11] Shea, K.; Aish, R.; Gourtovaia, M.: Towards Integrated Performance-Driven Generative Design Tools, Automation in Construction, 14(2), 2005, 253-264. http://doi.org/10.1016/j.autcon.2004.07.002
[12] Terzidis, K.: Algorithmic Architecture, Architectural Press, Oxford, 2006.

© 2027 U-turn Press LLC, http://www.cad-journal.net