



**Politecnico
di Torino**

Politecnico di Torino

Master's Degree in Architecture for Heritage

A.Y. 2024/2025

Graduation Session September 2025

Reconstructing Architectural Complexity:

An Integrated HBIM Workflow for the Monumental Atrium of Palazzo Barolo

Supervisor:

Giovannini Elisabetta Caterina (DAD)

Candidate:

Wu Yushan

Abstract

Historic Building Information Modeling (HBIM) combines survey data with parametric modeling to support the digital reconstruction of architectural heritage. Yet its application becomes problematic when elements present irregular forms together with typological variation. This paper takes the atrium and staircase area of Palazzo Barolo in Turin as a case study and proposes a HBIM workflow centered on components: by combining semantic segmentation and the LoIN (Information Demand Level) framework, and selecting matching parametric strategies for different components.

The research focuses on three representative systems: columns, vaults and railings. The column system adopts a proportional method nested family based on "base diameter" as the global parameter, converting the classic rules into re-usable parameter logic. The vault is modeled using the "floor-hosted void families" generated by Dynamo, ensuring precise positioning, stable host interaction and typological flexibility, covering three specific case in this project: atrium, transition and staircase. The oblique components in the railings are not suitable for the native rotational family and need to undergo shear transformation in Rhino before being imported into Revit. This demonstrates that the HBIM workflow can adopt cross-platform geometric modeling when dealing with special cases, rather than forcibly maintaining the parametric logic within a single platform. All components are generated based on point cloud segmentation and orthophotos, and are supplemented by consistent naming and lightweight semantic encoding to ensure traceability.

The results show that this workflow achieves a balance between geometric accuracy and parameter control, and has the potential for migration in similar heritage environments. The methodological contribution lies in: establishing a bridge between "form-driven" and "type/rule-driven", adopting corresponding strategies for different components - such as the host-void cutting mode for arches and the nested family parameter transfer for columns - while the cross-platform processing of individual free-form shapes is only a local supplement and does not affect the stability of the overall framework. The proposed component-based HBIM method in this paper provides a reusable modeling path for heritage protection, architectural analysis, and future reuse.

Riassunto

L'Historic Building Information Modeling (HBIM) integra i dati di rilievo con la modellazione parametrica, offrendo un percorso affidabile per la ricostruzione digitale dei componenti del patrimonio architettonico. Tuttavia, l'applicazione dell'HBIM rimane complessa quando gli elementi architettonici presentano al tempo stesso geometrie irregolari e variazioni tipologiche. Questo studio, prendendo come caso di riferimento l'atrio e la scala monumentale di Palazzo Barolo a Torino, propone un flusso di lavoro HBIM basato sui componenti, che combina la segmentazione semantica con il quadro del *Level of Information Need* (LoIN), selezionando strategie parametriche adeguate per ciascun tipo di elemento.

La ricerca si concentra su tre sistemi rappresentativi: colonne, volte e balaustre. Le colonne sono ricostruite mediante una strategia a famiglie annidate, governata dal parametro del "diametro di base", che traduce le regole proporzionali classiche in logiche parametriche riutilizzabili. Le volte sono modellate tramite famiglie di vuoto ospitate da solai (*floor-hosted void families*) generate con Dynamo, garantendo posizionamento preciso, interazione stabile con gli elementi ospitanti e flessibilità tipologica in atrio, zone di transizione e scala. Per le balaustre, gli elementi inclinati—non compatibili con le famiglie native a rivoluzione—sono ottenuti attraverso trasformazioni di *shear* in Rhino e successivamente integrati in Revit, dimostrando che, in presenza di geometrie eccezionali, è possibile ricorrere a modellazioni cross-platform senza mantenere necessariamente la piena editabilità parametrica in un unico ambiente. Tutti i componenti sono derivati da segmentazioni di nuvole di punti e proiezioni ortofotografiche, accompagnati da una nomenclatura coerente e da una codifica semantica leggera per garantirne la tracciabilità. I risultati mostrano che il flusso di lavoro proposto raggiunge un equilibrio tra accuratezza geometrica e controllo parametrico, risultando al contempo trasferibile a contesti patrimoniali simili. Dal punto di vista metodologico, esso costruisce un ponte tra approcci "form-driven" e approcci basati su tipologie e regole, adottando strategie mirate per ciascun sistema—come il modello ospite-vuoto per le volte e la trasmissione parametrica nelle famiglie annidate per le colonne—mentre i trattamenti cross-platform di forme libere restano interventi locali senza compromettere la coerenza generale. La ricerca propone quindi un metodo HBIM component-based, riutilizzabile, a supporto della conservazione, dell'analisi e del riuso futuro del patrimonio architettonico.

Content

1. Introduction	1
1.1 Research Background	1
1.2 Research problems	2
1.3 Research objectives	3
1.4 Scope of the research	4
1.5 Case study: Palazzo barolo	4
1.6 Structure of the thesis	5
1.7 Acknowledge	6
2. Literature Review	7
2.1 Informative and geometrical modeling	7
2.2 From Segmentation to Modeling Strategies	8
2.2.1 Semantic segmentation in HBIM	8
2.2.2 Geometric and parametric approaches in HBIM application	8
3. Methodology	11
3.1 Strategies comparison: Geometric vs. Parametric	11
3.2 Modeling framework	13
3.3 Column	17
3.3.1 Method Evaluation and logic of proportional strategy	17
3.3.2 Proportional Logic and Nested Family Strategy for Columns	21
3.3.3 Parameters Transformation in Nested family	22
3.3.4 Proportional Control in Component Detail Modeling	23
3.4 Vault	24
3.4.1 Analysis of architectural geometric forms	24
3.4.2 Method Evaluation: Void Family vs. DirectShape	26
3.4.3 Hosted Family Types for Void Forms.	28
3.5 Railing	30
3.5.1 Normal railing system in standard component and in historical building ...	30
3.5.2 Treatment of Special-Geometry Inclined Balusters	31
4. Implement and Result	34
4.1 Column	34
4.1.1 Two types Column in Atrium and Staircase	34
4.1.2 Modeling for column system	38
4.2 Vault system	47
4.2.1 Final Modeling Outcomes and Comparative Assessment	47
4.2.2 Vault atrium(VAU_ATR)	50
4.2.3 Vault intermediate(VAU_INT)	61
4.2.4 Vault Staircase(VAU_STA)	67
4.3 Railing	71
4.3.1 Segmented Elements By Revit system	71
4.3.2 Normal Baluster and Oblique Baluster	73
4.4 Outcome after all	76
5. Discussion	80
6. Conclusion and Recommendations	82
Bibliography	84
Appendix	93

1. Introduction

1.1 Research Background

Historic Building Information Modeling (HBIM) (Murphy et al., 2009) is an emerging methodology within the broader domain of digital heritage that integrates traditional documentation methods (such as archival research and architectural drawing) with digital techniques including laser scanning, photogrammetry, and parametric modeling. Unlike conventional BIM used for new construction, HBIM must accommodate irregular geometries, data gaps, and historically specific forms and materials. Its goal is not only to model the geometry of heritage architecture, but to embed it with semantic meaning, historical stratification, and spatial logic.

For example, UNESCO (2003) explained in the Charter on the Preservation of Digital Heritage that digital tools are needed to protect cultural resources. ICCROM also emphasizes this idea in its report *The Digital Imperative* (Van Malssen et al., 2021), which described digital preservation as a key step for saving collective memory. More recently, ICOMOS and ICCROM (2023) published guidance for recovery after disasters and conflicts, and they included digital documentation and modeling as part of this process. These documents show that HBIM is not only a technical choice, but also follows international heritage policies. Recent case studies such as the Basilica di Collemaggio (Brumana, Della Torre, et al., 2018), Santa Croce (D'Agostino et al., 2024) in Florence, and Palazzo Pitti (Bonora et al., 2023) in Florence have explained how HBIM can serve as a platform for integrated collaboration, which enables architects, historians, engineers, and conservators to co-develop and manage knowledge about built heritage.

In this context, parametric modeling has become a widely adopted strategy in HBIM workflows, particularly suited to the reconstruction of classical and typologically organized architectural elements. By encoding scale logic, geometric constraints, and hierarchical relationships, tools such as Revit¹ (especially in system families) and Dynamo², the visual programming language based on its API³, enable the generation of structured digital components that reflect formal and construction intent. There are some case studies that proved the feasibility of this approach in modeling historically complex systems—such as vaulted ceilings reconstructed through generative rules and adaptive thickness profiles (Liberotti & Gusella, 2023), or Greco-Roman column assemblies derived from laser-scanned profiles and encoded as reusable parametric families (Elsaid et al., 2019; Scianna et al., 2020).

¹ Revit is a Building Information Modeling (BIM) authoring software developed by Autodesk. It provides tools for creating parametric 3D models combined with non-geometric information, enabling coordinated design, documentation, and analysis across disciplines. Accessed from <https://help.autodesk.com/view/RVT/2026/ENU/>

² “Dynamo is a graphical programming interface that lets you customize your building information workflow. Dynamo is an open source visual programming platform for designers. It is installed as part of Revit along with Revit specific programming nodes..” accessed from https://help.autodesk.com/view/RVT/2026/ENU/?guid=RevitDynamo_Dynamo_for_Revit_html

³ Application programming interface (API) is a connection between computers or between computer programs. Here means the connection between Revit and Dynamo.

These strategies enable not only geometric precision but also semantic clarity, allowing for the reuse, adaptation, and analysis of components across typologically similar architectural contexts.

From a technical standpoint, HBIM relies on frameworks such as the Level of Information Need (LoIN; ISO, 2024) to define the fidelity, purpose, and scalability of models. LoIN addresses the type, amount, and quality of information that should be embedded within a model to meet documentation and management goals in heritage contexts. Rather than focusing solely on geometric precision--what Banfi (2017) refers to as Grades of Generation (GoG), LoIN emphasizes purposeful modeling tailored to analysis, communication, or conservation.

The present study builds upon this tradition by developing a component-based HBIM model of Palazzo Barolo, focused on the staircase and atrium zone. Special attention is given to parametric modeling methods that allow not only geometric reconstruction but also typological classification and semantic enrichment.

1.2 Research problems

Although Building Information Modeling (BIM) has become a standard tool in architectural design and documentation, its native modeling environment often proves insufficient when applied to the geometrical complexity of historic architecture. Elements such as elliptical vaults with irregular boundaries, non-orthogonal junctions between oblique walls, or oblique decorative balusters exceed the representational capabilities of default Revit modeling tools (Brumana et al., 2019; Moyano et al., 2021).

The complexity is further amplified in projects requiring fine control of family nesting, cross-family parameter binding, and adaptive shape logic. In the column system, for example, classical proportional rules require multi-level formula inheritance across nested families. Vault forms--especially those with lunettes or three-centered arches--often fail to conform to Boolean operations in Revit, leading to reliance on Dynamo-generated voids (Medina, 2020). Similarly, oblique balusters demand a geometry that rotates or shears along non-orthogonal planes, which Revit's native family templates cannot support.

In such cases, relying solely on native family types and editing commands leads to rigid approximations, fragmented geometries, or excessive manual workarounds. To address these challenges, this study integrates a visual programming language (VPL) environment--specifically Dynamo--to generate geometry that cannot be directly modeled in Revit. This VPL-based approach enhances the precision, consistency, and adaptability of complex geometric configurations within a heritage modeling context. (Hitech CADD Services, 2019; Marcello, 2022)

Apart from geometric constraints, the three core family types in Revit - system families, loadable families, and local families - cannot directly handle many complex shapes in the user interface. When non-standard families are used to build these geometries, semantic information is often lost. This happens because the chosen family for some specific elements in historical buildings does not match the Revit system family, so parameters cannot be recorded correctly and the data cannot be linked. Many heritage elements--such as vaults with non-rectangular bases, obliquely decorative balusters, or columns with a typical classical proportions--cannot be accurately represented by using existing family templates due to the limitation of software's system. For example, no default family exists for multi-centered vault typologies (and even the normal simple vault), and standard baluster families restrict the rotation and deformation of elements along inclined host surfaces. (Autodesk Community Forums, 2014; Autodesk Help, 2025)

Moreover, managing these irregular forms requires a high degree of parameter control and family nesting, which significantly increases modeling complexity. The definition of shared parameters, the establishment of nested reference geometries, and the transfer of dimensional control across multiple family levels all pose technical challenges, particularly when dealing with non-standard shapes and spatial arrangements. These issues are further compounded when attempting to ensure semantic consistency across different categories of architectural components. (Quattrini et al., 2015)

Beyond geometric and parametric challenges, the encoding of semantic information within BIM environments remains a persistent issue in heritage modeling. While parametric families can define dimensional behavior and geometry, they rarely reflect Typological hierarchies, historical classifications, or functional roles of the architectural elements they represent. For instance, a column family may contain size parameters but fail to convey whether it belongs to a Doric, Ionic, or Composite order, or what its contextual meaning may be.

Taken together, these issues point to a methodological gap: there is a lack of integrated strategies that allow for the coordinated management of geometry, parameter logic, and semantic interpretation in the modeling of heritage architecture. This gap defines the core technical and conceptual challenge that the present study seeks to address.

1.3 Research objectives

The main objective of this study is to construct a parametric modeling workflow capable of addressing the geometric and Typological complexity of selected heritage components from Palazzo Barolo, through the integration of BIM tools--specifically, a Revit- and Dynamo-based workflow focused on classical architectural systems.

This study specifically aims to:

Construct structured parametric families for columns, vaults, and balustrades, based on Typological decomposition and nested relationships;

Apply proportion-driven modeling strategies informed by historical architectural treatises;

Develop a reusable and transferable logic applicable to heritage contexts with comparable architectural typologies.

By approaching these aims, the research proposes a modeling framework that ensures the accurate reconstruction of complex geometries, while at the same time supporting their semantic interpretation and enabling future re-usability.

1.4 Scope of the research

This research focuses on the parametric reconstruction of selected architectural components in Palazzo Barolo, namely the columns, vault system, and decorative balustrades. These elements were selected for their representative geometric complexity and their structural and ornamental significance within the palace's circulation scheme. The study excludes analysis of internal material layers, structural behavior simulations, or building services (mechanical/electrical systems). Instead, it adopts the Level of Information Need (LoIN) framework defined in ISO 7817-1:2024 to define the granularity and accuracy of the digital models. The modeling is based on point cloud data and orthophoto projections and limited to external and interior surface geometry. The primary objective is to establish a reusable and semantically structured parametric modeling workflow.

This method is oriented to heritage modeling practitioners, architectural heritage scholar, and digital archiving professionals. It aims to deliver parametric modeling that are structurally coherent semantically interpretative, and procedural sustainable thereby supporting future analysis, communication and long-term conservation efforts.

1.5 Case study: Palazzo barolo

Palazzo Barolo stands at the corner of Via delle Orfane and Via Corte d'Appello in Turin. It is considered an important example of Baroque and Rococo architecture linked with the aristocracy of the city. The palace was first built in the late seventeenth century, when Count Ottavio Provana di Druent, a high court officer under Vittorio Amedeo II of Savoy, transformed an earlier residence into a new monumental house. The commission was assigned to Gian Francesco Baroncelli, who reorganized the residence with a monumental layout. His design placed the atrium at the center and introduced a forked staircase directly connected to it. This spatial choice was unconventional for the late seventeenth century, since staircases were usually set along the side wings, and it marked the atrium as the true focus of the palace. (Fenoglio, 1928)⁴

⁴ Giulio Fenoglio, "Il Palazzo dei Marchesi di Barolo," Torino: Rivista mensile municipale, vol. VIII (1928): 164 – 171, accessed July 16, 2025, <https://www.museotorino.it/resources/pdf/books/528/files/assets/common/downloads/page0164.pdf>.

The building was inherited by the Falletti di Barolo family in the early 18th century. In 1727, under Ottavio Giuseppe Falletti, the interiors were redesigned by Benedetto Alfieri to reflect the Rococo taste of the time. The palace remained the family's residence until the extinction of the lineage with the death of Giulia Falletti di Barolo in 1864. According to her will, the property was donated to the Opera Pia Barolo, a charitable foundation she had established. Today, restoration initiatives supported by the Compagnia di San Paolo and the Fondazione CRT have opened the historical apartments of the palace to the public.⁵

This layered architectural evolution, spanning over two centuries, makes Palazzo Barolo an exemplary case for heritage-oriented parametric reconstruction. The palace shows a mix of spatial complexity, unusual construction rules, and stylistic changes from different periods. These features make it a suitable case for applying and testing advanced HBIM methods. Beyond its formal and historical significance, Palazzo Barolo also holds a unique position within the urban fabric of Turin. The building's axial organization, central monumental staircase, and sequential vaulted ceilings reflect a spatial strategy that merges aristocratic representation with urban permeability. These characteristics—while widely discussed in historical literature—have rarely been analyzed through the lens of digital reconstruction. As such, this project does not only serve as a technical modeling case but also as a cultural and spatial interpretation of Baroque heritage through contemporary tools.

1.6 Structure of the thesis

This thesis is organized into six chapters and an appendix, moving from the background and theoretical framework toward the modeling implementation and the broader implications of the research.

Chapter 1 introduces the background of the study, outlining the HBIM research context, the main problems addressed, and the objectives. It also explains the methodological approach adopted in this work.

Chapter 2 presents a literature review that situates the research within current scholarship. It discusses the development of HBIM, with a focus on parametric modeling strategies applied to heritage case studies, and examines frameworks such as the Level of Information Need (LoIN), which inform decisions about modeling depth and information management.

Chapter 3 defines the methodology. It explains how different strategies—geometric, parametric, and hybrid approaches—were considered, and sets out the rationale for choosing void-based parametric families as the main modeling framework. It also introduces the specific tools (Revit, Dynamo, Rhino) and explains how they are combined in the workflow.

Chapter 4 illustrates the implementation. This chapter applies the methodology to the selected case study components, describing the modeling steps in detail and presenting the resulting parametric families for vaults, columns, and balusters.

Chapter 5 provides the discussion. It evaluates the strengths and weaknesses of the adopted workflow, compares the outcomes with existing approaches in HBIM

⁵ Opera Barolo. (n.d.). Il Palazzo. Accessed from July 16, 2025, from <https://www.operabarolo.it/palazzo-barolo/>

research, and considers the potential re-usability of the method in other heritage contexts.

Chapter 6 presents the conclusions. It reflects on the research objectives and summarizes how they were achieved. The chapter also identifies the contributions of the study and highlights possible directions for further investigation.

Chapter 7 contains the bibliography and references, gathering all the sources cited throughout the thesis.

Overall, the structure is designed to guide the reader from the general background of HBIM to the specific modeling solutions tested in this study, and ultimately to the evaluation of their broader significance.

1.7 Acknowledge

The research data and digital acquisition were part of an ongoing project, directed by Roberta Spallone and Marco Vitali, which is currently going on with the precious collaboration of Prof. Concepción López González (who joined the group thanks to the international collaboration project “Nuevas tecnologías para el análisis y conservación del patrimonio arquitectónico”, funded by the Ministry of Science, Innovation and the University of Spain)

2. Literature Review

2.1 Informative and geometrical modeling

The traditional notions of Level of Development (LOD) and Level of Information (LOI) originate respectively from the AIA E202 BIM Protocol (American Institute of Architects, 2008) and the British PAS 1192-2:2013 specification (British Standards Institution, 2013), distinguishing between geometric completeness and metadata richness. Within HBIM scholarship, the additional concept of Level of Geometry (LOG) was introduced to address the geometric variability and modeling challenges specific to heritage workflows. The need to manage non-standardized data-including various inputs such as laser scans, orthophotos, archival drawings, and descriptive reports -- led to the formal proposal of LOG within an extended LOD framework (Lombardini & Cantini, 2017). LOG in this context functions to differentiate modeling strategies based on data quality, reconstruction method, and intended fidelity. This structure was subsequently formalized by Brumana et al. (2019), who aligned LOG thresholds (LOG 300, 400, 500) with the broader framework of Grades of Generation (GOG) and Grades of Accuracy (GOA), facilitating clearer typological segmentation and reconstruction strategies. More recently, the concept of Level of Information Need (LoIN), as defined in ISO 7817-1:2024, consolidated these dimensions into a unified, purpose-driven framework that links modeling depth to documentation intent, delivery phase, and actor roles (Bonora et al., 2023; ISO 7817-1, 2024).

Semantic segmentation in HBIM is not merely a geometric decomposition process, but a semantically driven structuring operation that precedes modeling decisions. It directly aligns with the Level of Information Need (LoIN) framework defined in ISO 7817-1:2024, which organizes modeling requirements through six contextual dimensions: why (purpose), who (actors), what (objects), how (methods), when (phases), and where (delivery environments) (ISO 7817-1, 2024). Within this structure, segmentation becomes a way of filtering raw spatial data into functionally or typologically meaningful parts—columns, vaults, lunettes, or balusters—before any geometric modeling occurs (Brumana, Della Torre, et al., 2018; Quattrini et al., 2015). Empirical studies have demonstrated that segmentation decisions are tightly coupled with the selection of modeling strategies. Zhao et al. (2024) developed a weakly supervised model (SQN-DLA, Semantic Query Network based on Dual Local Attention) for semantically segmenting heritage building point clouds, providing classification structures that directly informed HBIM component generation. Giuliani et al. (2024) proposed stratified segmentation of masonry structures using image analysis, guiding how individual layers were modeled based on semantic hierarchy. Zhao et al. (2023) provided a systematic review of semantic segmentation techniques for cultural heritage point clouds, reinforcing their foundational role in heritage-oriented modeling pipelines. Together, these studies support the idea that segmentation, when driven by LoIN logic, serves not as neutral data preparation but

as a strategic act that determines granularity, typological integrity, and the selection between mesh-based or parametric modeling methods.

2.2 From Segmentation to Modeling Strategies

2.2.1 Semantic segmentation in HBIM

In HBIM, semantic segmentation is not equivalent to geometric modeling; rather, it serves as a crucial interpretive layer that precedes the creation of any parametric families. It breaks down heritage buildings into meaningful architectural components such as columns, vaults, walls, and decorative elements, based not merely on geometry, but on typological, functional, and morphological reasoning (Parisi et al., 2019). Werbrouck et al. (2020) emphasize that segmentation constitutes a preliminary organizational logic that informs model structure before any geometric generation. This process facilitates the structuring of parametric libraries in software environments like Revit, where segmented elements can be transformed into reusable families with coherent naming rules, parameter hierarchies, and semantic grouping.

For instance, Werbrouck et al. (2020) define segmentation as a central step in their scan-to-graph framework, translating raw point cloud data into semantically structured graphs that directly inform HBIM object generation. Similarly, Parisi et al. (2019) demonstrate how semantic partitions can be used to organize modeling logic by coupling parameters with classification taxonomies. Croce et al. (2021) propose a semiautomatic segmentation-to-HBIM pipeline in which architectural typologies drive object classification and model enrichment. Sanseverino et al. (2022) further integrate point cloud segmentation with orthophoto analysis to generate accurate, georeferenced parametric families. Moyano et al. (2022) note that existing classification standards such as Omniclass, Unifomat II, and Uniclass 2015 are not fully compatible with heritage architecture. Their workflow includes restructuring classification layers and adapting semantic codes based on national vocabularies, with mappings extended to IFC categories.

2.2.2 Geometric and parametric approaches in HBIM application

The distinction between geometric and parametric approaches in HBIM first appears implicitly in studies focusing on modeling complexity. Brumana et al. (2018) used the phrase “geometric complexity” to describe the modeling of vaults and lunettes in the Basilica di Collemaggio, where shape irregularity prevented the use of reusable parametric types. In contrast, Banfi (2020) explicitly contrasted “geometric modeling,” based on mesh or NURBS surfaces, with “parametric modeling,” which relies on rule-based libraries and repeatable logic. Banfi (2019) further associated geometric approaches with workflows driven by morphological accuracy, using tools such as Rhino for direct modeling, especially when typological consistency could not be assumed. On the other hand, Quattrini et al. (2015) defined parametric modeling in HBIM as a process of encoding semantic and typological relationships into predefined object families. Aubin (2013) exemplified this logic in Renaissance-style

components, where classical rules guide the creation of column families with nested parameters.

In HBIM, a geometric-driven approach is often applied when dealing with architectural elements that exhibit complex or irregular geometry not suited for rule-based modeling. Allegra et al. (2020) describe how unique elements at the Castle of Maredolce, such as the pleated pavilion of the Aula Regia, were modeled in Rhinoceros and imported into Revit as generic families, bypassing parametric definitions due to their uniqueness and morphological complexity. Similarly, Costantino et al. (2023) reconstructed the church of San Nicola in Montedoro using Rhino and Grasshopper by slicing point clouds into geometric sections, then building NURBS surfaces without embedding them in native Revit family logic. Another important case is presented by Diara and Rinaudo (2019), who reconstructed a rib vault based on LiDAR scans using manual loft and patch modeling in Rhino. Although the final geometry was later exported to BIM-compatible formats solid via STEP through platforms such as FreeCAD, the initial process was fully morphology-driven and not based on parametric objects. Yang et al. (2019) also report a mesh-to-HBIM strategy at the St-Pierre-le-Jeune Church, where triangulated surfaces were thickened and imported into BIM environments without typological structuring or parameterized reuse. These examples demonstrate that geometric modeling workflows--often described as “morphology-driven,” “scan-to-mesh,” or “non-parametric”--serve as critical alternatives to parametric logic when dealing with heritage structures whose geometry cannot be generalized.

Parametric modeling in HBIM not only allows for geometric control through dimensions and formulas, but also supports the creation of structured libraries of reusable components. These libraries encode typological rules, geometric variations, and placement logic, serving as the operational core of reusability in HBIM workflows. In ArchiCAD, ArchiRADAR provides GDL-based parametric vault objects such as the volta a doppia stella, where users can adjust curvature, dimensions, and materials within a typology-aware family structure (ArchiRADAR, n.d.). Edificius offers a built-in library of vaulted components, allowing users to select from various typologies—such as barrel, sail, pavilion, and polygonal cross vaults—and adjust key parameters directly within the BIM environment, without external modeling tools (ACCA Software, n.d.). In contrast, Autodesk Revit lacks predefined vault families, forcing users to create custom parametric families or repurpose system components. Medina (2020) modeled the ground-floor groin vaults of Notre-Dame by adapting the “roof” family system, illustrating a functional parametric substitution. Capone et al. (2019) applied rule-based generation in Grasshopper to build adaptive rib vault families with trapezoidal bases, linking ideal geometric logic with real-world deformation. Similarly, Calvano et al. (2023) used Grasshopper to generate mass families from B-Rep geometry and applied “roof by instance” mappings to incorporate them as parameter-controlled vaults in Revit.

A crucial contribution is offered by Croce et al. (2021), who proposed a semi-automatic workflow from semantic point cloud segmentation to the instantiation of predefined HBIM families. Their method uses supervised classification and geometry–type matching to populate parametric libraries, reinforcing the idea that the parametric system serves both as a modeling strategy and a tool for semantic structuring. These studies also underline the essential role of libraries—not merely as repositories of components, but as the fundamental mechanism enabling typological consistency and re-usability. In Natta’s (2024) doctoral research, this logic is extended through the integration of Rhino.Inside.Revit and Human UI. A customized interface within Revit allows users to select from predefined vault typologies and input geometric parameters, facilitating more structured control and management of vaulted elements during modeling.

In HBIM studies, geometric and parametric modeling are not treated as opposite choices but are often used together. The decision usually depends on the kind of architectural element. Irregular and unique parts are better handled with geometry-based methods, while regular and typologically stable ones are more suitable for parametric tools. Both rely on segmentation, yet their aims are different: geometry seeks to rebuild formal complexity, while parametric modeling focuses more on typological order and reuse.

3. Methodology

3.1 Strategies comparison: Geometric vs. Parametric

In HBIM (Historic Building Information Modeling), there are two main modeling strategies: the Geometrical approach and the Parametric approach. These two methods differ from how the geometry is generated and managed (see Table 3-1). This section refers to Banfi's (2017) concept of Grade of Generation (GoG), which clarifies how the geometric approach is generated. Geometrical Approach: This method generates shapes based on point clouds. Geometry may be reconstructed by slicing or extracting boundaries from point cloud-derived meshes, or by producing non-parametric curves after typological comparison. It focuses on geometric accuracy, but usually lacks parametric control. The GoG can range from 3—where simplified or abstracted curves are transformed through extrude, sweep, or revolve operations to 10—where mesh boundaries or slices extracted from point data are re-lofted to rebuild complex geometry. Parametric Approach: This method uses tools like Revit, Dynamo, or Grasshopper to build models using parameters, formulas, and algorithmic logic. It allows for easier editing, reuse, and integration of semantic information (when it was in the same editing environment). The GoG typically ranges from 4 to 8.

Table 3-1 Difference between Geometric approach and Parametric approach

Aspect	Geometrical Approach	Parametric Approach
GoG Level	3–10	4–8
Generation method	Curve fitting, surface modeling	Parameter formulas, algorithmic scripts
Editability	Low (requires manual changes)	High (controlled by parameters)
Reusability	Low	High
Best suited for	Freeform, historically deformed elements	Modular, rule-based architectural elements
BIM interoperability	Weak (geometry only)	Strong (semantic and parametric data)
Information modeling	Added manually	Embedded during modeling
Examples	Scan-to-Mesh, Rhino NURBS, Boolean operations	Dynamo-based vaults, adaptive Revit families

For Types of geometrical modeling: Manual profile extraction is a traditional approach where basic profiles (e.g., vertical sections) are interpreted from point cloud data. These profiles are often semantically classified in advance and then used to build simple 3D forms such as revolved solids. According to Banfi's GoG classification, this method corresponds to low GoG levels (e.g., GoG 3), because although it results in a 3D object, it lacks parametric control and represents only formal characteristics without internal logic.

Scan-to-Mesh refers to the generation of high-resolution 3D meshes directly from point clouds, often using tools such as Autodesk ReCap. These meshes are especially valuable in the reconstruction of damaged, deformed, or richly detailed surfaces. According to Banfi (2017), when these mesh models are further processed through NURBS fitting-such as reconstructing surface edges or slicing through sections for geometric logic-they may correspond to GoG 10.

These models are not parametric but offer very high levels of geometric fidelity and are especially suited for visualization, simulation, or structural analysis workflows.

Parametric modeling strategies vary depending on the platform and complexity of the architectural element. In the context of HBIM, parametric modeling is especially valuable for systematically controlling geometry, adapting to Typological variations, and embedding information. The following Table 3-2 presents key categories of parametric modeling approaches used in heritage-related applications:

Table 3-2 Parametric modeling approaches used in heritage-related applications

Method	Platform	Features	Application
GDL ⁶ object libraries	ArchiCAD with GDL scripting	Parametric objects with adjustable dimensions	Standard vault families
System family modeling	Edificius	Built-in tools to modify curvature, thickness, materials	Fast modeling of simple vaults
Manual family with voids	Revit	Manual creation of void geometries for vaulted forms	Custom complex vaults
Visual Programming (VPL)	Dynamo, Grasshopper + Rhino.Inside	Algorithmic control using parameters and logic rules	Irregular vaults, adaptive geometry, typological variation
Main Benefits: Fully compatible with BIM standards (e.g., IFC, LOIN); High reusability and flexibility; Supports Scan-to-BIM workflows with semantic enrichment.			

This study focuses exclusively on the parametric approach, recognizing that geometrical and parametric modeling follow fundamentally different logics, offer distinct levels of reusability, and serve different purposes. Parametric modeling is chosen specifically for its ability to support the reuse of modeling processes and the reuse of geometry after typological or semantic analysis. Rather than integrating different modeling logics, this approach prioritizes the consistency and adaptability of a parameter-driven workflow that aligns with both analytical reconstruction and BIM-oriented information management.

Based on the comparative analysis above, this study adopts a parametric modeling strategy as its exclusive approach. The following sections detail how this strategy was applied to selected architectural components from Palazzo Barolo.

⁶ “GDL(Geometric description language) is a parametric programming language,...GDL provides solutions for creating rules for the object’s parameters and creating graphical UI for the elements. These objects are called library parts.” (Graphisoft, n.d., para. 1). Accessed August 22, 2025, from <https://gdl.graphisoft.com/reference-guide/gdl-definition/>

3.2 Modeling framework

The parametric modeling in this study is guided by three main objectives. First, the models aim to remain faithful to the geometry of the original architecture. Second, they are structured to allow parametric control, making them adaptable for reuse and for handling typological variations. Third, whenever possible, semantic information is embedded into the digital components. These goals follow the logic of the Level of Information Need (LoIN) framework (ISO 7817-1:2024), which defines how much detail and what kind of information should be included in the model, depending on the documentation and analysis requirements, and "one purpose of defining the level of information need is to prevent delivery of too much information". (ISO 7817-1:2024, p.2)

Prerequisites						Level of information need		
Why	When	Who		What		How		
Purpose	Information delivery milestone	Actor (information receiver)	Actor (information provider)	Object	Breakdown structure			
						Geometrical information	Detail	
							Dimensionality	
							Location	
							Appearance	
							Parametric behaviour	
						Alphanumerical information	Identification	
							Information content	
						Documentation	Set of documents	

Figure 3-1 Information need table for this study (reproduced from ISO 7817-1:2024, Table B.3)

To contextualize the modeling goals within a standardized information framework, this study adopts the Level of Information Need (LoIN) structure proposed by ISO 7817-1:2024.

Table 3-2 translates this structure into the specific requirements of this project, defining the expected purpose, object granularity, geometric detail, and alphanumerical scope for the HBIM components modeled.

Although no formal shared parameters were implemented, a consistent naming convention was employed to encode typological and procedural logic across families. This lightweight strategy ensures a degree of semantic traceability, even in the absence of embedded metadata.

This LoIN interpretation also clarifies the modeling boundaries adopted in this study-focusing on external and structural geometry, excluding materials or

behavioral simulations, and emphasizing reuse and adaptability through parametric control.

Table 3 Level of Information Need (LoIN) applied to component-based parametric modeling in Palazzo Barolo

Category	Content
Purpose(why)	Reconstruction and semantic enrichment of classical architectural components in HBIM
Delivery Stage(when)	Post-survey modeling phase
Actors(who)	Developed by author, Intend for: Heritage scholars, HBIM practitioners, digital archiving professionals
Object(what)	Vault/ Column/ Balustrade
Geometric information(How)	Detail: Medium Dimensionality: 3D Location: Accurate Apperance: Geometric appearance only; no texture or material properties assigned(which could be in the further work) Parametric behaviour: High.
Alphanumerical info	No embedded parameters were used. However, family and type names were encoded systematically to reflect typology, construction logic, and model variants (e.g.,COL_ATR, VAU_LUN_H1).
Documentation	Point cloud, orthophoto, historical references (treatises and archive drawings)

In order to implement this strategy effectively, each architectural component is evaluated according to its typological characteristics, geometric regularity, and level of integration within the building system. These criteria inform both the selection of modeling methods (e.g., nested families, void-based systems, adaptive components) and the assignment of shared parameters.

The toolset used throughout this process primarily consists of Autodesk Revit, enhanced by Dynamo for parametric generation and custom logic implementation. In certain cases involving complex deformations or oblique geometries, Rhino and Grasshopper are employed for their advanced modeling flexibility and geometry transformations.

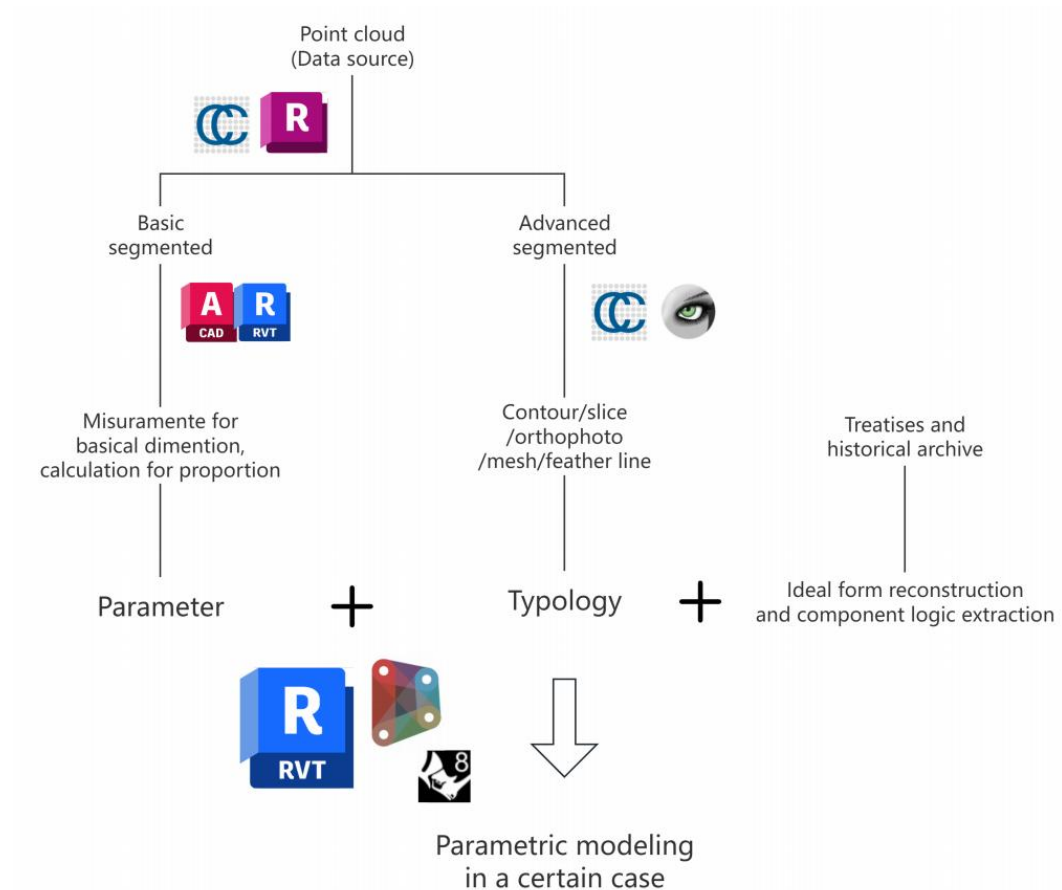


Figure 3-2 Workflow of parametric modeling in this study

The diagram above (Figure 3-2) illustrates the core modeling framework adopted in this study, which combines segmented point cloud data with typological knowledge and dimensional parameters to enable accurate and reusable parametric reconstruction of heritage components. As mentioned in 1.7, the point cloud and digital acquisitions used in this project were provided within the ongoing research project, directed by Roberta Spallone and Marco Vitali, with the collaboration of Prof. Concepción López González.

The process begins with the segmentation of the point cloud model. A basic segmentation, typically performed in software such as ReCap and CloudCompare, divides the dataset according to architectural components—columns, vaults, balustrades—and allows for dimensional extraction using CAD tools like Revit or AutoCAD. These measurements serve as the basis for defining proportional parameters, including heights, diameters, or curvature radii.

In parallel, a more advanced segmentation is conducted. This includes the generation of orthophotos, extraction of contour lines or feather curves, and surface reconstruction via tools such as CloudCompare or MeshLab. This layer of analysis enables the classification of geometric typologies, referencing formal logic found in historical construction treatises and architectural manuals.

The integration of these two sources—parameter control from basic measurements and typological logic from advanced geometric interpretation—is further supported

by historical documentation, which informs idealized component structures and semantic rules (e.g., column orders, vault classifications). By synthesizing these three axes (Parameter + Typology + Historical Logic), the modeling process achieves both geometric precision and semantic fidelity.

This multi-source strategy is then implemented using Revit, Dynamo, and Rhino, each serving different functions depending on the component's complexity. The result is a consistent, component-based HBIM model that reflects not only the surveyed geometry but also the historical rationale behind architectural forms.

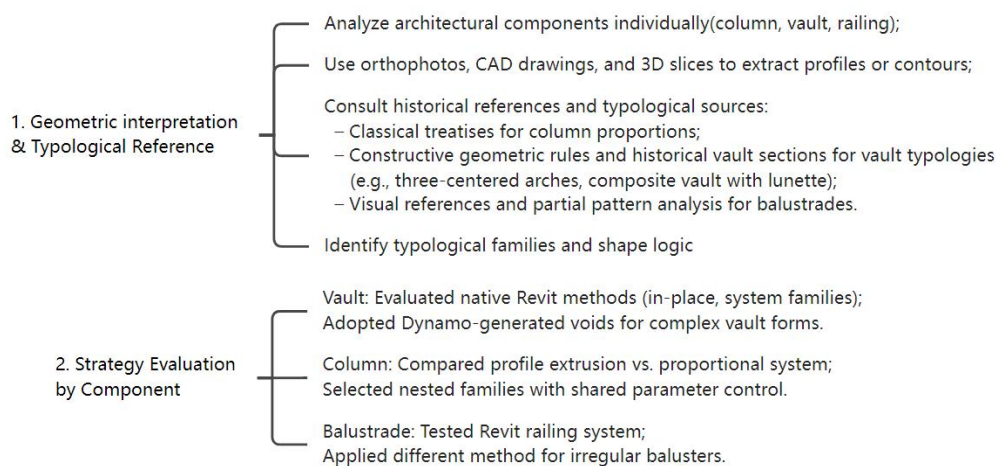


Figure 3-3 Component-based modeling strategy: geometric interpretation and method evaluation

Based on the integrated framework described above, the modeling process proceeds through two complementary stages. The first stage focuses on the geometric reading and typological classification of each architectural component, using both digital documentation (e.g., orthophotos and 3D slices) and historical references (e.g., treatises on proportions and vault construction). This phase ensures that each element is understood not merely as geometry, but as a rule-based architectural type.

The second stage involves a component-specific evaluation of modeling strategies, testing and comparing different Revit and Dynamo-based methods according to each element's typological and geometrical complexity. This comparative approach allows the selection of appropriate techniques for each family—whether nested parameterization, geometry-driven voids, or adapted railing workflows—ensuring both modeling accuracy and semantic traceability.

The following sections detail how these strategies were applied to three key elements—columns, vaults, and balustrades—through parametric workflows developed in Revit and Dynamo.

3.3 Column

3.3.1 Method Evaluation and logic of proportional strategy

In this study, the column modelling method adopts the strategy of “*Primary–Proportional Control Strategy*”. A global parameter serves as a control variable for all the dimensions and is consistently applied across nested families. This strategy enables the structure assembly of the components and establishment of dimensional linkages within a nested family system in BIM software.

This method differs from directly *tracing section profile*⁷ extracted from the point cloud for modeling. The advantages of this method include the capacity to partially reconstruct the construction logic and proportional system among column components in historical architecture, while also re-usability of family files in future projects.

The proportional system used in this study is mainly based on the analysis and construction rules described by “The classical Orders of Architecture”(Chitham, 2005). In that book, each order, such as Doric, Ionic, or Composite, has a relatively stable set of proportions (Figure 3-4 left). For example there are standards for the ratio between the column height and the base diameter, the height ratios of the base and plinth, and the proportions between the lines of the capital moldings. These classical proportions have developed over history to form the basic logic of column components, and they provide a measurable reference for parametric modeling.

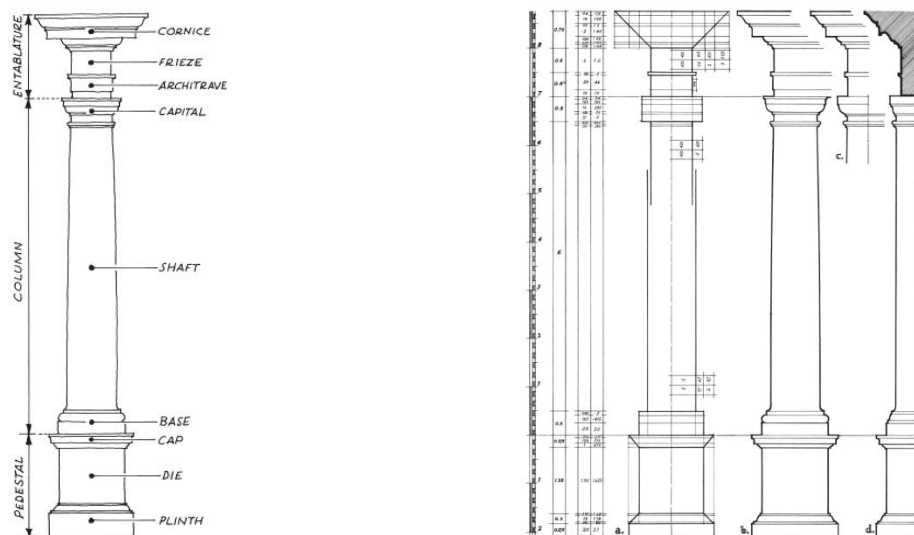


Figure 3-4 Main parts of the order, reproduced from Chitham (2005, p.209 left, p.57 right)

⁷ A 2D outline extracted from the point cloud, often representing a vertical or horizontal cut through the object

In the actual modeling process, this study refers to these historical standards and selects the *Base Diameter*⁸ as the main control parameter for the entire column. From this parameter, all the other dimensions of the base, shaft, capital, and the associated molding lines are calculated. For example (see Figure 3-4 right), the capital height is set as $\text{Base Diameter} * 0.5$, the shaft height as $\text{Base} * 6$, and the base height as $\text{Base} * 0.5$, resulting in a total column height of $\text{Base} * 7$. All these dimensions could be linked via Revit's formula system within the main family and progressively associated in nested families.

The choice of the Base Diameter as the key global parameter follows both traditional theory and practical experience in digital modeling. In the study of classical orders, Chitham(2005, p. 23) explains that all proportions are set out as decimal parts of the column's base diameter. This rule makes the calculation process more straight forward and ensures consistent scaling across different components. A similar logic is later applied in digital workflows. Aubin and Milburn(2013, p. 39), for example, adopt the Base Diameter as the reference parameter in "Renaissance Revit", where it drives the proportional control of the shaft, base, and capital inside nested families.

Although the reference book provides a systematic model of classical column orders, the columns in this case belong to a baroque-period historical building and do not strictly follow classical standards. Therefore, during the early modeling stage, the proportional parameter system needed to be adjusted using measurements from the point cloud and orthophoto⁹ generated via the FARO As-Built plugin, which converts point cloud data into geometrically corrected 2D projection images. Unless otherwise specified, the term orthophoto in this thesis refers exclusively to this type of architectural image. This made it possible to balance the "historical standards" with the actual measurements of the building.

In contrast, the tracing section profile method(Figure 3-5) from the point cloud can reproduce the current shape more directly, but it lacks structural logic, which cannot be reused, and it is hard to control with unified parameter. This makes it unsuitable for large-scale or systematic family construction.

⁸ Base Diameter: the column diameter at its base. In this project, it is used as a global parameter to govern the nested families in correct proportion.

⁹ Orthophoto refers here to a 2D projection image extracted from point cloud data using FARO As-Built. This term will be used consistently throughout the thesis to indicate such projection views, unless otherwise specified.



Figure 3-5 Tracing section profile from column in project

Therefore, the Primary – Proportional Control Strategy is preferred in this study, as a comprehensive strategy to achieve both geometric restoration, construction logic, and family system management.

At the beginning of the project, an attempt was also made to build the columns by tracing directly from orthophotos or point cloud sections, in order to quickly get the shape and height information. However, this method showed limitations clearly in practice, especially when dealing with local deformations and errors that are extremely common in historical buildings as mentioned before. For example, due to ground settlement, structural loads, or later construction changes, columns of the same type might have a height difference in actual measurements. Eventhough, during the modeling stage, the reference level couldn't be change or even slightly adjusted, which made correction more difficult.

If the whole column shape is generated by direct tracing, it is very difficult to adjust the reference planes or unify the proportional logic later. This can easily lead to misalignment between parts or confusion in the structural logic. In contrast, the proportion-based modeling method allows the whole column to be divided into several sub-elements and controlled by a unified global parameter.

Even if there is a need to change the height of some parts, it can be corrected consistently by changing local parameter formulas or moving the reference plane. This makes the system more flexible and adaptive. Therefore, although direct tracing can quickly create a basic mass at the beginning, its limitations do not meet the needs for later fine control and batch copying.

It should be noted that the proportion-based modeling strategy is mainly applied to column parts with higher geometric regularity and more unified types, such as the base, shaft, and the capital. Within the scope of this study, the more complex decorative components, such as floral ornaments or carved patterns on the capital, were not rebuilt using proportional methods. These decorative parts usually show strong individual differences and handmade features, which make them more suitable for reconstruction through mesh or direct modeling. Although it is possible to build these elements as parametric families In Revit, this study doesn't focus on them because of their complexity.

*Table 3-4 Comparison between
Primary-Proportional Control Strategy and Direct Tracing Method for Column Modeling*

Criteria	Primary – Proportional Control Strategy	Direct Tracing from Section Profile
Modeling logic	Based on predefined proportions linked to a global control parameter (Base Diameter)	Based on tracing 2D profiles from orthophotos or point cloud slices
Control system	Formula-driven dimensional control via nested families and global parameters	Manual shape tracing without consistent parameterization
Geometric consistency	High: components scale proportionally and stay aligned	Low: prone to misalignment due to manual segmentation or irregular shapes
Flexibility and adaptability	High: local parts can be adjusted without breaking the global logic	Low: hard to modify once geometry is fixed
Re-usability	High: parametric families can be reused in other projects	Low: model is case-specific and hard to generalize
Suitability for historical deformation	Allows correction through proportional adjustments and reference plane shifting	Sensitive to irregularities and deformation in scan data
Application scope	Ideal for components with repetitive geometry (e.g., base, shaft, capital)	Better for unique or highly ornamental features
Efficiency in batch modeling	Supports systematic family creation and control	Manual and time-consuming for large-scale components
Limitations	Less effective for free-form, highly detailed ornament	Lacks construction logic; poor scalability
Software support	Requires careful formula setup and family management in BIM software	Simpler in geometry tools, but less robust in BIM

A comparative summary of the two modeling methods is presented in Table 3-4. As shown, the Primary--Proportional Control Strategy offers greater consistency, flexibility, and re-usability for systematic reconstruction, while the Direct Tracing

method may be useful for certain decorative exceptions but is generally unsuitable for structured parametric workflows.

3.3.2 Proportional Logic and Nested Family Strategy for Columns

In this study, the column modeling strategy refers to the classical principles summarized by Chitham (2005), with a clear separation of the base, shaft, and capital and pedestal parts. These elements are organized using a nested family structure inside the Revit system family of columns. The overall proportions of the column are guided by classical standards, which provide reliable rules for the height and width relationships among the three main elements. These references help maintain architectural consistency and create a measurable, parametric framework in the BIM environment. This approach also supports better visualization and easier editing, because each part of the column can be managed separately but still stays connected to the main parameter.

In Revit, the shaft is usually placed with its elevation controlled by an offset from the base reference level. The base and capital are then linked to the shaft through nested families, sharing the same main control parameter, for example the base diameter. All size changes are passed through Revit's formula system, so the nested elements can stay consistent. This nested strategy makes it possible to adapt the column to different floor heights, levels, or structural changes without rebuilding the entire model.

Figure 3-6 redrawn from Chitham (2005) can be included here to show the typical proportion logic and construction of classical columns, and to illustrate how these parts are associated in the nested family system. These images help to explain which parameters are connected, and how the reference planes support the vertical stacking of the elements. In addition, the figures can make clear which geometric references and constraints are needed to align the base, shaft, and capital correctly in 3D space. As a result, this method provides a practical solution to combine traditional proportional rules with flexible parametric modeling in Revit.

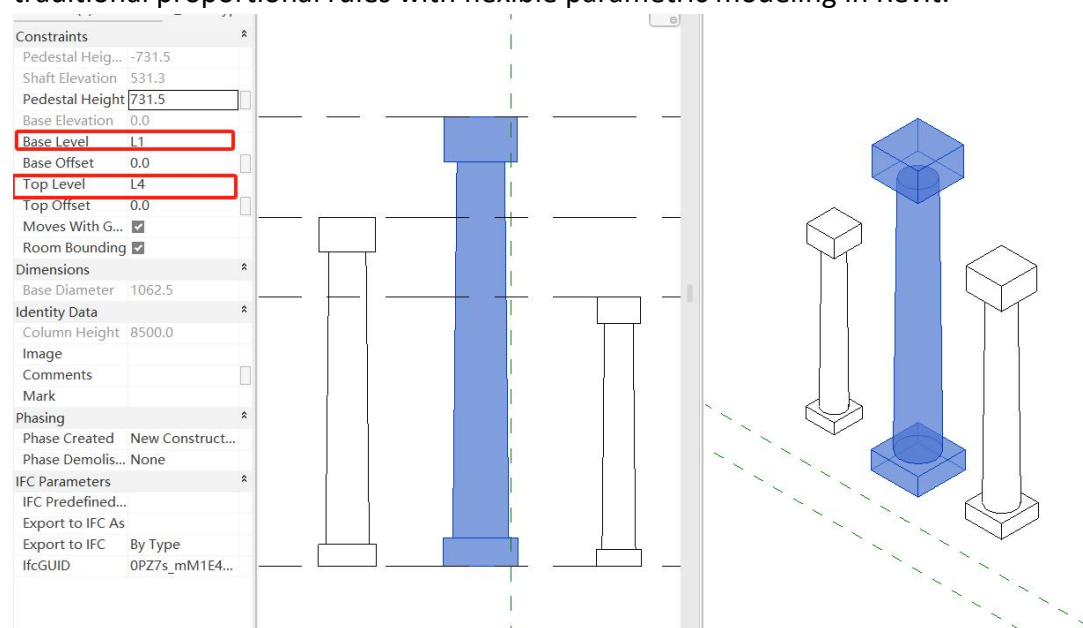


Figure 3-6 Schema of the column, redrawn and reproduced from Renaissance Revit (Aubin & Milburn, 2013, Figure 3.25), modified to reflect the parameter structure used in this study.

3.3.3 Parameters Transformation in Nested family

In this study, the complete modeling process proposed in the book "Renaissance Revit (2013)" was adopted for the column modeling. This method, as a parametric tutorial for modeling classical architectural columns, provides clear structural decomposition, proportion control, nested family organization, and parameter management strategies. It was fully absorbed and applied in the reconstruction of the columns of Palazzo Barolo.

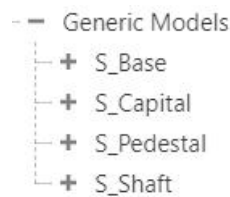


Figure 3-7 Family structure tree diagram (Column main family nested Capital/Shaft/Base/Pedestal)

Within this method framework, the columns are decomposed into four logical components: Capital, Shaft, Base, and Pedestal, and each is built as an independent Generic Model family. These families are then nested to form the main family. The core logic is to select Base Diameter as the unified control parameter, thereby deriving the total height of the column and the dimensions of each sub-component at a fixed ratio. This choice is not only because Base Diameter is the "modular unit" in column construction, but also because in Revit, the total height of the column is controlled by Base Level and Top Level, so the elevation is determined by the system parameters, while the diameter can serve as a stable core for the construction ratio.

To transfer parameters between the host family and nested components, this study uses the Associate Family Parameter¹⁰ function in Revit. This operation allows a parameter -- such as the Base Diameter -- defined in the parent family to be bound to corresponding parameters in nested sub-families. Without this association, nested elements retain default dimensions and do not scale with the host, breaking the parametric logic, even the modeler has already input the formula among the needed parameters. Otherwise, if a certain component is not bound, it will remain in its original size after loading and cannot automatically scale with the parameters, resulting in incorrect proportions, geometric misalignment, and even errors in the system when the project is loaded(see Figure 3-8).

¹⁰ "In Revit, the Associate Family Parameter function allows parameters in a host family to be linked with those of nested families, so that type or instance values can be controlled from the project environment. Parameters must be of the same type to be associated (Autodesk, 2022)."

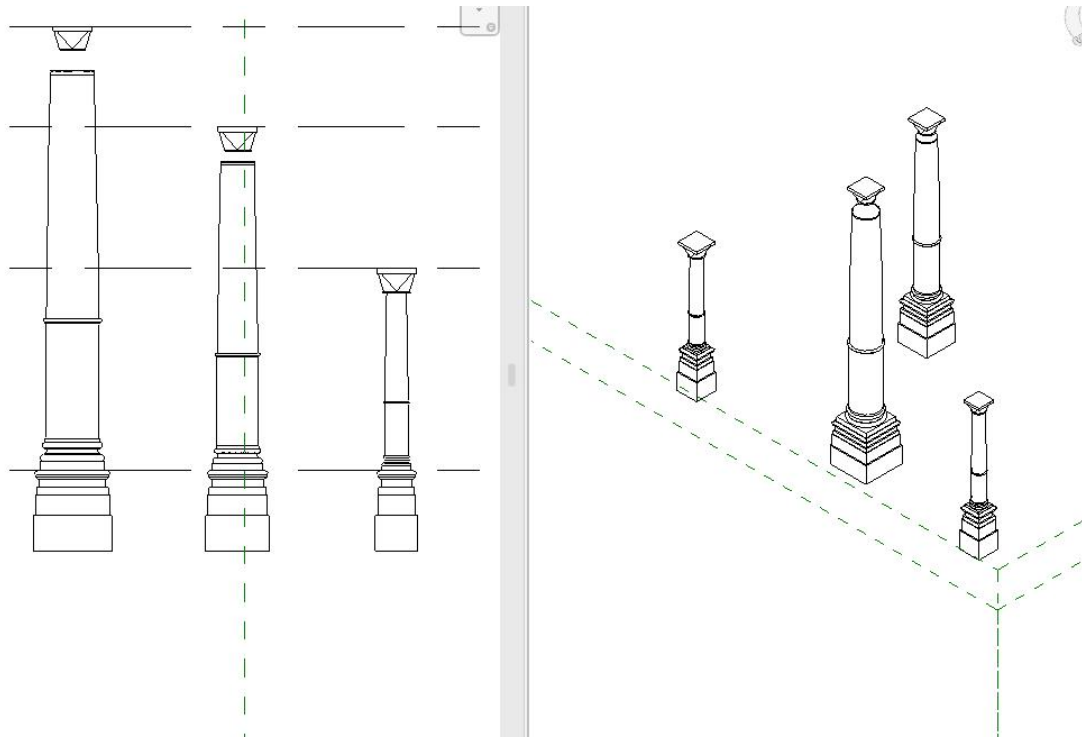


Figure 3-8 Situation without associated parameter(capital)

To ensure the consistency of parameter logic, this study further formulated the following nested management principles in the actual modeling process:

- All nested families uniformly adopt Base Diameter as the key parameter name to avoid name conflicts;

- Each nested family is verified for parameter binding in its type parameter panel after being completed;

- Multiple types are set in the main family for testing to ensure consistent deformation responses of nested components;

- For nested components that need to be enabled/disabled (such as column bases or carvings), type condition formulas are used for visibility control.

Although this method is highly structured, due to the fact that Revit itself does not support the automatic recursive transmission of parameters, the entire process still requires modelers to perform step-by-step manual operations and be supplemented by logical checks. The completeness of nested family management directly affects the stability, controllability, and later scalability of the modeling.

3.3.4 Proportional Control in Component Detail Modeling

In the construction of column families, detailed parts such as the base and capital usually contain many curved moldings and complex profiles. The modeling accuracy of these details directly affects the visual quality and structural expression of the whole column. To achieve parametric control of these detailed elements, this study uses a combined strategy of unified proportion rules and external supporting references.

For proportion control, the dimensions of all detailed components are based on the “Base Diameter” as a key parameter. Each size is calculated as a multiplier coefficient, and expressed through formulas in the nested families, in order to guarantee consistent proportion relationships among components, and improves the re-usability and scalability of the families. This was particularly relevant for historic buildings, where many pieces appear similar but still show small variations.

Revit’s Profile families are restricted to 2D work planes and cannot verify vertical section geometry. Because of this, it was often hard to check the correctness of complex profiles directly in the modeling environment. To address the issue, this study used external references, including orthographic views created with the Faro AsBuilt plugin and section outlines measured in AutoCAD. These drawings were brought into the family editor and applied as sketch guides, which helped to improve the accuracy of the modeled profiles.

Although this workflow can achieve a high-fidelity digital reconstruction, the modeling process still relies on manual calculations, parameter management, and collaboration between multiple tools. It is a high-cost, high-precision modeling path with complete information that require precise representation. As mentioned, this modeling process remains highly labor-intensive. As demonstrated in Brumana's HBIM case study of the Basilica di Collemaggio (2018), the development of high-fidelity parametric models without generative tools required over 600 hours of expert BIM modeling, in addition to 200 hours of preparatory procedures and the assistance of students over 120 working days. Even with improved workflows, later iterations still involved 300+120 hours of professional input. These figures support the assertion that high-precision digital reconstruction is a high-cost modeling path, particularly when applied to historic components with complete geometric information.

3.4 Vault

3.4.1 Analysis of architectural geometric forms

The formal classification of vaulted structures in historical architecture has long been an essential part of architectural education and construction practice. One of the most influential 19th-century pedagogical treatises in Italy, *L'arte di Fabbricare* by Giovanni Curioni (1868), systematized a vocabulary of vault types such as volta a botte (barrel vault), volta a padiglione (pavilion vault), volta a vela (sail vault), volta a crociera (groin vault), and volta a conca (basin or cloister vault)(see Figure 3-9). These typologies were not only geometrical distinctions but were directly linked to masonry techniques, load-bearing behavior, and proportional reasoning in architectural construction.

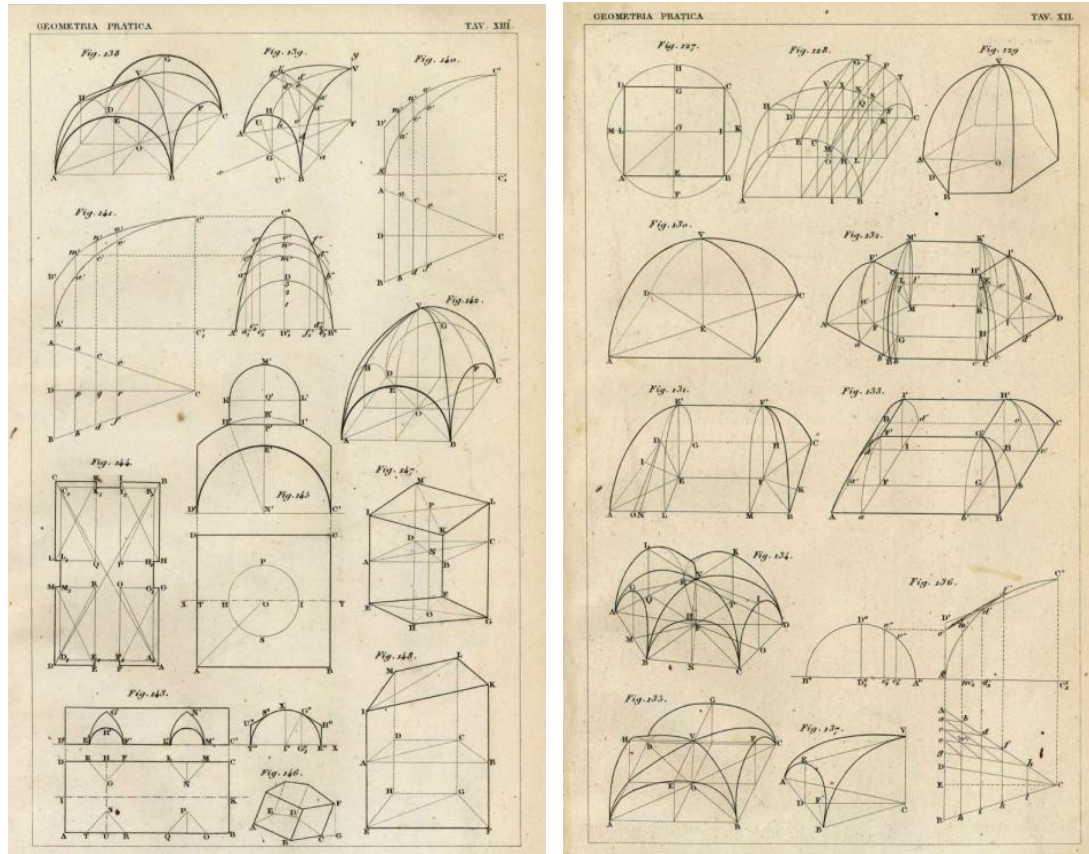


Figure 3-9 Geometrical principles for vault construction, reproduced from Giovanni Curioni, *L'arte di Fabbricare* (Torino: Negro, 1868), Tav. XII – XIII.

Diagrams include pavilion, sail, and groin vaults based on descriptive geometry.

Building on this tradition, recent research has explored ways to translate classical vault forms into digital frameworks. A key example is the doctoral thesis of F. Natta(2024), which applied VPL-to-HBIM (Visual Programming Language to Historic Building Information Modeling) methods to organise these forms into a systematic digital structure. As shown in Figure 3-10, both simple and composite vault typologies are summarised in a readable way, serving as a digital interpretation of historic geometric logic, which was adapted for contemporary parametric modeling workflows.

These combined typological references provide the theoretical foundation for the vault modeling strategy in this study. The specific vaults in Palazzo Barolo may not strictly conform to textbook typologies, but they could be understood as combinations or geometric variants of these standard forms. This modeling approach relies on the principles of classical vault logic. It is then adapted to address the irregular conditions and structural differences typical of heritage reconstruction.

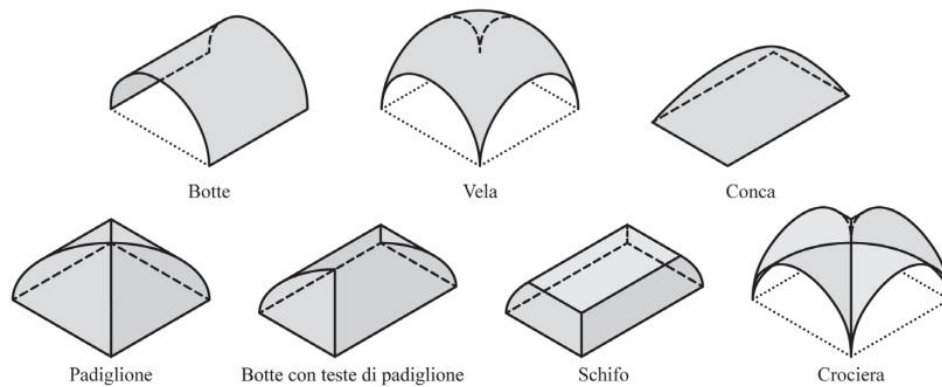


Figure 3-10 Typologies of simple and composite vaults digitally reconstructed through a VPL-to-HBIM pipeline. Reproduced from F. Natta, PhD Thesis, Politecnico di Torino, 2024

This typological framework forms the conceptual foundation of the vault modeling methodology developed in this research. While the vaults of Palazzo Barolo do not replicate these types identically, their geometry can be interpreted as variants or hybridizations of these canonical forms. The detailed classification and project-specific adaptation are discussed in Chapter 4.2.

3.4.2 Method Evaluation: Void Family vs. DirectShape

During the modeling of this project, to achieve the complex geometry of the vault and the requirement for parametric control, a comparative evaluation of two typical Revit-based modeling approaches is in the following part: Hosted void families¹¹ and DirectShape¹². There are significant differences between these two methods in terms of modeling workflow, parametric control, re-usability, host interaction, geometric complexity, BIM data integration, performance and visual output. 错误！未定义书签。 presents a detailed comparison to prove the selected optimal solution.

As an initial trying, the vault geometry was modeled using the node (FamilyType.ByGeometry¹³) in Dynamo to create parametric solid families. However, due to the large number of separated and not-continuous elements, it was extremely difficult to combine them in to a unified solid. Boolean operations¹⁴ - such

¹¹ Hosted family is a family in Revit that must be placed on or within another system element(eg., wall, floor, face).

¹² "This element type can store arbitrary geometry obtained from import operations or calculation in either a project or family document." Autodesk(2018).DirectShape. *Revit API Developer Guide*.

DirectShape is a static geometry element in Revit, often used to represent complex forms imported via Dynamo or API scripts. It does not support native parameters or host interactions, making it unsuitable for parametric control.

¹³ FamilyType.ByGeometry is a node in Dynamo for Revit that enables the creation of a loadable Revit family (.rfa) directly from a solid geometry. The node requires a geometry input and a specified family template path. It is often used to automate family generation for complex shapes, but the resulting geometry is static and cannot be parameterized or hosted.

¹⁴ Boolean operation refers to a geometric operation in 3D modeling that combines or modifies solid forms through union (addition), subtraction (cutting), or intersection (overlap). In Revit and Dynamo, Boolean functions are used to create complex volumes or to generate void forms that interact with host geometry.

as union, subtraction, or intersection of solids - often failed and the resulting solids could not be easily reused or controlled. As a result, this approach was abandoned in favor of using DirectShape. Multiple solids were imported in batches and grouped automatically by node inside the Revit project environment as an integrated geometry to simplify management, also without any need of positioning.

The vault geometry was initially constructed by generating surfaces and applying a Thicken operation to convert them to solid forms. But due to the non-continuous and segmented nature of these surface, the resulting solid could not be unified through Boolean operations in Revit, the same problem as FamilyType.ByGeometry.

While DirectShape is effective for representing highly complex or free-form geometry, particularly through Dynamo or the Revit API¹⁵, it has several critical limitations. It cannot be nested into families, doesn't support for type or instance parameters, and has no interactive relationship with system families such as floors or walls. Further more when the geometry is dense or finely detailed, DirectShape often produces triangulated mesh surfaces, leading to visible faceted edges in 3d Views and 2d documentation, which reduces both clarity and aesthetic quality.

In contrast, void families, particularly those using floor-hosted templates or created via node(FamilyType.VoidByGeometry) in Dynamo, offer robust parametric control, alignment constraints, and direct host interaction. Although void elements do not carry their own materials, they visually inherit the materials of the host elements they cut, such as floors. These families can be reused, nested, and fully integrated in to Revit's family system.

Based on these evaluations, the void family approach was ultimately selected as the main method of modeling. This method provides a more stable, controllable, and BIM-compliant workflow for vault reconstruction. It balanced the need for geometric flexibility with the requirements of precision, performance, and documentation integrity.

Table 3-5 Comparison between hosted void family, and DirectShape

Aspect	Void family (Hosted)	DirectShape (Solid)
Modeling Method	Created in family editor using void forms and host interaction/or use FamilyType.VoidbyGeometry with hosted family template, adjust the	Imported into the project via dynamo or API as static geometry.

¹⁵ Revit API (Application Programming Interface) is the official programming interface provided by Autodesk for automating, extending, or customizing Revit functionalities. It allows developers to access Revit's geometry engine, parameters, families, and modeling logic through .NET-based languages (such as C# or Python). Many Dynamo nodes operate by invoking underlying API methods.

	host-cut by manual	
Parametric control	Fully supports Revit parameters, constrains, and formulas, highly flexible	Static geometry, no native parameter support
Re-usability	Can be saved as family in void cut, reused, and nested.	Complex shapes require multiple solids and separate assembly, could be created by grouped
Host interaction	Can cut system elements (e.g., floors)	No interaction with hosts
Geometric complexity	Moderate flexibility within family environment	Very high; Supports complex or free-form shapes
BIM data support	Supports materials ¹⁶ and type/instance parameters	Limited metadata; geometry only
Performance for hardware	Lightweight; efficient even with multiple instances	Heavy geometry may degrade performance, especially with multiple shapes
Visual quality / 2D Output	Clean geometry, smooth surfaces, no triangle edges	Visible triangle edges in views and documents
Recommended for Vault	Yes. Precise, parametric, and stable	No, Can't be controlled when the geometry is combined with multiple solid and can't be in a family

3.4.3 Hosted Family Types for Void Forms.

After void families were chosen as the main modeling solution, the next task was to decide which hosted type could work best. This decision was influenced by multiple constraints: the vault geometry was reconstructed based on orthophotos, and it needed to be placed in relation to existing walls and floors that is already modeled in the project environment. For this reason, precisiuous positioning was important. It was not derived directly from point cloud coordinates, but from a framework established in the BIM context.

Two Revit family types were evaluated (Table 3-6): face-based and floor-based templates. Not-host family was excluded in this comparison completely, because the voids which was created in Revit can't cut the system family elements, such as floors or walls, which have already loaded and posited in the project.

Face-based families can be attached to almost any planar surface in the model, which makes them very flexible, but it also means a big cost: they are not so stable while attached on the host family. A known issue by the platform of users, is that the instance in the project environment rotates by twice the intended angle during placement or copy-rotate operations. As noted by Revit users: "When I rotate an item in Revit using the rotate command and I want to make a copy, the new instance of the item appears 2x the angle that I entered." (Autodesk Community Forum, 2019)

¹⁶ The void's material appearance is derived from the host element it cuts, such as a floor or wall.

This behavior leads to misalignment and makes precise placement almost impossible.

Furthermore, face-based void families must be carefully modeled in the correct orientation before insertion, and the base of the void must be aligned with the face host after rotation. During placement, the user must select the floor surface from bottom to top, following a reversed modeling logic. Such complexity often results in angular errors, flipped geometry, or even host loss. Due to these constraints, face-based families are unsuitable for vault modeling where accuracy and reliability of host interaction are essential.

Instead, floor-based family shows more stability and reliability. Although floor based families cannot be rotated around the vertical axis(Z-axis), once the element was placed in the project environment, the limitation contributes its reality. As every element was created in their own orientation, it is better not to rotate them in the project, avoiding to make misalignment. This constraint also simplifies the loading process, because each instance would maintain the same spatial relationship with the host, avoiding the rotation bug which is found in face-based families

The only things to do is to align the void base to the floor reference plane(base), and make the family horizontal center inside the void projection on the horizontal plane. So it is better to predefined the thickness of host in the family as it is same as the corresponding floor. In this project, it is also convenient to be consistent with wall, floor, and other components, without any reverse transformation.

In terms of family management, the floor-based approaches require creating separate families for each unique lunette, as the fixed orientation is already modeled before loaded. But, if the rotation behavior and the host worked correctly, the same geometric logic could be represented using a single family with different types or instance parameters.

And for position of the quadrilateral bottom edges of some irregular-shaped vault, the adaptive family is used as the container of four reference point, which arrange the named reference point in the project environment as a group, to mark the springing line and zone of bottom. But these adaptive components is only used for the position, which don't contains a real void form. Due to the limited of system in Revit, void form created inside the adaptive family can't interact with system family. Therefore, the most important logic of modeling "cut host by void" doesn't make sense.

The final void geometry is firstly created as a solid, based on the reference point in the project. Then it is converted to void form by node (FamilyType.VoidByGeometry¹⁷), with the floor-based family template. Finally, adjust the thickness of the host in the family environment, and align the base to the

¹⁷ Same as FamilyType.ByGeometry, but it creates void geometry forms in families, used specifically for cutting host elements such as walls or floors.

reference level, and make sure the origin of the family is in the horizontal zone of the void form.

Table 3-6 Void family performs in floor-based and face-based family

	Floor-based	Face-based
Cut system family?	Yes	Yes
Alignment and Stability	The same angle as it was created, always hosted on the floor	Not accuracy along to the edges, once the face is cut by other geometry, it might lost host.
Rotation behavior	No rotation along Z axis, but fixed orientation avoids misalignment	Bug: input rotation angle is doubled(e.g., 30° becomes 60°).
Amount of Family types (in this project)	One lunette per family, fixed orientation	Reusable via types or instance parameters, if orientation works correctly
Modeling Workflow	Simple: place and align	Complex: requires pro-rotation in family, reverse insertion logic in project.
Suitability for vault reconstruction	Perferred: stable, repeatable, easier alignment for vaults cutouts	Not recommended due to instability and rotation error

All geometry reconstructions presented in this study are focus on external surface data captured by Point Cloud. Internal structural layers and materials are not available. and remain subject to interpretative uncertainty.

3.5 Railing

3.5.1 Normal railing system in standard component and in historical building

In the Revit railing system, three primary family types work together to satisfy structural expression, repeatability, and parametric management requirements:

Baluster Family

This represents the typical repetitive vertical elements of the railing, usually small posts placed at fixed intervals according to the railing system's distribution rules. These balusters primarily serve protective and structural functions, and are automatically arrayed through parameters such as spacing and distribution mode defined within the railing type.

Baluster Post Family

Used at the start, end, or turning points of a railing, the baluster post provides enhanced support. These posts are typically larger and structurally stronger than regular balusters, and their positions are automatically identified by the railing

system at corners or endpoints, allowing different post family types to be inserted in combination with balusters.

Baluster Panel Family

This family is applied between balusters as an infill element, such as glass panels or metal grilles, serving both protective and aesthetic functions. It can be specified within the railing panel settings, with parametric definitions for height, thickness, and materials, supporting flexible appearance customization.

These three families are combined within a Revit railing type and managed together, so that the repeating vertical elements, strengthening nodes, and intermediate panels can work in a coordinated and systematic manner while remaining independently replaceable or adjustable.

In historical buildings, the balusters (vertical posts) also display a wide variety of shapes and styles. The examples shown in Figure 3-11 summarize the typical shapes of different historical periods, including single-chamber, double-chamber, square-column, and complex-decorated styles, among others. These styles often changed with regional styles and classical preferences, but the basic logic didn't change. In the most cases the balusters were arranged in repetition, their sections keeps slender for protection, and the proportions adjusted in a way that created smooth transition (Donghi, 1935). Such historical reference also support the definition of standard baluster families in Revit, allowing modern digital modeling to follow the idea of grouped arrangement and modular generation.

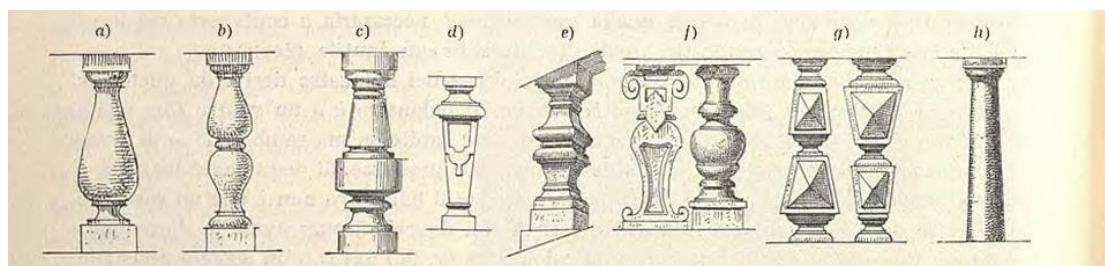


Figure 3-11 Variations of baluster forms historically used in Italian architecture, adapted from Donghi (1935, p.194).

3.5.2 Treatment of Special-Geometry Inclined Balusters

In this study, the staircase railing system requires a set of inclined balusters (oblique balusters) arranged according to the same repetitive logic as standard baluster families, yet with a geometry distinct from conventional vertical or rotationally symmetric members. As illustrated in Figure 3-12, historical balusters generally maintained a continuous symmetrical profile generated through revolved solids, while their upper and lower connections were adapted to the slope of the stair through trimming operations. This results in a spatial configuration where the

baluster appears sheared or inclined as a whole. Such geometries cannot be fully achieved using standard Revolve or Sweep commands in Revit, and are therefore classified as non-standard baluster components.

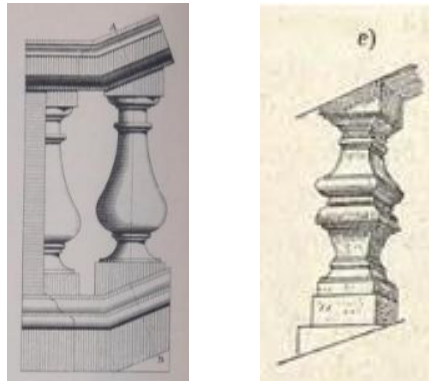


Figure 3-12 Examples of common baluster profiles in historical architecture, and their profile-end trimming when adapted to an inclined staircase.

Left: adapted from Chambers (1968, p. 102); right: adapted from Donghi (1935, p. 194).

But these oblique balusters still need to be placed in a repetitive sequence and aligned with the stair treads, it doesn't matter whether the treads are flatten or sloped. The Revit railing system would assume vertical arrangement of baluster elements, and the family editor does not support custom rotation or shear of revolved geometry. This makes the direct generation of oblique elements in Revit particularly challenging.

Therefore, this research categorizes them as a form of "special geometry" and explores three alternative solutions for modeling and BIM integration:

1. Contour slicing and lofting in Dynamo, where the standard horizontal baluster is sliced into uniform-height contours and reconstructed into an inclined solid through loft operations;
2. Shear transformation in Rhino/Grasshopper, where the standard horizontal geometry is generated and then globally transformed using a shear node to precisely match the stair slope;
3. Void-based sweep or extrusion within Revit, creating a cutting volume to manually trim a standard baluster, approximating the inclined shape.

The choice among these methods depends on requirements for geometric accuracy, ease of editing, and long-term maintenance. Compared to conventional balusters, the geometry of oblique(sheared) balusters require additional steps for transferring of geometric formats among different software programs, which means involved more cross-platform modeling workflows, intermediate formats (such as SAT¹⁸), or

¹⁸ SAT (Standard ACIS Text) is a file format based on the ACIS geometric modeling kernel, supporting the exchange of NURBS or B-rep solids between CAD/BIM platforms.

manual adjustments, in order to achieve alignment with the slope and preserve historical authenticity.

It is also worth noting that while Revit supports importing SAT geometry to integrate complex freeform solids, these imported elements generally behave as static geometry without native parametric controls, which limits their adaptability and connection to the railing system's family parameters. Among available exchange formats, SAT is preferable because it preserves NURBS(Non-Uniform Rational B-Splines)-based solid geometry, whereas formats such as OBJ¹⁹ or STL²⁰ are triangulated meshes that not only lose volumetric consistency but also introduce unnecessary triangulation edges in Revit's views, compromising the visual clarity and reducing the overall readability of the model.

The specific implementation steps and comparative evaluation of these three solutions are discussed in detail in Section 5.4.3.

¹⁹ OBJ is a widely used polygon mesh exchange format, originally developed for 3D graphics, which represents geometry as a collection of planar faces (usually triangles).

²⁰ STL (Stereolithography) is a mesh-based file format commonly used in rapid prototyping and 3D printing workflows, storing only tessellated triangular surfaces without solid or parametric data.

4. Implement and Result

4.1 Column

4.1.1 Two types Column in Atrium and Staircase

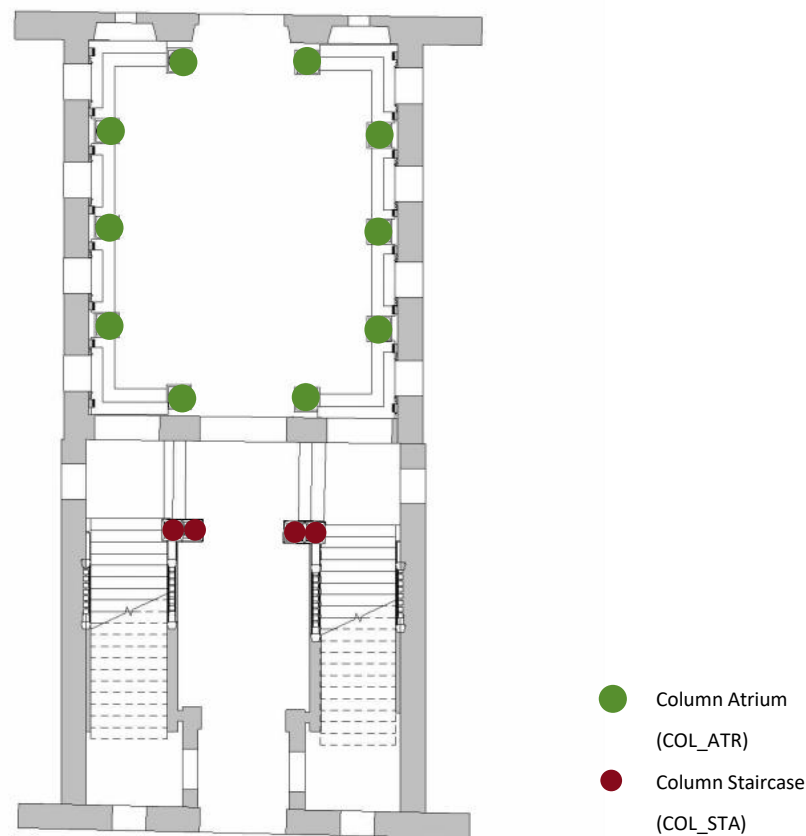


Figure 4-1 Two types of column in atrium and staircase

During the parametric stage, this study first classified the columns based on their spatial distribution in the architectural plan. Two categories were defined: Atrium Columns(COL_ATR) and Staircase Columns(COL_STA) (see Figure 4-1). These two types differ from their spatial position, and besides their proportion. As the atrium columns played a structural role, and the ones which is in staircase were more decorative and thin to provide good sense of higher spatial atmosphere.

To facilitate logical segmentation and proportional control in the subsequent Revit family construction, a basic decomposition of each column was carried out based on the point cloud data, identifying the boundaries of the capital, shaft, base, and, where applicable, the pedestal, as illustrated in Figure 4-2. This segmentation was based on height ranges and approximate geometric boundaries, rather than fine

decorative details, providing a reference framework for proportional parameterization.





Table 4-1 summarizes the core proportional parameters for the two column types: both adopt a total height of approximately Base Diameter × 8, but differ in the distribution of capital and shaft heights. The COL_ATR features a relatively shorter capital and a taller shaft to emphasize the continuity of the arcade, while the COL_STA has a more prominent capital and a slightly shorter shaft to enhance the visual focus and structural presence of the staircase area. In addition, the pedestal height is subject to project-specific adjustments.




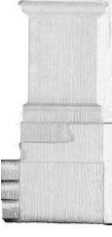
Through this segmentation and proportional analysis, the subsequent nested family components can be developed based on a unified master parameter (Base Diameter), while still preserving the distribution characteristics of historical construction. This approach supports a flexible yet historically accurate digital reconstruction workflow.



Figure 4-2 Segmented Column of Pointcloud in Atrium and in Staircase(type)

Table 4-1 segmented element in column (Pointcloud)

	COL_ATR	COL_STA
Capital		
Shaft		

Base		
Pedestal		
Note	Column= Base Diameter*8 Capital : Shaft : Base = 0.6 : 6.9 : 0.5 Pedestal(depends in project)	Column= Base Diameter*8 Capital : Shaft : Base = 0.8 : 6.7 : 0.5 Pedestal(depends in project)

Final Modeling Outcome

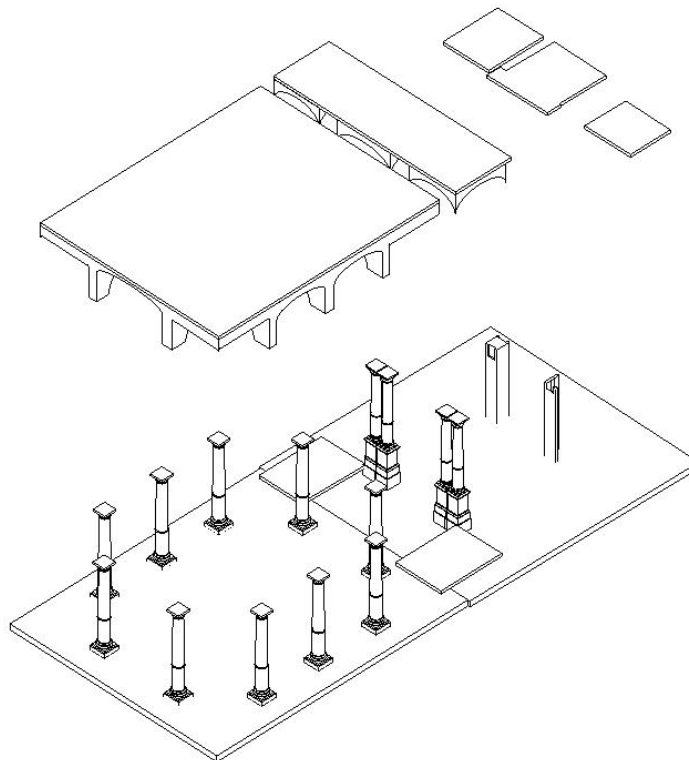


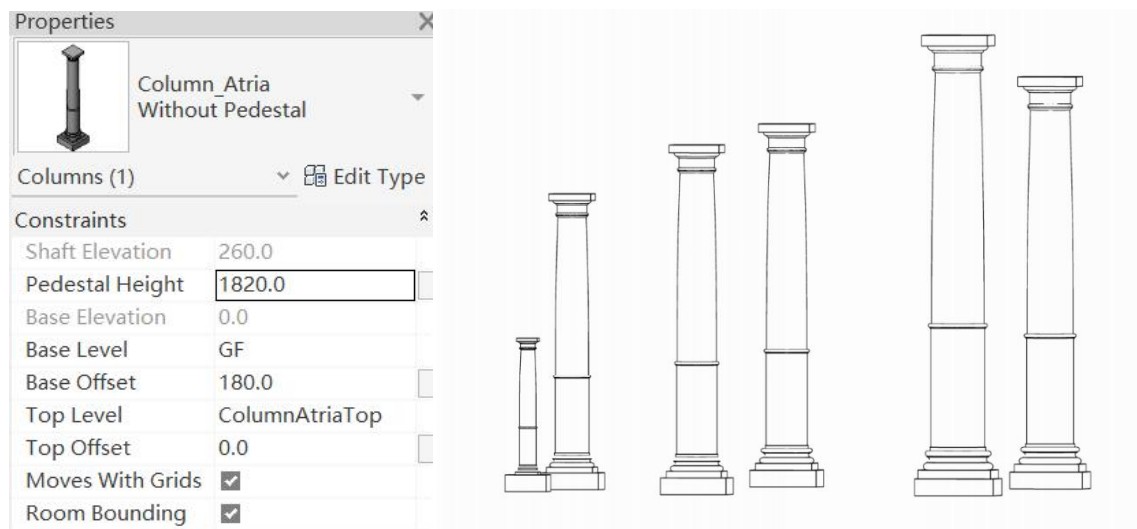
Figure 4-3 distribution of columns within the project

After completing the parametric segmentation of columns, designing the profiles, and organizing the family files, these components were loaded into the Revit project for testing and placement. Figure 4-3 illustrates the overall distribution of columns within the project, where both atrium and COL_STA are controlled by their respective parametric families and aligned precisely with the slab and wall nodes. Using the master parameter Base Diameter and instance parameters such as Shaft

Elevation, the columns can be flexibly adjusted across different levels and elevation conditions. The connection with the structural system confirms the feasibility of the family logic within the project coordinate system, ensuring portability and reuse. However, due to the resolution limitations of the point cloud in capturing fine sculptural details, the more intricate ornaments of the capitals were not fully recreated within the parametric family and are instead reserved for mesh models or later complementary modeling. Future research could explore using Dynamo or other automation scripts to further integrate the segmentation and parameter transfer of these complex details.

When the column family is loaded into the project environment, its instance properties, as shown in the left of Figure 4-4, include constraints for Base Level, Top Level, and their respective offsets. Therefore, it is only necessary to set the correct reference levels in the project. Meanwhile, the offset values can be fine-tuned if needed - for example, the pedestal height can be adjusted to match floor slopes or local elevation differences.

Because all elements in the family are consistently governed by the global Base Diameter parameter, even when the level is changed, the proportional logic is preserved, preventing any geometric distortion or constraint conflicts. The illustration demonstrates the correct behavior of the columns placed on different levels within the project, confirming that the combination of system reference levels and parametric controls ensures robust digital reconstruction(see the right of Figure 4-4).



*Figure 4-4 System column families under different height-level constraints in the project.
(Correct representation of constraints)*

4.1.2 Modeling for column system

Logic in column system family

The implementation of the parametric column family requires addressing the limitation that Revit's system column family only provides two reference levels-namely Base Level and Top Level. However, historical columns in this project are subdivided into at least three or four parts, including the base, shaft, capital, and in some cases a pedestal. To ensure proper vertical segmentation and maintain correct positioning of the shaft, an additional intermediate parameter named Shaft Elevation was introduced within the column family. This parameter was configured as an instance parameter and defined by the following formula: Shaft Elevation = Base Diameter * 0.5 (equals to Base height)

As illustrated in Figure 4-5, this parameter ensures that the shaft segment can be flexibly adjusted on a per-instance basis while maintaining a consistent proportional relationship to the base diameter. This design improves adaptability when minor vertical deviations are encountered on-site.

Figure 4-6 shows how family parameters were created in the column family to handle these additional controls, establishing a clear linkage between the measured proportions of the historical columns and the parametric framework within Revit.

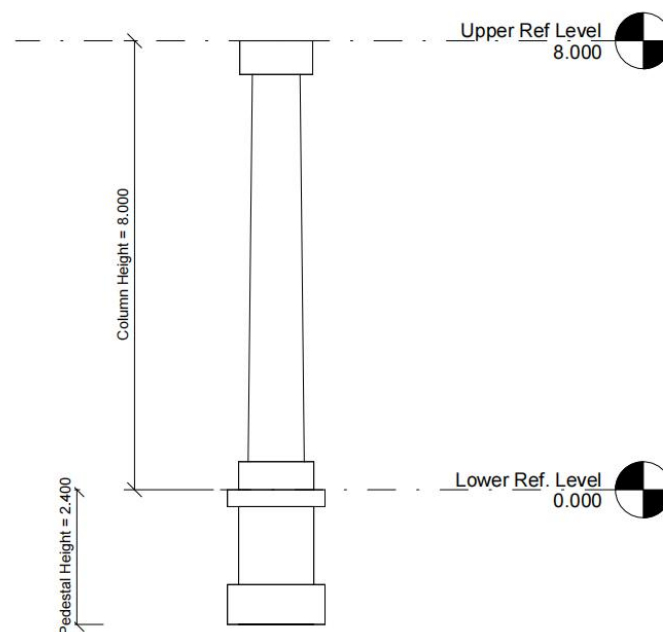


Figure 4-5 The two preset reference levels and the added measured family parameter (column height)

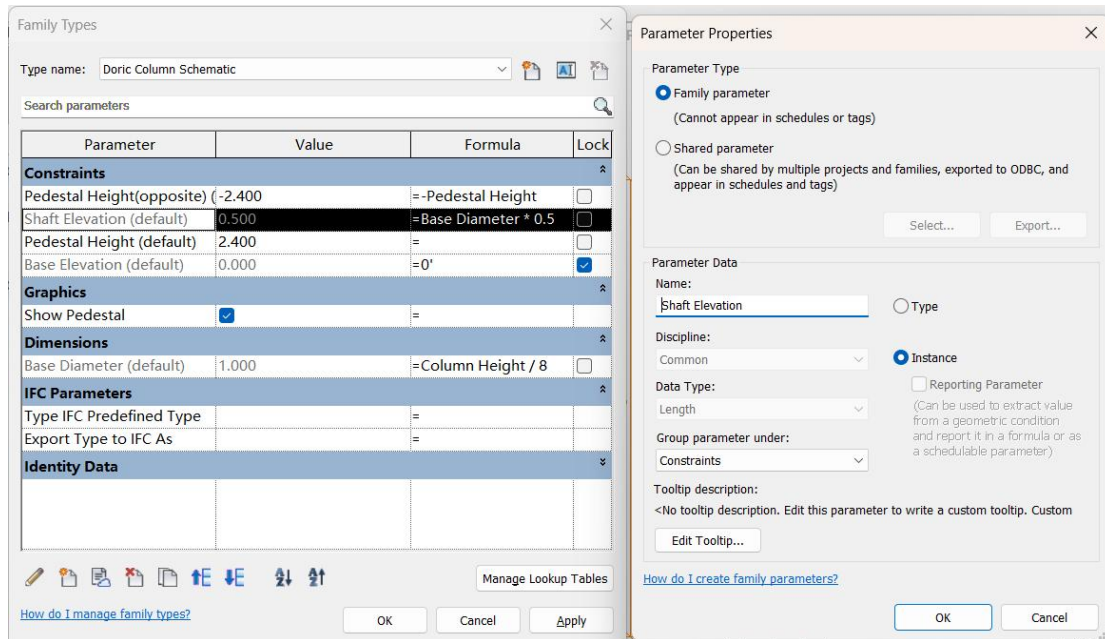


Figure 4-6 Create family parameters in column family

In terms of constraining the other sub-components, namely the base, capital, and pedestal, these were aligned and locked to the two existing reference levels provided by the system family, without requiring additional parameters. As shown in Figure 4-7 and Figure 4-8, these elements could be stably positioned simply by aligning their reference planes to the system's Base Level and Top Level constraints, then applying lock constraints to fix them in place.

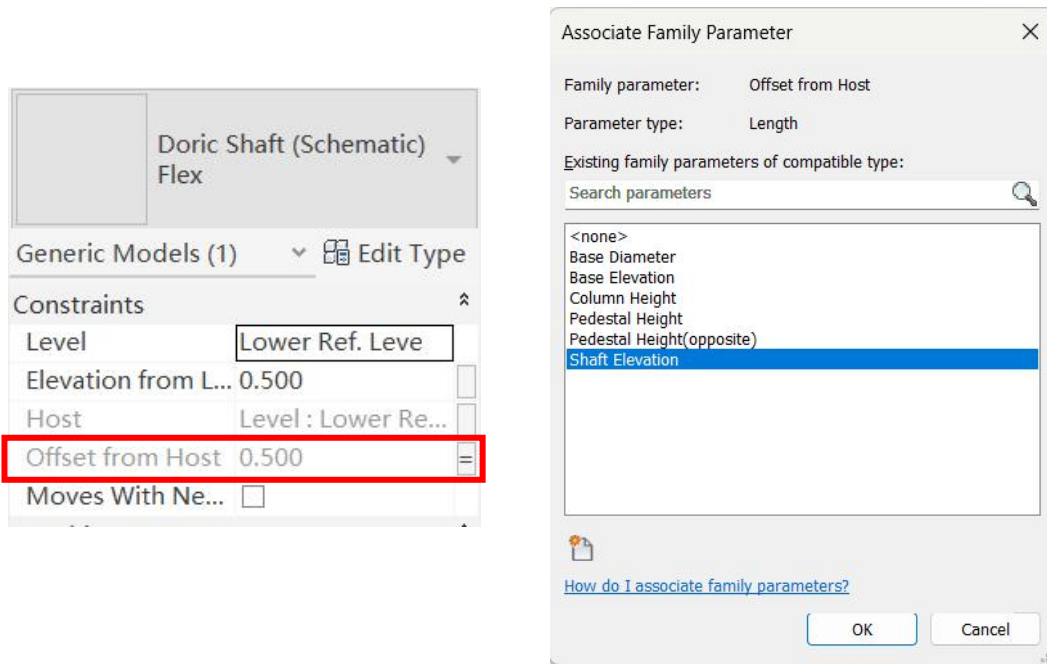
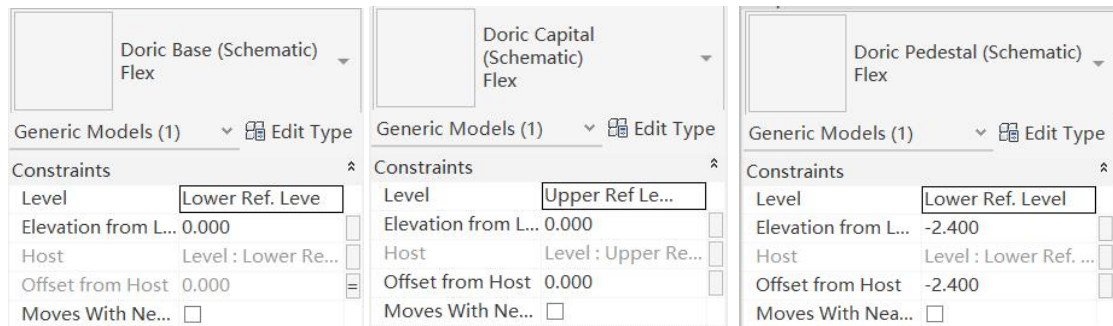
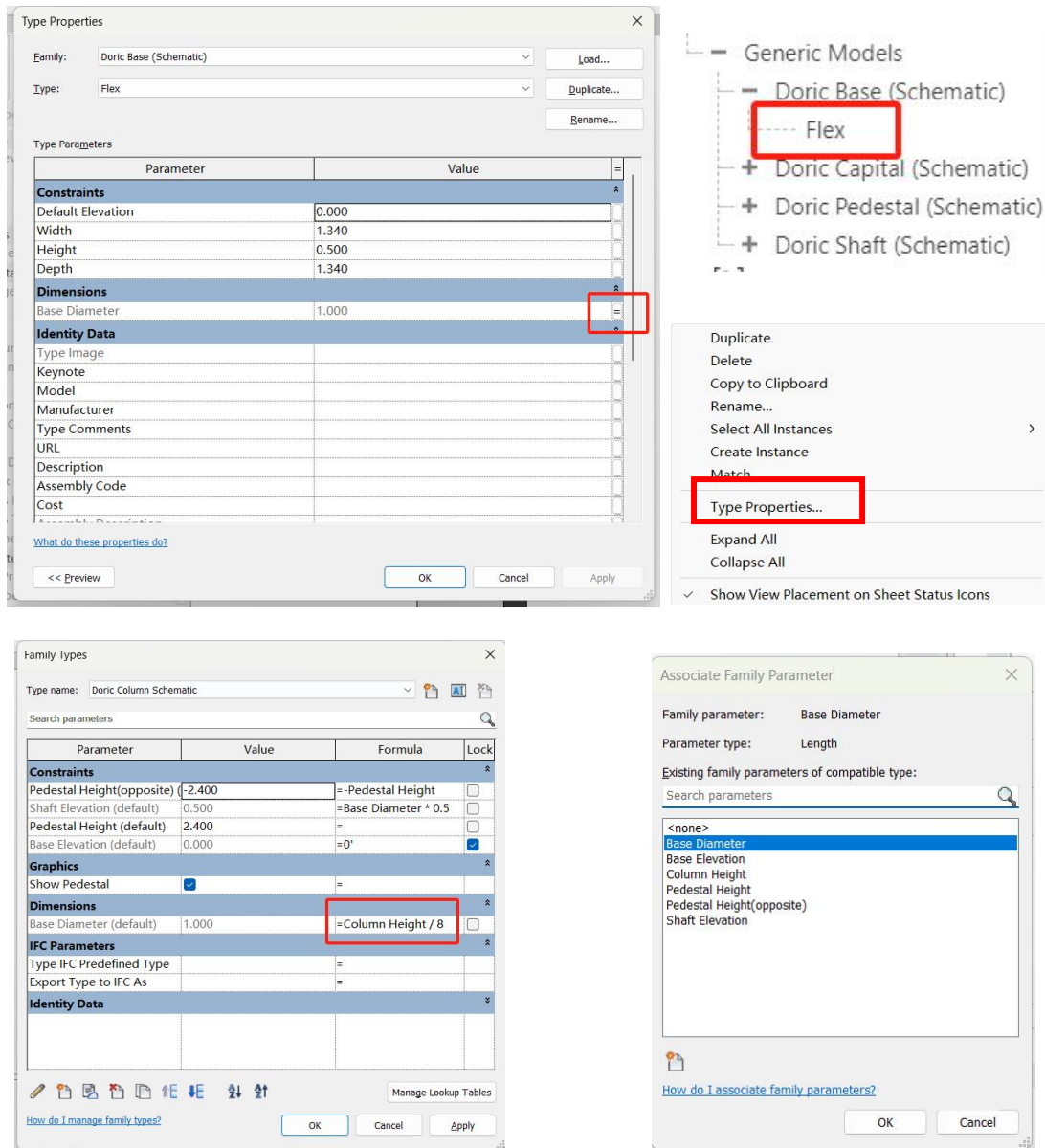


Figure 4-7 Parameter mapping between instance-level constraints and family level associations in revit



*Figure 4-8 Constraints for Base, Capital, and Pedestal
(aligned to Lower Ref. Level, Upper Ref. Level, and Lower Ref. Level respectively)*

For each nested family loaded into the column family (e.g., the nested shaft, base, or capital families), it was necessary to open its type properties and link its pre-defined “Base Diameter” parameter to the one defined in main column family. As shown in Figure 4-9, this link guarantees that any changes made at the main family level is passed down automatically to each nested component. Otherwise, without this step, nested families would retain their default dimensions and fail to scale with the column as expected (see Chapter 3, Figure 3-8).



*Figure 4-9 Associated Base Diameter Type Properties
(Base diameter parameter in family type already created with formula)*

Logic in Nested family

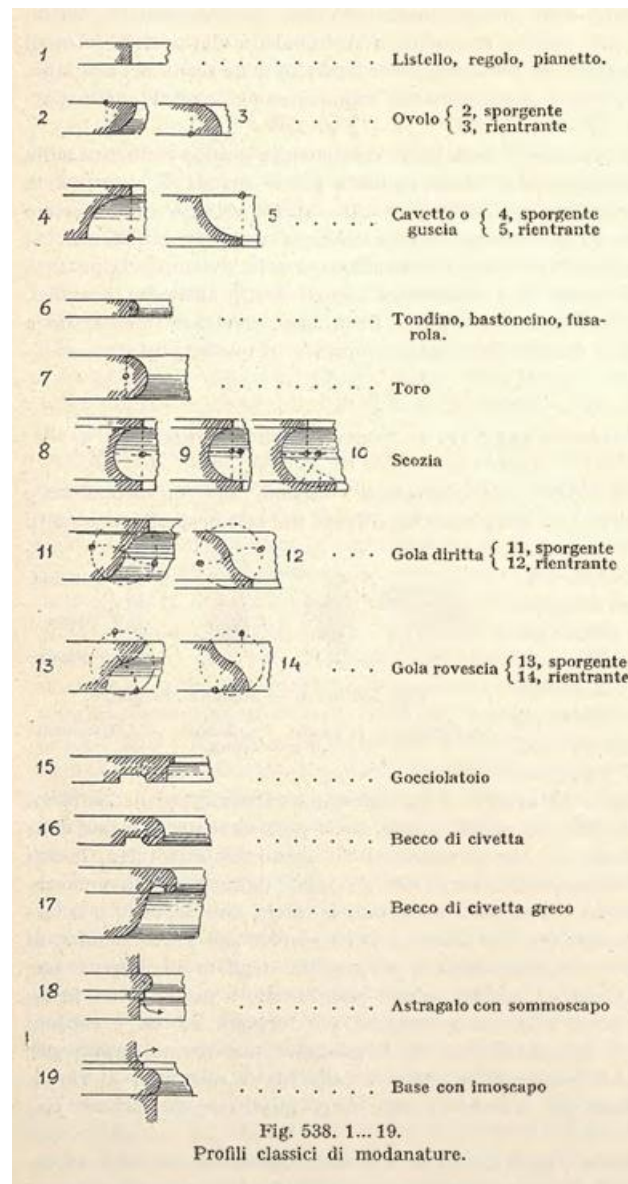


Figure 4-10 Classical moulding profiles extracted from Donghi (1935, p. 154).

Before discussing the nested family approach, it is worth briefly recalling the role of architectural ornament, particularly moldings, which represent one of the most widespread categories of architectural decoration. As described by Donghi (1935), classical moldings were standardized profiles used to articulate architectural orders and to define transitions between structural parts, from base to cornice. These profiles, while seemingly simple, exhibit a parametric logic of curvature, fillets, and projections, designed to generate consistent yet flexible decorative elements across different scales of architecture (Donghi, 1935).

This traditional logic of composing standard profiles resonates strongly with the Revit concept of nested families, where a simple repeated geometry can be parameterized and recombined to form more complex architectural systems. In this

sense, the nested family strategy adopted in this study can be seen as a digital reinterpretation of the compositional rules historically applied to classical mouldings and similar decorative elements.

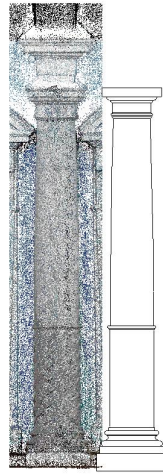


Figure 4-11 pointcloud data in dxf and modeling

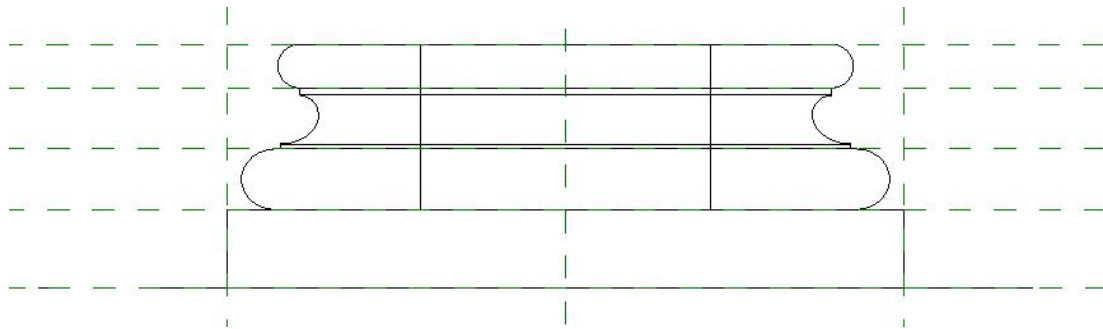


Figure 4-12 Section of Base

For this component's parametric control, it is necessary to calculate the proportional relationships between each vertical (Y-direction) element and the Base Diameter. Since the total base height is predefined, it can be conveniently subdivided into simpler fractional proportions (for example, $0.09 + 0.125 + 0.125 + 0.16$) relative to the overall $0.5 * \text{Base Diameter}$ height. This approach supports consistent management of parameters in the family and simplifies the proportional calculations.

In the reconstructed base of this study, the composition from bottom to top includes the plinth, a lower torus molding, a scotia (with fillet), and an upper torus. This layered sequence is characteristic of classical architecture: the plinth provides the raised platform and horizontal reference, while the lower and upper torus elements help visually anchor and consolidate the base. The scotia in between, with its concave curvature, introduces shadow contrast and enhances the sense of rhythm and proportional harmony.

Therefore, in the modeling process, it is only necessary to create the following components:

1.a plinth modeled as an extrusion, with constraints for its height and its plan dimensions, and with its bottom center aligned to the family's central reference point;

2.a cylindrical extrusion whose height is defined by subtracting the plinth's height, and which is placed on top of the plinth to serve as the reference for retrieving profile edges during the sweep operation.

3.On the perimeter of this cylindrical base, the following moldings can be positioned(see Figure 4-13):

- (1)a torus at the top;
- (2)a scotia in the middle;
- (3)another torus at the lower part.

Among these, the top and lower torus moldings can be created as the same type with variable instance parameters for vertical positioning or scale, in order to improve modeling efficiency and reduce the redundancy of family definitions.

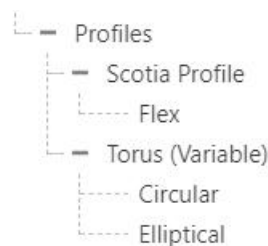


Figure 4-13 Nested Profile families in Column Base family

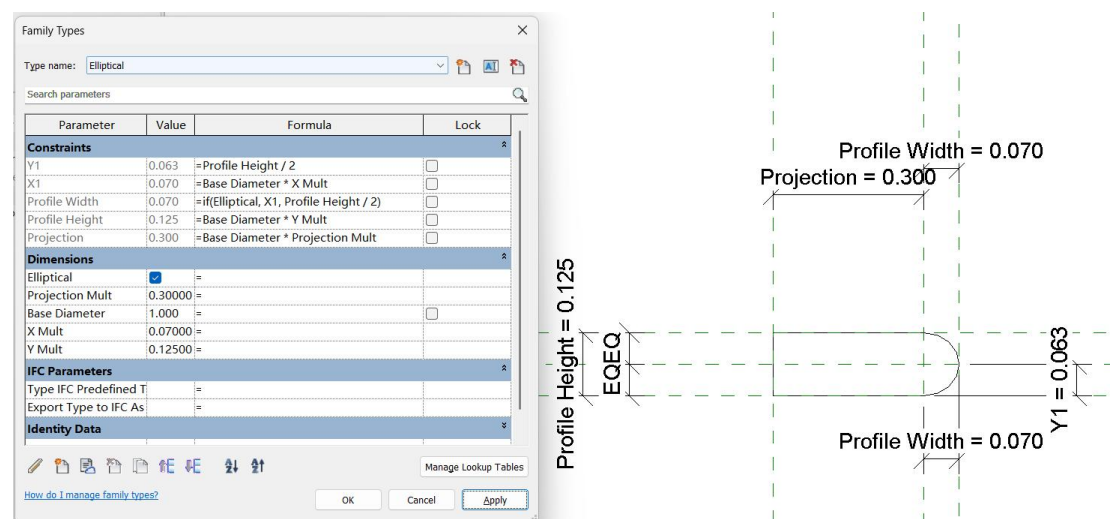


Figure 4-14 Parameters in Torus Profile family

Figure 4-14 shows a parametric profile family for a molding that can switch between elliptical and circular shapes. The dimensions are linked proportionally to the Base

Diameter, with multiplier parameters such as X Mult, Y Mult, and Projection Mult. A boolean parameter named Elliptical allows switching of the profile geometry. Formula-driven constraints keep the width, height, and projection in proportion, which allows the same family to scale consistently within various details.

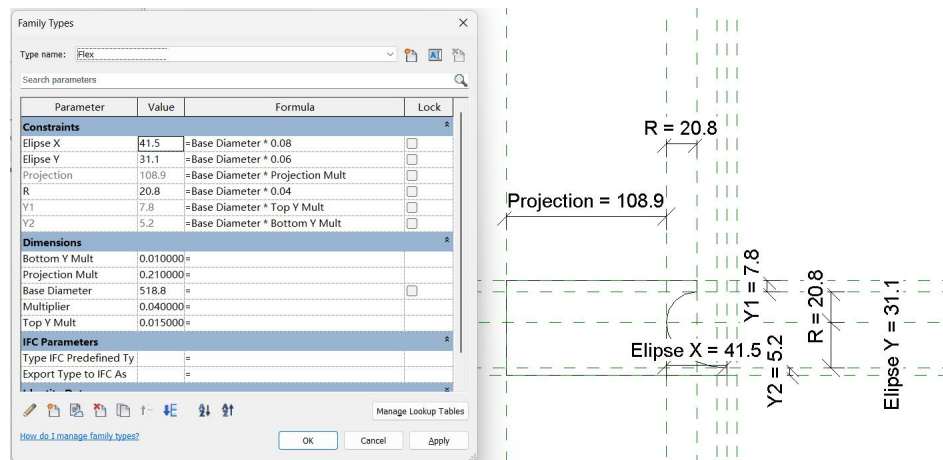


Figure 4-15 Parameters in Scotia Profile family

Figure 4-15 shows a flex-profile family used to control the geometry of column base or capital moldings in a parametric way. In the “Family Types” panel on the left, the profile's dimensions are driven by the global parameter “Base Diameter”, with additional multipliers (e.g., X/Y Mult, Projection Mult) to automatically adjust its geometry. The main parameters are summarized as following (see Table 4-2):

Table 4-2 Short description of parameters in Scotia Profile family

Parameter	Description
Ellipse X / Ellipse Y	Semi-major and semi-minor axes of the elliptical portion of the profile
Projection	Horizontal projection distance
R1 / R2	Corner radii to ensure smooth transitions in the curve
Y1 / Y2	Vertical positioning of the profile's midpoint or split, enabling symmetry
Top Y Mult / Bottom Y Mult	Proportions of the top and bottom the profile height, supporting shape variety

The sketch on the right of Figure 4-15 identifies projection, radii, and elliptical dimensions relative to the reference planes. This configuration allows the profile to be scaled to various column diameters and molding shapes. At the same time, it reduces the effort needed to maintain multiple separate family files. When all profile is done, load all of them into column base family and open the type property to link the base diameter to guarantee the global parameter transforming correctly. Once all the profiles have been completed, they should be loaded into the column base family. Within the type properties, it is necessary to associate their local Base Diameter parameters with the global Base Diameter of the parent family to ensure that the master parameter propagates correctly across all nested profiles. Following the same logic as in the previous section, family parameters were created within this profile family (for example, the vertical offset of the profile, see Figure

4-16), and then associated to instance parameters. This approach ensures that the nested profile components can be flexibly positioned while maintaining consistency with the overall column geometry. And also association of base diameter is needed (in type properties, see Figure 4-17).

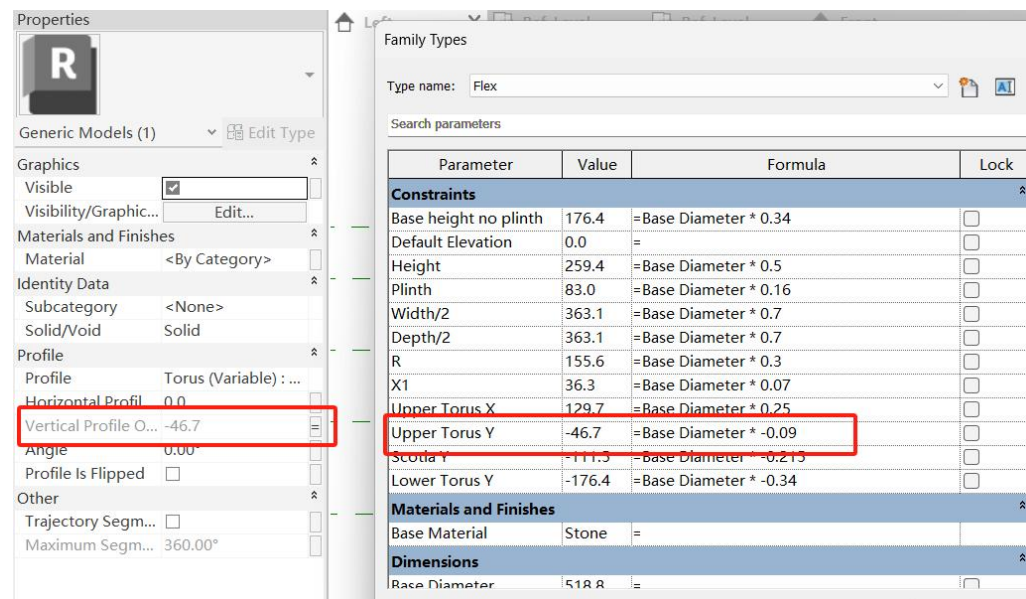


Figure 4-16 Associated Vertical Profile offset to Upper Torus Y

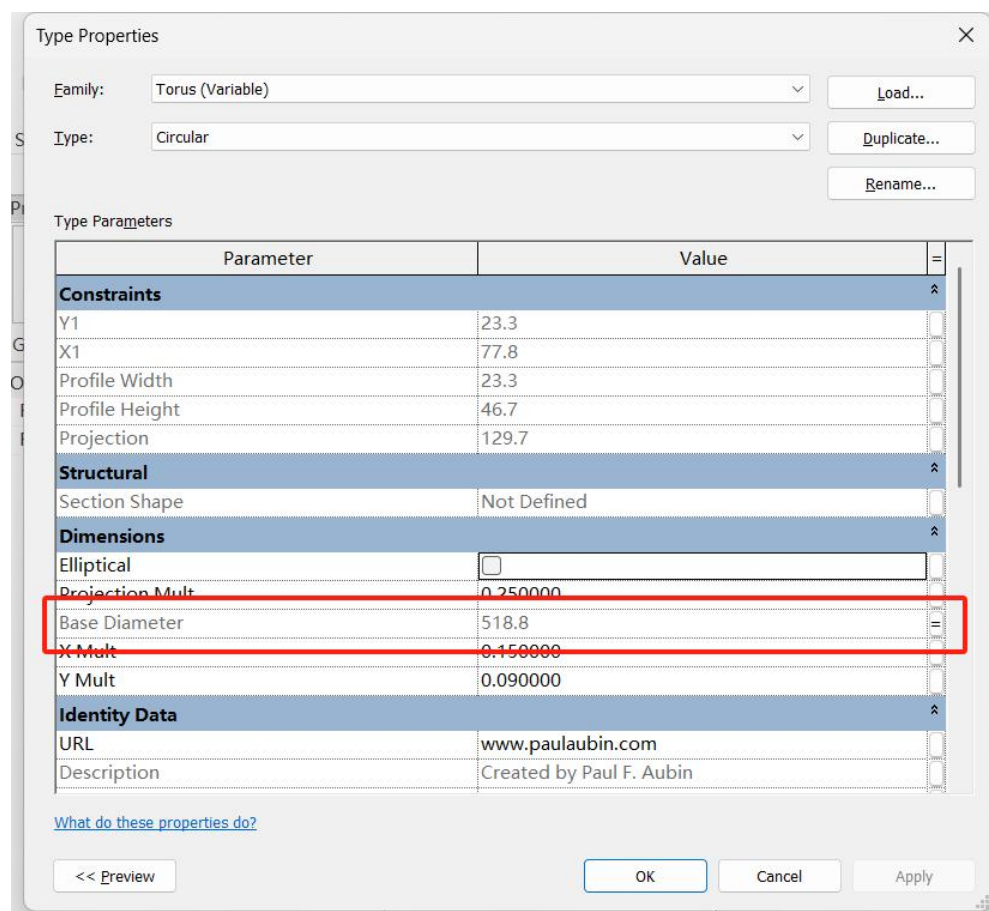


Figure 4-17 Associated Base Diameter in Type Properties

4.2 Vault system

4.2.1 Final Modeling Outcomes and Comparative Assessment

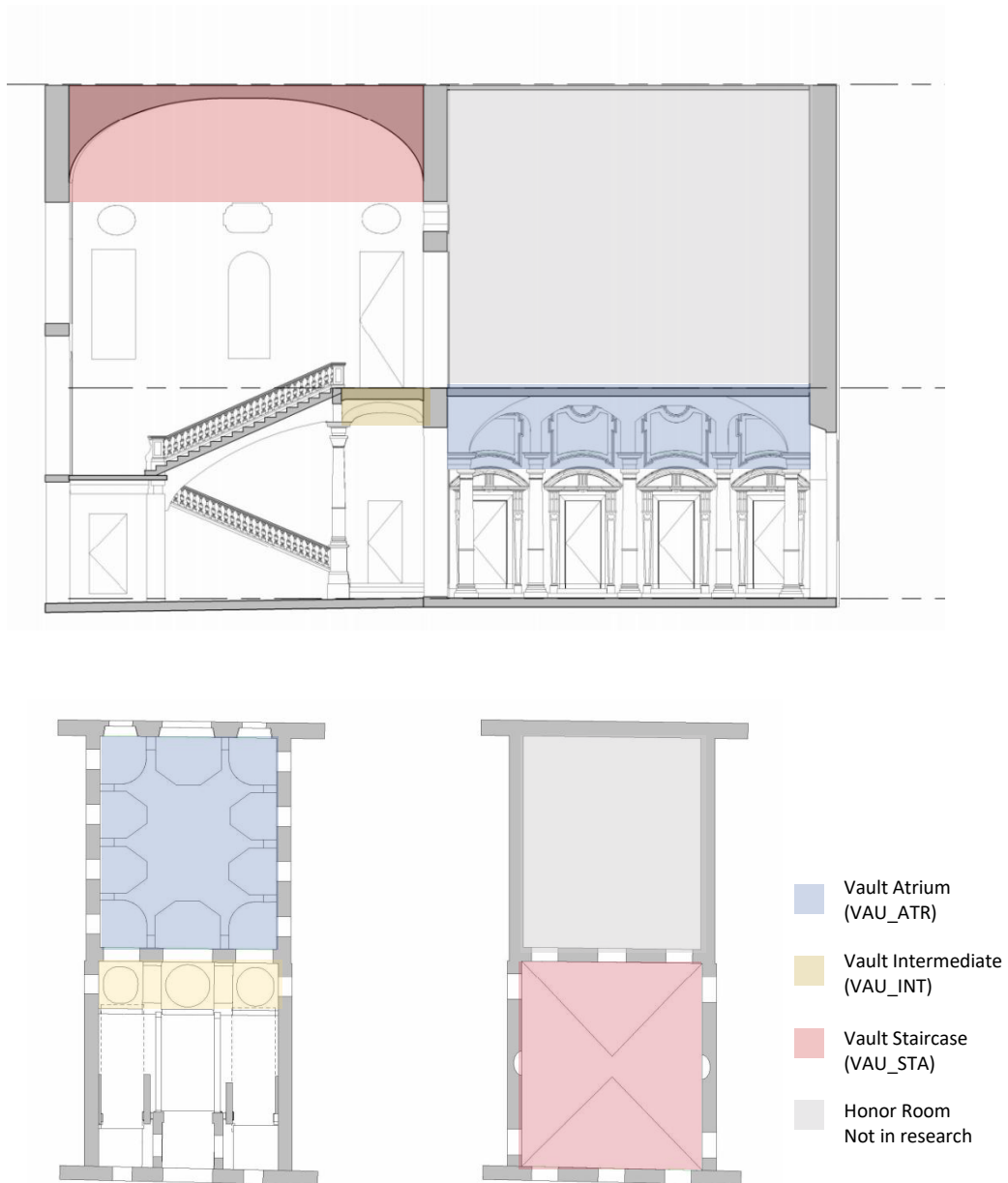


Figure 4-18 Vault position in Reflected Ceiling Plan



Figure 4-19 Segmented vault system from Point Cloud(CloudCompare)

In this project, three distinct vaulted systems were identified and modeled based on their spatial location and structural characteristics. To streamline management and clarify component identity, each vault was encoded using the prefix VAU_ followed by a type abbreviation:

VAU_ATR: the atrium vault, located above the central honor space, composed of a main barrel vault and ten surrounding lunettes;

VAU_INT: the intermediate vaults, located in the corridor between the atrium and the grand staircase, consisting of two lateral bays (Type A and Type B) with transitional geometric roles;

VAU_STA: the staircase vault, located above the staircase connecting to the Honor Room, with a barrel-vault-with-cloister-end configuration.

Although the three types differ in scale, complexity, and in the logic of integration, they were all produced within a single parametric framework built on floor-hosted void families. The following paragraphs provide a comparative analysis, focusing on geometric characteristics, strategies of control, and specific issues that arose during implementation.

The VAU_ATR has a surface similar to the combination of barrel and conca(on surface), and 10 non-standard Barrel lunettes. Therefore, it requires the highest degree of geometric abstraction and parameter control. In contrast, the geometric shape of the VAU_STA is relatively regular and is reconstructed through a

combination of scanning and lofting, requiring an extremely low degree of approximation. The VAU_INT between atrium and staircase although having a relatively simple geometric structure, introduces unique main interaction and direction adjustment, making it an important transitional case in this method.

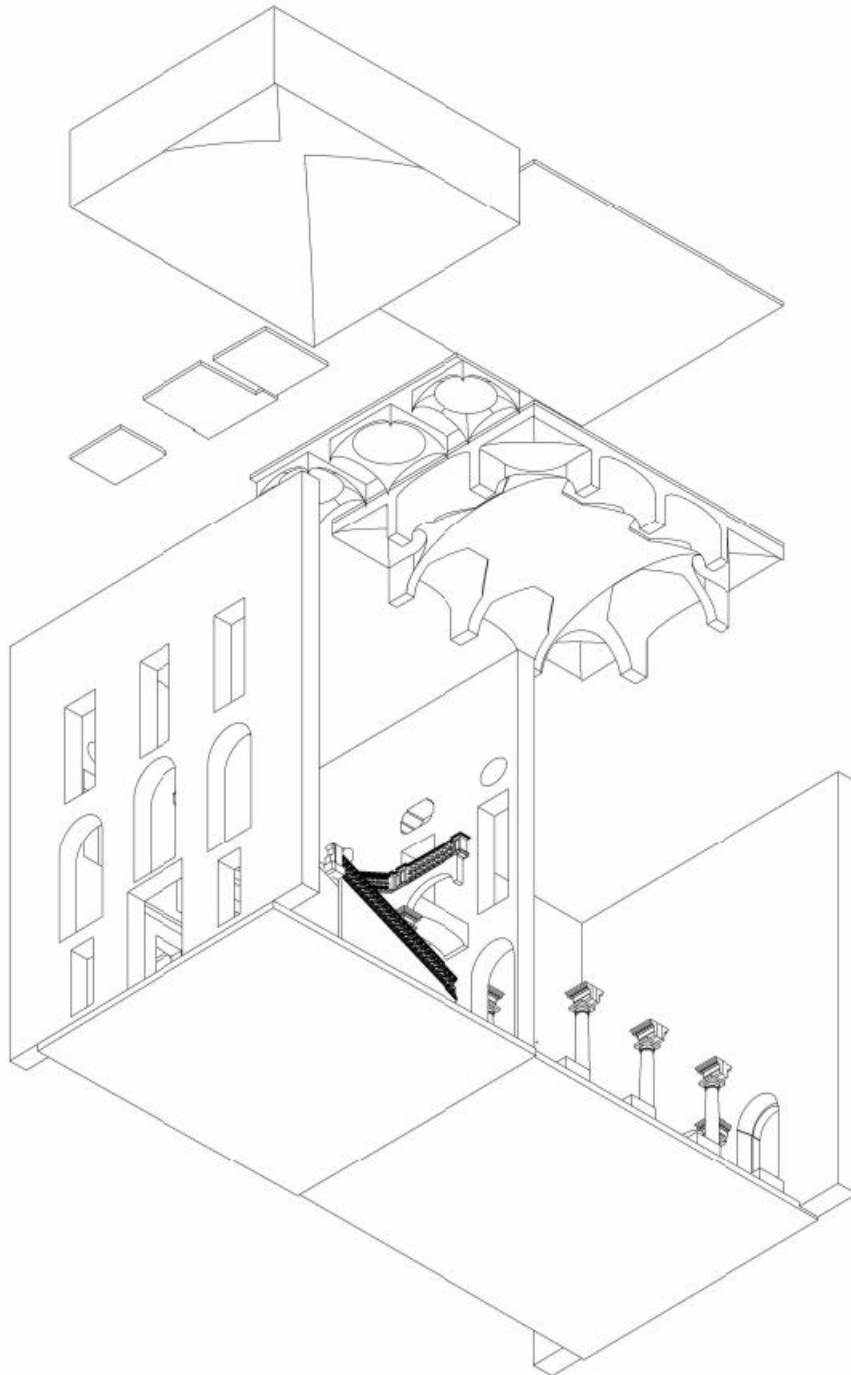


Figure 4-20 Floor displace

The following Table 4-3 summarizes the key differences of each dome and the modeling strategies adopted:

Table 4-3 Summary of geometric and modeling strategies across vault system

Feature/Vault	VAU_ATR	VAU_INT (between Staircase and Atrium)	Staircase Vault
	Barrel vault with cloister end (Structure)+edge or corner lunette(not single barrel shape)	Central vault(Type 1): Ellipsoidal pendentive Lateral vaults(Type 2): So-called “spherical panels built on arcs of variable shape”	Barrel vault with cloister end (clear shape)
Control points used	None (mainly by axis)	4	6
Surface/solid strategy	Loft, Extrude	revolve,Patch	Extrude, Loft by rule
Family type	Metric Generic Model floor-based		
Host and cut geometry	Floor and Cut with free-form void (Transformed by geometry), semi-auto offset and cut by manual.		
Challenge	The judgment is made based on structural logic modeling or based on surface fit.	The form is simple, but the geometric definition is not unique.	Standard historical structure, with highly consistent geometric/surface representation

Note: Family type and host strategy are shared across all three vaults, enabling consistent integration and Cut Geometry operations within the same BIM framework. The following sections (4.2.2 to 4.2.4) provide a detailed account of the modeling process for each vault type respectively - VAU_ATR, VAU_INT, and VAU_STA - focusing on their geometric segmentation, parameter framework, and implementation through floor-hosted void families.

4.2.2 Vault atrium(VAU_ATR)

Final Modeling Outcome

The final model of VAU_ATR (Figure 4-22) reproduces the geometry of the vault with precision. Each void family instance was set up with shared parameters, so that it records not only the geometric rules but also semantic data such as orthophotos or linked drawings. In this way, the model can be used for both visual checks and documentation tasks. Because the voids are organized as independent families, their position, geometry, and materials can be modified later without affecting the rest of the assembly. This structure makes the system easier to maintain and allows updates

or replacements when new survey data becomes available. However, as the focus of this study is on the modeling process, these aspects are not further discussed here.

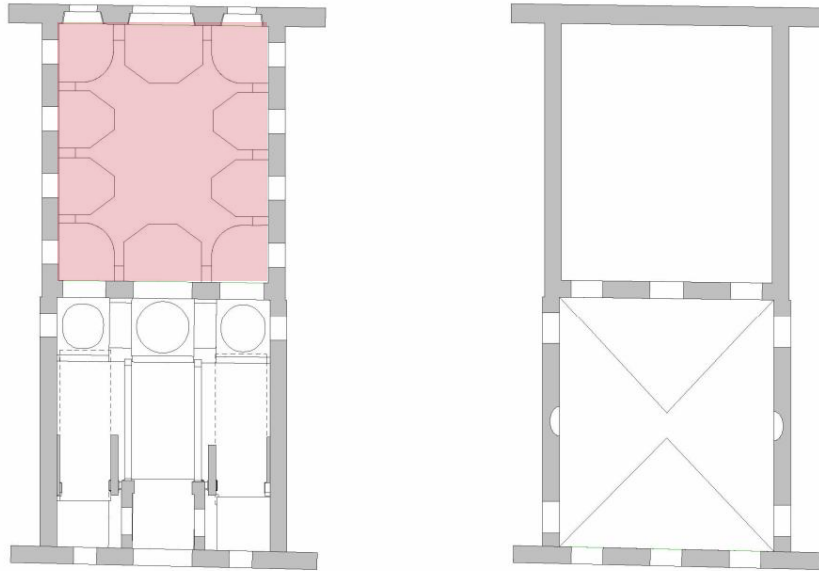


Figure 4-21 VAU_ATR in section and reflecting ceiling plan

Figure 4-22 and Figure 4-23 illustrate the outcome. The lunettes connect with the central barrel through cut operations that follow shared reference planes, keeping the arrises continuous and the junctions clear. Since each void remains modular, adjustments can be made with limited disruption. Vaults and other components follow the same type logic, which means that the Dynamo script/node can be reused in different projects by simply changing the parameter inputs. This ensures that the reconstruction is accurate for the present study while remaining adaptable for future heritage modeling.

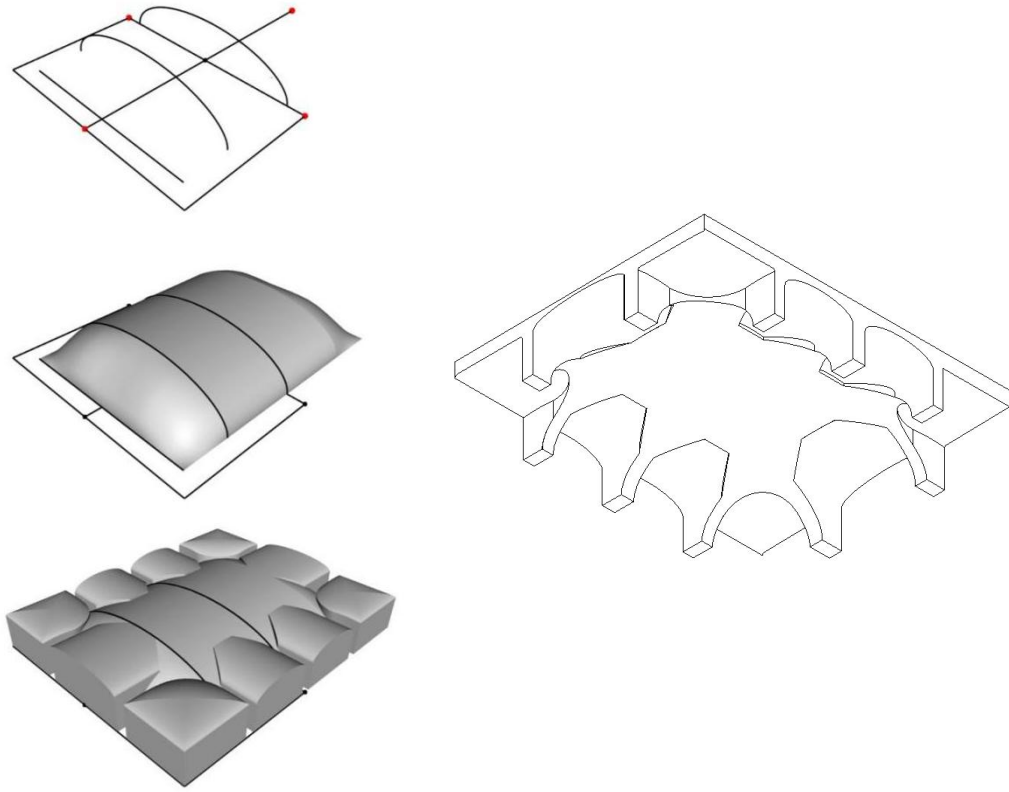


Figure 4-22 Generation of VAU_ATR as solid(left) and VAU_ATR Result Modeled with Floor Hosts and Cutting Voids(right)

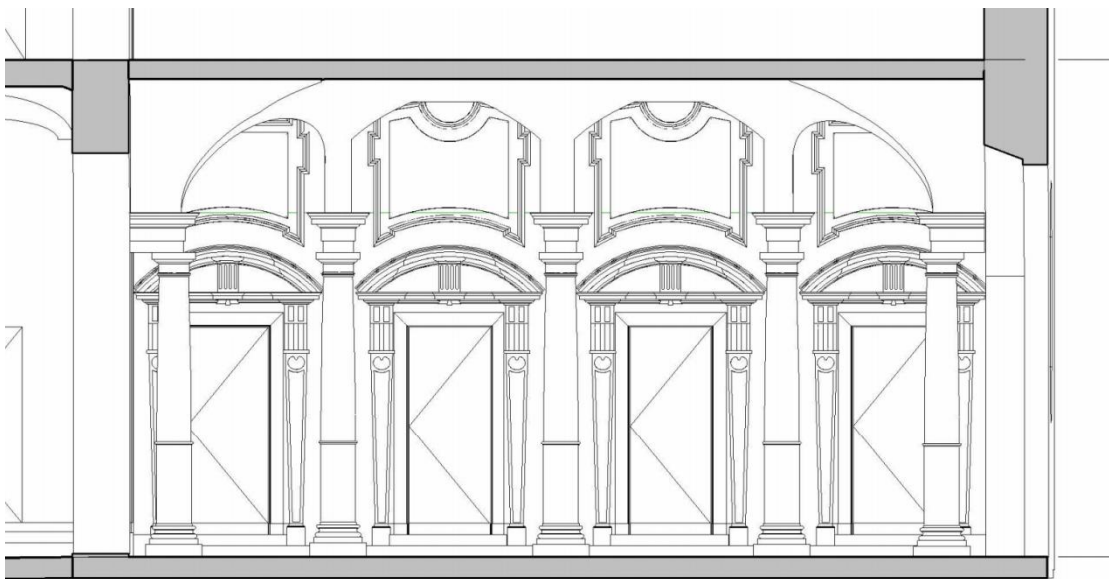


Figure 4-23 VAU_ATR Result in section

Geometric Composition

The vault solid was reconstructed based on orthophoto and point cloud. Its geometry can be described as a central barrel vault with ten lunettes surrounding. As the structure consists of multiple repeated components, the strategy of modeling is to create the lunette and the barrel separately and transform as void forms. Especially for lunettes, were created separately but generated in series(see section Batch Creation of Void Family with Dynamo).

Figure 4-24 shows the segmented point cloud of the VAU_ATR, which could be distinguished clearly of the components. This segmentation result supports the subsequent classification and geometric abstraction.

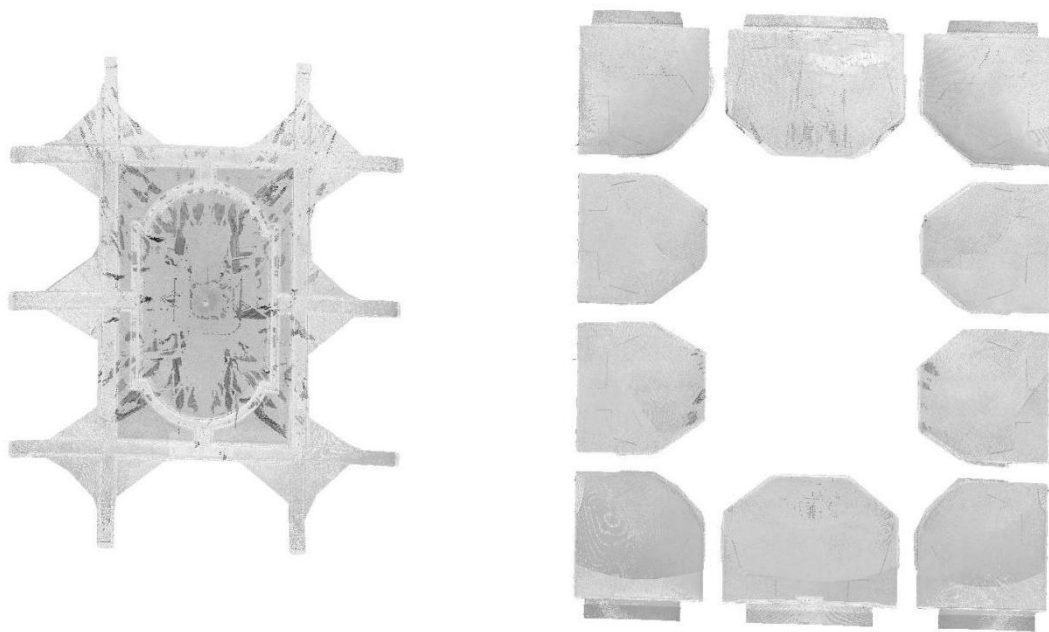


Figure 4-24 Segmented PC: barrel with rib(left), 10 lunette(right)

Rib profile and Edge loft

Although the Vault barrel shown in the Figure 4-26 looks like a typical barrel vault in the section, the actual geometry is more complex. Orthographic section analysis and curve reconstruction using CAD system shows that the structure includes an additional edge extension, forming a loft-like surface, that connects to the central barrel.

These side forms resemble half pillow shaped geometries, with their vertical section connecting seamlessly to the semi-elliptical surface of the central barrel. Particularly, the top vector is not performed horizontal as usual, but oblique. Here the 'vector' refers to the direction of the intersection line between the pillow-like geometry and the plane formed by the barrels extrusion direction and the vertical axis. This oblique

vector results in a discontinuous curvature at the connection, where the tangents are not aligned.

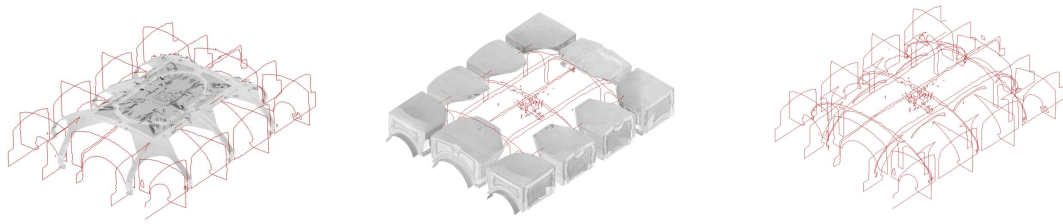


Figure 4-25 Sliced coutour in CloudCompare

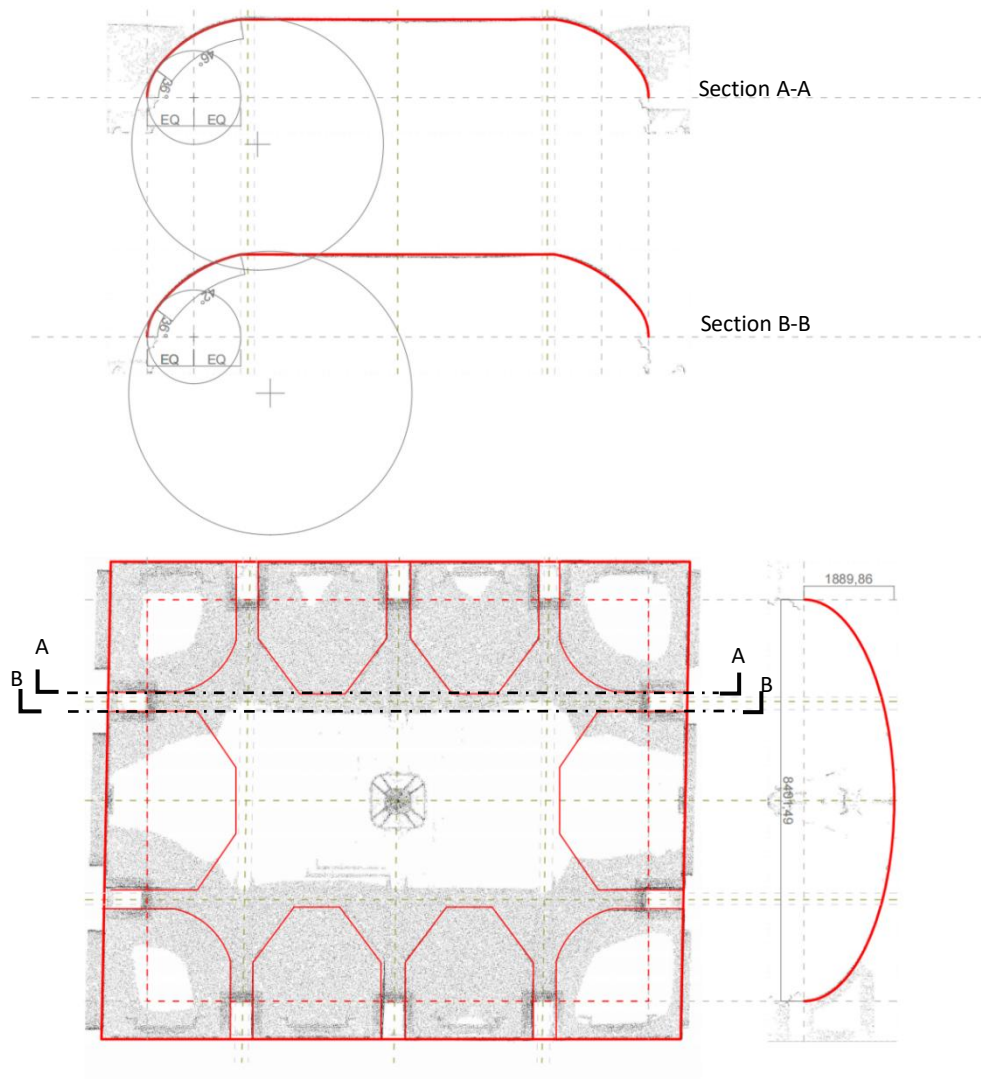


Figure 4-26 Form of barrel in plan and section

Further simulation in CAD system shows that the rib profile consists of two connected arcs. The first arc starts at the edge and sweep upwards by approximately 36° (by simulation), centered at the $1/4$ the length of the span. Then it transitions into the second arc with tangent continuity, forming a smooth curve. Although its shape is quite similar to the three-center or multi-center arched structure, it does not extend all the way to the very top of the arch. To be precise, it is more like a partial application of the multi-center logic, and its function is to adjust the degree of curvature between the edge-extending part and the cylindrical part. Structures like the three-center arch (also known as the handle arch or segmented arch), and elliptical vaults, which are multi-center arched structures, were widely used in Renaissance and Baroque architectural styles (Duvernoy, 2015). This approach makes the building's outline more gentle and ensures the structural integrity. These forms are typically composed of three or more arcs with different radii and centers, joined tangentially to form continuous profiles. This method is often found in historical vault construction, balanced structural feasibility with visual continuity. For a comprehensive discussion of multi-centered geometry in architecture, see Migliari, R. (2009). *Geometria descrittiva - Volume II - Tecniche e applicazioni*, section 2.2.3 "profile ovale".

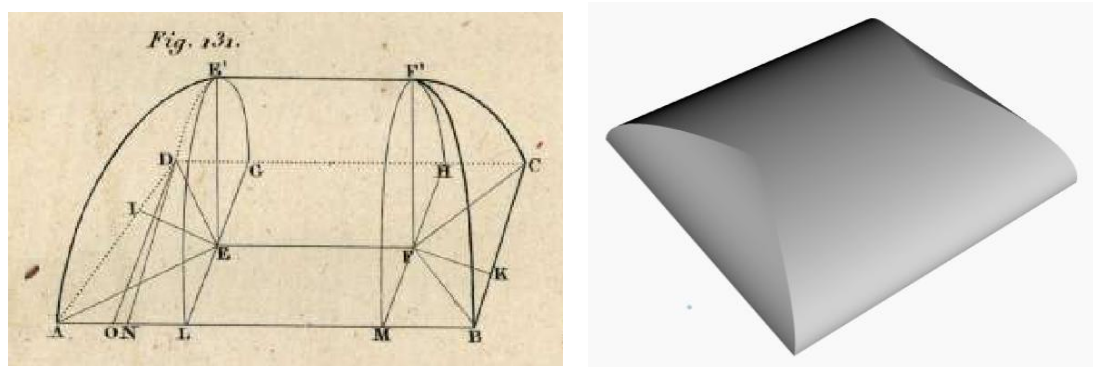


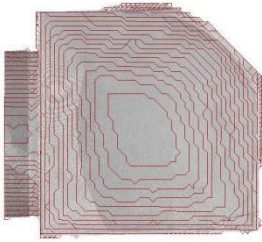
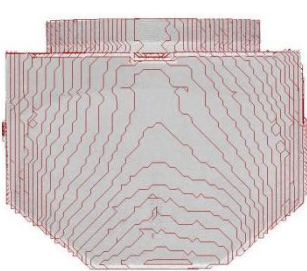
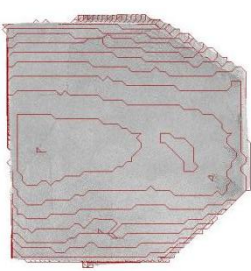
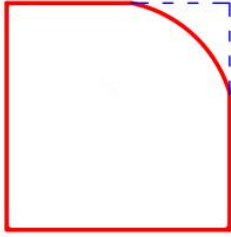

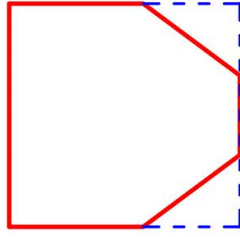
Figure 4-27 Left: Vault structure: barrel vault with pavilion end (*volta a botte con teste di padiglione*). reproduced from: Curioni, G. *Geometria pratica: applicata all'arte del costruttore. L'arte di fabbricare (III)*, Tav. 131.

Right: structure assumption in this project

Although this vault visually exhibits the characteristics of a "Barrel with conca" structure, from the perspective of structural rationality and historical construction practices, it is more likely that the original construction was a "Barrel with cloister ends" type (Figure 4-27 shows the reference of barrel with cloister and the right one shows which fits in this project. Its appearance might have been caused by the extensive use of decorative layers and the resulting concealment of the structure. (Elet, 2020)

Classification and parametric strategy for lunettes

Table 4-4 three types of lunette

Ortho image with Contour Export from CloudCompare			
Element Name	Lunette Corner	Lunette Edge-Vertical	Lunette Edge-Horizontal
Element Encoding	Void-Lun-C#	Void-Lun-V#	Void-Lun-H#
Amounts	4	2	4
Geometric analysis			
Parameter control in Dynamo (Not represented in family)	Length/Width Height Rise Fillet Radio (Except the positioning parameter)	Length/Width Height Rise Chamber X/Y (Except the positioning parameter)	Length/Width Height Rise Chamber X/Y (Except the positioning parameter)

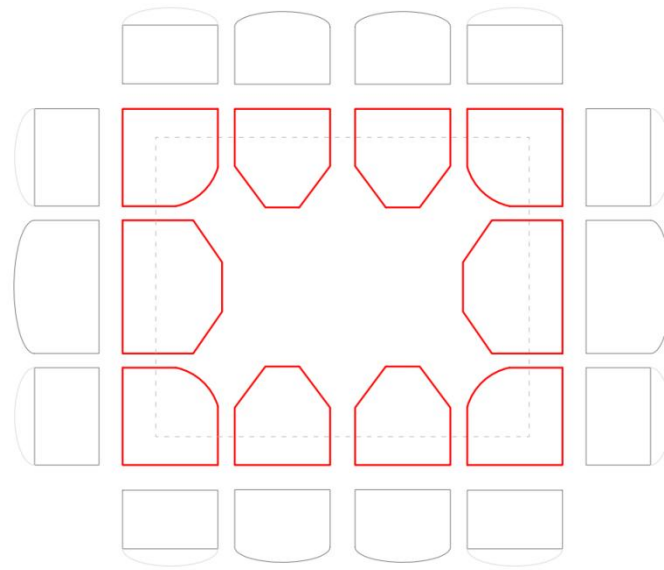


Figure 4-28 Form of lunette in plan

Based on geometric characteristics(Figure 4-28), the lunettes were classified into three types(Table 4-4): corner, edge-horizontal, and edge-vertical. Each type has slightly different parameters to control, including length/width, height, lift amount, and such as "chamfer X/Y" circular radius or deformation values. To facilitate model management and data extraction, all void forms have been assigned shared parameters, such as "graphic", which enables linking of orthophotos corresponding to each segmented element. The classification logic, encoding, and parameter are detailed in the accompanying tables and diagrams.

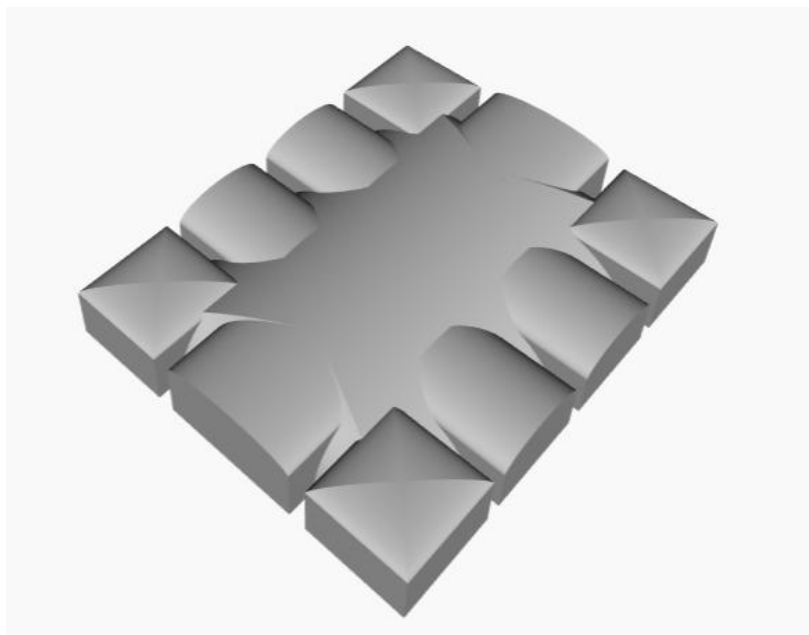


Figure 4-29 Structural hypothesis of the whole vault without stucco, derived from Dynamo showing the underlying geometry used for parametric reconstruction.

Batch Creation of Void lunette Families with Dynamo

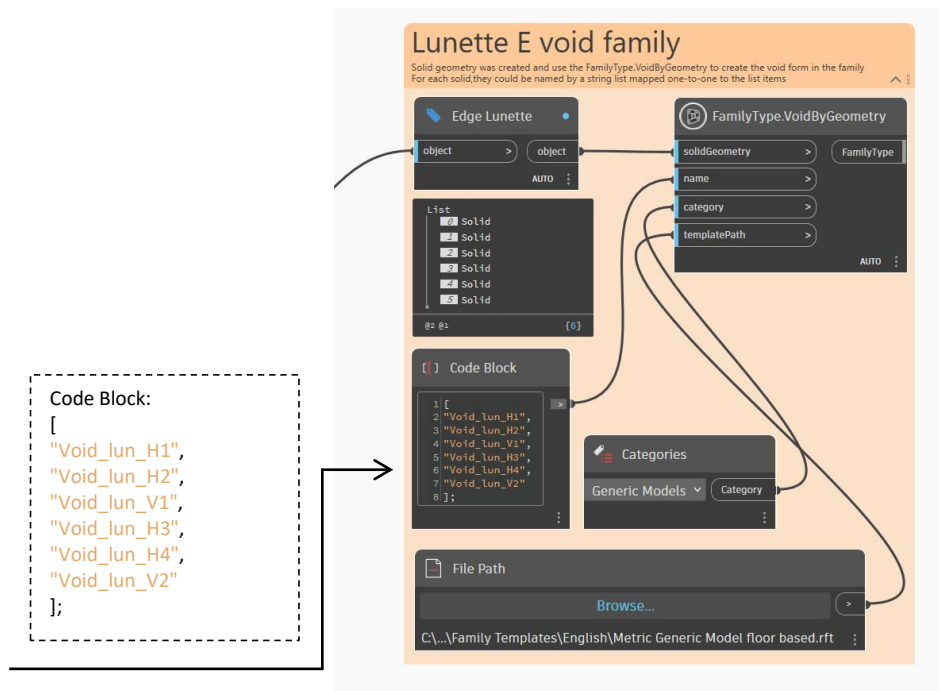


Figure 4-30 batch-created lunette family

To avoid manually creating numerous family files and to maintain consistent spatial logic between components, Dynamo was used for batch generation of void families. A predefined list of string names (e.g., "Void_Lun_H1", "Void_Lun_V2") enabled the mapping of each void element to its designated location. The Dynamo graph manages the iteration process, spatial transformations, and family creation, culminating in the use of the Dynamo node(FamilyTypes.VoidByGeometry) to convert solid geometries into family instances. Once generated, these void families can be loaded into the Revit project in batches and managed systematically by their names and indices²¹.

Figure 4-31 illustrates the complete Dynamo graph developed for automating the modeling of vault elements within the atrium. The workflow begins with the selection of reference elements from the Revit project environment, which serve to locate each vault geometry precisely in 3D space. A central point is established as the coordinate base, from which transformation parameters for each element are derived. The core of the process involves list structuring and batch logic operations: geometries are categorized into corner lunettes, edge lunettes, and the central barrel, each undergoing filtering, translation, and alignment. These steps ensure that each geometry is spatially accurate before conversion. Finally, the solids are passed

²¹ Note: The term "indices" in this context refers not only to the naming convention (e.g., "H1", "V2"), but also to the positional identifiers used in the Dynamo list management process. During batch generation of void families, a series of list transformations - such as Transpose, FirstItem, Combine, Join, and Map - are applied. Maintaining a consistent and traceable index structure is essential for ensuring that each geometric element is matched with the correct spatial transformation, host alignment, and family instance. The index effectively links geometry, placement logic, and family output across the entire vault system.

through the node(FamilyTypes.VoidByGeometry) to generate separate family files. This structure enables scalable, repeatable generation of void forms, although coordinate resetting at the family level necessitates additional handling at the placement stage. The modularity of this graph ensures that the logic can be extended to additional vault types or reused in similar projects.

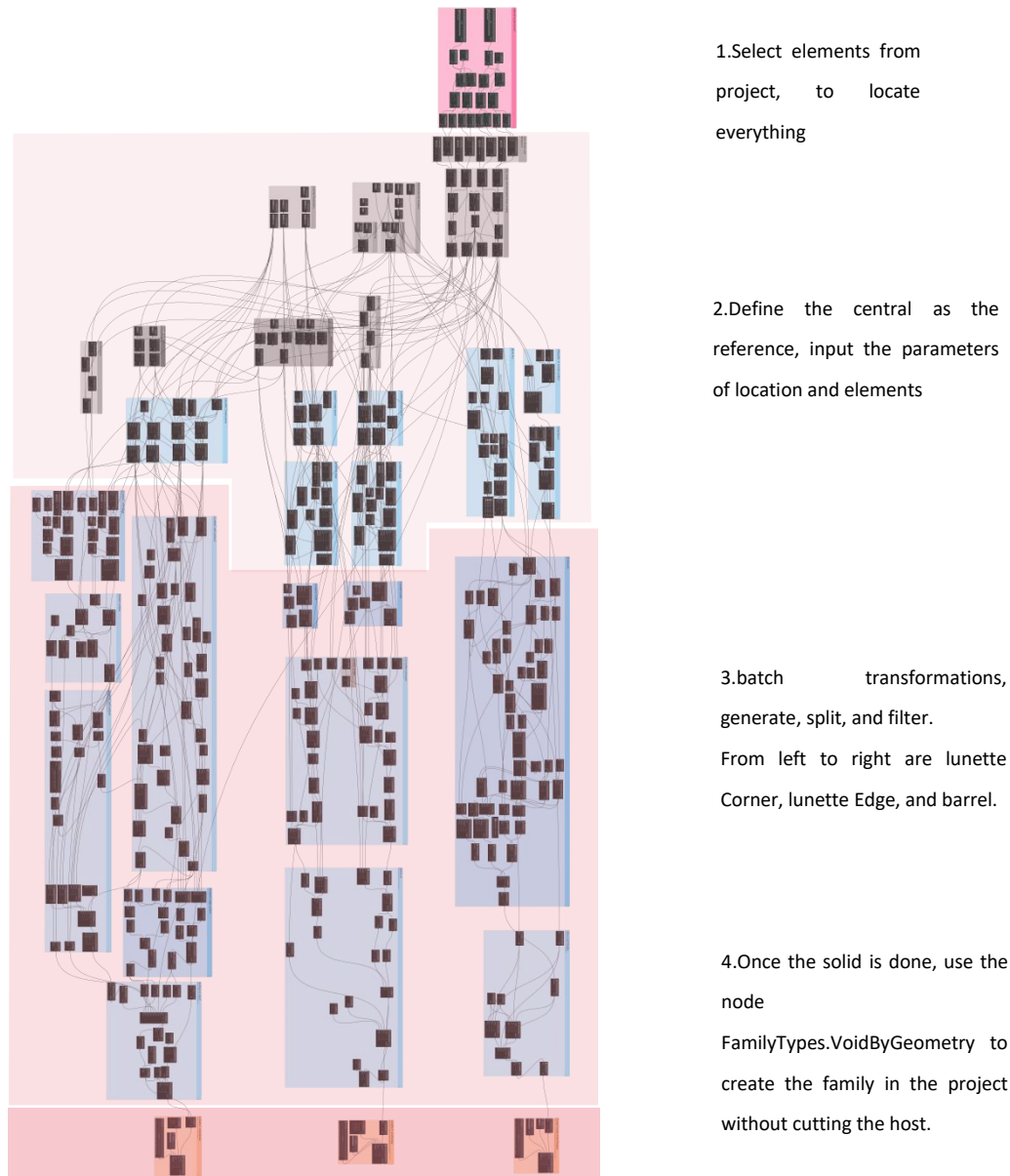


Figure 4-31 Dynamo Nodes of VAU_ATR

Integration of Floor Hosts and Manual Cut Workflow

For modeling stability, all void families were created using the floor-based family template. Although this limits the ability to rotate instances in the project, it greatly enhances placement precision and host consistency. Each void is embedded into a floor host whose thickness matches its horizontal projection. The floor provides

rendering material, structural layer control, and cutting support, while the void sculpts the spatial geometry.

Since the Revit API does not support programmatic execution of Cut Geometry operations, and because the internal coordinate reset issue²² caused by the (FamilyTypes.ByGeometry) node, the void was created over than the surface ($z \geq 0$), a PyRevit²³-assisted script was used to batch-move the voids into place. Unfortunately, while batch placement is achievable, the Cut operation still needs to be performed manually within the Revit UI. This limitation is acceptable in the current workflow, as the manual cut requires only a single click per instance and ensures precision. This approach is simple, efficient, and fully acceptable within the workflow.

²² A known issue has been reported in the Autodesk Revit Forum, where users noted that “solid imported by Dynamo doesn’t cut on sections” (Autodesk Community, 2024).

²³ pyRevit is an open-source IronPython-based scripting environment for Autodesk Revit. It extends Revit’s native interface by allowing custom toolsets, buttons, and scripts ... is widely used for automating repetitive tasks, batch processing families, and interacting with Revit’s API without full plugin development. (pyRevit, n.d.)

4.2.3 Vault intermediate(VAU_INT)

Final Modeling Outcome

The three vaults analysis in this study are located along the ground-floor corridor, connecting the atrium and the grand staircase to the Honor Hall(Figure 4-32). they act as transitional supports, connecting two major circulation spaces and precisely defined the spatial hierarchy.

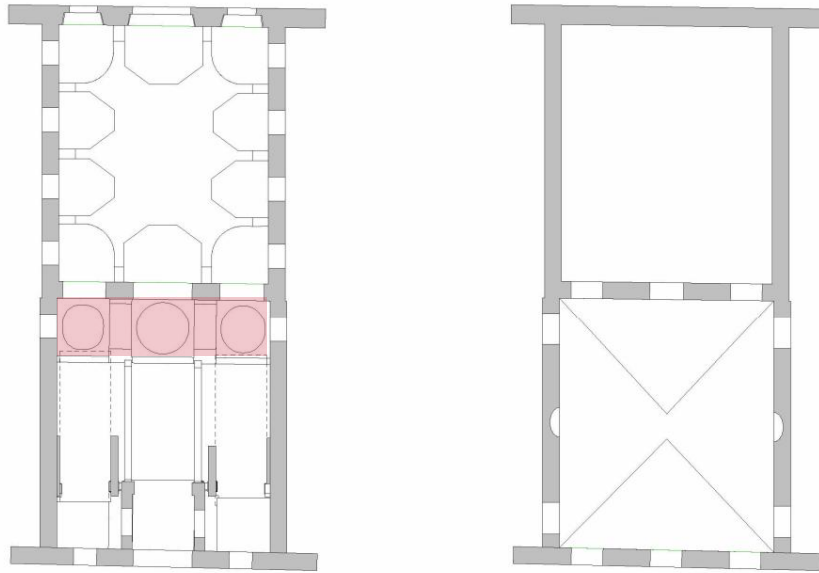


Figure 4-32 VAU_INT in reflecting ceiling plan

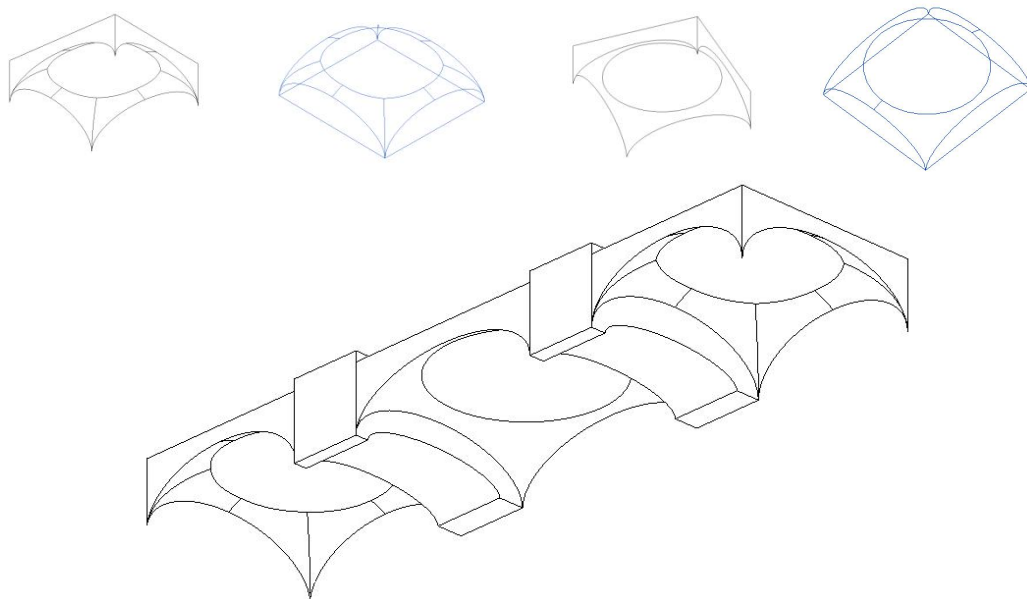


Figure 4-33 VAU_INT Type1&2 with Host(left) and without host(right)
Vault Combination together(lower)


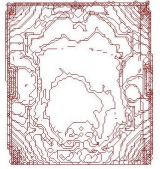
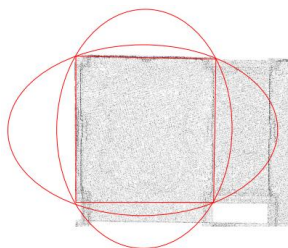
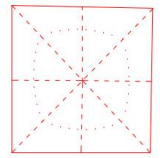
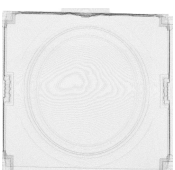
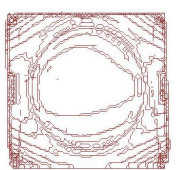
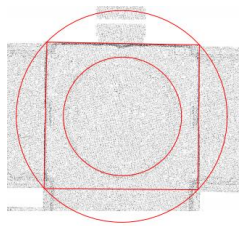
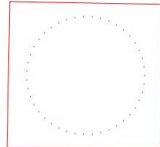
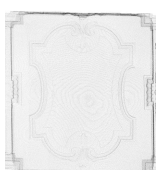
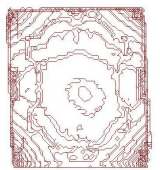
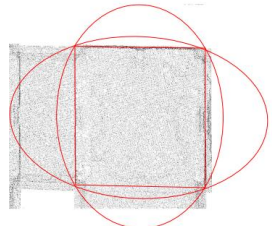
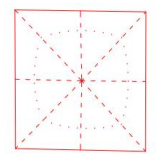
Geometric shape analysis

These three vault could be classified in 2 types based on geometry and complexity(Table 4-5).

Central vault(Type A): Ellipsoidal pendentive

Lateral vaults(Type B): So-called “spherical panels built on arcs of variable shape”

Table 4-5 Vault intermediate (VAU_INT)

	Pointcloud orthography	Contour cloudcompare	By Geometric analysis	Curve extract for modeling
TypeA: North Vault				
TypeB Central Vault				
TypeA: South Vault				

The central vault(TypeA) adopts a simple ellipsoidal pendentive geometry. It is constructed by intersecting an ellipsoid with a vertical prism extruded from a square base. This approach results in a smooth, regular surface that efficiently channels loads to the corners. In some historical cases, the surface is always covered with stucco work visually, which minimize the visibility of structure joins.

To simplify the generation, the geometry is revolved by a elliptical arc. The parametric data requires:

Four points location, for the base dimensions, the length of the side etc.

Vertical curvature: The minor axis of the elliptical arc.

The height of the stucco. (for visual)

The second vault type, as shown in the Table 4-5, is modeled with reference to the concept of “spherical panels built on arcs of variable shape” as described by Curioni in Algorithmic Modelling as a Key Tool for Ribbed Vault Geometry.(Bagnolo et al., 2022)

This approach differs from strictly hemispherical geometry by allowing the curvature of the arcs defining the vault's surface to vary, resulting in an ellipsoidal or hybrid vault form that remains geometrically controlled yet adapts to irregular constraints. The geometric analysis in this case identifies a curvature profile derived from an adjustable arc family, enabling a more flexible interpretation of historical vaulting strategies. The parametric data requires:

- Four points location,
- a diagonal arc across the bay
- a central apex arc
- four vertical edge arcs rising from corners.
- The height of the stucco.

Curve Extraction and Surface Construction

Instead of following a conventional scan-to-mesh workflow or slice-based lofting from raw sections, this study adopts a method based on geometric interpretation. Curves are not extracted at fixed intervals. They are chosen with reference to architectural rules, such as structural axes, spatial symmetry, and known vault types. The aim is to rebuild not only the surfaces, but also the underlying geometric logic, so that the model can be reused and kept consistent within a parametric framework.

For Type A (ellipsoidal pendentive vaults), the process begins with three points that define a circle on the horizontal plane. This circle provides the center for an elliptical arc. The major radius comes directly from the circle, while the minor radius is set by a fourth, higher point that marks the vault's apex. With these references, an arc is revolved around the vertical axis to create the main vault body. The interior volume is then isolated by extruding the base rectangle upward and intersecting it with the revolved shape. Finally, a horizontal cut is applied at the height of the stucco layer to trim the top and obtain the final volume.

For Type B (variable-curvature vaults), three types of elliptical arcs are extracted from the same four corner points:

- Four vertical elliptical arcs rising from the base, one at each corner;
- A diagonal elliptical arc, forming a 90° sector, with its major axis equal to the length from center to corners, and its minor axis inferred from the estimated vault height;
- A central apex arc, whose tangent aligns with the vectors from the vault center to the midpoints of each boundary arc.

These three arc types define a closed spatial boundary, which is then passed to node (Surface.ByPatch) to generate the vault shell.

For Type B, the three curves connect to each other as a closed loop (not a plane loop) and then insert for node(Surface.ByPatch). This operation produces a continuous curved surface representing the vault, which adapted to local geometrical variation while maintaining the coherence.

However, while the resulting surface appears correct within Dynamo geometry engine, it often encounters interoperability issues when transferred to Revit. In particular, surfaces created by node(Surface.ByPatch) or Solids created by node(Solid.ByJoinedSurface) may fail to register as valid BRep solids in Revit, due to the following issues:

- Boundary discontinuity or mismatched tangency,
- Invalid or open edge loops,
- Non-manifold topology²⁴

These issues prevent Revit from recognizing the geometry as a closed solid, and consequently void elements generated from such geometry cannot perform boolean operations (e.g., Cutting system families).

Indeed, surfaces generated through sweep or revolve maintain simpler topological structures and are more compatible with Revit's solid kernel. However, while such geometric primitives are effective for ensuring Revit compatibility, they are often insufficient for capturing the nuanced curvature and irregular forms found in historical architecture. Many vaulted systems, particularly those with asymmetrical springing, compound curvature, or non-canonical profiles, demand more expressive modeling strategies beyond sweep- or revolve-based forms.

In this context, freeform surface construction remains essential despite its limitations, as it allows for a more accurate semantic representation of the built form.

Integration of Floor Hosts and Manual Cut Workflow

To guarantee that the void is placed correctly and properly cuts through the host geometry, it is recommended to align the base of the void form with the base of the floor.

To ensure accurate alignment, it is better to pre-adjust the thickness of the floor in the family editing environment so that it matches the thickness of the host floor in the project before loading it. As described previously, edit the existing family already loaded in the project environment by node (FamilyType.VoidbyGeometry) (Instance not yet created), and reload it after modification. Then create the instance, insert it to ensure it to be hosted on the floor.

²⁴ While Autodesk does not provide fully detailed public specification for BRep validation in Revit, both developer experience and forum reports indicate that Revit's geometric kernel will reject solids that violate standard BRep integrity rules- such as closed manifold topology continuous surface normals, and valid edge loops. (Autodesk Community, 2023)

Vault Modeling Based on Reusable adaptive Point Logic

As mentioned in Section 3.2.3, adaptive components can deal with irregular geometry by means of adaptive points. However, Revit's system has a key limitation: the voids created inside adaptive families cannot perform Boolean cuts on system families. In this project, the adaptive component is therefore used only as a container. It organizes four reference points that mark the springing line and the lower zone of the vault, but it does not contain any real void that could cut the host. To overcome this problem, a Dynamo node group was developed. By selecting a new set of four adaptive points inside the project, the same cutting logic can be applied to different quadrilateral vault configurations. This makes it possible to reuse the modeling approach and reduces the need for manual rework. The Dynamo routine reads the four points as its input, generates the base geometry, and then creates the voids that cut the system families. In this way, both accuracy and parametric control are maintained.

The Dynamo graph(Figure 4-34) begins by selecting four adaptive points from a placed reference family (Select Model Elements), which serve as spatial anchors for the vault geometry.

From these points, boundary lines are generated by node(Line.ByStartPointEndPoint) and used to construct a reference circle by node(Circle.ByThreePoints). This circle guides the construction of elliptical arcs by node(EllipseArc.ByPlaneRadiiAngles), which define the curvature of the vault shell. The resulting arc is revolved into a three-dimensional solid by node(Solid.ByRevolve), which is then trimmed using planar cutting surfaces by node(Geometry.Split) to isolate the internal vault volume. Finally, the processed solid is converted into a Revit-compatible void element using node(FamilyType.VoidByGeometry), with its family category and template path specified for correct insertion.

They are following this logic:

1. Geometry Input and positioning
2. Coordinate reset and alignment
3. Geometry construction
4. Void family conversion

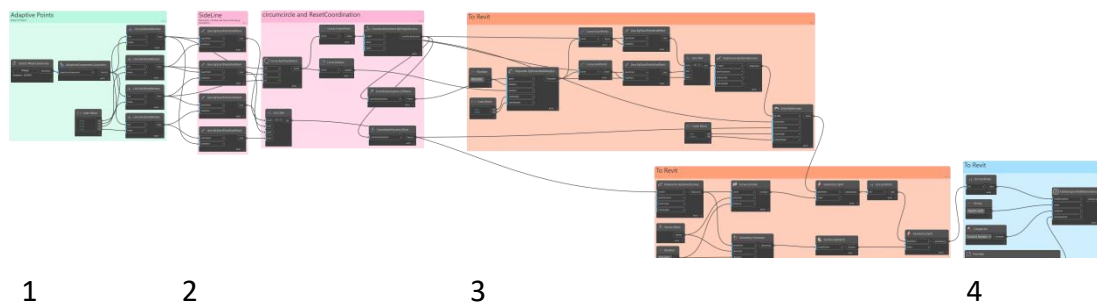


Figure 4-34 Dynamo Graph for Parametric Construction of Type A Vault

In this alternative Dynamo definition(Figure 4-35), the modeling process begins by selecting four adaptive points from a placed reference family (Select Model

Elements), which serve as geometric anchors. Straight lines are created between them using node(Line.ByStartPointEndPoint), followed by the generation of elliptical arcs node(EllipseArc.ByPlaneRadiiAngles) along both the diagonals and side edges. A central top arc is constructed via node(Arc.ByStartPointEndPointStartTangent), aligning tangentially with the spatial direction of the bounding curves. These three types of arcs are organized into a list structure and passed into node(Surface.ByPatch) to create a freeform surface bounded by the arcs. Multiple surfaces are then joined via node(Solid.ByJoinedSurfaces) to form a closed solid, which is trimmed using node(Geometry.Split) and finally converted into a void family through node(FamilyType.VoidByGeometry), with its category and template path defined for integration into the Revit environment.

They are following this logic:

1. Geometry Input and positioning
2. Vector Construction and support geometry
3. Geometry construction
4. List management and Boolean preparation
5. Void family conversion

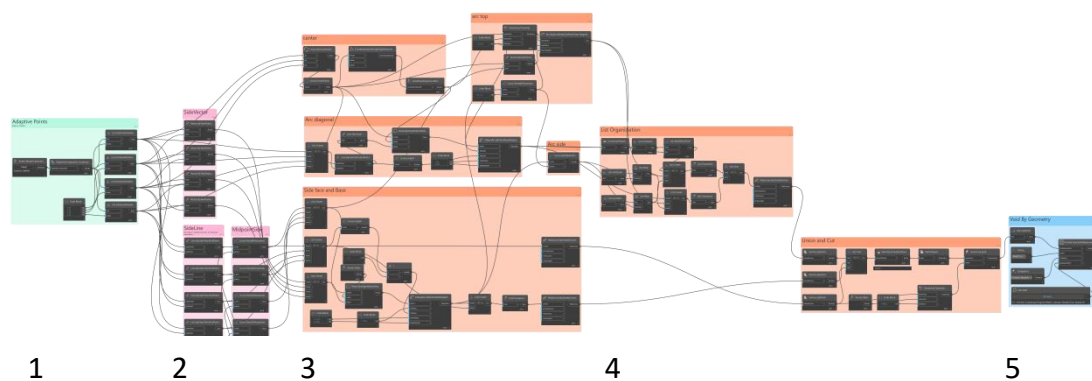


Figure 4-35 Dynamo Graph for Parametric Construction of Type B Vault

4.2.4 Vault Staircase(VAU_STA)

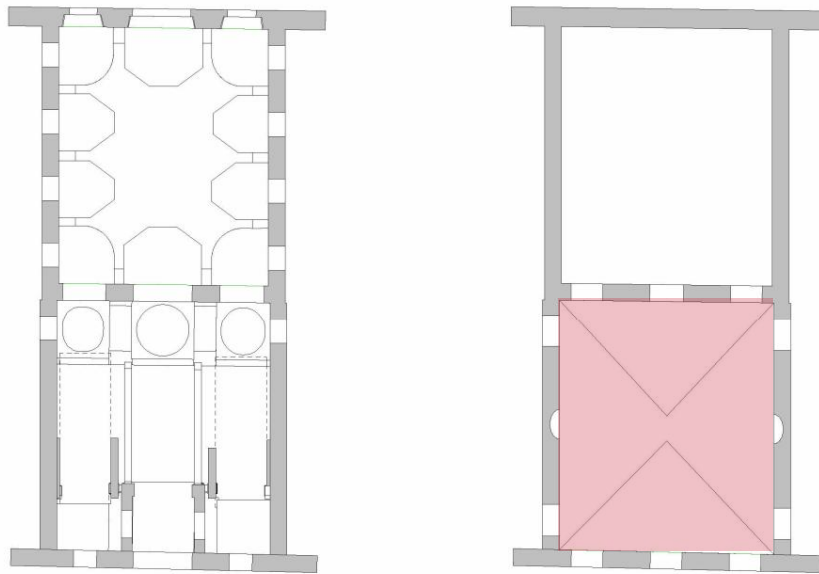


Figure 4-36 Vault staircase in reflecting ceiling plan

Final Modeling Outcome

The final model of the staircase vault successfully integrates both the cloistered and barrel components into a coherent and geometrically faithful digital reconstruction. Through the combination of sweep, extrusion, and loft operations, the hybrid curvature of the vault was accurately captured. The use of a six-point adaptive family enabled precise placement of the reference geometry within the existing architectural context, with each point aligned directly to previously modeled structural elements such as walls and floor slabs.

Once generated, the solid was converted into a void form using node (FamilyTypes.VoidByGeometry), embedded within a floor-based family template to ensure system host interaction. After placement, a manual Cut Geometry operation was applied in the project environment to sculpt the ceiling slab, producing a crisp spatial articulation between vault and host.

The result achieves both visual continuity and material consistency with the surrounding architectural framework. The geometric logic is fully traceable, parameterized, and reproducible, and the modular family-based structure ensures maintainability and future extensibility. Together, these outcomes demonstrate that even geometrically readable vaults—such as cloistered barrel types—benefit from a structured, data-driven modeling approach when reconstructed within a BIM context.

Geometric shape analysis

Unlike the other vaults studied in this project, the ceiling of the staircase connecting the atrium to the Honor Room presents a geometry that is visually straightforward to interpret. Despite the presence of elaborate stucco decoration, the underlying structure remains clearly legible and can be confidently identified as a barrel vault with cloister ends. While this initial assessment is visually evident, it was further verified through sectional analysis, using slices created from as-built orthophotos and geometric simulation performed in CAD system.

The horizontal cross-section consists of two elliptical arcs connected by a central straight segment, forming a continuous and smooth projection that conforms to the typical geometry of cloister vaults. These vaults can be understood as the result of two intersecting barrel vaults, producing a lowered intersection profile—a condition that distinguishes them from groin vaults, where the intersection creates raised ridges.

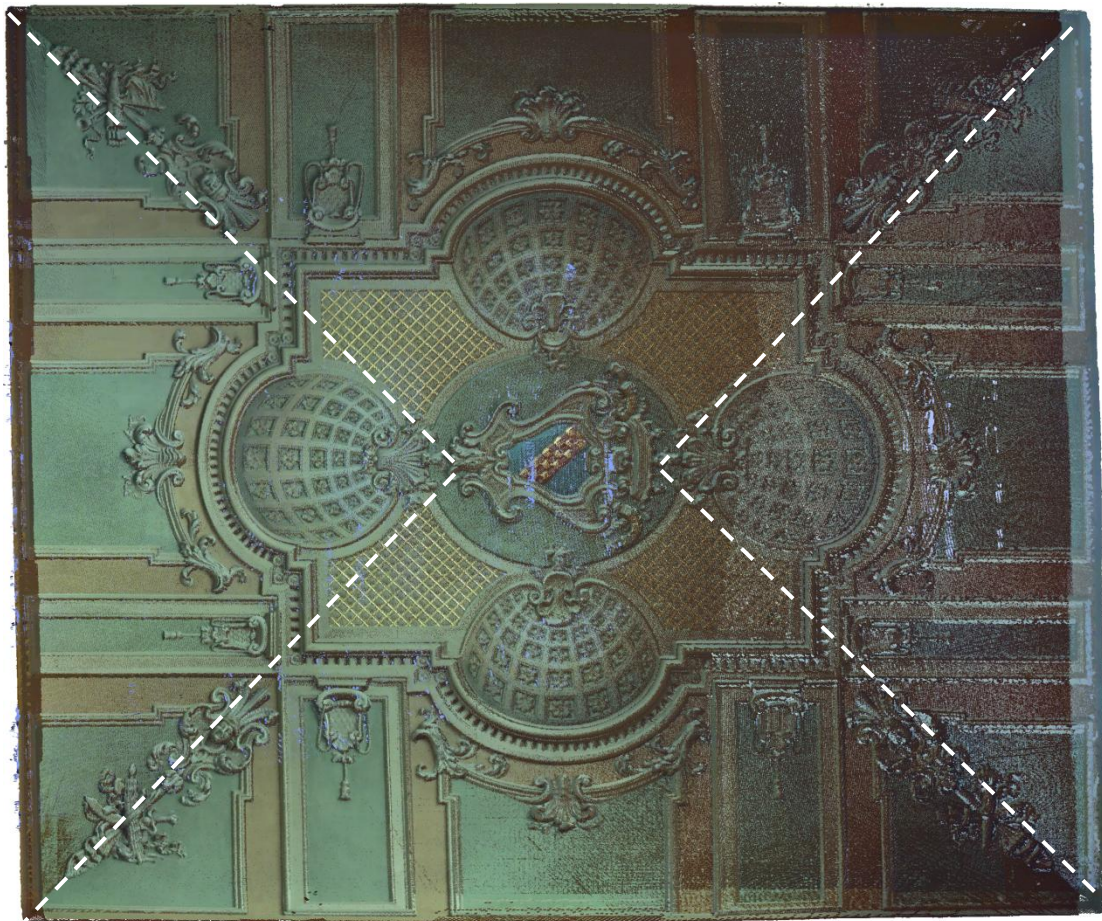


Figure 4-37 Clear line for cloister edge

Curve Extraction and Surface Construction

For traditional modeling methods, using standard regular quadrilaterals or rectangles makes it very easy to create arches.

In historic buildings, the base edges of vaults are rarely true rectangles. Several reasons explain this irregularity. Manual construction was never perfectly precise, and over time thrust and settlement have caused further deformation. In some cases, builders also made intentional adjustments to fit the vault into the surrounding structures. Because of these conditions, the four corner points of the springing line are usually not aligned on a single horizontal plane. Opposite sides may differ in length, or even show slight curvature. For this reason, standard rectangular templates cannot be applied. Instead, the base edge is defined by four independent points that form an irregular quadrilateral. This point-driven strategy follows the geometry visible in the point cloud and gives a representation closer to the actual built form.

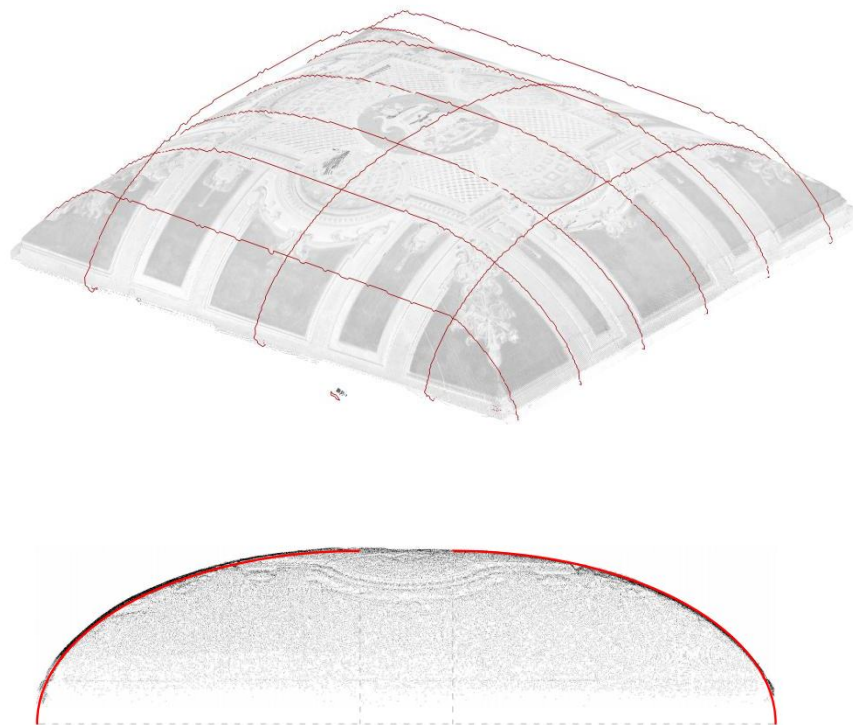


Figure 4-38 Geometric decomposition of Vault staircase section

The modeling of this vault, despite its geometrically recognizable form, still requires a composite construction logic to reflect its differentiated surface structure. The cloistered portion of the vault was modeled using sweep and extrusion, while the barrel segment was generated through Loft by Rule. Specifically, the top horizontal line and the two long base edges were extracted as the first guiding set, and the two lateral curves formed the second guiding set, enabling the surface to be split and generated as two symmetrical lofted parts.

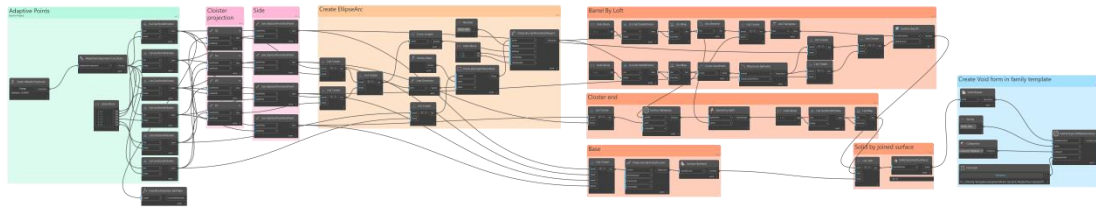


Figure 4-39 Dynamo node for vault staircase

The essential geometric parameters required for reconstruction are limited to six points: the four corner points of the base, and the projection and height of the apex point. The base points determine the overall plan footprint, while the projection point of the apex on the base plane (XY) and its vertical offset (Z) define the dome's rise. The apex height was acquired either from orthographic projection in CAD system or directly measured in Revit using native annotation tools.

As a result, the Generic Model Adaptive family described earlier can still be used, but with one small change: the adaptive component is expanded from four points to six. The order of these points is fixed inside the family, so Dynamo can recognize them consistently during reconstruction.

In this context, placing the adaptive points is relatively straightforward. The project already contains an architectural framework, including walls and floor slabs. Each reference point can therefore be aligned directly to these system families with the Align (AL) command or the standard snapping tools in Revit. This reduces uncertainty in positioning and makes the overall process more accurate. In practice, the adaptive component functions as a spatial container that relies on the existing geometry, giving the modeler an intuitive and repeatable way to set the points.

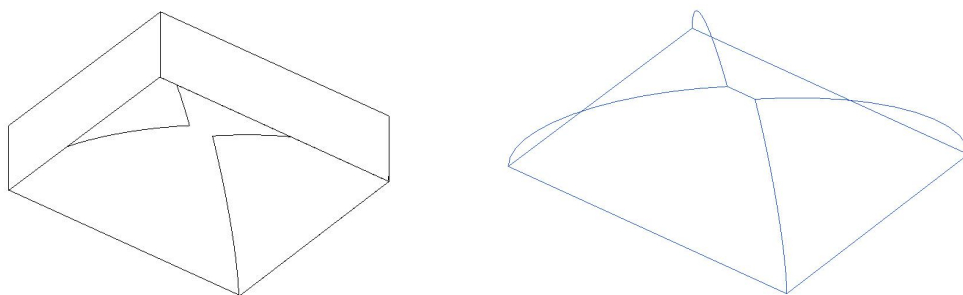


Figure 4-40 Vault Staircase with Host(left) and without host(right)

4.3 Railing

4.3.1 Segmented Elements By Revit system

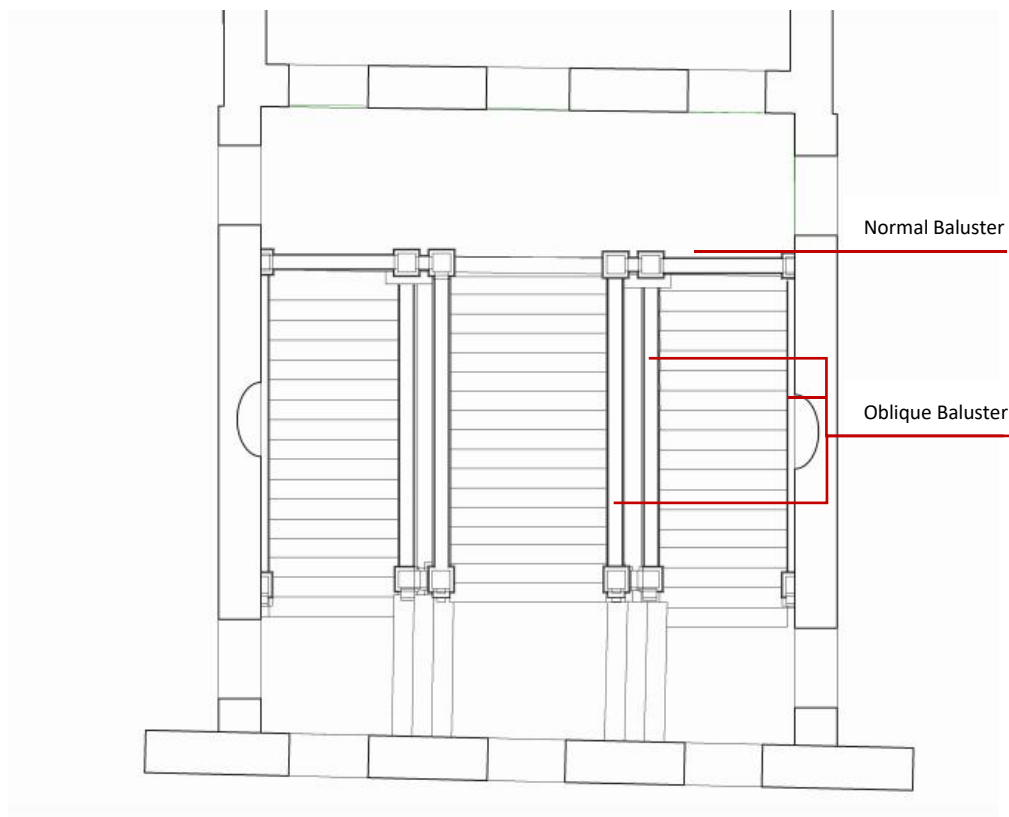
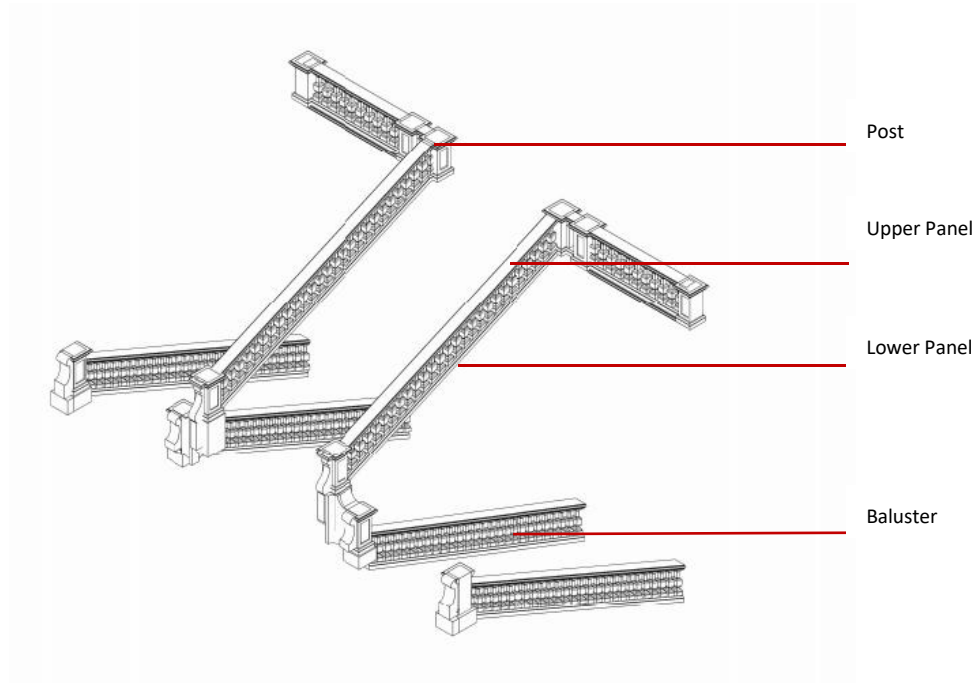


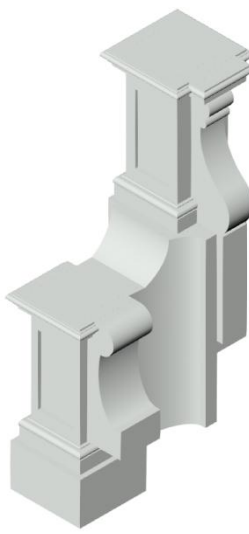



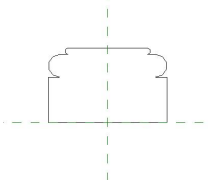
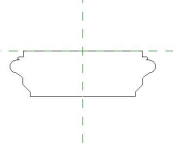


Figure 4-41 Location of each element

In this project, the balustrade system was composed of two types of balusters and four corner/end posts with slight geometric differences, in addition to the upper and lower panel elements. These were carefully positioned to follow the structural logic and historical references of the staircase. Figure 4-41 illustrates the distribution and placement of each element within the balustrade layout: Four posts at corners and ends, with minor variations to adapt to the local geometry; Upper and lower panel; Oblique and Normal baluster. This combination aimed to maintain both geometric consistency and visual coherence across the entire stair railing.

Table 4-6 Segmentation by the Revit Railing System

Element				
Family Template	Posts 1 Generic model	Posts 2 Generic model	Posts 3 Generic model	Posts 4 Generic model
Element				
Family Template	Oblique baluster Baluster	Normal baluster Baluster	Lower panel Baluster-Panel	Upper panel Baluster-Panel

To further clarify the organization of these elements within the Revit environment, the balustrade components were segmented according to the internal logic of the Railing System (Table 4-6). This classification reflects the three family categories introduced in Section 3.5.1: Balusters, including both the normal and oblique variants, were modeled as Baluster families; Panels were defined as Baluster-Panel families for the upper and lower horizontal elements, represented in practice as profile-based components; Posts were represented as Generic Model families, subdivided into four slightly different types to adapt to corner and end conditions.

It should be noted that, according to the Revit Railing System, the Baluster Post Family is normally part of the baluster system and should have been used in this context. However, in the staircase of Palazzo Barolo, the posts display a high degree of decorative complexity and contain multiple asymmetric ornamental attachments. While the system requires the posts to behave symmetrically within the overall balustrade layout, their individual geometry is not symmetrical. As a result, the Baluster Post family template could not be adopted without significant distortion. Instead, the posts were modeled as Generic Models, which allowed the overall symmetry of the railing system to be preserved while accommodating the individual irregularities of each post.

4.3.2 Normal Baluster and Oblique Baluster

Final modeling outcome

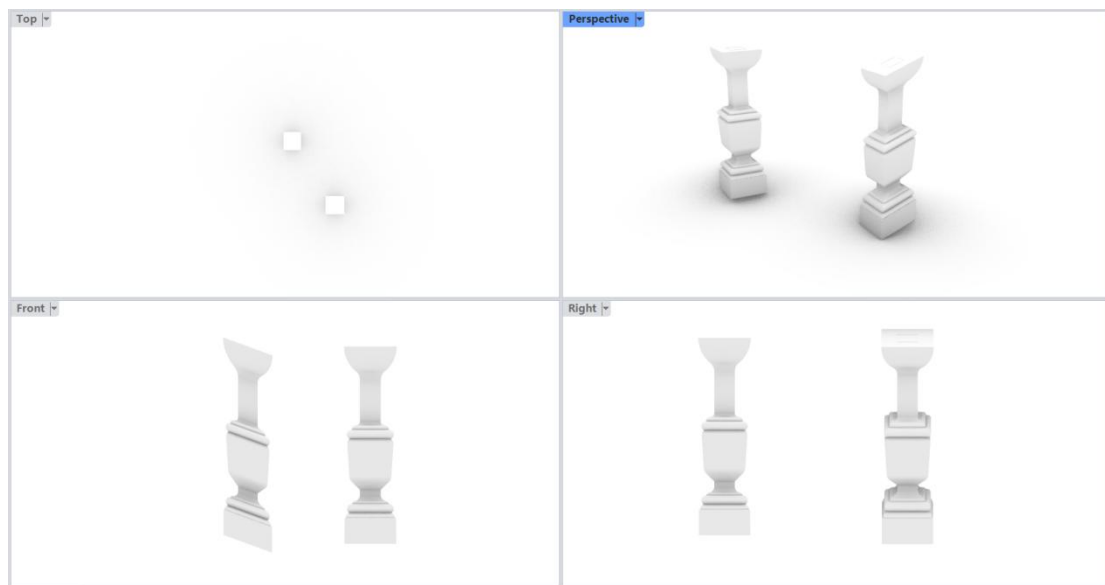


Figure 4-42 Outcome of two types of baluster in rhinoceros 8

The final modeling outcome of both the normal and the oblique balusters is shown in Figure 4-39, based on the parametric design process carried out in Rhinoceros 8.

Normal Baluster by revolve or sweep

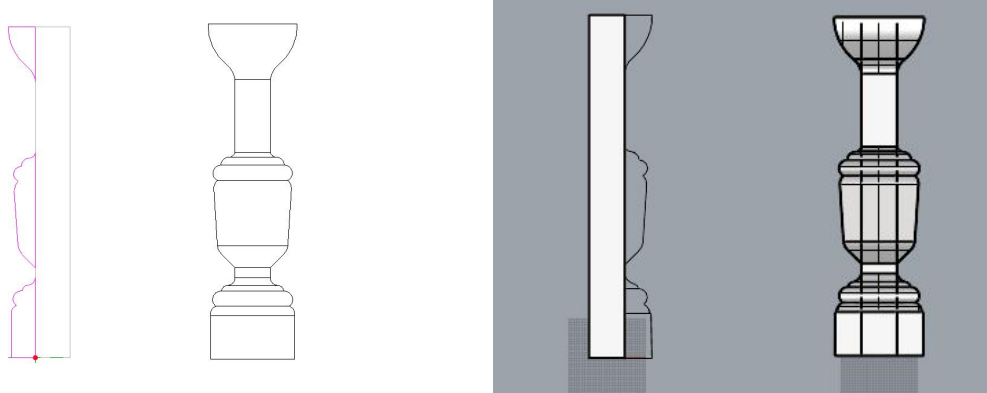


Figure 4-43 Normal Baluster Modeling by Sweep in Revit(left) and Rhinoceros(right)

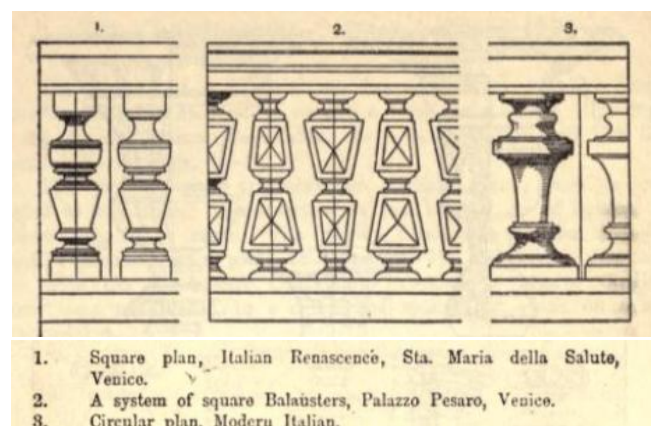


Figure 4-44 Examples of historical balustrade systems with square and circular profiles.

Left: square-plan balusters from the Italian Renaissance, Sta. Maria della Salute, Venice;

center: a system of square balusters at Palazzo Pesaro, Venice;

right: circular-plan balusters in modern Italian style.

reproduced from Meyer (1900, pp. 224 – 225).

Three method of Oblique Baluster

For the oblique post, it can be interpreted as an evolved form of the standard vertical post, using the same profile revolved around a vertical axis. However, the oblique version undergoes an additional transformation by rotating around the X-axis on the XZ-plane at an inclination of approximately 20–30 degrees (depends on the ramp). This results in a coordinate system where the XY and YZ planes remain perpendicular, while the XZ plane is no longer orthogonal. The initial geometry is then scaled proportionally within this transformed frame.

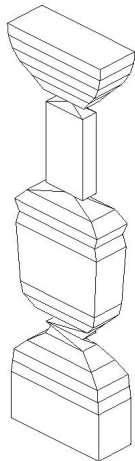
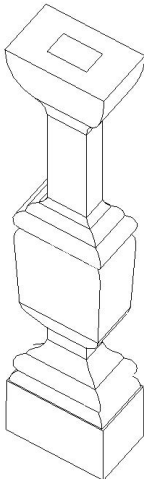
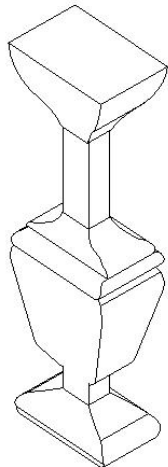
Due to the complexity of this transformation logic, achieving the most accurate parametric representation directly in Revit proved impractical. Therefore, Rhinoceros was employed to carry out the geometric transformation in a more flexible and precise way, by applying a simple shear or rotation operation to approximate the oblique shape.

In practice, three methods for generating the oblique baluster were evaluated (as explained in section 3.3.2):

- Contour slicing and lofting in Dynamo
- Shear transformation in Rhino/Grasshopper
- Void-based sweep or extrusion within Revit

Among these, the Rhino-based workflow was chosen for the final model, due to its better control of geometry precision and simpler management of the inclined coordinate system.

Table 4-7 Comparison with three modeling method of oblique baluster

Method	Contour slicing + loft (Dynamo)	Shear transform (Rhinoceros)	Void-based trimming (Revit)
Result			
Advantages	Simple to automate; parametric in Dynamo	Accurate; preserves NURBS; good control	Quick; stays within Revit environment
Limitation	Low detail if slices are few; cannot directly shear	Requires fine subdivision of profiles for sweep to shear	Approximate result; possible curve distortions
Interoperability	Needs SAT export/import	Needs SAT export/import	No external export

The characteristics of the three alternative methods are summarized in Table 4-7: Contour slicing and lofting in Dynamo: This method is comparable to extracting slices of a mesh generated from defined positioning points, then reconstructing a smooth surface through lofting. Since Dynamo does not support shear transformation

directly, the slicing strategy is adopted instead. However, if too few slices are generated, the extracted contour features will be insufficient, causing the resulting loft surface to lose critical details and fail to capture the decorative features of mouldings. Additionally, this method requires exporting the geometry as SAT and reimporting it into Revit for further integration.

Shear transformation in Rhino/Grasshopper: This approach is accurate and effective because Rhino preserves the NURBS geometry throughout the shear transformation, maintaining perfect curve continuity. By exporting and importing via SAT, the NURBS characteristics of the profiles are also retained within Revit. The main limitation is that the initial sweep geometry must be subdivided with sufficient detail (not slicing contours, but rather the detailed segmentation of the moulding profiles) to guarantee a successful shear transformation.

Void-based sweep or extrusion within Revit: This method can be performed entirely in the family environment of Revit. A solid is extruded and trimmed by a void extrusion manually, without requiring Dynamo or Rhino. The process is simple and direct, which only provides an approximate representation and cannot reliably preserve the smooth curves of the decorative profiles. And in many case it results in geometrically distorted or jagged edges.

After testing this comparative evaluation, the Rhino-based shear transformation was selected as the final modeling solution. It gave the best results in preserving the architectural moulding elements and allowed the decorative features to be reconstructed with greater accuracy.

4.4 Outcome after all

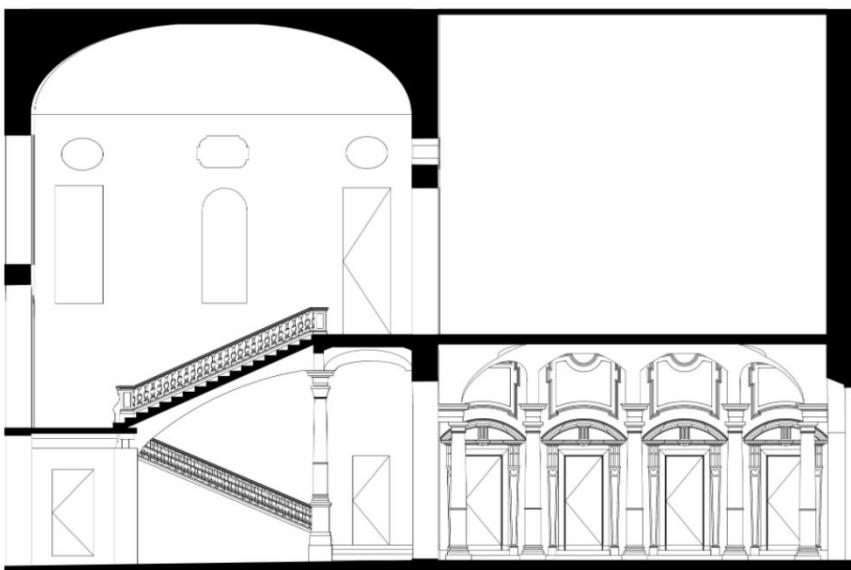
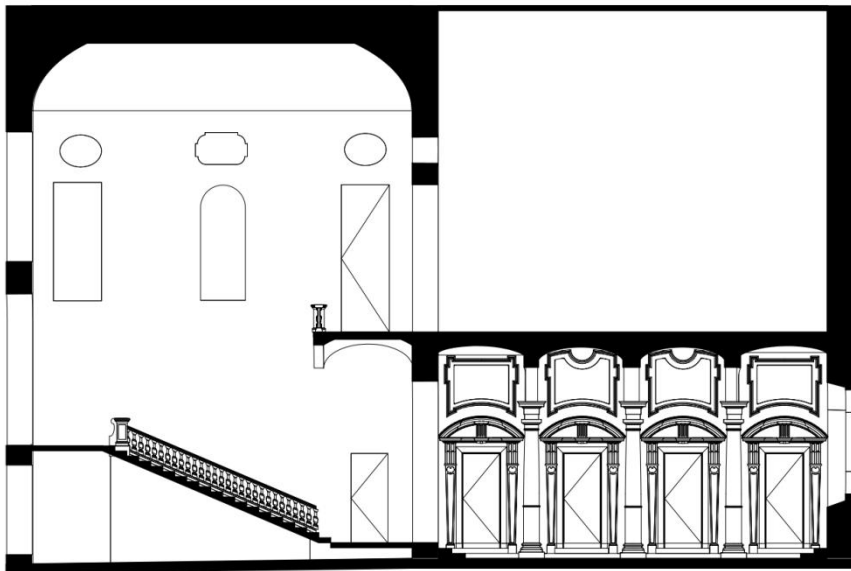
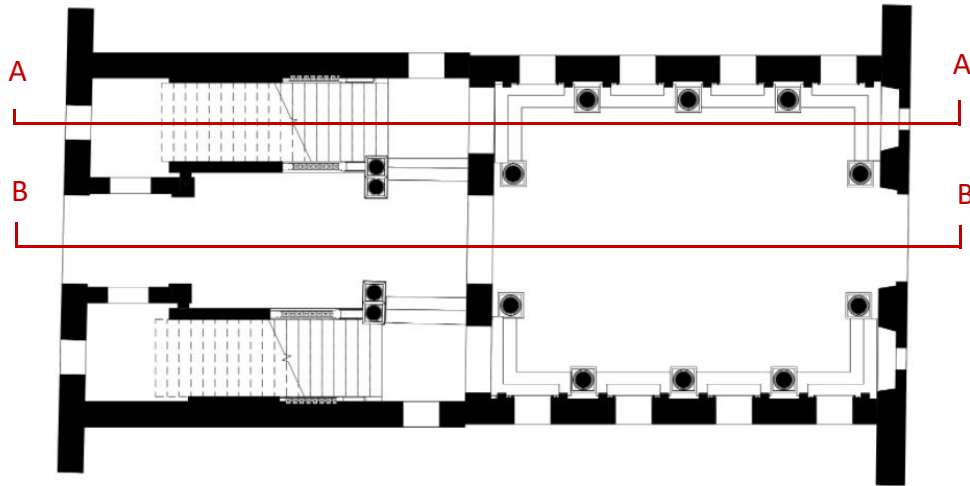


Figure 4-45 Section of the outome



Figure 4-46 Render in Atrium

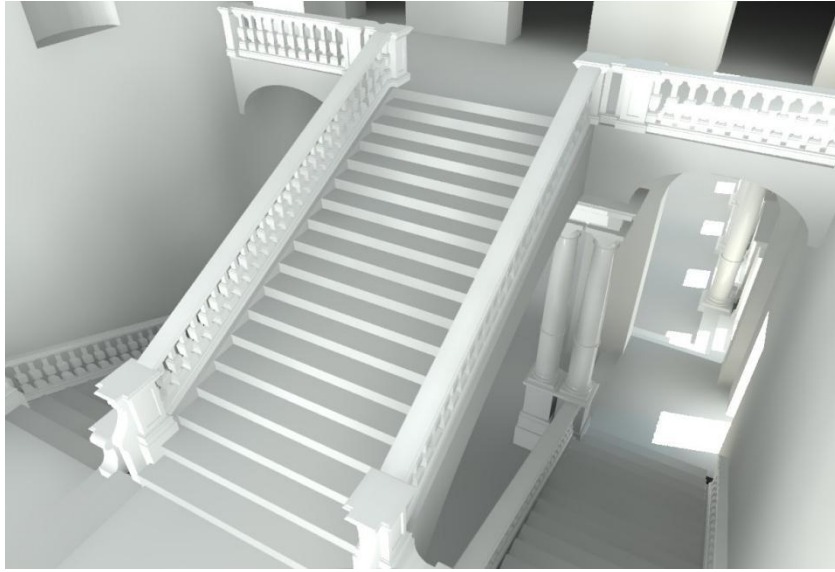


Figure 4-47 Render in staircase

5. Discussion

The modeling strategy for the column clearly shows the potential of using proportional systems to express knowledge in parametric modeling. By using the Base Diameter as the main control parameter, the traditional proportional rules of classical column types were transformed into a set of formulas. These formulas allowed not only the creation of geometry, but also preserved the construction logic. This modeling method, which centers on “proportion – logic – nested components,” can be seen as a way to digitize and structure historical architectural norms. It offers a useful path for semantic modeling and typological classification of heritage components. Although the column component has a clear parametric structure, it relies on deep nesting. All proportional control depends on parameter transfer from the main family to the nested families. When a user needs to adjust detailed proportions, such as the ratio between the base and the shaft, or change a local profile like the base molding, they must enter and edit the nested family directly. This increases the difficulty for users who are not experienced. While deep nesting helps to keep geometric consistency, it also brings high maintenance cost in model delivery, teaching, and teamwork situations.

The vault components are classified into three main position types: VAU_ATR (atrium), VAU_INT (intermediate), and VAU_STA (staircase). Each type represents to a different structural form and geometric feature. This classification follows the historical distinction between “simple vaults” and “composite vaults”, which helps each type maintain a clear modeling logic and parameter framework. The complex vaults VAU_ATR is decomposed and reconstructed from a barrel and multiple lunettes. Because of the Simplicity if the lunette structure but its large quantity(10 pieces), these parts are controlled by the same parameters such as span, rise, and fillet radius, and all values follow a consistent naming rule (e.g., Void_Lun_H1), and the generated in the same logic. So, if the geometric is in the same logic, what we need is to position it and get the input parameter, without reconstruct the logic. In this way, the UI or the component library are in need. For the load of geometry, all voids are built as floor-based families, which can be placed in floors of different thickness. Their positions are defined by adaptive point families or model lines, which do not take part in geometry creation, making it easier to keep construction logic separate.

During the modeling process of the arch, the main technical issues stem from the geometric incompatibility brought by the native Revit environment of Dynamo for Revit. When creating complex surfaces or solids using nodes such as Surface.ByPatch or Solid.ByLoft, the generated geometry often has problems such as non-manifold boundaries, open loops, or normal flips. For this reason, Revit can not recognizes them as valid BRep entities. The more complex geometry (such as hyperboloids), the higher the failure rate, which limits the universality of this method. The most direct result it causes is that the void family cannot perform the "cut geometry" operation. Even if the correct family template and valid entities are used, users still need to manually move the void form and cut the host in the Revit family environment,

which reduce the automation and repeatability of the process. Unlike the method of generating geometry in other software and implanting it into Revit or the DirectShape import method, the arch modeling in this study focuses on integrating typological classification, host interaction, and void cutting logic. The form of the arch is not only reconstructed based on point cloud data but also achieved through grouping, segmentation, and parametric operations. This process reflects the digital expression of historical construction logic and lays the foundation at the model level for future work such as arch typological classification, component packaging, and semantic coding.

The modeling of the oblique baluster(complex geometry) completely failed in the native Revit environment. This is because its geometry is based on a revolved shape followed by a shear transformation, which is not supported in either Revit or Dynamo. Attempts in Revit to use sweep combined with void cutting could not create accurate curved shapes. In Dynamo, methods like lofting from sliced contours also lacked precision. In the end, the workable solution was to model the shape in Rhino(other software or platforms), to apply the shear transformation to the revolved solid in NURBS, and to export it as a SAT file(with logical curve) into Revit. This component became the only one in this study that had to be fully removed from the parametric modeling process and based on other modeling software, which shows a clear boundary of Revit's geometric capabilities. Although the oblique baluster does not use parametric control, it expresses the geometric features of historical form accurately through NURBS surfaces. This approach shows another path for "non-parametric semantic modeling." The goal here is not to allow adjustable dimensions, but to keep the identity and decorative structure of the component. This suggests that in HBIM, decorative or non-repetitive components should follow a strategy of "high shape fidelity first, structure logic second," to complement the parametric modeling of structural elements.

Although the modeling strategies developed in this research follow a component-based logic, a reusable library of parametric components was not established. This is mainly because the dataset used in this study was relatively small and fragmented. For each typology—whether columns, vaults, or balusters—only one or two instances were available for modeling. As a result, it was not possible to define consistent patterns or variation ranges that would support the construction of a typology-wide family library. Instead of creating a database of parametric types, the modeling focused on reconstructing each component in detail and evaluating the logic behind its geometric and semantic structure. This limitation reflects the nature of the source data, which came from a specific case study and did not include a broad or statistically rich sample. In future work, expanding the dataset and collecting more examples of each architectural type could make it possible to create a structured, searchable HBIM component library.

6. Conclusion and Recommendations

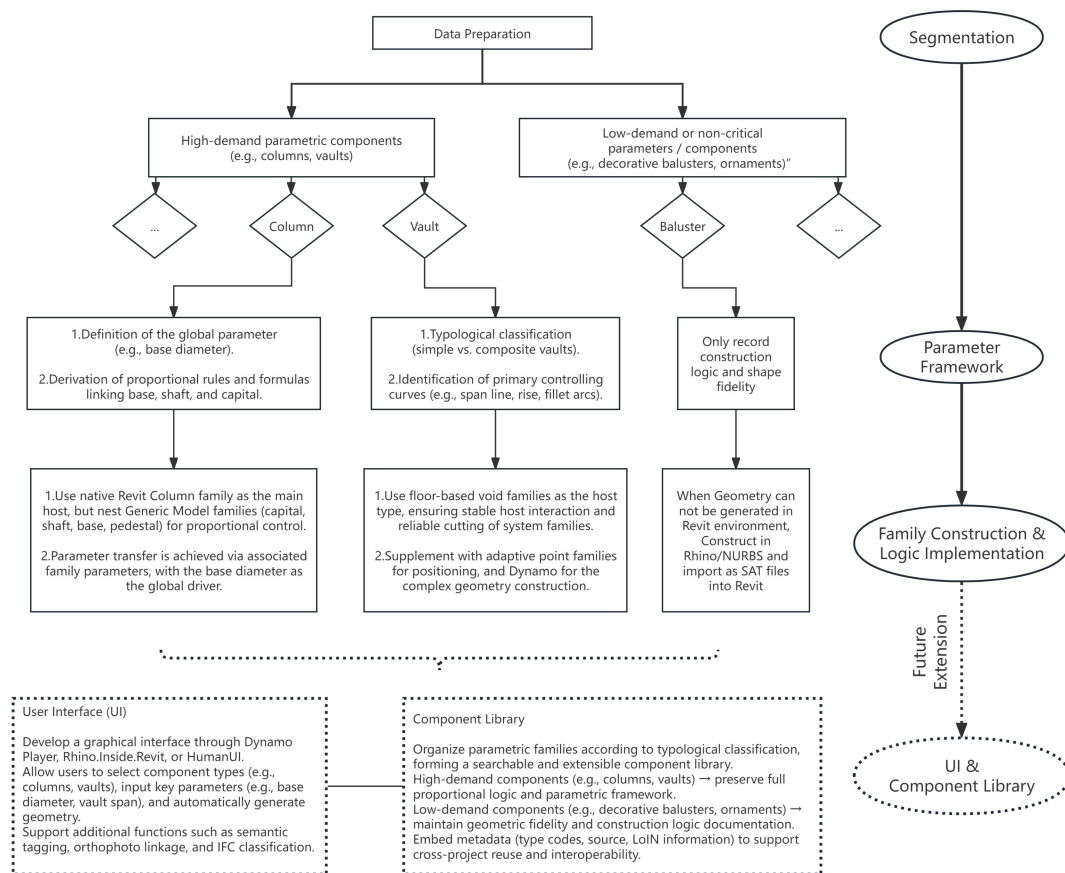


Figure 6-1 The Integrated HBIM Workflow

This study tested a multi-layered modeling workflow for reconstructing historical architectural components in a BIM environment. Figure 6-1 gives an overview of the process. The diagram outlines both the common HBIM framework and the specific strategies used for columns, vaults, and balusters. The approach brings together typological classification, parameter structuring, and geometry-based logic. With these tools, the team was able to construct complex vault systems, proportion-based columns, and balusters with ornamental detail (or other low-demand parameter element). Although the steps varied for each component type, the same methodology was kept throughout, so that the models remained semantically traceable and adaptable to different scales and formats.

The workflow follows a component-based HBIM logic. After segmentation, each element type is assigned to either a high-demand parametric path (columns, vaults) or a low-demand path (balusters and ornamental details). These two branches require different levels of parameterization and construction logic.

As shown in the Figure 6-1, three specific element are discussed:

The column system was modeled with a proportional method. A single base diameter served as the main control, and other measurements—such as the shaft height and the capital—were calculated from it through formulas. In Revit this logic

was applied by using the native Column family as the host, while the base, shaft, capital, and pedestal were added as nested Generic Model families. Parameters were passed from the host to the nested parts so that the sizes stayed linked, though this also required careful handling to avoid instability.

For vaults, the first step was to sort them into typological groups, separating simple from composite forms. Their geometry was then defined by a few main curves, such as the span, rise, and the arcs of the fillet. Floor-based void families were chosen to generate the forms, because they cut into the host floor directly and remain stable in most cases. This solution helped to avoid problems that often appear when trying to build solids in Dynamo or when importing surfaces from Rhino into Revit.

For oblique balusters and other decorative elements with low parametric demand, full parameterization was not necessary. Instead, the focus was placed on preserving construction logic and geometric fidelity. The components were modeled externally in Rhino which use NURBS transformations and then imported into Revit as SAT files. This solution avoided the limitations of Revit's family editor while ensuring that the distinctive historical form of the balusters was accurately represented.

The original research questions focused on how to embed semantic meaning within the geometry of historic elements and how to build reusable, parameter-controlled families for different structural and decorative types. These objectives have been partially fulfilled. The vault families demonstrated clear parametric logic based on geometric subdivision and typological grouping. The column system translated historical proportional rules into nested family logic. Even in the case of balusters, where parametric control was not feasible, geometric fidelity was maintained through an alternative pipeline.

However, due to the limited number of instances for each component type, this research did not develop a reusable library of parametric families. The modeling focused on reconstructing individual elements rather than generalizing typologies, which limited the scalability of the results across broader heritage contexts.

Academically, this research contributes to the development of HBIM by offering a structured modeling methodology that links architectural typology with parametric logic. It shows how classical rules and geometric hierarchies can be transformed into reusable digital assets. In practice, the method supports future workflows that require both semantic richness and model scalability, particularly in heritage conservation projects where documentation, classification, and reconstruction must coexist in a single environment.

However, the research also encountered technical limitations, especially in dealing with non-standard geometries within the Revit environment. The lack of native support for operations such as shear transformations, and the instability of Dynamo-generated solids, remain challenges. Future work may focus on building geometry preprocessing pipelines, extending parametric logic to other heritage components, and developing a reusable library that integrates geometry, metadata, and typological knowledge, especially when broader datasets become available..

Bibliography

Books and Reference

ACCA Software. (n.d.). *Disegni volte e cupole di molteplici tipologie scegliendo tra tantissimi oggetti.*

<https://www.acca.it/edificius-bim-modeling-volte-e-cupole>

Allegra, V., Di Paola, F., Lo Brutto, M., & Vinci, C. (2020). SCAN-TO-BIM for the MANAGEMENT of HERITAGE BUILDINGS: The CASE STUDY of the CASTLE of MAREDDOLCE (PALERMO, ITALY). *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLIII-B2-2020*, 1355–1362.

<https://doi.org/10.5194/isprs-archives-XLIII-B2-2020-1355-2020>

American Institute of Architects. (2008). *Document E202—Building information modeling protocol exhibit*. AIA.

ArchiRADAR. (n.d.). *Volte 3D parametriche per ArchiCAD.*

<https://www.archiradar.it/it/oggetti/volte-3d-parametriche-archicad>

Aubin, P. F., & Milburn, A. (2013). *Renaissance Revit: Creating classical architecture with modern software*. G3B Press.

Autodesk. (2022). *Associate Family Parameters [Manual]*. Autodesk.

<https://help.autodesk.com/view/RVT/2022/ENU/?guid=GUID-E2E6B19B-E47F-4F35-9F6D-1273A3248A50>

Autodesk Community. (2023, April 18). *BRepBuilder fails without explanation.*

<https://forums.autodesk.com/t5/revit-api-forum/brepbuilder-fails-without-explanation/td-p/11905308>

Autodesk Community. (2024). *Solid imported by Dynamo doesn't cut on sections.*

<https://forums.autodesk.com/t5/revit-structure-forum/solid-imported-by-dynamo-doesn-t-cut-on-sections/td-p/12609256>

Autodesk Community Forum. (2019). *Rotate & copy doubles rotation angle.*

<https://forums.autodesk.com/t5/revit-architecture-forum/rotate-amp-copy-doubles-rotation-angle/td-p/9063514>

Autodesk Community Forums. (2014). *Loading railing family.*

<https://forums.autodesk.com/t5/revit-architecture-forum/loading-railing-family/td-p/6378465>

Autodesk Help. (2025). *About the different kinds of families.*

<https://help.autodesk.com/view/RVT/2025/ENU/?guid=GUID-403FFEAE-BFF6-464D-BAC2-85BF3DAB3BA2>

Bagnolo, V., Argiolas, R., & Vanini, C. (2022). Algorithmic Modelling as a Key Tool for Ribbed Vault Geometry. *Nexus Network Journal*, 24(1), 147–166.

<https://doi.org/10.1007/s00004-021-00570-z>

Banfi, F. (2017). BIM orientation: Grades of generation and information for different type of analysis and management process. *The International Archives of the*

Photogrammetry, Remote Sensing and Spatial Information Sciences,

XLII-2/W5, 57–64. <https://doi.org/10.5194/isprs-archives-xlii-2-w5-57-2017>

Banfi, F. (2019). HBIM GENERATION: EXTENDING GEOMETRIC PRIMITIVES AND BIM

MODELLING TOOLS FOR HERITAGE STRUCTURES AND COMPLEX VAULTED

SYSTEMS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W15*, 139–148.

<https://doi.org/10.5194/isprs-archives-XLII-2-W15-139-2019>

- Banfi, F. (2020). HBIM, 3D drawing and virtual reality for archaeological sites and ancient ruins. *Virtual Archaeology Review*, 11(23), 16.
<https://doi.org/10.4995/var.2020.12416>
- Bonora, V., Meucci, A., Conti, A., Fiorini, L., & Tucci, G. (2023). Knowledge representation of built heritage mapping an ad hoc data model in ogc standards: The case study of pitti palace in florence, italy. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-M-2-2023, 281–288.
<https://doi.org/10.5194/isprs-archives-XLVIII-M-2-2023-281-2023>
- British Standards Institution. (2013). *PAS 1192-2:2013—Specification for information management for the capital/delivery phase of construction projects using building information modelling*. BSI.
- Brumana, R., Banfi, F., Cantini, L., Previtali, M., & Della Torre, S. (2018). Generative HBIM modelling to embody complexity (LOD, LOG, LOA, LOI): Surveying, preservation, site intervention—The basilica di collemaggio (l’aquila). *Applied Geomatics*, 10(4), 545–567. <https://doi.org/10.1007/s12518-018-0233-3>
- Brumana, R., Banfi, F., Cantini, L., Previtali, M., & Della Torre, S. (2019). Hbim level of detail-geometry-Accuracy and survey analysis for architectural preservation. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W11, 293–299.
<https://doi.org/10.5194/isprs-archives-xlii-2-w11-293-2019>
- Brumana, R., Della Torre, S., Previtali, M., Barazzetti, L., Cantini, L., Oreni, D., & Banfi, F. (2018). Generative HBIM modelling to embody complexity (LOD, LOG, LOA, LOI): Surveying, preservation, site intervention—the Basilica di Collemaggio

(L'Aquila). *Applied Geomatics*, 10(4), 545–567.

<https://doi.org/10.1007/s12518-018-0233-3>

Calvano, M. & others. (2023). Script VPL in Grasshopper per la generazione di una famiglia di massa da B-Rep e applicazione dell'istanza parametrica di tetto alla massa per la modellazione della volta. In L. Carlevaris & G. M. Valenti (Eds.), *Digital & documentation. Reading and communicating cultural heritage* (pp. 170–183). Pavia University Press.

Capone, M., & Lanzara, E. (2019). SCAN-TO-BIM vs 3D IDEAL MODEL HBIM: PARAMETRIC TOOLS TO STUDY DOMES GEOMETRY. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W9, 219–226.

<https://doi.org/10.5194/isprs-archives-XLII-2-W9-219-2019>

Chitham, R. (2005). *The classical orders of architecture* (2nd ed). Architectural Press.

Costantino, D., Pepe, M., & Restuccia, A. G. (2023). Scan-to-HBIM for conservation and preservation of Cultural Heritage building: The case study of San Nicola in Montedoro church (Italy). *Applied Geomatics*, 15(3), 607–621.

<https://doi.org/10.1007/s12518-021-00359-2>

Croce, V., Caroti, G., De Luca, L., Jacquot, K., Piemonte, A., & Véron, P. (2021). From the Semantic Point Cloud to Heritage-Building Information Modeling: A Semiautomatic Approach Exploiting Machine Learning. *Remote Sensing*, 13(3), 461. <https://doi.org/10.3390/rs13030461>

Curioni, G. (1868). Geometria pratica: Applicata all' arte del costruttore: ... / Giovanni Curioni. In *Geometria pratica: Applicata all' arte del costruttore: ...* Negro.

D'Agostino, P., Antuono, G., & Elefante, E. (2024). *Algorithmic approaches for HBIM*.

The great cloister of the opera di santa croce in florence (pp. 539–548).

Diara, F., & Rinaudo, F. (2019). FROM REALITY TO PARAMETRIC MODELS OF

CULTURAL HERITAGE ASSETS FOR HBIM. *The International Archives of the*

Photogrammetry, Remote Sensing and Spatial Information Sciences,

XLII-2/W15, 413–419.

<https://doi.org/10.5194/isprs-archives-XLII-2-W15-413-2019>

Donghi, D. (1935). *Manuale dell'architetto. Vol. 2: La composizione architettonica*.

Parte II: Decorazione od estetica architettonica. UTET.

<http://digit.biblio.polito.it/id/eprint/5643>

Duvernoy, S. (2015). Baroque Oval Churches: Innovative Geometrical Patterns in

Early Modern Sacred Architecture. *Nexus Network Journal*, 17(2), 425–456.

<https://doi.org/10.1007/s00004-015-0252-x>

Elet, Y. (2020). Stucco as Substrate and Surface in Quattrocento Florence (and

Beyond). In A. R. Bloch & D. M. Zolli (Eds.), *The Art of Sculpture in*

Fifteenth-Century Italy (1st ed., pp. 283–313). Cambridge University Press.

<https://doi.org/10.1017/9781108579322.019>

Elsaid, M. E., Ayoub, M., & Hassan, H. (2019). Scan-to-Building Information Modelling

vs. HBIM in Parametric Heritage Building Documentation. *IOP Conference*

Series: Earth and Environmental Science, 397(1), 012015.

<https://doi.org/10.1088/1755-1315/397/1/012015>

Fenoglio, G. (1928). Il palazzo dei marchesi di barolo. *Torino: Rivista Mensile*

Municipale, VIII, 164–171.

- Giuliani, F., Gaglio, F., Martino, M., & De Falco, A. (2024). A HBIM pipeline for the conservation of large-scale architectural heritage: The city Walls of Pisa. *Heritage Science*, 12(1), 35. <https://doi.org/10.1186/s40494-024-01141-4>
- Hitech CADD Services. (2019). *How revit dynamo improves BIM workflows*. <https://www.hitechcaddservices.com/news/how-revit-dynamo-can-improve-bim-workflows/>
- ICOMOS and ICCROM. (2023). *Guidance on post- disaster and post- conflict recovery and reconstruction for heritage places of cultural significance*.
- ISO 7817-1. (2024). *ISO 7817-1:2024—Building information modelling—Level of information need*. <https://www.iso.org/standard/78171.html>
- Liberotti, R., & Gusella, V. (2023). Parametric Modeling and Heritage: A Design Process Sustainable for Restoration. *Sustainability*, 15(2), 1371. <https://doi.org/10.3390/su15021371>
- Lombardini, N., & Cantini, L. (2017). Non-standardized data in the BIM process. The management of construction systems data in the cultural heritage conservation. *Atti Di Convegno*. <https://hdl.handle.net/11311/1045041>
- Marcello. (2022). *Dynamo workflow to recreate complex geometry from imported SAT file*. <https://forum.dynamobim.com/t/dynamo-workflow-to-recreate-complex-geometry-from-imported-sat-file/81912>
- Medina, A. (2020). *Parametric modeling of vaults for Notre Dame in revit*.
- Migliari, R. (2009). *Geometria descrittiva*. Città Studi Edizioni.
- Moyano, J., Carreño, E., Nieto-Julián, J. E., Gil-Arizón, I., & Bruno, S. (2022). Systematic approach to generate Historical Building Information Modelling

- (HBIM) in architectural restoration project. *Automation in Construction*, 143, 104551. <https://doi.org/10.1016/j.autcon.2022.104551>
- Moyano, J., Gil-Arizón, I., Nieto-Julián, J. E., & Marín-García, D. (2021). Analysis and management of structural deformations through parametric models and HBIM workflow in architectural heritage. *Journal of Building Engineering*, 45, 103274. <https://doi.org/10.1016/j.jobbe.2021.103274>
- Murphy, M., McGovern, E., & Pavia, S. (2009). Historic building information modelling (HBIM). *Structural Survey*, 27(4), 311–327. <https://doi.org/10.1108/02630800910985108>
- Natta, F. (2024). *Modellazione computazionale per l'interpretazione geometrica di sistemi voltati complessi fra teoria e realizzazione* [PhD Thesis, Politecnico di Torino]. <https://hdl.handle.net/11583/2991327>
- Parisi, P., Lo Turco, M., & Giovannini, E. C. (2019). The Value of Knowledge Through H-BIM Models: Historic Documentation with a Semantic Approach. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W9, 581–588. <https://doi.org/10.5194/isprs-archives-XLII-2-W9-581-2019>
- pyRevit. (n.d.). *pyRevit: An open-source scripting extension for Autodesk Revit*. <https://github.com/eirannejad/pyRevit>
- Quattrini, R., Malinverni, E. S., Clini, P., Nespeca, R., & Orlietti, E. (2015). From TLS to HBIM: high quality semantically-aware 3D modeling of complex architecture. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W4, 367–374. <https://doi.org/10.5194/isprsarchives-xl-5-w4-367-2015>

- Sanseverino, A., Messina, B., Limongiello, M., & Guida, C. G. (2022). An HBIM Methodology for the Accurate and Georeferenced Reconstruction of Urban Contexts Surveyed by UAV: The Case of the Castle of Charles V. *Remote Sensing*, 14(15), 3688. <https://doi.org/10.3390/rs14153688>
- Scianna, A., Gaglio, G. F., & La Guardia, M. (2020). HBIM data management in historical and archaeological buildings. *Archeologia e Calcolatori*, 31(1), 231–252. <https://doi.org/10.19282/ac.31.1.2020.11>
- UNESCO. (2003). *Charter on the preservation of the digital heritage*.
- Van Malssen, K., Tandon, A., & Hazejager, K. (2021). *The digital imperative: Envisioning the path to sustaining our collective digital heritage*. ICCROM / AVP / NISV.
- Werbrouck, J., Pauwels, P., Bonduel, M., Beetz, J., & Bekers, W. (2020). Scan-to-graph: Semantic enrichment of existing building geometry. *Automation in Construction*, 119, 103286. <https://doi.org/10.1016/j.autcon.2020.103286>
- Yang, X., Lu, Y.-C., Murtiyoso, A., Koehl, M., & Grussenmeyer, P. (2019). HBIM Modeling from the Surface Mesh and Its Extended Capability of Knowledge Representation. *ISPRS International Journal of Geo-Information*, 8(7), 301. <https://doi.org/10.3390/ijgi8070301>
- Zhao, J., Hua, X., Yang, J., Yin, L., Liu, Z., & Wang, X. (2023). A Review of Point Cloud Segmentation of Architectural Cultural Heritage. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, X-1/W1-2023, 247–254. <https://doi.org/10.5194/isprs-annals-X-1-W1-2023-247-2023>

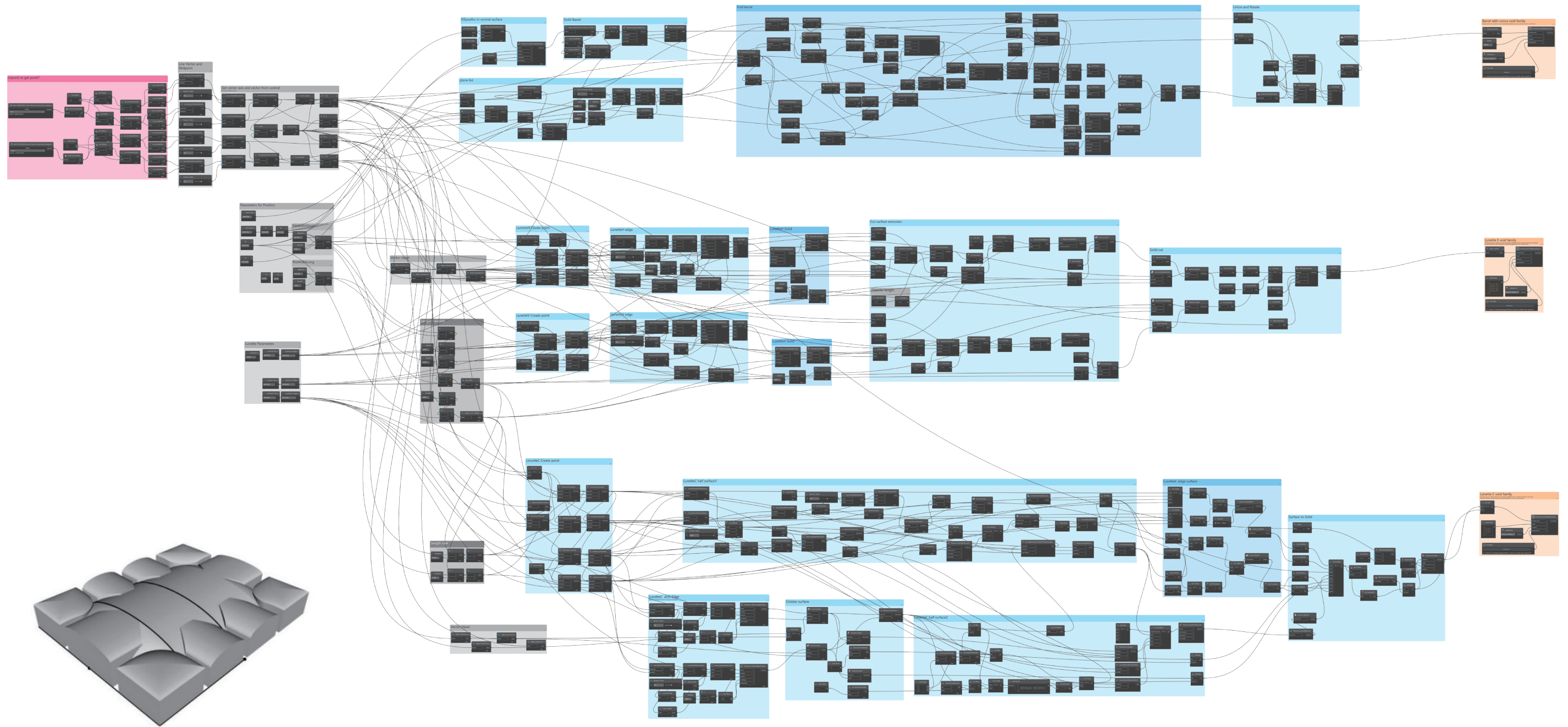
Zhao, J., Yu, H., Hua, X., Wang, X., Yang, J., Zhao, J., & Xu, A. (2024). Semantic segmentation of point clouds of ancient buildings based on weak supervision. *Heritage Science*, 12(1), 232. <https://doi.org/10.1186/s40494-024-01353-8>

Appendix

The appendices provide enlarged views of Dynamo node graphs used in the vault construction process.

Each diagram corresponds to a specific vault type or construction stage described in the main text, with step-by-step breakdowns of geometry creation, list management, and family generation.

Appendix-1	VAU_ATR (Vault in Atrium) Barrel Vault with conca end and mutiple lunette- General description
Appendix-2	VAU_ATR (Vault in Atrium) Barrel Vault with conca end
Appendix-5	VAU_ATR (Vault in Atrium) Lunette Vertical and horizontal (Not at corner)
Appendix-8	VAU_ATR (Vault in Atrium) Lunette at Corner
Appendix-11	VAU_INT_1 (Vault in Intermediate position Type 1) So-called “spherical panels built on arcs of variable shape”
Appendix-13	VAU_INT_2 (Vault in Intermediate position Type 2) Ellipsoidal pendentive
Appendix-14	VAU_STA (Vault in staircase) Barrel Vault with Cloister



General Overview

In addition to the explanations of the four main functional modules (Data Input, Parameter Pre-processing, Geometry Construction, Output to Revit Family), the P1 diagram includes a detailed enlargement of Module “Geometry Construction” to illustrate the geometric generation process in greater depth. This section also provides a separate table of list-processing nodes used in the workflow. These nodes (e.g., List.Create, List.Combine, etc.,) represent general-purpose data-handling methods applied across all Dynamo scripts in this study. Since their functions remain consistent throughout the workflows, they are documented here once in P1 and will not be repeated in the appendices for subsequent diagrams.

General Dynamo Graph Overview: VAU_ATR

Data input

Reads existing reference geometry such as model lines, adaptive points, or placement markers from the project. This ensures that all subsequent geometry construction is aligned to the actual site conditions.

Parameter Input & Pre-processing

Description: Inputs dimension values (e.g., span, rise, radius) and computes derived values such as midpoints, vectors, or axis directions to guide geometry generation. This step ensures that the geometric logic is fully parametric and adaptable.

Geometry Construction

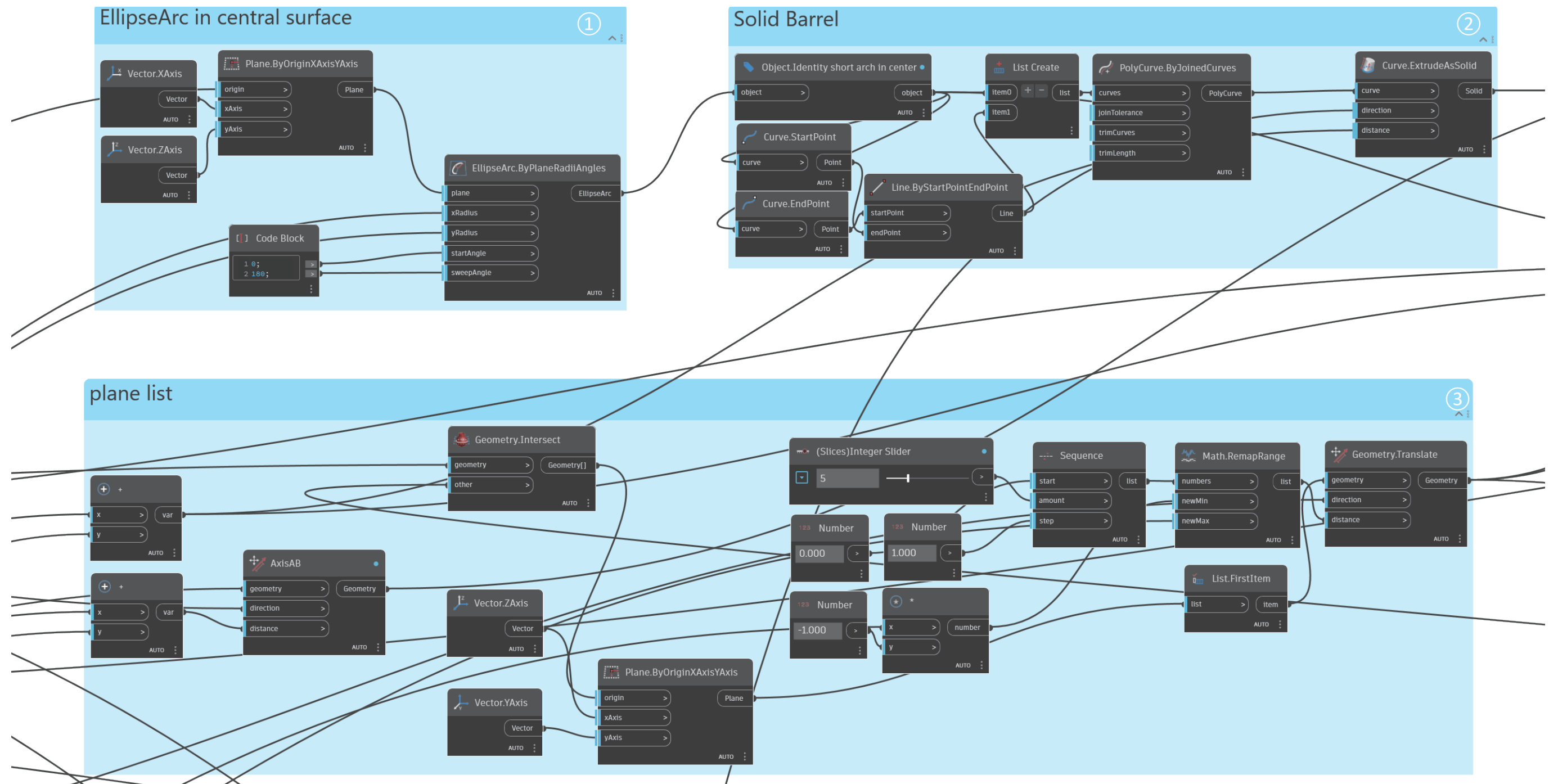
Generates base profiles and transforms them into solids through lofting, revolving, or patching operations. At this stage, the geometry is still a Dynamo object, not yet integrated into the Revit family system.

Output to Revit Family

Applies family templates (e.g., floor-hosted void) to create BIM-compatible components. These can then be loaded into the project for placement and interaction with host elements.

List Node used in this project

Node Name	Function Description
List.Create	Creates a list from multiple input items.
List.Combine	Applies a function across multiple lists simultaneously.
List.Transpose	Swaps list levels to reorganise the data structure.
List.Flatten	Flattens nested lists into a single-level list.
List.GetItemAtIndex	Retrieves a specific item from a list by index.
List.FirstItem	Returns the first element in a list.
List.LastItem	Returns the last element in a list.
List.RestOfItems	Returns all elements except the first item.
List.Join	Merges multiple lists into a single list.
List.Map	Applies a given function to each item in a list.
List.DropLastItem	Removes the last element from a list. (Clockwork package)



This Dynamo workflow is organised into three main functional clusters, each corresponding to a specific stage in the generation of the intermediate vault geometry.

1.EllipseArc in Central Surface

Defines the central reference plane using (Vector.XAxis) and (Vector.ZAxis), combined in (Plane.ByOriginXAxisYAxis). Creates the main elliptical arc with (EllipseArc.ByPlaneRadiiAngles), driven by radius and angle parameters from a code block.

2.Solid Barrel

Identifies the short arch curve at the vault's centre, extracts its start and end points, and joins them into a closed

PolyCurve via (PolyCurve.ByJoinedCurves).

Extrudes the polycurve into a solid using (Curve.ExtrudeAsSolid).

Together with Step 1, this sequence defines the geometric logic for constructing the central section of the barrel vault.

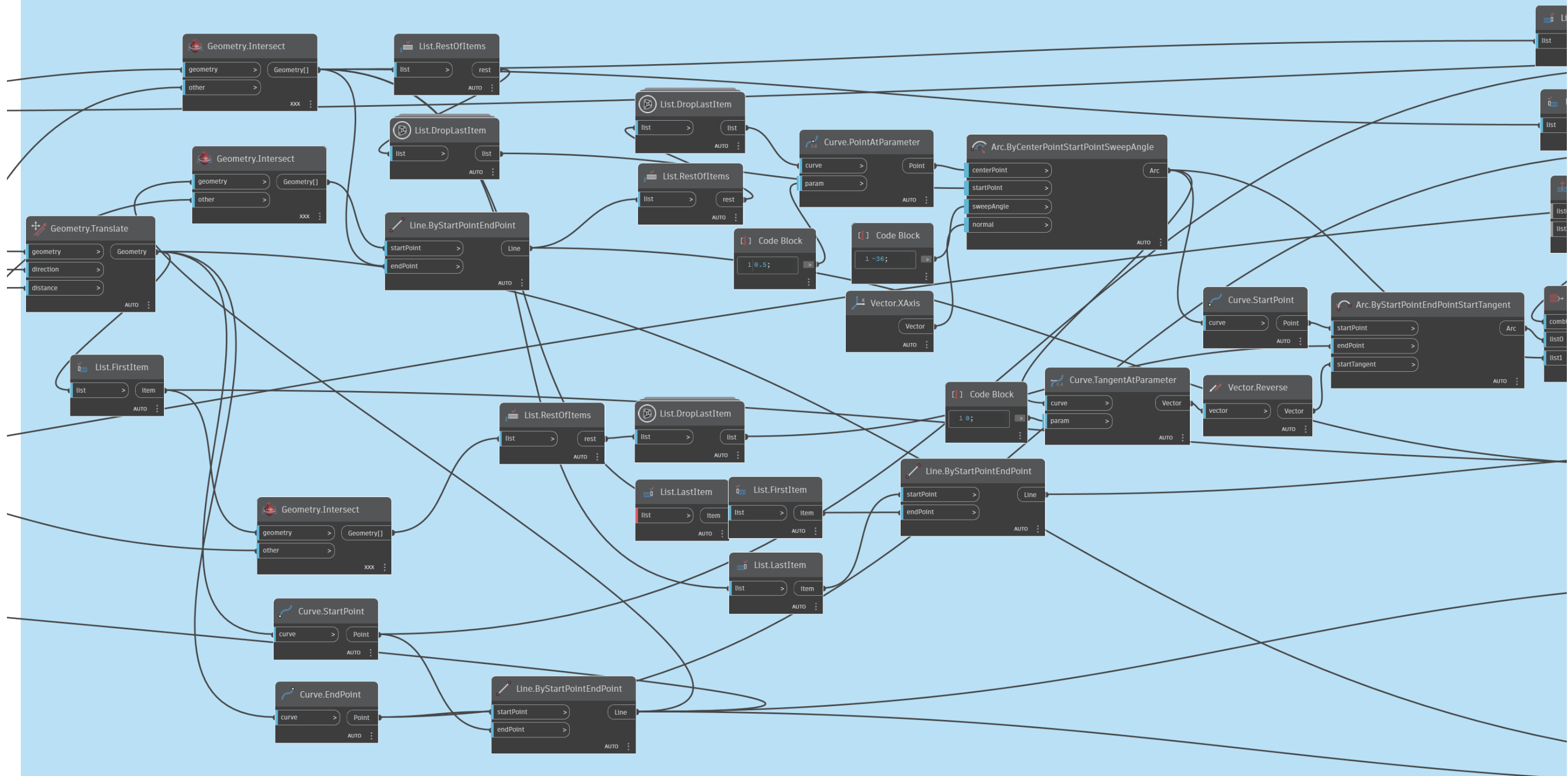
3.Plane List for End Geometry Construction

Generates a set of planes using (Sequence), (Math.RemapRange), and (Geometry.Translate), oriented by (Vector.YAxis) and (Vector.ZAxis). Uses (Geometry.Intersect) and AxisAB to align planes for slicing and subsequent curve generation.

These planes are later used to construct multiple multi-centred arcs which are lofted to approximate a conca-like shape forming the end of the barrel vault. As described in Section 4.2.1 of the main text, this geometry is derived directly from scanned point cloud data rather than typological references.

General Dynamo Graph Overview: VAU_ATR



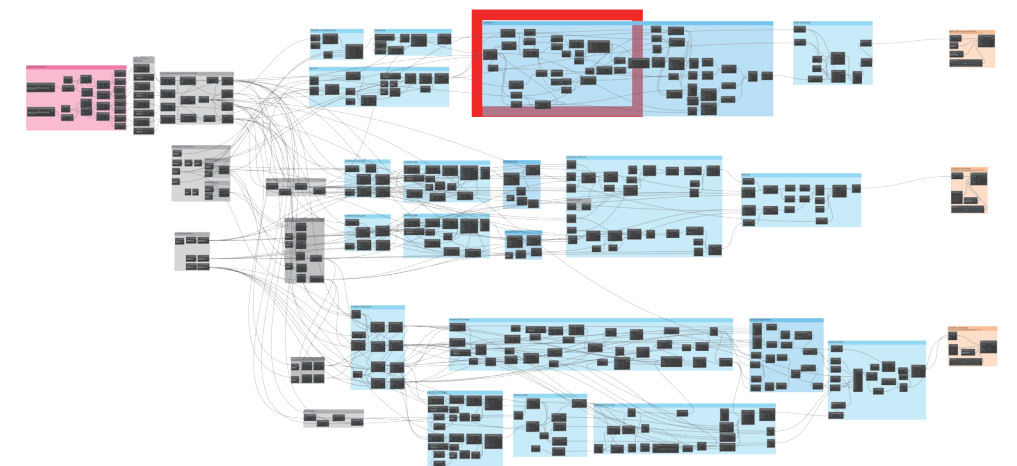


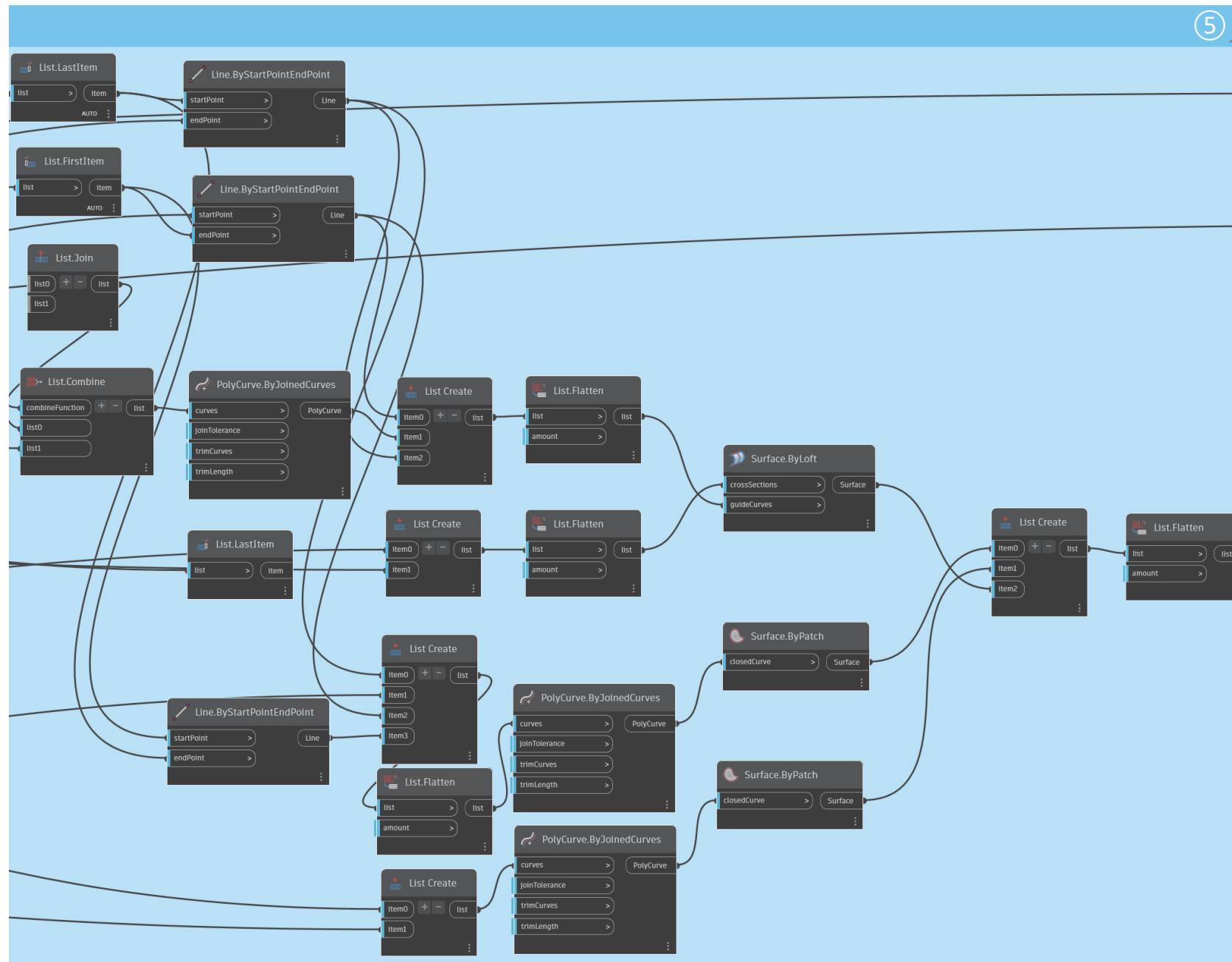
4. Arc Construction on each plane

This step calculates the intersection points between each reference plane and the target curves, using them as base points.

Two arcs are constructed on each plane: the first defined by a known centre point, start point, and sweep angle; the second defined by a known start point, start-point tangent, and end point.

Key nodes used in this step: Geometry.Intersect, Curve.PointAtParameter, Arc.ByStartPointEndPointStartTangent, Arc.ByCenterPointStartPointSweepAngle.



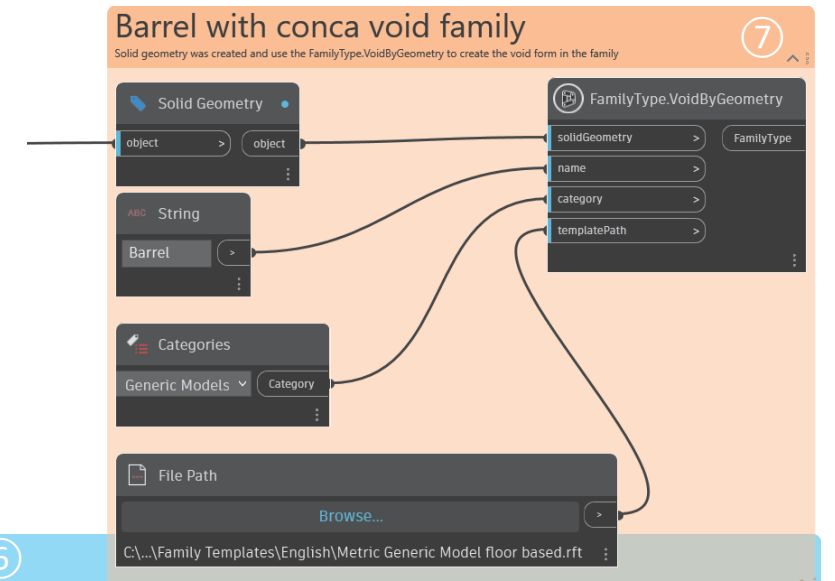


5. Solid construction of Barrel with conca
 Manages the above curve lists to loft a surface, patches two additional planes, and creates a list of three surfaces without multi-level nesting.

Key nodes: Surface.ByLoft, Surface.ByPatch, List.Flatten.

6. Rotate to complete the vault
 Rotate the geometry along the central Z-axis, flatten the list, and union all elements into a single solid.

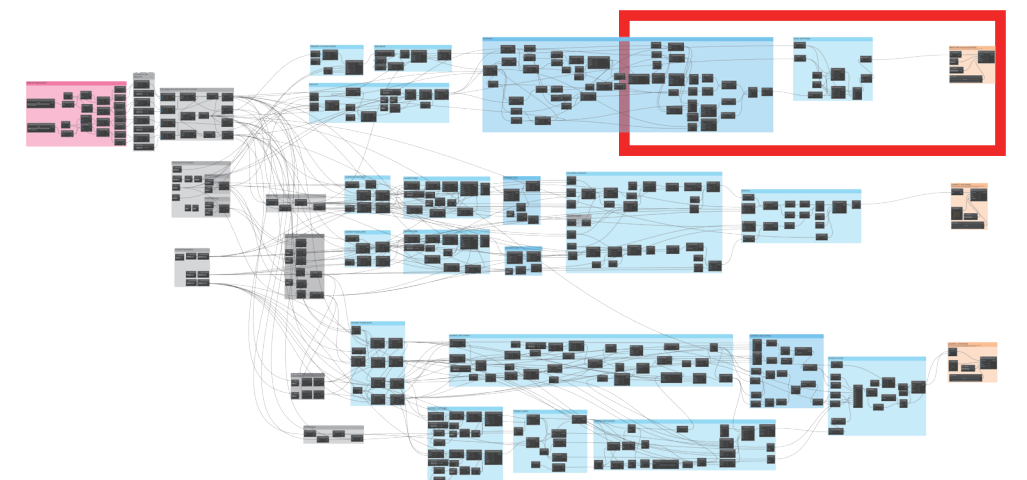
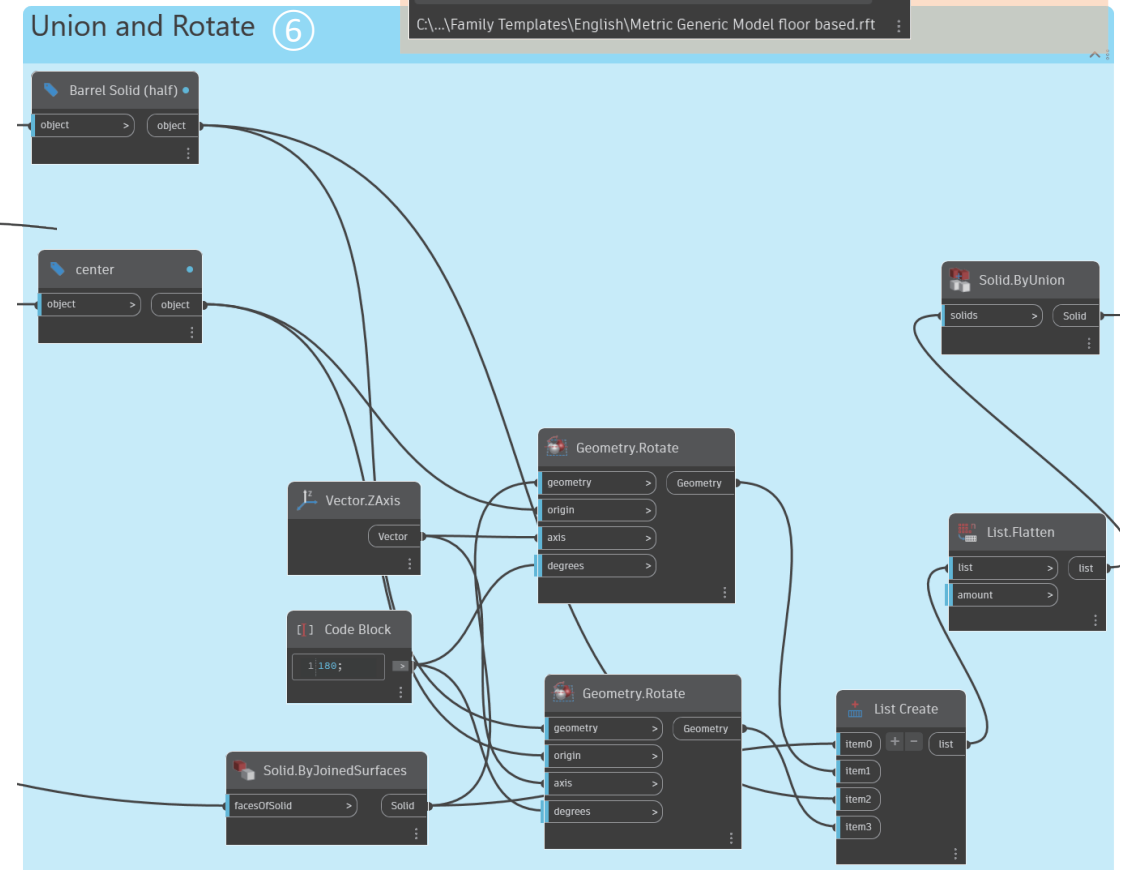
Key nodes: Solid.ByJoinedSurfaces, Solid.ByUnion, Geometry.Rotate.

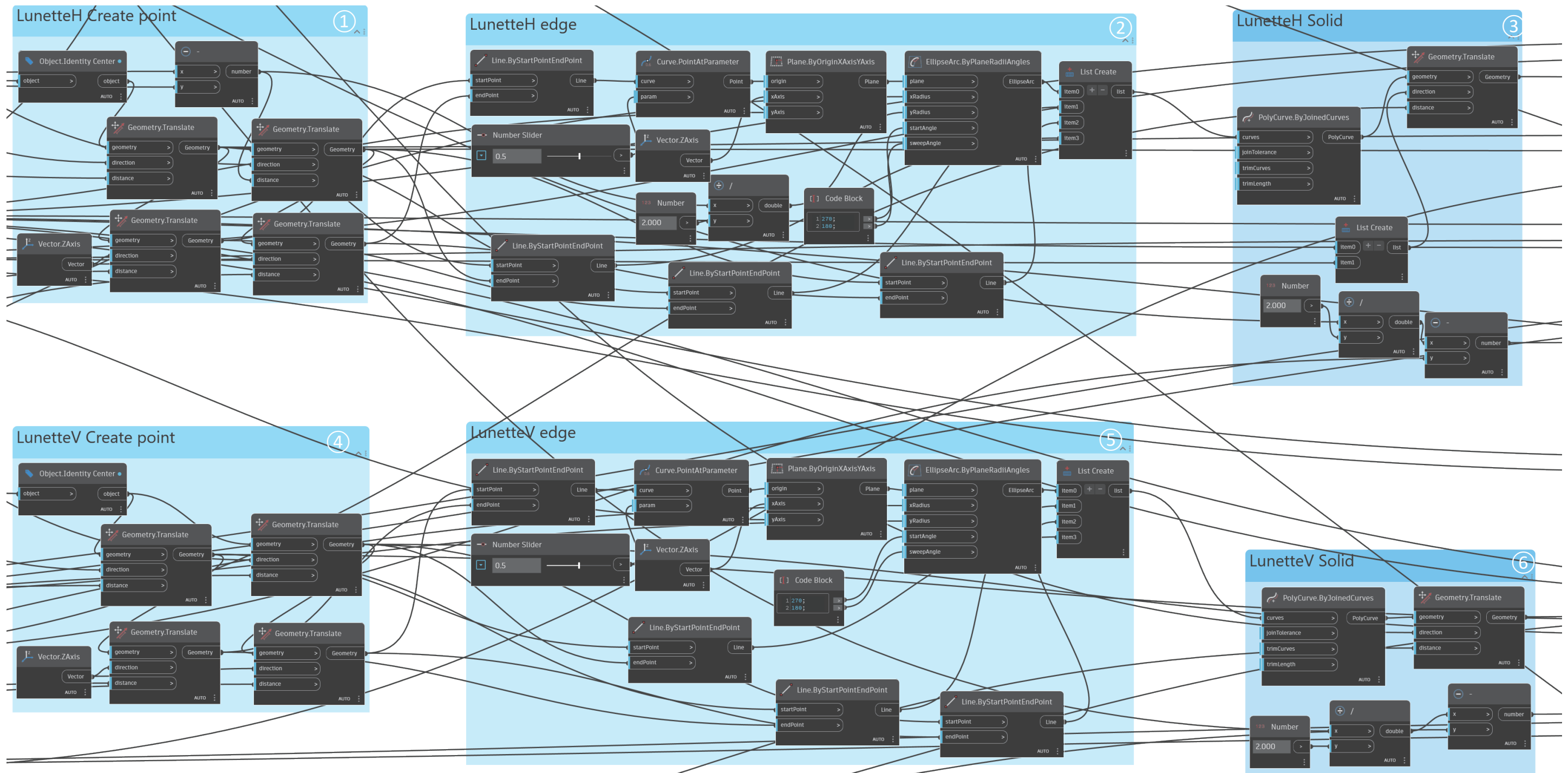


7. Create Family in the Project (Output to Revit Family)

This step is at last of the workflow by converting the completed solid geometry into a floor-hosted void family. The process assigns the family name and category, specifies the template path, and prepares the family for placement within the Revit project environment.

Key nodes: FamilyType.VoidByGeometry, Categories, File Path, String.





This graph generates the normal (non-corner) lunettes as barrel-type segments; cutting/boolean operations are performed later in the workflow.

1.LunetteH – Create Point

Define and position control points for horizontal lunettes.

Key nodes: Object.Identity, Geometry.Translate, Vector.ZAxis.

2.LunetteH – Edge

Build edge curves on the working plane (lines + ellipse arcs).

Key nodes: Line.ByStartPointEndPoint, Curve.PointAtParameter, Plane.ByOriginXAxisYAxis, EllipseArc.ByPlaneRadiiAngles

3.LunetteH – Solid

Join edges and generate a provisional solid for horizontal lunette.

General Dynamo Graph Overview: VAU_ATR

Key nodes: PolyCurve.ByJoinedCurves, Geometry.Translate.

4.LunetteV – Create Point

Define and position control points for vertical lunettes (same logic as H).

Key nodes: Object.Identity, Geometry.Translate, Vector.ZAxis.

5.LunetteV – Edge

Build edge curves for vertical lunettes on the working plane.

Key nodes: Line.ByStartPointEndPoint, Curve.PointAtParameter, Plane.ByOriginXAxisYAxis, EllipseArc.ByPlaneRadiiAngles

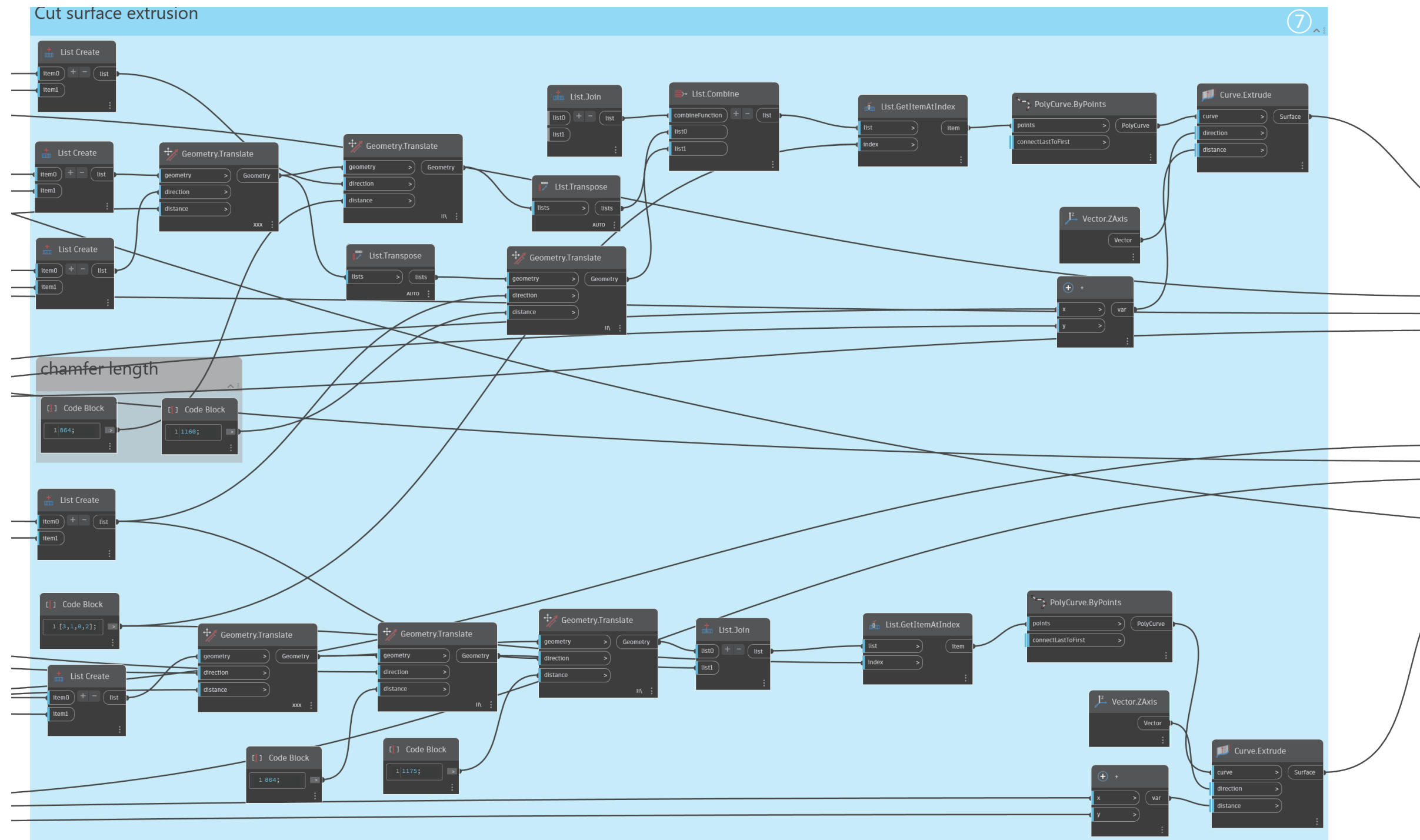
6.LunetteV – Solid

Join edges and generate a provisional solid for the vertical lunette.

Key nodes: PolyCurve.ByJoinedCurves, Geometry.Translate.

Note: Outputs here are pre-cut solids representing standard barrel-vault lunettes. Subsequent steps perform trimming(geometry construction) and void conversion.



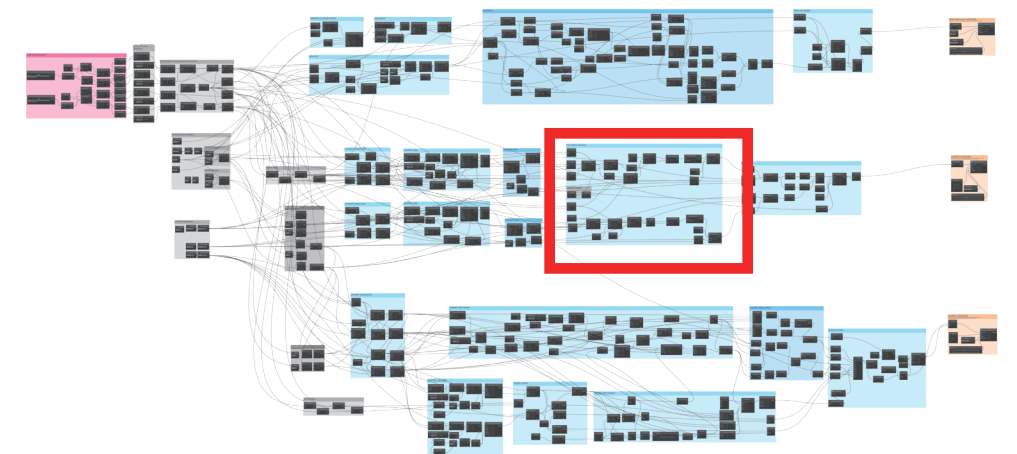


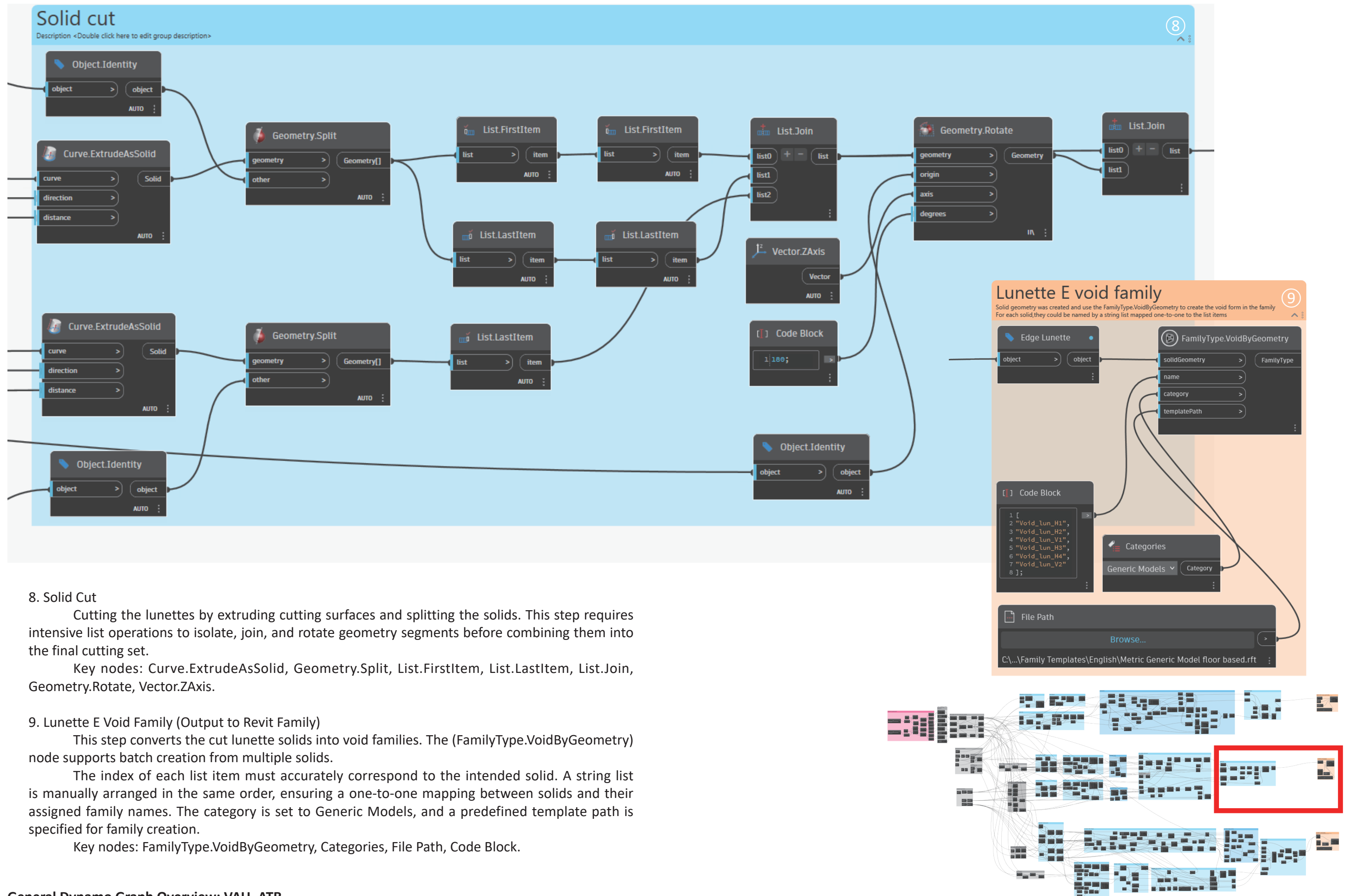
7. Cutting the normal lunette

Basic logic: extrude surfaces and subtract from lunette solids.

Real difficulty: list management. Need to handle intersecting lists with many operations (Transpose/Filter/Merge/Combine).

Step uses large number of nodes. Most for data organising and alignment.





8. Solid Cut

Cutting the lunettes by extruding cutting surfaces and splitting the solids. This step requires intensive list operations to isolate, join, and rotate geometry segments before combining them into the final cutting set.

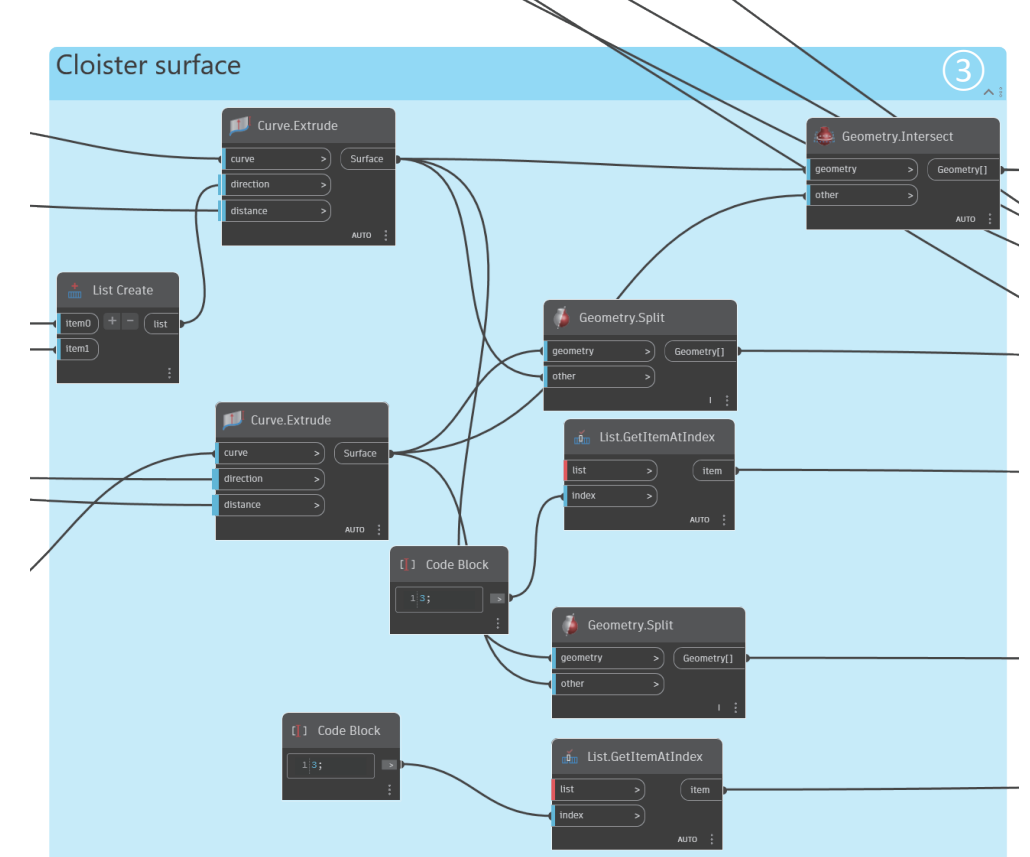
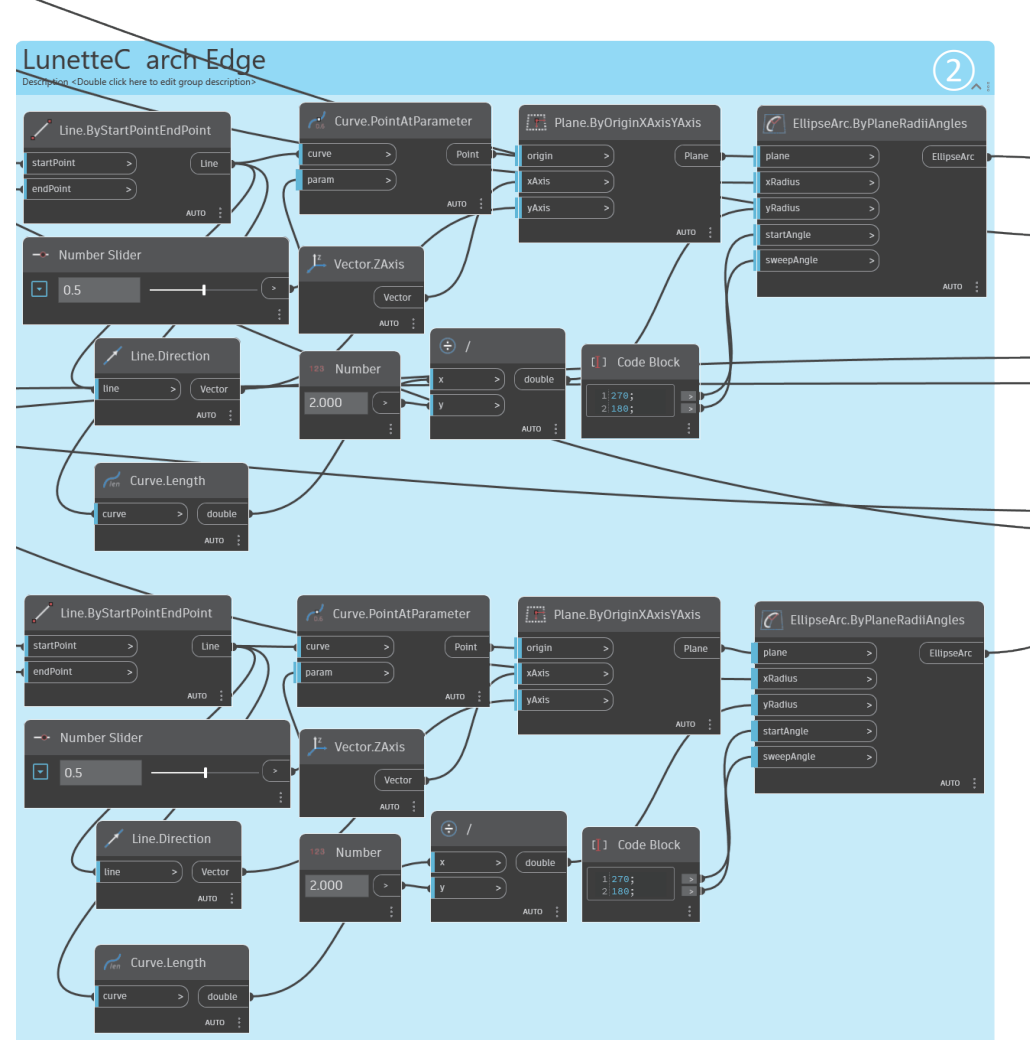
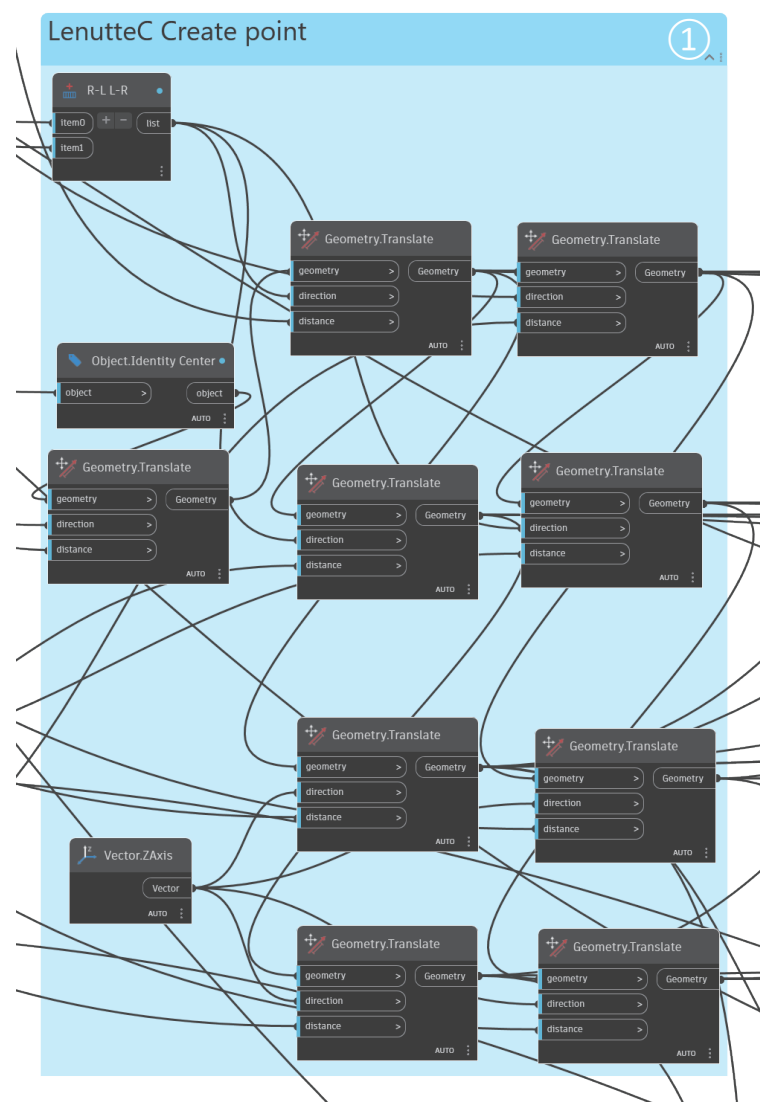
Key nodes: Curve.ExtrudeAsSolid, Geometry.Split, List.FirstItem, List.LastItem, List.Join, Geometry.Rotate, Vector.ZAxis.

9. Lunette E Void Family (Output to Revit Family)

This step converts the cut lunette solids into void families. The (FamilyType.VoidByGeometry) node supports batch creation from multiple solids.

The index of each list item must accurately correspond to the intended solid. A string list is manually arranged in the same order, ensuring a one-to-one mapping between solids and their assigned family names. The category is set to Generic Models, and a predefined template path is specified for family creation.

Key nodes: FamilyType.VoidByGeometry, Categories, File Path, Code Block.



This is the most complex lunette construction due to the stucco decorations. The structure geometry is hiding, which makes its typology impossible to identify precisely. The shape is approximated as a combination of half a cloister and half a lofted surface.

1.Point Positioning

Control points for the corner lunette are located using Object.Identity and multiple Geometry.Translate operations, establishing the reference positions in relation to the vault's central axis.

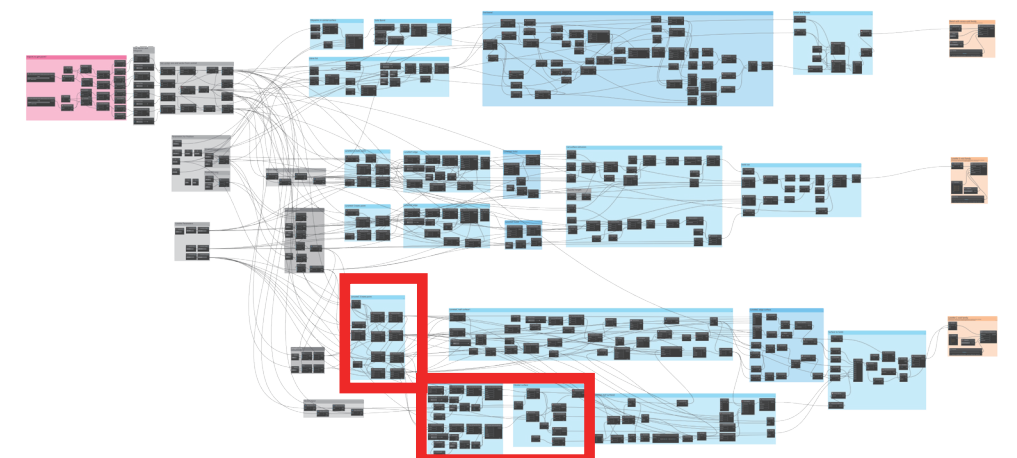
2.Lunette Arch Edge

Edge curves are defined on working planes using Node(Line.ByStartPointEndPoint), (Curve.PointAtParameter), (Plane.ByOriginXAxisYAxis), and (EllipseArc.ByPlaneRadiiAngles). These curves form the perimeter of the lunette arch.

3.Cloister Surface

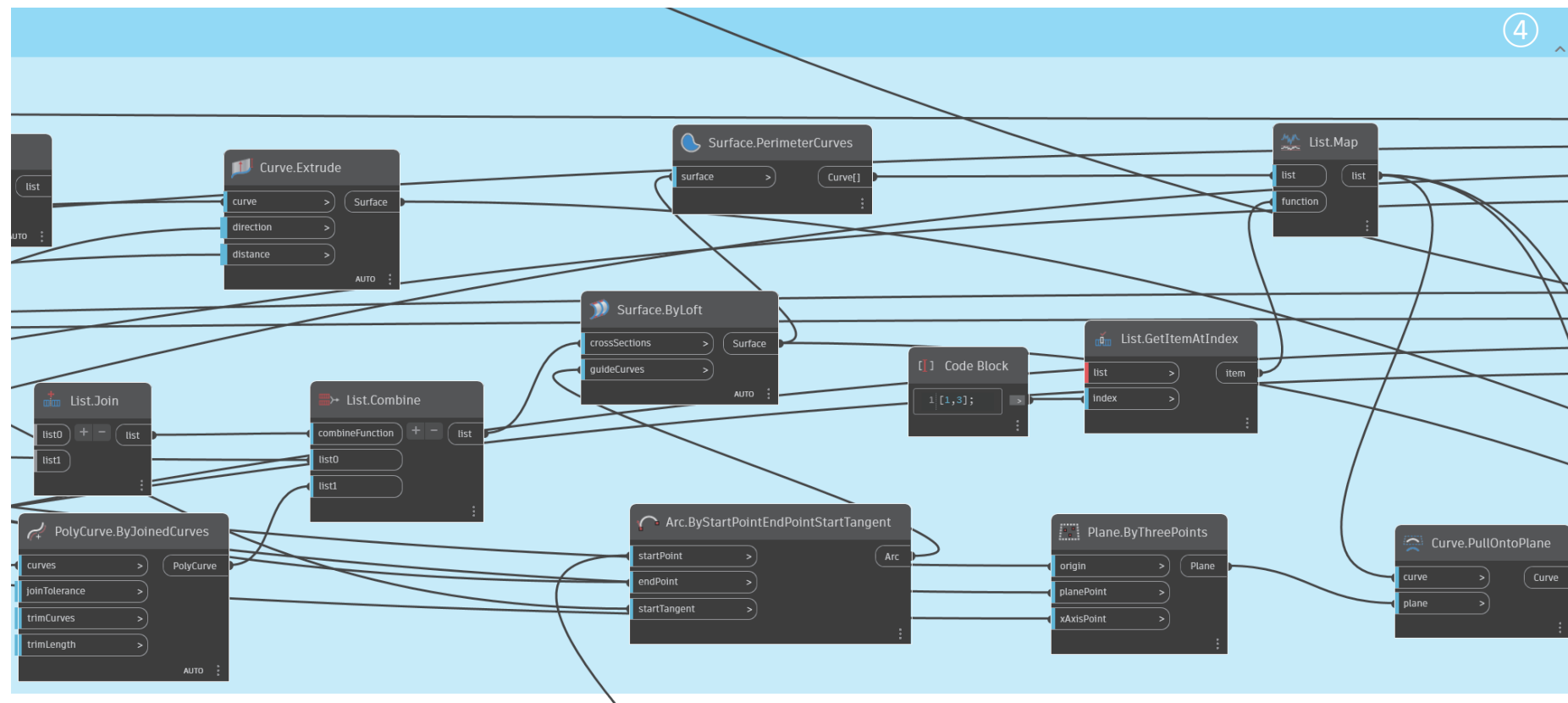
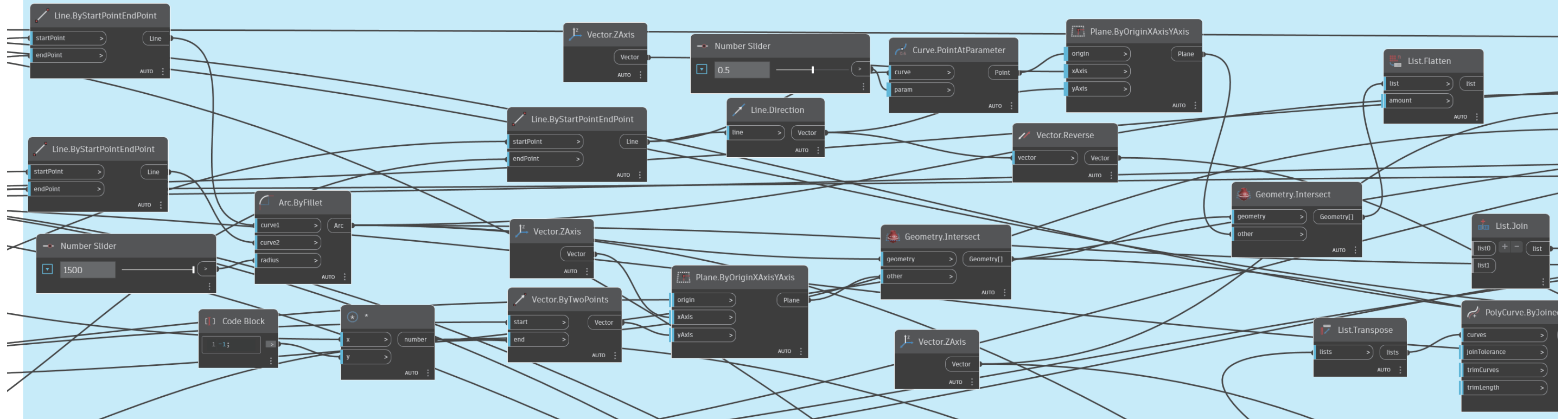
Surfaces are extruded from curves (Curve.Extrude) and processed through Geometry.Intersect and Geometry.Split to isolate the target geometry. List.GetItemAtIndex is used to select the correct segments for the final assembly of the lunette surface.

General Dynamo Graph Overview: VAU_ATR



LunetteC half surface1

4



a. Intersection and Base Geometry Definition

Calculate intersections between guide curves and reference planes. Extract and organise the intersection points into structured lists.

b. List Management for Loft Preparation

The organization of list Node(List.Transpose), (List.Combine),(List.GetItemAtIndex) arranges cross-section curves and guide curves in the correct sequence. Each set of curves is associated with its corresponding reference plane.

c. Loft Generation and Edge Refinement

Generate the lofted surface using (Surface.ByLoft).

Since lofting may produce edges that are not perfectly coincident, re-extract the side edges using (Surface.PerimeterCurves).

d. Edge Remapping for Planarity

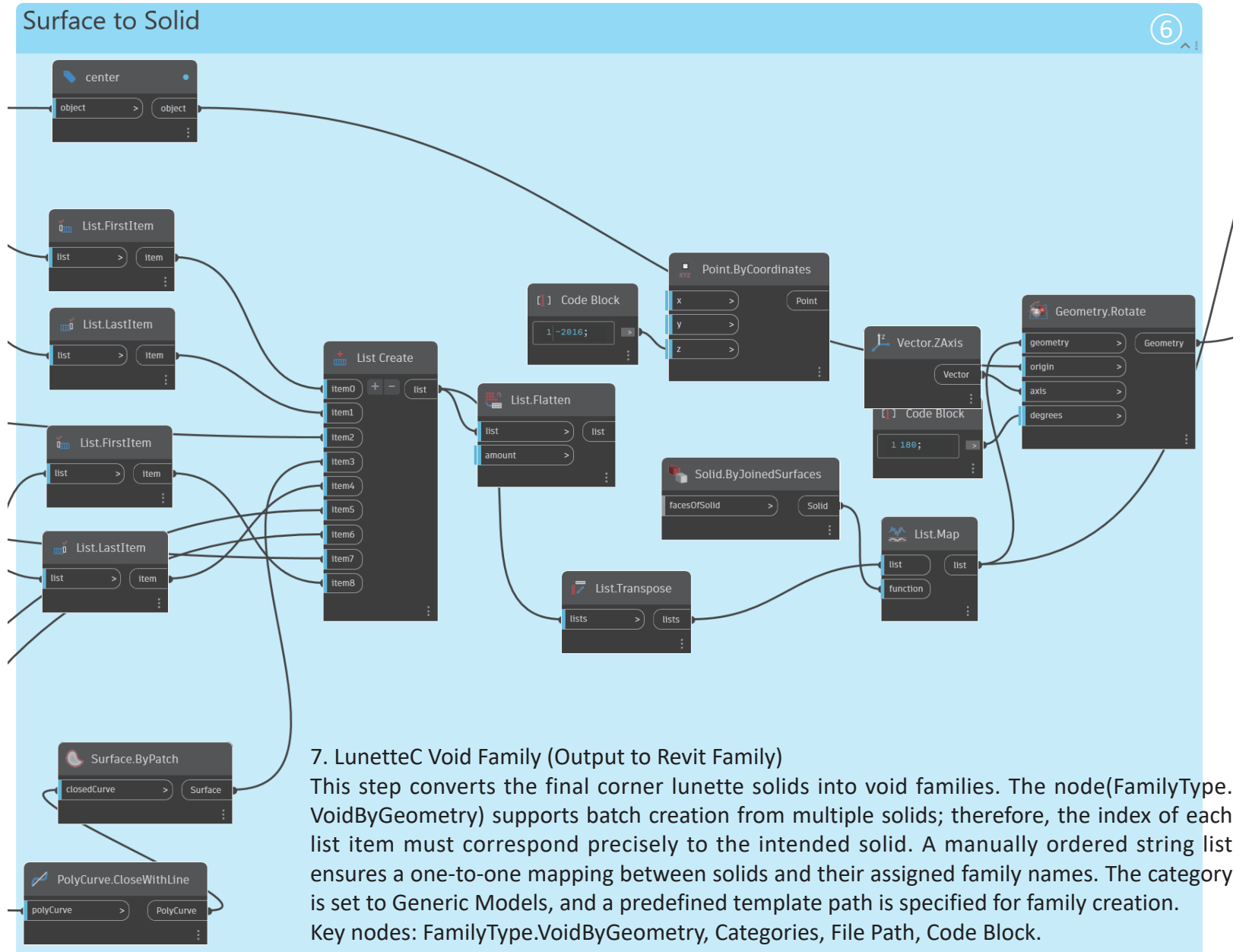
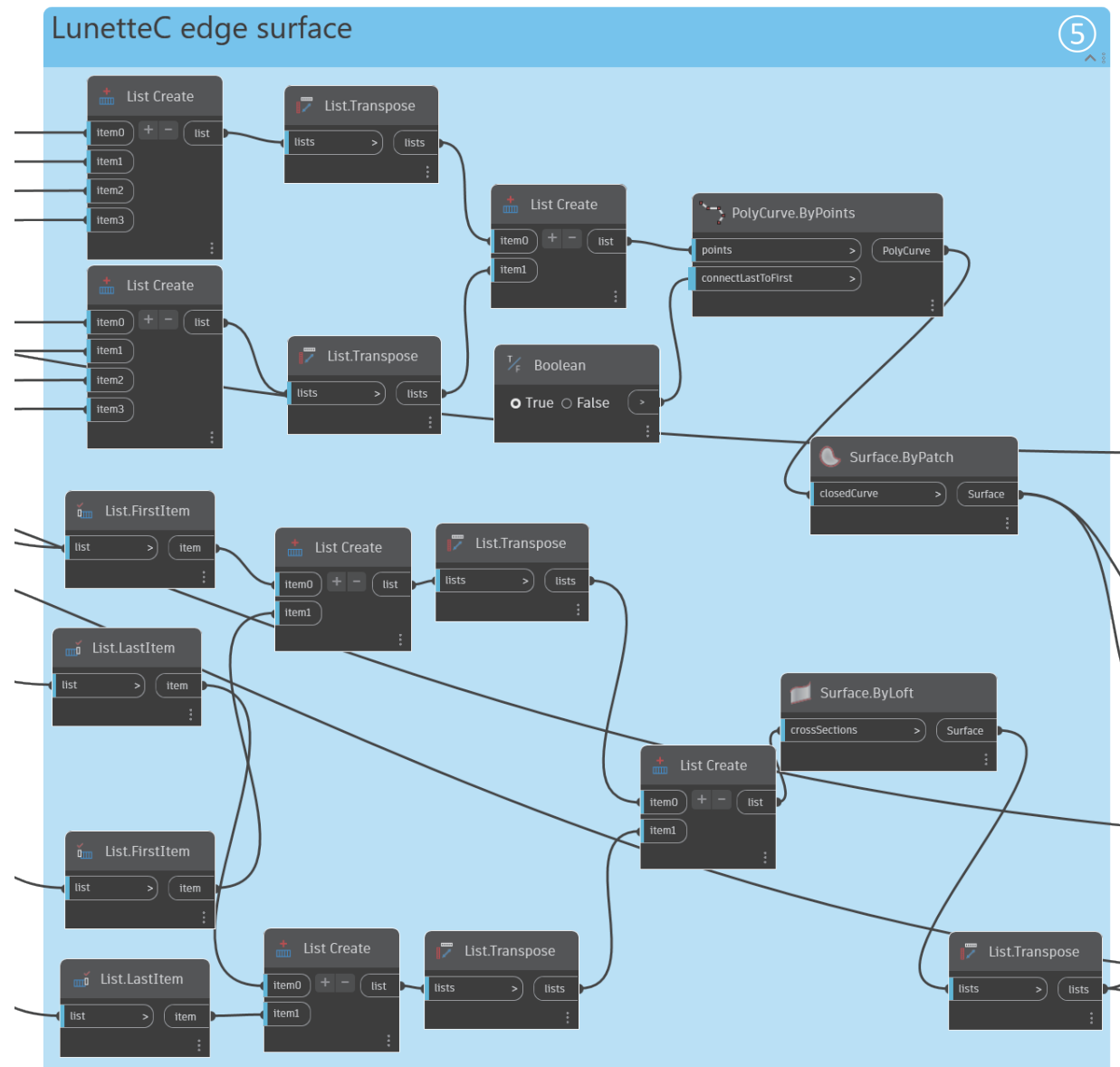
Project or pull the extracted edges back onto the intended reference plane using (Curve.PullOntoPlane). Replace the original edges in the surface definition to ensure an approximately planar side boundary.



4. Half-Loft Arch Construction (LunetteC)

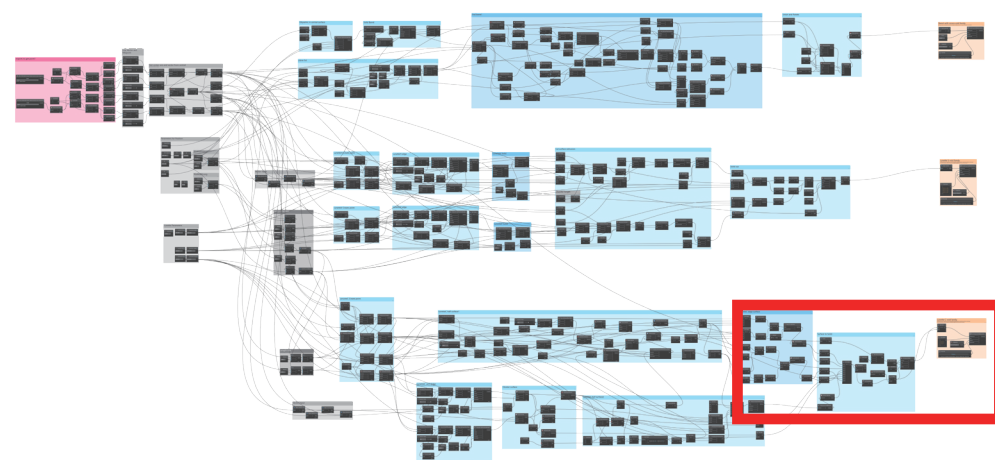
This stage constructs half of the lofted arch for the corner lunette. Due to the complexity of intersecting lines, defined plane shapes, and the need for precise alignment, the workflow involves intensive list management. The process includes:

General Dynamo Graph Overview: VAU_ATR



7. LunetteC Void Family (Output to Revit Family)

This step converts the final corner lunette solids into void families. The node (FamilyType.VoidByGeometry) supports batch creation from multiple solids; therefore, the index of each list item must correspond precisely to the intended solid. A manually ordered string list ensures a one-to-one mapping between solids and their assigned family names. The category is set to Generic Models, and a predefined template path is specified for family creation. Key nodes: FamilyType.VoidByGeometry, Categories, File Path, Code Block.



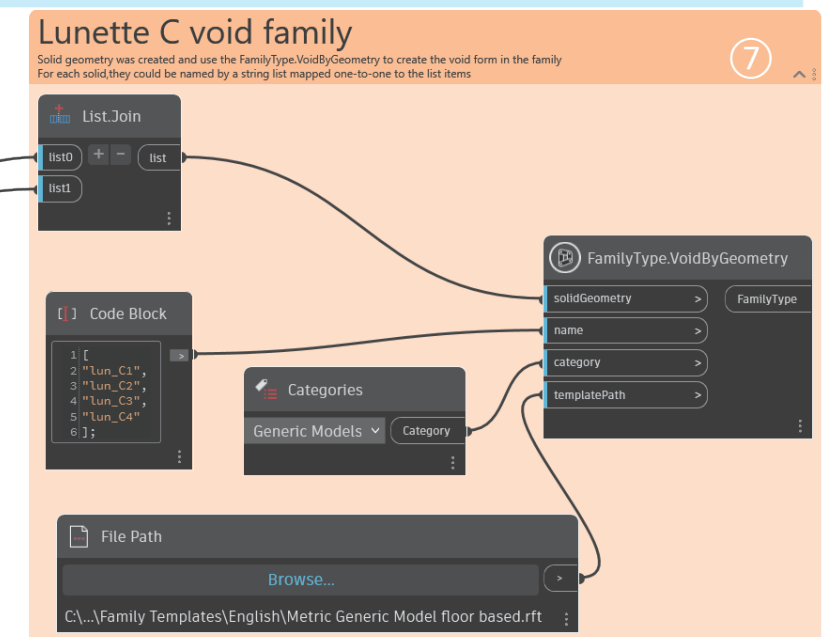
General Dynamo Graph Overview: VAU_ATR

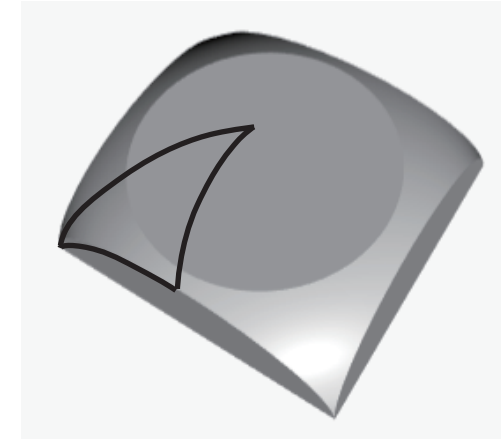
5. LunetteC Edge Surface

Constructs the edge surfaces for the corner lunette. Multiple sets of points are created and transposed into the correct sequence (List.Create, List.Transpose). Profiles are generated (PolyCurve.ByPoints), patched (Surface.ByPatch), and lofted (Surface.ByLoft) to form the edge surfaces. This step involves extensive list restructuring (List.FirstItem, List.LastItem, List.Transpose) to maintain correct alignment between curve sets and surface operations.

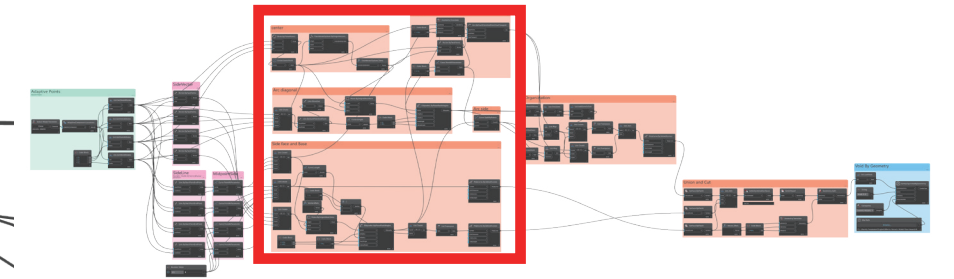
6. Surface to Solid

Converts the assembled surfaces into solids and prepares them for integration. Surfaces are combined into closed profiles (PolyCurve.CloseWithLine, Surface.ByPatch) and joined into solids (Solid.ByJoinedSurfaces). Node(List.flatten) and Node(list.transpose) ensures proper geometric grouping before rotation (Geometry.Rotate) into position. The node of list organization (mentioned in the first page of appendix) (List.Map) applies transformations uniformly across all solid instances.

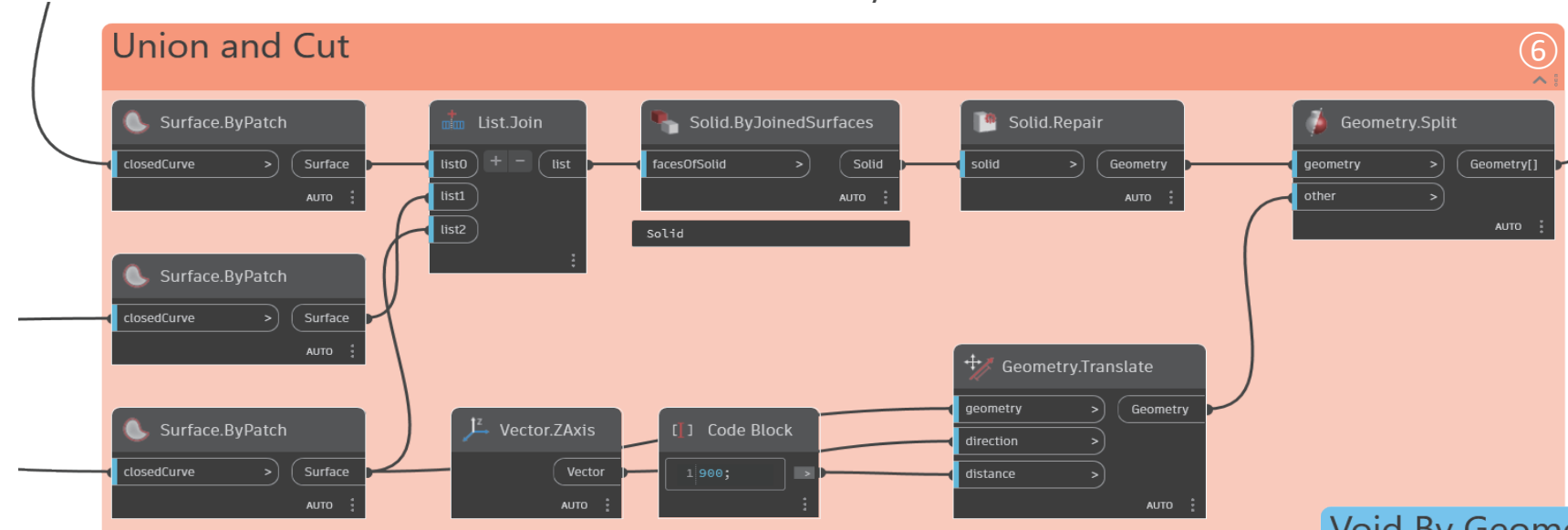
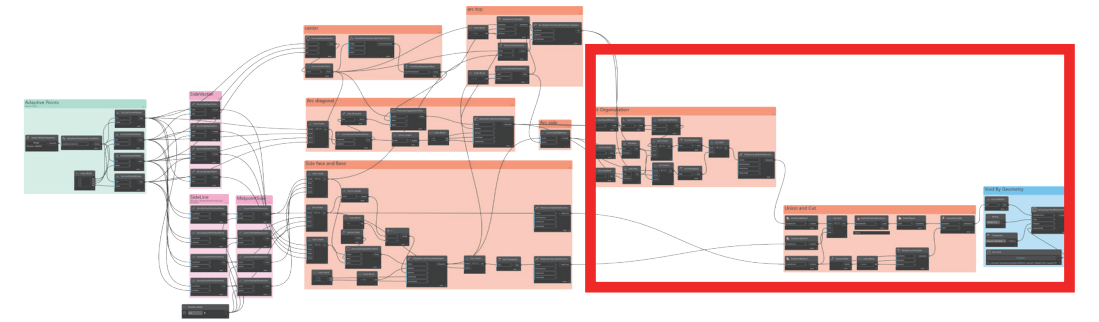
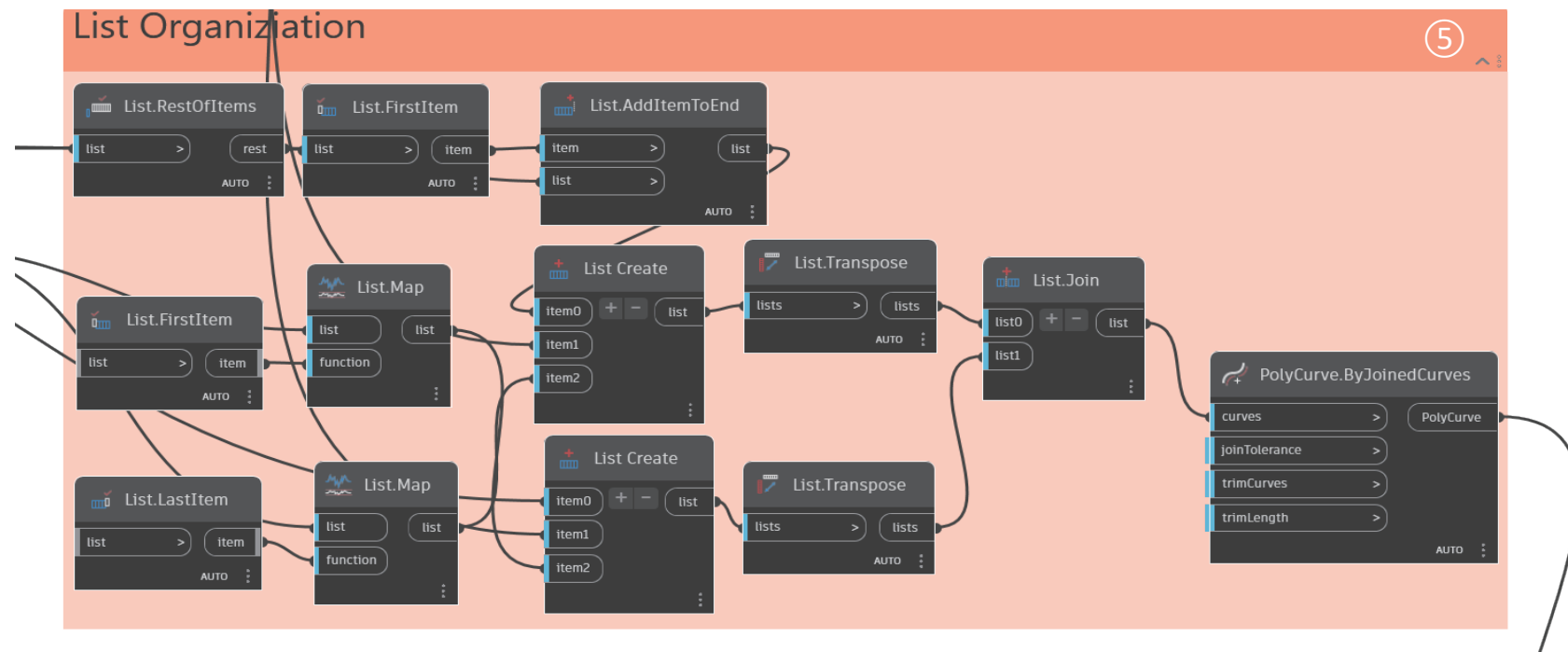




Data input
Geometry Construction
Parameter Input & Pre-processing
Output to Revit Family



- 1. Centre Definition**
 Define the central axis and reference points from adaptive geometry.
 Key nodes: `Circle.ByCenterPointRadius`, `Geometry.Translate`.
- 2. Diagonal Arcs**
 Generate arcs along diagonal planes to connect the centre with the vault corners.
 Key nodes: `EllipseArc.ByPlaneRadiiAngles`, `Plane.ByOriginXAxisYAxis`.
- 3. Top Midline Arcs**
 Create arcs along the vault's top midline, ensuring symmetry with the central axis.
 Key nodes: `Arc.ByCenterPointStartPointSweepAngle`, `Vector.XAxis`.
- 4. Side Boundary Arcs**
 Build side arcs using ellipse arcs or standard arcs, depending on profile geometry.
 Key nodes: `Line.ByStartPointEndPoint`, `EllipseArc.ByPlaneRadiiAngles`.



5. List Organization

Organises multiple curve sets for surface generation. Lists are restructured to ensure the correct pairing of cross-sections and guide curves before joining into closed profiles.

Key nodes: List.RestOfItems, List.FirstItem, List.LastItem, List.AddItemToEnd, List.Create, List.Transpose, List.Map, List.Join

6. Union and Cut

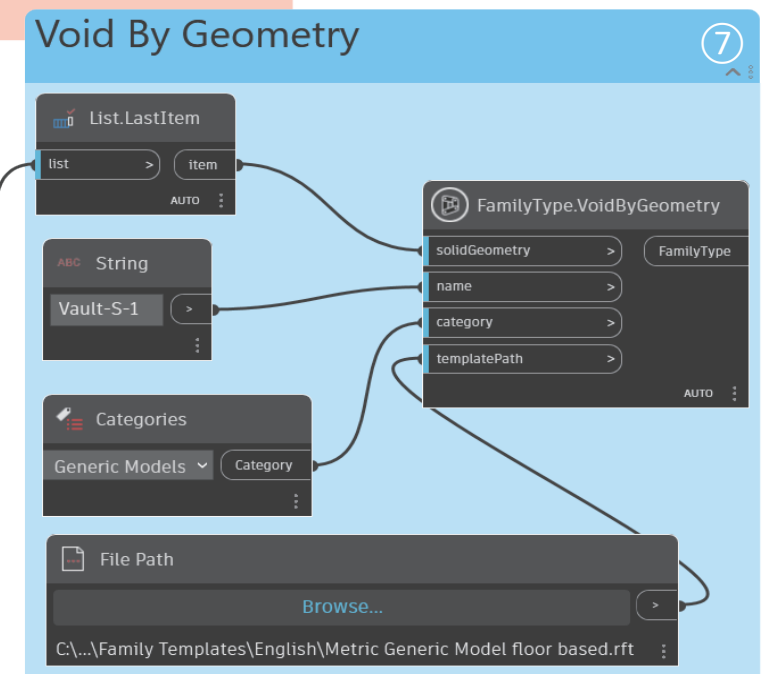
Patches closed curves into surfaces, combines all surfaces into a single solid, and repairs the geometry before applying top-face cutting operations.

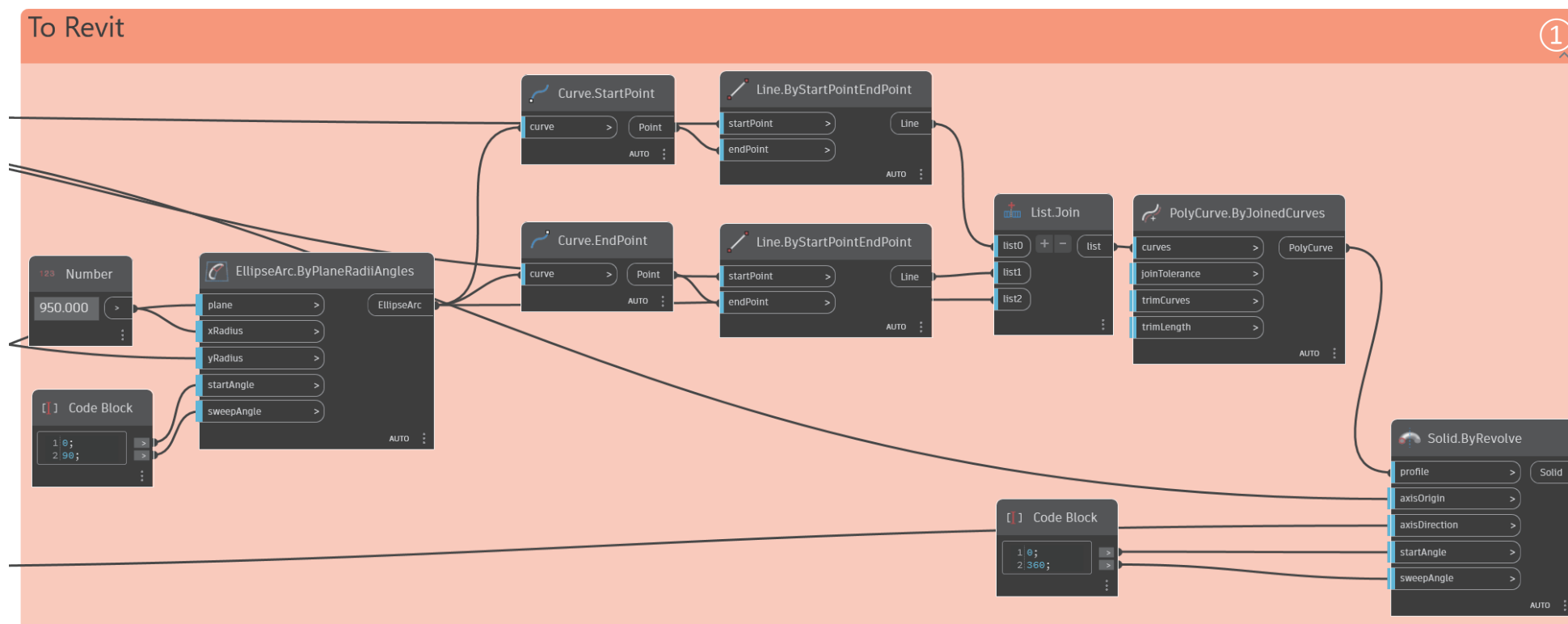
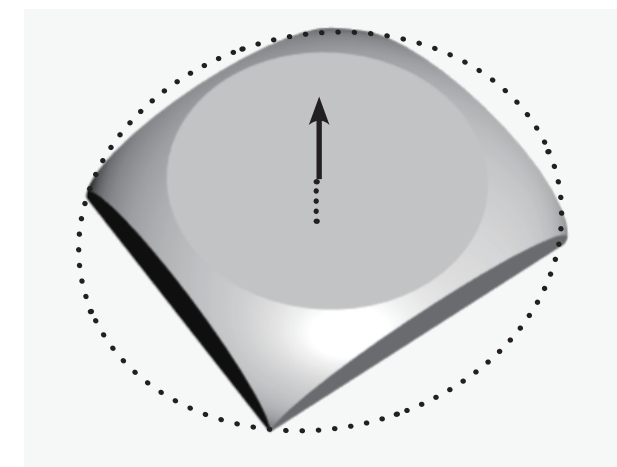
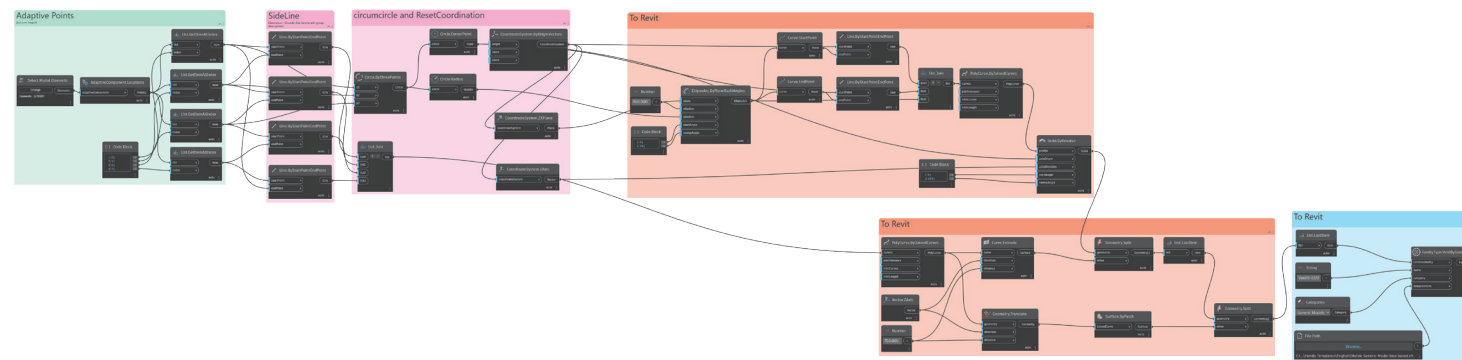
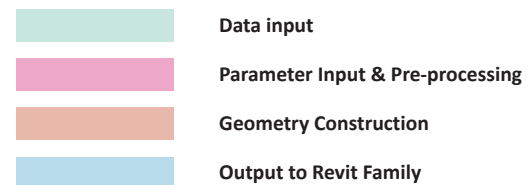
Key nodes: Surface.ByPatch, Solid.ByJoinedSurfaces, Solid.Repair, Geometry.Split

7. Family create

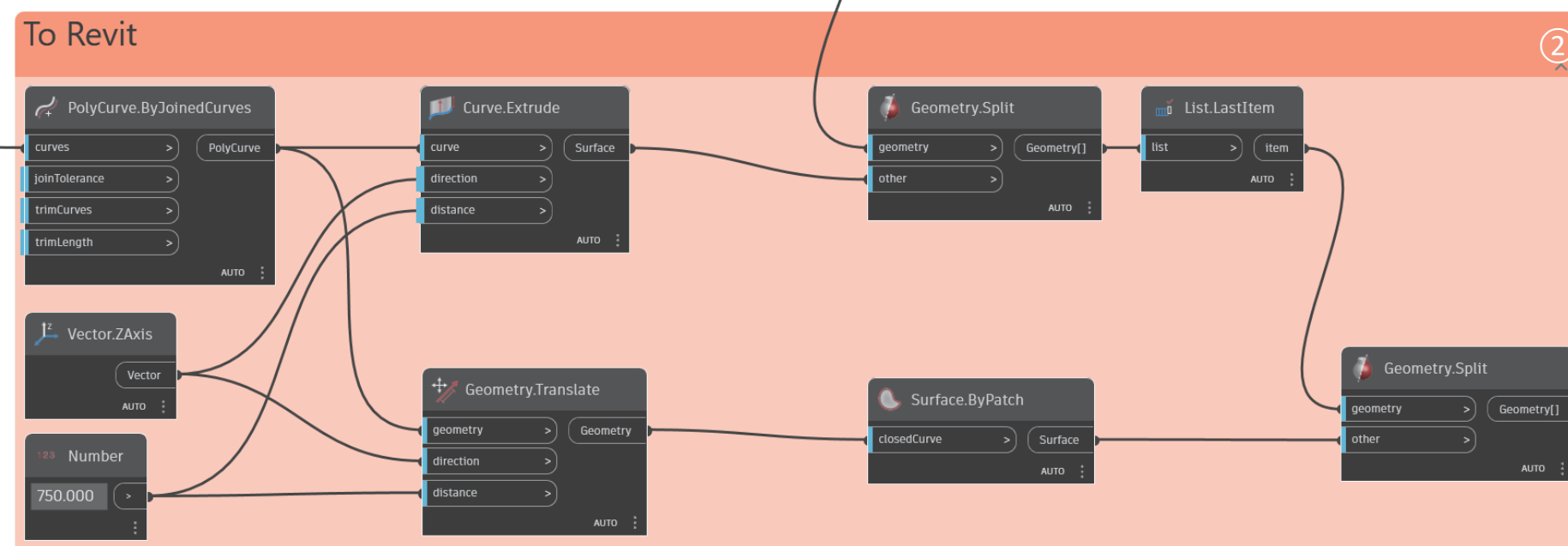
As demonstrated in previous examples, the FamilyType.VoidByGeometry node is used here to convert the final solid geometry into a void family.

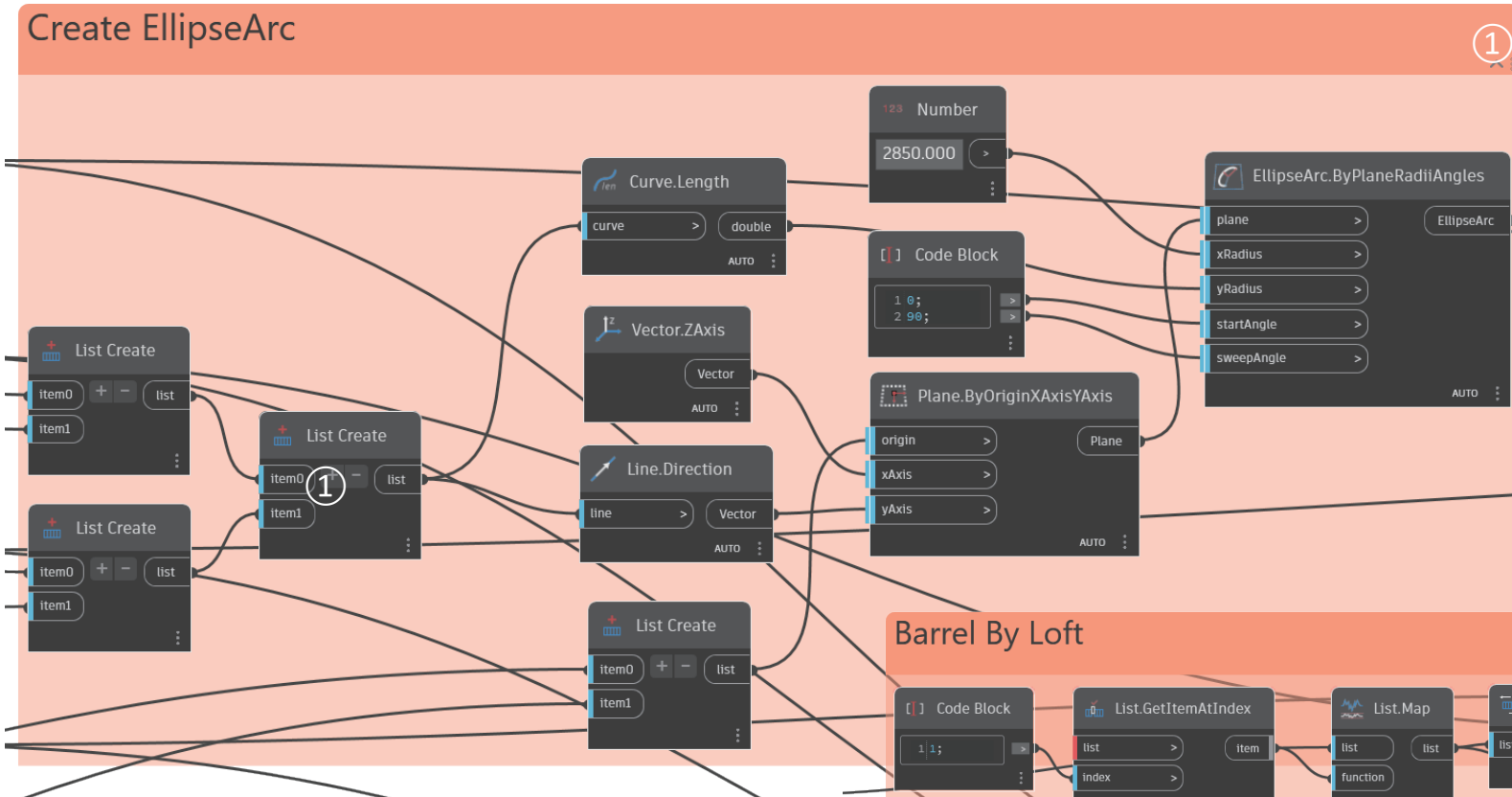
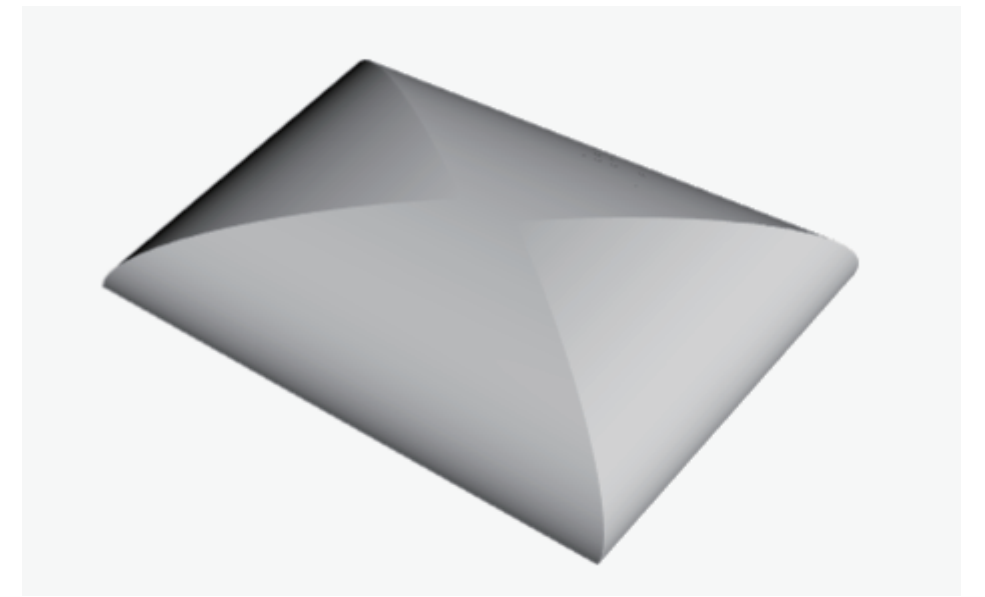
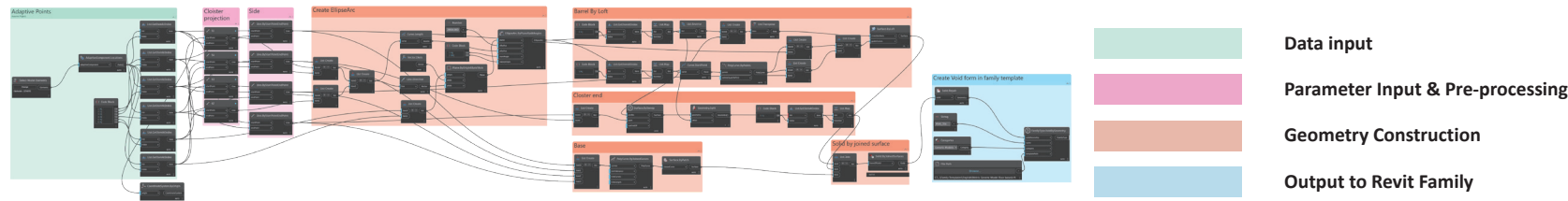
Key nodes: FamilyType.VoidByGeometry



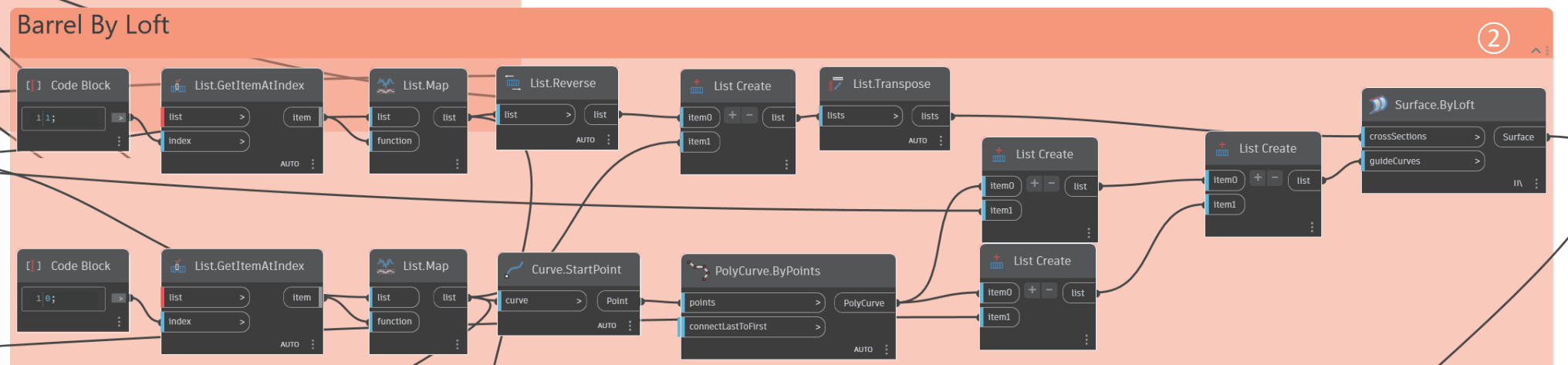


- 1.Revolved Half-Ellipsoid**
 Construct a half-ellipsoid by revolving an ellipse arc around its axis.
 Key nodes: **EllipseArc.ByPlaneRadiiAngles**, **Solid.ByRevolve**.
- 2.Cutting Operations**
 Use planes formed by the edge curves and the Z-axis to cut the half-ellipsoid, and apply an additional top cut.
 Key nodes: **Geometry.Split**, **Surface.ByPatch**.

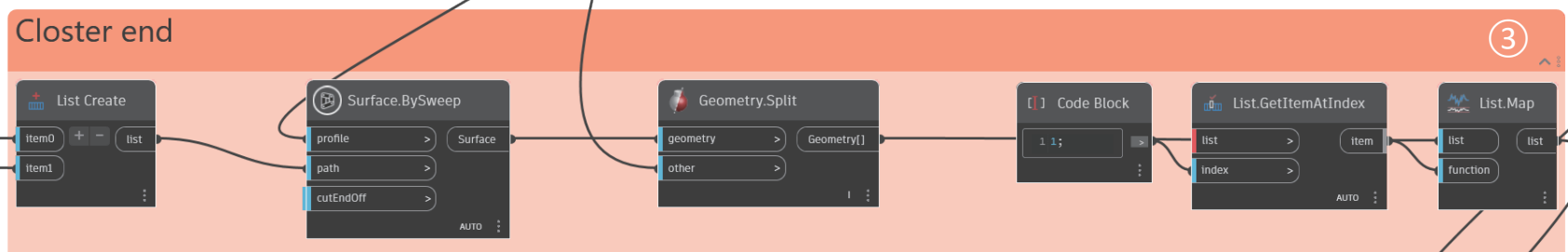




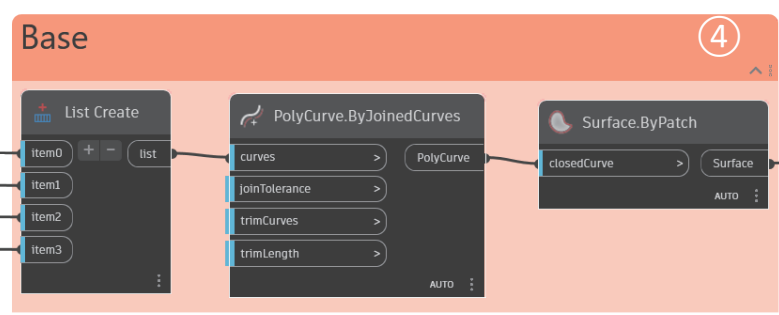
1.Create Ellipse Arc
Define the elliptical profile for the vault using geometric references and parameters for radii, start, and sweep angles.
Key nodes: List.Create, Vector.ZAxis, Line.Direction, Plane.ByOriginXAxisYAxis, EllipseArc.ByPlaneRadiiAngles.



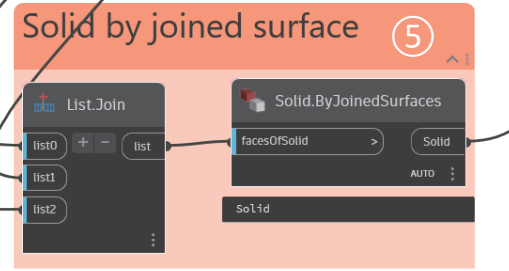
2.Barrel by Loft
Organise cross-section curves and guide curves via list operations, then loft them into the main barrel surface.
Key nodes: List.GetItemAtIndex, List.Map, List.Reverse, List.Create, List.Transpose, Curve.StartPoint, PolyCurve.ByPoints, Surface.ByLoft.



3.Cloister End
Sweep the cloister-end profile along a defined path and split it with intersecting geometry to match vault boundaries.
Key nodes: List.Create, Surface.BySweep, Geometry.Split.



4.Base
Join base boundary curves and patch them into a planar surface.
Key nodes: List.Create, PolyCurve.ByJoinedCurves, Surface.ByPatch.



5.Solid by Joined Surfaces
Combine the barrel, cloister end, and base surfaces into a single solid.
Key nodes: List.Join, Solid.ByJoinedSurfaces.

General Dynamo Graph Overview: VAU_STA