TU WIEN Informatics

Prozedurale Formfaltung

Einbindung von Architekturdetails in 3d Vorentwurfs-Flächenmodelle

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Masterstudium Visual Computing

eingereicht von

Dipl.-Ing. Galina Paskaleva
Matrikelnummer 09626500

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Christian Huemer

Wien, 22. Februar 2023

Galina Paskaleva	Christian Huemer

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Procedural Shape Contraction

Integration of Architectural Details into 3d Conceptual Models

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Master's programme Visual Computing

by

Dipl.-Ing. Galina Paskaleva
Registration Number 09626500

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Christian Huemer

Vienna, 22nd February, 2023

Galina Paskaleva	Christian Huemer

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Galina Paskaleva

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 22. Februar 2023

Galina Paskaleva

Kurzfassung

Diese Arbeit präsentiert eine neue Methode zur Einbindung von 2d Ausführungsdetails aus dem Bereich der Architektur in ein 3d Vorentwurfsmodell mit dem Ziel, ein detailliertes ausführungsreifes 3d Flächenmodell zu erstellen. Das Ziel ist, daraus ein Gebäude in den aus den Ausführungsdetails hervorgehenden Materialien und unter Anwendung der typischen Bauprozesse bauen zu können. Im ersten Schritt extrahieren wir manuell jene 1d Merkmale aus dem 2d Detail, die einen direkten Einfluss auf das resultierende 3d Modell haben. Weiter ermitteln wir die *Sharp Features* des 3d Vorentwurfsmodells, um daraus das Skelett des Gebäudes zu generieren, das aus begrenzenden Kurven und deren Schnittpunkten (Ecken) besteht. Anschließend richten wir die aus den Ausführungsdetails gewonnenen *Feature Collections* an das Gebäude-Skelett entsprechend des Architekturdesigns aus. Das Ziel ist, ein 2-Manifold mit Begrenzung für jede Ecke des 3d Modells aufzubauen. Wir vervollständigen das detaillierte 3d Flächenmodell durch das Aufspannen von *Ruled Surfaces* zwischen den konstruierten 2-Manifolds an jeder Ecke. In dieser Arbeit fokussieren wir uns auf den Algorithmus für die Konstruktion der Ecken-2-Manifolds: Nach dem Ausrichten der *Feature Collections* an das Gebäude-Skelett berechnen wir für jedes *Feature* ein *Rich Descriptor* mittels geometrischer Beziehungsfunktionen. Weiter ermitteln wir für jedes *Feature* aus allen betroffenen Nachbar- *Features* ein Adjazenzgraph. Wir wenden das *Procedural Contraction* an allen *Feature Collections* an der gleichen Ecke des Gebäude-Skeletts schrittweise an. In jedem Schritt errechnen wir ein *Preservation Score* für jedes *Feature* und löschen jenes mit dem niedrigsten Wert, worauf alle Adjazenzgraphen und *Preservation Scores* für alle andere neu ermittelt werden. Diese schrittweise Kontraktion ergibt *Ruled Surface* Segmente, die schlussendlich zu einem 2-Manifold mit Begrenzung zusammengeschweißt werden. Als Evaluierung entwickelten wir ein Prototyp des *Procedural Contraction* Algorithmus in MatLab, den wir an 170 Detailkombinationen testen und die Ergebnisse hier graphisch präsentieren.

Abstract

This work introduces a method for integrating 2d construction documentation-level architectural details into a 3d conceptual model of a building to produce a detailed surface model. The goal is to generate geometry that can be built in the designated material using the appropriate standardized techniques. In the first step, we subject each 2d detail to manual 1d feature extraction to determine those shapes that have an influence on the 3d model. We also extract the sharp features of the 3d model to obtain the building's skeleton, consisting of edge curves and corners. We perform an alignment of the feature collection obtained from each detail with the 3d skeleton, in accordance with the architectural design. Our goal is to build a 2-manifold with a boundary at each corner of the 3d skeleton. Spanning ruled surfaces between neighbouring corner manifolds completes the final surface model. In this work we focus on the algorithm for constructing the corner manifolds: After the alignment of the feature collections with the 3d skeleton is performed, we calculate a rich descriptor, based on geometric relationship functions, for each feature. In addition, we construct its adjacency graph, containing all other features whose descriptor will change in case this feature is discarded. We then apply simultaneous procedural contraction to all feature collections affecting the same corner of the 3d model. In each step a preservation score is calculated for all features, based on their descriptors. The feature with the lowest score is discarded and the descriptors and adjacency graphs for all others recalculated. This contraction produces ruled surface segments that are eventually stitched together into a 2-manifold with a boundary. We evaluated the algorithm by building a prototype in MatLab and testing it on 170 detail combinations.

Contents

CHAPTER 1

Introduction

The process of planning, construction and management of buildings has gone through considerable changes over the last two decades. The very view of the building has become much more holistic. Buildings have complex life-cycles encompassing multiple iterative and interlocking phases. In addition, the time constraints, engineering standards [RM10, FYK10], production and energy efficiency requirements [Cor01], along with the aesthetic aspect of the building place substantial demands on the tools available for planning. Among these demands is that the generation, modification and visualization of geometry is suitable to the needs of each designer or engineer involved during the various planning stages [YWR09], from conceptual design to construction and facility management. This requires a differentiation at least in regard to the *level of geometric detail*, the level of decision making (from a single vertex on a 3d surface to an entire building), the *dimensionality* (2d, 3d, 4d, etc.), and the amount of non-geometric information available [JTT+15]. At the same time, communication among the various planers [Cor01] and interoperability with existing tools [DM13] is essential for a productive workflow.

1.1 Current Developments

Some current efforts in the structuring and management of building data have led to the development of *building information modelling (BIM)*. An example of an open BIM specification can be found at the site of buildingSmart International (bSI) [Bui13] - the Industry Foundation Classes (IFC), which is also an ISO-Standard (ISO 16739:2013) since its release IFC4 in 2013. This data model provides an object oriented interface for template-based data generation and exchange between *CAD tools*. It covers geometry, spatial relationships, geographic information, quantities and properties of building elements and more [YWR09]. It also enables the detailed documentation, management and analysis

of data [DM13]. AutoDesk REVIT[1] and ArchiCAD BIM[2] [DM13] are among the tools partially implementing it (organic shapes are not supported). In order to be certified by bSI, a partial implementation needs to conform to a Model View Definition (MVD), which defines a pre-approved subset of the specification. A complete implementation is extremely time-consuming as the specification contains hundreds of classes that are subject to adaptation to current norms and engineering developments. The specification contains multiple redundancies in order to accommodate the way different users organize and manage information. In this regard, it exhibits flexibility similar to annotation-based applications, e.g. involving *shape annotation* [ARSF09], which rely on a dynamic, often user-specific, *ontology*.

Another consequence of the demand for and introduction of all-encompassing data models are CAD tools that attempt to provide all possible functionality for all planning stages and tasks (e.g., AutoDesk REVIT) [YWR09] or components for various different tasks (e.g., Architectural Desktop ADT, AutoCAD Mechanical, AutoCAD Electrical, etc.). The learning curve for all these applications is very steep [Cor01] and the interoperability still under development. In practice, as soon as the building design has non-standard requirements, the use of multiple tools is still necessary. For example, the architect may need to use both ArchiCAD and Rhinoceros[3] with the Grasshopper[4] plugin to produce *free-form surfaces* in a semi-automated fashion. The HAVC (heating ventilation air-conditioning) and MEP (mechanical electrical plumbing) planner may use C.A.T.S.[5] for most of the project, but also DDS-CAD[6] for the calculation of the summer overheating. The building physicist may need to use ArchiPHYSIK[7] to calculate the standard energy certificate (Energieausweis), but resort to EXCEL or MatLAB to perform the necessary simulations during the planning of a *plus-energy building*.

Since the development of new technologies (e.g., new materials and constructions) and the adaptation of building codes to include their use is a continuous process [Cor01], the continuous integration of all tasks a designer may need to perform in one single CAD tool is not sustainable. This becomes particularly apparent in cases where the designer requires a *rich environment* where, for example, gravity, the load-bearing capacity of materials, the thermal conductivity of constructions, etc. are all simulated [PMAS11]. The situation is similar when the balance between user input and automation needs to be dynamically adaptable [FYK10]. What the *BIM model* offers, however, is an exchange format that can carry all the data necessary for performing any task - e.g. different representation of the same geometry. For example, the IFC4 specification offers a CSG primitive, an extruded solid, a surface model, a B-rep model, and a tessellation for the same basic shape (see Example E.2 in [Bui13]). A small application for a specific task

[1] Autodesk 2018: Revit. https://www.autodesk.com/products/revit/overview
[2] Graphisoft, Nemetschek 2018: ArchiCad. https://www.graphisoft.at/archicad/
[3] Rhinoceros 6: https://www.rhino3d.com/
[4] Grasshopper - algorithmic modeling for Rhino: https://www.grasshopper3d.com/
[5] C.A.T.S. Software: http://www.cats-software.com/de/
[6] Data Design System, Nemetschek 2018: https://www.dds-cad.de/
[7] ArchiPHYSIK, A-NULL 2018: https://www.archiphysik.at/

that reads, modifies, and saves only the part of the data it needs, would be easier to implement and maintain.

1.2 Problem Statement

One of the major consequences of the above-described developments in Architecture, Engineering and Construction (AEC) is the production of multiple digital models. However, these models do not represent one continuous progression of the building's life-cycle. Instead, they are snapshots of the view each domain expert has on the built structure at a particular project phase [Cer11]. For example, the architectural and structural models are often completely separate digital entities that can be viewed together via collaborative BIM platforms, such as BIMCollab1[8], but not edited collaboratively. This often results in the repetition of work steps, including the modeling of the 3d geometry. This in turn, exhausts too many resources and reduces the time available for the evaluation of the design after the input of various domain experts. For example, the input of the building physicist involves wall compositions and connecting details between them. These have a profound impact on the final building design. However, due to time and resource constraints, in most cases, no complete model of the building including all these details can be constructed.

The typical digital form of an architectural concept is a 3d model represented by a collection of curves and surfaces (e.g., non-uniform rational B-spline or NURBS). It also contains the resulting triangle mesh and is generally produced by a CAD tool. The NURBS representation is both parametric and structured [PLH+05] - relatively few control points influencing a large number of vertices on the corresponding triangle mesh. This provides the user with a simple and flexible editing mechanism and is quite successful in the early stages of building design, when support of creativity is essential [PMAS11], and large modifications occur often and have to be performed quickly. From now on we will refer to this type of model (also depicted in Figure 1.1 (b)) as *conceptual 3d model*.

The *conceptual 3d model* is very well suited to solving ill-structured problems [MBRM+16] in an ill-structured domain [MW09]. It does not need to be perfect in order to be useful [MBRM+16]. However, it reaches its limits during the transition from conceptual design to design development, when the tasks become well-defined and find their finalised geometric representation in the form of the *construction documentation*. It is in 2d by default and exhibits the highest level of detail - as seen in architectural details, detailed work plans, or construction structure drawings [YWR09]. In this planning phase the input from the structural engineer, the building physicist, the HVAC and MEP planner, and others has to be integrated in the geometry. *2d drawings* are still the standard for this task, as they are easy to construct [PMAS11], show the true size and shape of the depicted elements, are subject to standardization [RM10], and can incorporate large amounts of additional information (dimensioning, text descriptions) [Cor01]. The main drawback of a 2d drawing is that it requires interpretation (due in part to the

[8] `https://www.bimcollab.com/` (last accessed 2022-10-5)

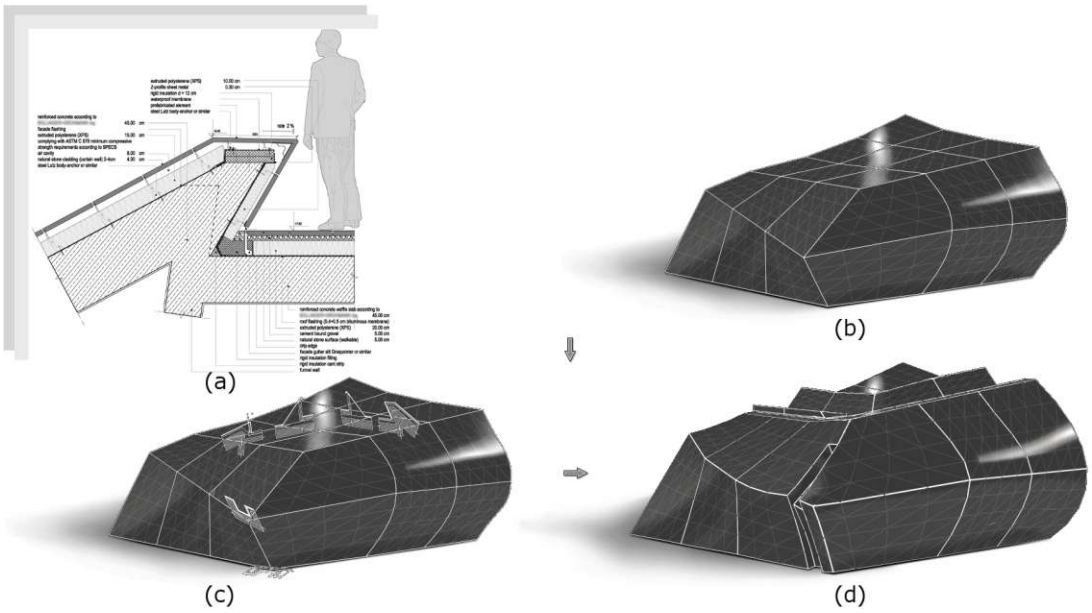


Figure 1.1: Problem outline: The input consists of a number of details (a) and a conceptual 3d model (b). The user (e.g. the architect) assigns each detail to the edge of the conceptual 3d model it was developed for. Subsequently, an algorithm performs the geometric alignment (c) and constructs the resulting buildable 3d model as a 2-manifold (d).

large number of specialized symbols and hatch patterns) and is difficult to visualize in relation to the 3d building model [Cor01]. Automated reconstructions of 3d geometry from such drawings have been demonstrated to work when they depict a complete object [WZY+10], but not when they depict a small part of a complex entity. From now on we will refer to the type of 2d architectural detail drawing that is incorporated in the construction documentation (also depicted in Figure 1.1 (a)) simply as *detail*.

Visualizing the impact of all details when applied to the geometry of the conceptual 3d model, especially when it contains free-form surfaces or corners where more than 3 edges intersect, is only possible by generating a detailed 3d surface model that integrates the geometrical information contained in both the details and the conceptual 3d model (see Figure 1.1 (c) and (d)). However, building this model by hand is at present so time-consuming [LWW08, MM11, dSIR17], it is either not performed at all, or only very superficially. Subsequently, the applicability of the chosen details to the conceptual 3d model is only truly tested on site by the building contractor. Due to cost and time constraints, corrections at this late stage are limited to modifications of the already chosen details that do not compromise functionality.

This work proposes a method, as outlined in Figure 1.1, for the automatic integration the information contained in the detail into the geometry of the conceptual 3d model, which will allow multiple detail configurations to be tested interactively. The output should

be a 3d surface model that contains only geometry that can be built in the designated materials and using the appropriate building technologies. It should also be a *2-manifold*, so that various CAD tools can use it as input - e.g. for simulating load distribution or heat flow. From now on we will refer to this type of model as *buildable 3d model*.

1.3 Research Questions

In order to develop this method, we need to determine the necessary preconditions that need to be fulfilled by both the conceptual 3d model and the details. Further, we intend to devise a robust method for the geometric application of the detail geometry to the relevant parts of the conceptual 3d model. Finally, we need to devise an algorithm for combining the 3d geometry resulting from the application of the details in a deterministic and robust manner. We intend to evaluate the resulting algorithm on real-world use cases from the AEC domain, including free-form structures. Our main focus will be on points where multiple details converge, such as the corners of the conceptual model.

This is a problem of high complexity, as it involves input from various experts concerning structure, wall compositions and the connecting details, where most concerns converge. For example, the geometric impact of the steel structural skeleton, designed by the structural engineer, in combination with the fire-proof cladding and thermal insulation designed by the building physicist, can be such that the architect has to go back to the original conceptual 3d model and make significant adjustments. An algorithm that reveals the necessity for this in the firs place has to allow the automatic integration of expert input into the architectural design, and involves multiple steps. Therefore, we need to answer the following research questions:

RQ1 Which geometric information has to be extracted from a *detail*, so that it can be consistently applied across the entire *conceptual 3d model*?

RQ2 What conditions have to be fulfilled by the *conceptual 3d model*, or by the result of its pre-processing, so that the detail information can be applied to it correctly in the context of the architectural domain?

RQ3 After we apply the corresponding details to adjacent model edges, the algorithm produces the *buildable 3d model*, the so-called as-designed model. Form the point of view of the AEC industries, what degree of feasibility has this model?

RQ4 How much time is saved, on average, by applying our method as compared to the manual modeling of the as-designed model?

After answering the above-listed questions we expect the following results:

R1 A modeling guide for the typical 2d construction documentation level detail, in order to reduce the involvement of human agents in the pre-processing effort,

R2 A modeling guide for the conceptual 3d model to facilitate the extraction of salient features, such as edges and corners,

R3 The abstract outline of the Procedural Shape Contraction algorithm,
R4 The prototypical implementation of the fully automated Procedural Shape Contraction algorithm and a set of test cases.
R5 The evaluation both of the feasibility and of the time-saving aspect of the proposed method.

1.4 Methodological Approach

Both the detail and the conceptual 3d model need to be adapted to the requirements of the task. On one hand, we need to decrease the level of complexity of the detail while, at the same time, retaining the relevant information - this means *feature extraction* and *geometric relationship* calculation. On the other hand, the conceptual 3d model at the end of the conceptual design phase is often a polygon soup as a result of rapid changes or multiple imports and exports between CAD tools. Consequently, the model is supposed to be a *manifold*, but almost never is [BVGP09]. We need to repair it and extract its underlying 3d structure - the surfaces, edges and corners that correspond to building surfaces, edges and corners. Once we have that 3d structure we can insert the 2d information extracted from the detail in its proper context and in the correct position and orientation. Therefore, our methodological approach consists of the following steps:

1. **Literature Review: Chapters 2 and 3**
   According to the procedure for performing systematic reviews outlined in [Kit04], we examined the state-of-the-art in 2d and 3d geometry analysis, repair, segmentation and generation. The result served as the basis for the development of the Procedural Shape Contraction algorithm.
2. **Data Selection.**
   At first, we selected a minimal set of architectural details and conceptual 3d models and extracted the parts relevant for this work. Throughout the development of the theoretical framework and of the algorithm, we added both details and conceptual 3d model configurations iteratively, in order to include configurations we had not considered at the beginning.
3. **Algorithm Design and Evaluation**
   This part of the work was conducted as a case study according to the guideline proposed in [RH09], i.e., as a *software engineering hypothesis evidence search*. Our goal is not to just produce a set of ruled surfaces along the edges of the conceptual 3d model , but to model their intersection at each corner - the point where 3 or more building edges meet, procedurally. Intersecting surfaces ruled by 1d shapes means discarding segments of 1d shapes sequentially while managing adjacency and geometric relationships between them. We define this process as *procedural contraction*.
   (a) **Software Engineering Hypothesis: Chapter 4**
   This includes all steps involved in the development of the algorithm: the

automatic alignment of the details to the corresponding conceptual model's extracted edges and corners; the development of the contraction rules; and the method for step-wise 2-manifold generation. In addition, it includes the formulation of our expectations.

(b) **Data Collection: Chapters 5, 6, and 9**
The data we selected for the development and the evaluation of the algorithm stems from real-world architectural projects. In the end stages of the evaluation, we added several more complex configurations in order to improve the robustness of the algorithm.

(c) **Data Analysis: Chapter 7**
The contraction rules were applied to all use case models and the results were measured in terms of geometric correctness and in terms of feasibility in a construction project context.

(d) **Hypothesis Evaluation and Reporting: Chapters 8 to 11**
The research questions we formulated in the previous section were answered. Further, various aspects of the validity of the case study were analysed and possible improvements collected.

CHAPTER 2

Related Work: Understanding 2d Geometry

In this chapter we review some of the methods for describing, editing, and generating 2d geometry. Our focus is on complexity reduction and on data structures that allow procedural modification or generation of 2d and 1d shapes.

2.1 Extracting geometric information

Many details, particularly in historic buildings, are only available as scanned images. A 2d drawing in the form of a raster image may require *noise removal*, followed by *pattern recognition* to extract basic 2d shapes (e.g., regions) and 1d shapes (e.g., polylines and circle arcs) [WZY+10]. *Visibility decomposition* [FP14] or *contour tracking* can be used to extract connected regions in simpler drawings; *skeletonization* - to find a curve's medial axis in more complex ones. However, dashed lines and leading lines (e.g., connecting annotations with geometry) may be misclassified as noise by those same procedures [YWR09]. Even in 2d vector drawings created by a CAD application repair of disjointed lines, overlapping vertices, or false intersections may be necessary.

The extraction of *semantic* content can be automated only in a well-defined domain [MW09]. A floor plan, for example, contains walls that can be recognized as a pair of parallel lines of the same thickness and on the same layer. In a façade section, on the other hand, a pair of parallel lines denotes a material layer, while the material type can be expressed as a hatch pattern or a color. In all cases, recognizing symbolic annotations is a bottleneck, as even domain-specific symbol libraries can be very extensive [YWR09].

Once the 2d geometry has been coupled with semantic content - through pattern recognition, machine learning, or user input - there are various approaches to its management and modification.

2.2 Shape Grammars

In architecture, shape grammars can be used for shape analysis, querying, description, and generation [KST+09, MG13, HE22] and, possibly, verification [Pat12]. *Shape grammars* are an extension of 1d string grammars into the second and third dimension [WW93, Jan95]. They can be context-free or context-sensitive. A shape grammar is defined as a quadruple $\langle N, T, R, I \rangle$ where N is a set of non-terminal shapes, T is a set of terminal shapes (procedural or modelled) with $N \cap T = \oslash$, R is a set of derivation rules, and $I \subseteq N$ is a non-empty set of initial shapes [WWSR03]. An *attributed grammar* uses attribute matching to select the next rule to apply. [WWSR03] also use a *control grammar* for attribute propagation. Attribute matching and parameterized or stochastic rules give both high-level and low-level control over the result [OS09], depending on their position in the derivation tree. In this manner they allow design variation while, at the same time, preventing a combinatorial explosion of rules [LWW08].

The final model is the result of the evaluation of the derivation tree. The set of admissible rules can be arbitrarily restrictive, which in the case of *split grammars* [WWSR03] ensures that each child node is contained within the parent node. This makes split grammars well suited for incorporating the constraints typical for a building. Grammars can also integrate an arbitrary amount of 3d textured models as terminal shapes and, as a result, produce designs of very rich as well as very basic detail.

A fairly regular geometric pattern requires a small derivation tree (e.g., a façade elevation) while an irregular pattern (e.g., a detail) can give rise to extremely complex trees that can be managed only by an expert [WWSR03, IMAW15]. Shape grammars can be edited as a text (see the CityEngine[1]), which is not practical for the general user [Pat12], or as a graph (CityEngine, [LWW08]) by employing selection by semantic or parametric similarity, or by hand [LWW08], as well as by subgraph copying, exception node declaration [Pat12], and others. Graphs can visualize dependencies between rules [Pat12, GE18], accommodate multiple grammars (see the procedural skeletons generated during shape derivation in [IFPW10]), or support shape emergence [GE18], but can also suffer from visual clutter in more complex cases.

L-systems are grammars in which the rules are applied in parallel, and are well suited for describing branching self-similar structures. They can be stochastic, parameterized, context-free, or context-sensitive. An L-system can emulate growth without consideration of the geometric context [WWSR03], or include pruning as a form of interaction with the environment. [TMW02] demonstrate an application of L-systems for parameterized *mesh-growing* by associating a mesh face with a symbol that, in the next derivation step, replaces the face with a set of (connected) faces.

A major consideration when employing shape grammars is if the intended design lends itself to decomposition in well-defined *discrete components* [OS09] - for example, a detail depicting a steel-and-glass modular structure.

[1] CityEngine, Esri 2018: `https://www.esri.de/produkte/cityengine`

One alternative to grammars is pattern alignment and merging guided by *constraints*. [IMAW15] develop façade designs by utilizing *layers* with depth ordering, a ubiquitous technique in CAD tools, with each rectangular pattern occupying its own layer. The advantage of this method, compared to a shape grammar, is that it can manage overlapping shapes - e.g., a detail depicting a monolithic concrete structure. However, a high degree of regularity in the patterns is still required, and the set of constraints can produce unexpected results for the inexperienced user.

A general shortcoming of shape grammars in relation to our task of intersecting ruled surfaces by *procedural contraction* is their reliance on Euclidean *coordinate systems* and axis-aligned rule application. Alignment of a detail to the geometry of a conceptual 3d model means we need to adapt the detail and the procedural contraction rules to a different angle in each case (see Figure 1.1) - as the dihedral angle between the tangent planes of intersecting surfaces varies not just from edge to edge but even along the same edge. However, more recent publications show positive developments. [EHSF16] demonstrate the use of spherical and cylindrical coordinate systems with split rules and a robust transitioning between coordinate systems. Even more promising, [EPE18] show that grammar rules can be combined not just with affine transformations, but also with lattice transformations, which allows their embedding in an arbitrary (discrete) context.

2.3 Templates and Examples

Alignment and merging can also be applied after the geometry has been adapted to the design requirements through a combination of *affine transformations* and repetition [MM11]. The process can be aided further by *semantic annotation* of the existing shapes according to a domain-specific ontology [ARSF09]. CAD tools such as AutoDesk REVIT and ArchiCAD implement this concept in their extensive parametric template libraries (e.g., window and door types) [YWR09, DM13]. The user can define and modify templates, but cannot edit the constraints guiding their placement, when it is done procedurally (e.g., fan coils in a ventilation scheme). In AEC, compliance with templates can be used to verify an architectural model, e.g., in terms of fulfilling BIM [DM13, Cor01] or habitability and energy efficiency requirements [MdlCRGRM20].

The limitation of geometry generation and modification with the above-mentioned templates lies again in its reliance on coordinate systems, because the only geometric relationship a coordinate system encodes is the distance form a single origin. In situations where distances and angles to multiple objects play an important role *geometric relationship functions* offer a viable alternative [FE20]. Those are functions based on the calculation of distances and angles between geometric objects, including distance and angle ratios and more. [GJWW14] use them to produce *rich descriptors* for various shapes in the context of other shapes. A rich descriptor can hold a combination of geometric and semantic information either independently of the context or in relation to it. [BMP02] for example, use distance histograms to describe a point within a curve in relation to all other points on it. If the shape is integrated into an *attributed relational*

graph [LMV01], adjacency information relative to other shapes can be added to produce an even more meaningful description. In [FE20] the authors demonstrate the utility of such descriptors in generating valid building floor plans.

A major feature of a rich descriptor based on *geometric relationship functions* is its ability to manage the information necessary for the manipulation of geometry, even beyond the constraints of affine transformations. In their work [GJWW14] define the term *pose* as a tuple of a position and a direction vector. A pose can be used as one of the parameters in a geometric relationship function. The set of values a selected number of these functions take for the pose and each object in its neighborhood - e.g., polygon, curve, point, etc. is that pose's *rich descriptor*. This is relevant in an *example-based editing* scheme, where an initial pose is propagated to multiple poses with a similar descriptor relative to another neighborhood within a geometric context (e.g., propagating the furnishing of one room to all other rooms within the floor plan [GJWW14]). All geometric relationship functions are evaluated for the initial pose and neighborhood objects first. Each value is used for generating the level sets for the same geometric relationship function, evaluated for any possible pose and a candidate object. The pose is propagated to the maxima of the sum of all importance-weighted level sets.

The example-based approach finds its most powerful utilisation in machine learning approaches. [ADBW16] demonstrate the power of neural networks in Inverse Procedural Modeling (IPM) to infer entire procedural models including their parameters from appropriate training sets, which then provides a procedural model for generating new shapes. [BBL+17] give a comprehensive overview of deep learning techniques, naturally based on training sets, or examples, including learning the structure of the geometric data (e.g., manifold learning, including multi-dimensional scaling, Laplacian eigenmaps, various embeddings, and deep models) and learning the function of the data (e.g., similarity or correspondence between shapes, based on various transformation operations).

In Chapters 5 and 7 we will develop our own rich descriptor for the 1d shapes we need to manage during procedural contraction. As we have seen, it can define a shape in relation to multiple other shapes both in geometric and in semantic terms, and consequently, retain its descriptive power even under affine transformations or following the discarding of a neighboring shape.

CHAPTER 3

Related Work: Understanding 3d Geometry

As we mentioned in Section 1.4, the ideal *conceptual 3d model* would be a (closed) *2-manifold* embedded in 3d [BDK98]. However, 3d surface models built in a CAD tool are often just polygon soups. File formats that allow a polygon soup include IGES[1], DXF[2], and STL[3] [BDK98], among many others. This situation can be a consequence of the multitude of changes that occur in the conceptual design phase of the planning process [BK97], where one model may consist of parts copied and adapted from other models. It can also be the result of multiple exports and imports between other CAD tools [CCZ+15, XABR21].

In order to align each detail to its respective edge, we need to identify the structure of the conceptual 3d model: the edge curves (or 3d polylines), the dihedral angles between the tangent planes of the surfaces intersecting at each edge, and the corners where the edge curves intersect. From now on we will refer to this structure as the *3d building skeleton*. In this section we review some of the methods for mesh repair and segmentation as well as for feature extraction. Our focus is on the reliable extraction of the 3d building skeleton from the conceptual 3d model.

3.1 Mesh Repair

A *polygon soup* can exhibit the following defects: cracks, where two adjacent parametric surfaces sharing the same bounding curve were tessellated differently [BK97], degeneracies

[1]Initial Graphics Exchange Specification: `https://filemonger.com/specs/igs/devdept.com/version6.pdf`

[2]Drawing eXchange Format, Autodesk 2000: `https://www.autodesk.com/techpubs/autocad/acad2000/dxf/`

[3]STereo-Lithography: `http://www.fabbers.com/tech/STL_Format`

(T-junctions, zero volume solids), duplication of faces, missing faces, overlaps including dangling walls [BK97, Oh19] or more than 2 faces meeting at an edge [NT03, Oh19], inconsistent orientation, intersecting faces, etc. These defects can cause problems during finite element analysis, surface smoothing, automatic model simplification, various transformations [NT03, Oh19], manufacturing [Dum17], and during simulations (e.g., of heat flow or force distribution). They also produce visual artifacts (e.g., when applying radiosity) [ESV98]. The goal of repair procedures in general is not just a topologically sound manifold, fit for performing the algorithms mentioned above, but also the model envisioned by the designer. This makes user involvement at certain stages desirable [BDK98, Dum17, XABR21].

Surface-based repair algorithms can perform a *global consistency check* - e.g., if each face is oriented counter-clock-wise (CCW) when viewed from the outside [BDK98]. *Holes and open edges* can be detected through computing Jordan curves on the surface [BK97]. *Connected components* can be grown starting at any face [CBK20]. Subsequently, *duplicate polygons* can be detected using the adjacency graph of the component [BK97, CBK20]. This turns the model into a collection of 2-manifolds with boundaries - it does not repair cracks or intersections [NT03]. Various *hole-filling* algorithms can be applied - from Delaunay triangulation of the hole boundary polygon [XABR21] to surface interpolation based on radial basis functions, which produces an implicit surface over the hole that preserves C2 continuity across the boundary [Oh19]. Subsequently, the implicit surface is triangulated and stitched to the rest along the boundary of the hole [CC08, Oh19].

Even more elaborate algorithms perform *re-meshing*. For example, when the geometry represents a modular structure (e.g., steel-and-glass) a recalculation of the mesh as a planar quad mesh presents significant advantages: smaller number of edges (i.e., lower number of supporting beams), and lower node complexity (which gives a decided advantage during manufacturing) [SVS18]. The panelization can be performed using discrete differential geometry on a parameterized surface, which allows the application of geometric constraints adequate to the building material [Pot08, Dum17].

The conversion to a *volumetric model* avoids topological ambiguities inherent in polygonal models [SGY+21] and takes advantage of image processing techniques. Voxelization can be performed by applying 3d filters to the surface models to determine the inside and outside regions, or by calculating a distance map from the input polygonal model [NT03]. The standard algorithm used for *iso-surface extraction*, after modification of the volumetric model, is the marching-cubes algorithm [KBB16]. Voxelization, in addition to Laplacian-based contraction, or Voronoi diagrams, can also be used to extract the *curve skeleton* (a subset of the medial axis) of a surface model, producing a 1d representation that captures its topological characteristics and is easy to edit [CTO+10].

For the extraction of the *3d building skeleton* we do not need a manifold. A simple global consistency check to ensure the correct orientation of the face normals should provide sufficient repair as face normals can be used to determine the dihedral angles between the tangent planes of neighboring surfaces. Sharp feature detection, as we will see in the following sections, completes the extraction.

3.2 Mesh Simplification

The ideal case when simplifying a mesh would be the preservation of global features as well as important mini-structures [DZXL09], while, at the same time, discarding unnecessary fine detail [NT03] and noise - i.e., detecting features important to *visual perception*. According to [XC09], the initial object detection in human visual perception occurs on a coarse grained level where only the features facilitating figure-background segmentation for roughly four objects are considered. Object recognition occurs at a later stage, when the more fine-grained information has already been processed by the visual cortex. According to [OT06], initial scene recognition relies on the detection of similar features at multiple resolution levels and combining them in a spatial layout, which is sufficient for building a hypothesis. This hypothesis narrows down the possible interpretations of the high-resolution information in later stages by employing the mechanism of expectation [SE09]. Examining the mesh at different scales, e.g., *multi-scale mesh analysis* using blowing bubbles (spheres centered at the vertices) [AKM+06], or even a simple decomposition into a smooth base mesh and detail representation, encoded in the local frames of the base mesh [LSLCO05], can emulate a workflow similar to that of the human visual cortex, especially if the user is involved.

A side-effect of mesh simplification may be *topology simplification*. [NT03] mention a case of a cube with 300 small tunnels where mesh simplification has a dramatic effect on the topological genus of the model. Geometric simplification methods that allow topological changes include grouping vertices into clusters according to a uniform spatial grid (followed by merging the vertices and killing degenerate polygons), vertex merge and vertex split operations [ESV98], reducing the number of vertices in relatively flat regions, edge-collapse operators, face clustering [CCZ+15], cutting and feature extraction in models with high genus [CTJL10], α-hulls, etc. [ESV98] for example, use a method for topology simplification, similar to α-hulls, that is equivalent to rolling a sphere of radius α along a surface and triangulating over regions that it cannot reach. However, self-intersections and inconsistent orientation remain unaffected by such methods.

Mesh approximation approaches include calculating base vectors as functions of the mesh connectivity and representing the mesh as a combination of those vectors [SCOIT05], or using transformations to and from a topological space [KBB16]. There are numerous machine learning methods for feature-preserving de-noising [NALM19] or 3d texture detection and extraction [TBOW21].

Volumetric approaches (combined with *image processing* techniques [NT03]) can handle degeneracies, but are more complex [ESV98]. For example, applying *morphological operations* to volumetric representations can produce topology simplification: erosion followed by dilation widens holes, eliminates small features and disconnects components; dilation followed by erosion closes holes and connects disconnected parts. The distances used in these operations determine the level of change and can vary across the model. Morphological operations, however, destroy thin-shelled volume representations [NT03], such as walls in a building.

During such mesh simplification procedures, it can be beneficial to maintain hybrid models, e.g., a NURBS "ground truth" and a dynamic Boundary representation (B-rep) model linked by a correspondence graph, especially when the model is used for multiple purposes [CCZ+15], as is very often the case in AEC.

3.3 Segmentation

Mesh repair, simplification, and possibly analysis at multiple scales (to avoid sensitivity to noise and tessellation [AKM+06]), are closely coupled with *segmentation* and *feature extraction*. There are two major types of segmentation - *part segmentation* and *surface segmentation*. The first type has its origin in the study of human perception and is used in shape matching, retrieval, reconstruction, and reassembly. The second type partitions a surface into patches according to a criterion. Among its applications are geometry-image creation, mesh simplification, detail transfer, deformation, compression, collision detection, etc. [Sha08, AKM+06]. The general techniques to achieve both types of segmentation include region growing [TPT17], hierarchical (fuzzy) clustering [YW18], iterative and recursive clustering [TYPC20], spectral analysis, as well as implicit methods [Sha08, AKM+06]. Segmentation can be employed as a *reverse-engineering* algorithm as well [GSMCO09].

Segmentation is generally carried out according to chosen attributes of the mesh: *Fitting* to various primitives (cones, cylinders, rolling ball blends, quadrics, developable surfaces, etc.) in the least squares sense (and at varying scale) detects *smooth regions*. The analysis of the global variation in the vertex normal or the dihedral angle between faces, on the other hand, detects *sharp features* [WGJC18] and can be used for polygon clustering [CTJL10]. Segmentation according to curvature offers a fine balance between smooth and sharp. It also allows directional sensitivity by employing, for example, Curvature Co-occurrence Histograms (CCH) [WGJC18]. The extraction of the medial axis produces a *skeleton of the shape* [CTO+10, ZXK+19], while Shape Diameter Functions (SDF) provide a good distinction between thick and thin parts [SSCO08, HZK16]. In fact, while some methods concentrate on skeleton extraction only [ATC+08, SS22], multiple others couple mesh segmentation and skeleton extraction [SSCO08, ZXK+19]. Symmetry (detectable through sampling) or motion characteristics are also used [Sha08].

Segmentation is often handled as an *optimization problem* on 2-manifold (closed) meshes that accepts local extrema as solutions [Sha08]. The *constraints* may be of various types, according to the specific algorithm - cardinality constraints (e.g., maximal or minimal number of segments), geometric constraints (e.g., minimal or maximal admissible area of the sub-mesh, convexity, compactness, etc.), topological constraints (e.g., segments homeomorphic to a disc) [Sha08], or fixed feature points [CTJL10]. [BZA+16] lists a number of further measures of segmentation quality.

Furthermore, segmentation can be based on *semantic correspondence* between geometrically or topologically dissimilar meshes. That correspondence can be learned from user-guided examples by devising a similarity measure [BVGP09] or functional mapping

[FCSF17]. For instance, by utilizing these methods, a surface ridge interrupted by noise or scanning errors can be understood as an uninterrupted sharp edge.

3.4 Sharp Features

Feature extraction goes hand in hand with segmentation. Feature lines are powerful geometric *shape descriptors*. They are curves (or crest lines) on a surface that represent its visually important characteristics. They can be defined as the local extrema of principal curvature along corresponding principal directions [LZH+07], or as the set of points where the variation of the surface normal in the direction of the maximal change has a local extremum [OBS04]. The variation of the surface normal along the surface can also be used as the basis for a vector-valued image defined over the surface. Features can then be extracted by utilizing image processing techniques (e.g., various filters, edge detection algorithms, etc.) [LZH+07]. Further methods for sharp feature extraction include guided patch centroid normals [ALM19], spectral analysis, singularity theory and bifurcation theory [HPW05], resampling with variable density to capture sharp features, manual selection [CTJL10, LHL20], Markov random field labelling [ZGX+22], and various machine learning techniques. Among the latter, in [LS20] the authors present a method for learning the structure of shapes, while in [QYSG17] we find a hierarchical neural network performing multi-resolution sampling and grouping, which provides robustness of feature recognition even with regard to sampling irregularities and non-rigid transformations.

Sharp feature preservation techniques can result in rough and squiggly curves due to noise [CC08, HPW05]. In addition, local methods for curvature computation for discrete surfaces produce discontinuities in the estimated curvature because of non-uniformity of the local vertex neighborhoods [OBS04]. A mesh smoothed in pre-processing can exhibit smoother feature lines, but the process is time consuming and prone to producing distortions. Instead, smoothing of the extremalities can be applied [HPW05]. Global methods also include globally (computationally more expensive) or compactly supported *radial basis functions (RBF)*. They involve approximation of the mesh by a RBF surface, projecting the mesh vertices onto the surface, and estimating curvature tensors and derivatives at the vertex as those of the corresponding surface point. The detection of curvature extrema is performed using these estimates [OBS04]. After feature extraction, feature-sensitive re-meshing can be applied to the surface [XABR21]. It is to be noted, that all these methods can produce different feature classification depending on the scale of application [LZH+07].

The methods for mesh simplification, segmentation, and sharp feature extraction depend on knowledge about the represented shape's scale and the relative importance of topological and geometric features. User input is often needed as a guide [FCSF17]. In our case we have a *conceptual 3d model* constructed in pre-defined units (e.g., meters). The *detail* (also drawn in pre-defined units) gives us information about the minimal distance between any two distinct points - for example, the shortest line depicted in the detail.

The building code supplies the admissible tolerance for the chosen construction method - e.g., ÖNORM DIN 18202. This allows us to determine the maximal permissible error for any mesh simplification method we choose and the minimal size of the sharp features we extract.

3.5 Shape Generation

Before we conclude this chapter with the most important take-away, we give a brief overview of existing mesh generation algorithms and compare them to the goal of our work. Aside from the application of shape grammars to the 3d domain that we already discussed in Chapter 2, the most extensive research in the last few years has been dedicated to the application of machine learning techniques to shape generation. The applications are generally limited to a well-defined domain, as the performance of the underlying neural networks depends on the quality and completeness of the training set.

In [SGL+18], the authors rely on parsing input 2D or 3D shapes into a CSG tree using methods from Natural Language Processing (NLP), such as greedy decoding or beam search, to build a shape parser. Based on the similarity of the output to the input shape, a "reward" is calculated. However, the method is tested only on shapes that produce trees with less than 20 leaves. [KLMK19] present a string-based shape synthesis algorithm which is trained to recognise the relationships between building blocks in a pre-defined "castle domain". [LLZL21] relies on adversarial training on signed distance and occupancy fields of a class of shapes in order to learn the statistically common features of this class. Common to all three approaches is the inability to edit or fine-tune the result.

However, user input can be considered at various stages. For example, [KMG+21] demonstrates the application of user-defined pose and load-bearing constraints to pre-defined classes of geometric objects. In [JBX+20], the authors showcase a hybrid approach to shape generation where the advantages of procedural modeling and of machine learning are combined. Instead of using a shape grammar, they devise a custom imperative language for deterministic shape programming. This is followed by a stochastic neural network that learns to generate programs from existing ones. The user can edit the training set of shapes. Maybe the most direct involvement of the user is shown in [SGY+21], where high-resolution B-reps are generated from very low-resolution Minecraft-like input voxel models, and in [HKYM17] where 2d freehand sketches are converted to procedural model definitions.

All of the described techniques offer very little control to the user in the later stages of shape generation and rely on a well-defined shape class. What we aim to achieve in this work, however, is an iterative adaptation of 2d information to a 3d model whose final shape is unknown to the user. What is known are various properties of the *detail* in the context of the *conceptual 3d model*, from which we can derive a rich descriptor.

3.6 Rich Descriptors of 3d Surfaces

The result of feature extraction algorithms is a set of 1d curves that describe the most salient characteristics of the 3d surface - and can be used as one component of a *rich descriptor*. The expressiveness of this descriptor increases if the relationships between each two curves and between each curve and the 3d surface are also used [GSMCO09].

[GSMCO09] describe a method for extracting the 1d "wires" that define the 3d shape and using them for intuitive 3d shape manipulation. A crucial step in their algorithm is the establishing of the defining characteristics of the wires individually (e.g., rectangle) and as groups (e.g., co-planarity) that are to be preserved during manipulation by the user. The transformation is propagated to the underlying mesh by constraining *local frames* [LSLCO05], which are orthogonal vector triples computed from the discrete forms of the *1-ring neighborhood* of each vertex, and enable a decomposition of the representation into normal and tangential components.

Another example for using 1d curves as 3d *shape descriptors* is the work of [SAG+13]. In this case the descriptors are user-defined sketches that are used for *3d shape generation*, similar to [HKYM17]. However, here, by employing both geometric (e.g., parallelism, collinearity, perpendicularity, concentricity, coplanarity, etc.) as well as semantic (user-defined tags) relationship recognition, the algorithm directly translates the 1d curves into 3d geometry. The translation relies on automatic matching of 3d primitives (e.g., cylinders, spheres, etc.) to feature curves (non-view-dependent) and silhouette curves (view-dependent).

Descriptors of a 3d shape that are particularly well suited to both manipulation and generation result from the *hierarchical decomposition* of the shape into components [ZFCO+11]. The descriptor of a component itself can contain geometric properties directly observed in the data - curvature histograms, shape diameter histograms, principal component analysis (PCA) based descriptors, etc. [KCKK12]. The fitting of a 3d primitive can provide a cage-like *volumetric controller* [ZFCO+11]. Higher-order descriptors can also carry *adjacency* information (e.g., the number of components of one type that are adjacent to a number of components of another type) [KCKK12]. The hierarchy keeps the number of individual descriptors small and allows shape control on multiple scales.

In the next Chapter we develop the outline of our algorithm and determine the types of geometric and semantic information our rich descriptor needs to hold.

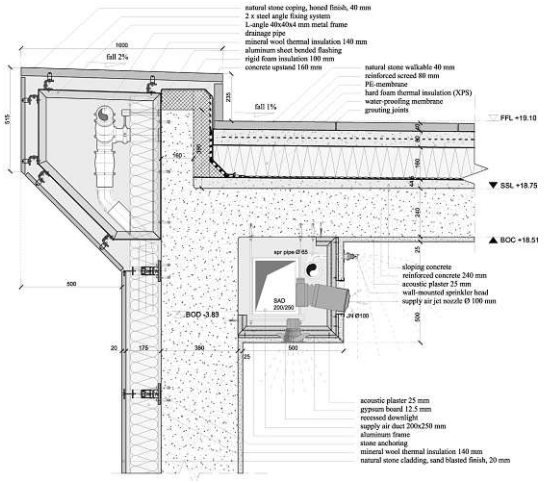
CHAPTER 4

Algorithm Outline

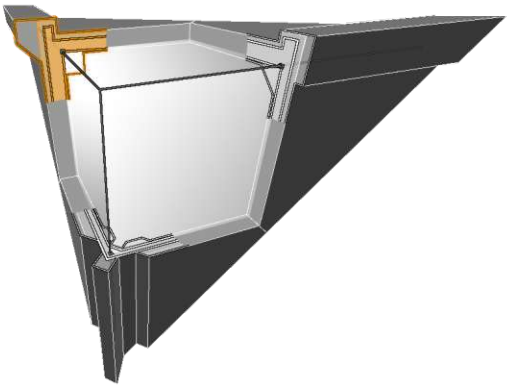
As discussed in Chapter 1, the typical 3d surface model produced by a CAD application during the conceptual phase of building development is a polygon soup. It lacks connectivity and semantic structure of any kind. If free-form surfaces are involved, not even domain-specific segmentation heuristics (e.g., a rule "walls are vertical") can be of any help. However, there are assumptions we can make. First, since *sharp features* at joints between walls, wall and ceiling, wall and roof, are common in architecture, mainly due to the types of building materials, we can rely on them being semantically significant. Second, architectural models are built in scale $1:1$ (see Figure 4.1a), even if the corresponding printed plans are produced in other scales. This allows us to perform smoothing or averaging operations to already extracted features because we can apply the standard tolerances for various materials [RM10] as admissible margins of error.

We can use dihedral angles between mesh faces to detect edges along sharp feature candidates and then apply piece-wise principal component analysis to those edges to extract a piece-wise linear representation of each sharp feature possibly describing a building edge - i.e. a polyline. Once we have obtained the approximations of the building's edges we can calculate the approximate position of its corners as a convex combination of the numerically determined points along each polyline that are at shortest distances from each other, or by looking for the local minima in the distance map calculated for all polylines. These steps lead to the extraction of the *3d building skeleton* - see the cube corner depicted in Figure 4.1b.

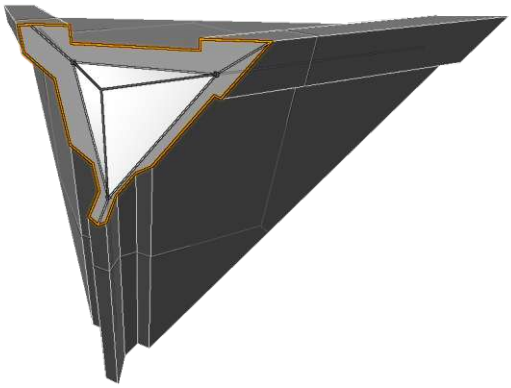
Since the *detail* that is to be applied to each corner is also in scale $1:1$ and subject to the same standard building tolerances as the 3d building skeleton, we can work with the already extracted linear approximations of the building's edges in the vicinity of each corner without compromising accuracy. This is of particular importance because, in order to provide enough space for all details to be intersected, we need to apply them at an appropriate distance from the corner (e.g., the diameter of the largest circumscribed



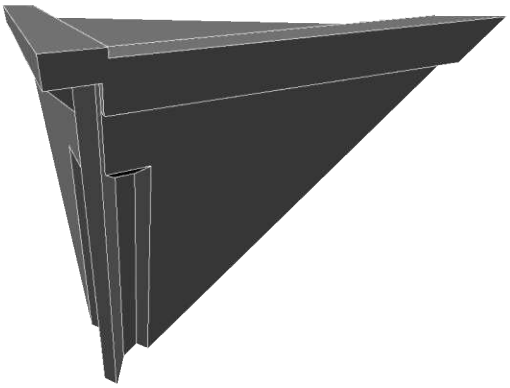
(a) An excerpt of construction documentation: a typical 2d architectural detail drawing.



(b) The details aligned with their respective edges of the 3d building skeleton derived from the conceptual 3d model (in this case - a box).



(c) Procedural contraction: the highlighted 1d shapes are processed sequentially according to their rich descriptors.



(d) The resulting corner of the buildable 3d model.

Figure 4.1: Applying a detail to a conceptual 3d model.

circle for the geometry of all relevant details) - see the positions of the details within the 3d building skeleton in Figure 4.1b.

After reviewing the available methods for 2d shape analysis in Chapter 2, it becomes clear that the complexity and variability of 2d architectural details (see Figure 4.1a) does not lend itself to translation into a *shape grammar* or any other representation relying on at least some degree of regularity. The application of *templates* is also limited due to the sheer volume of possible details. However, as we will see in Chapter 6, templates can be used to accommodate geometric variability in the same basic detail. As such details also carry the expert knowledge of the architect, building physicist, HVAC- and MEP-planer and others, the extraction of the 1d shapes that affect the geometry of the final 3d surface

model is best performed under expert supervision. We will elaborate further in Chapter 6 when we discuss the methods for establishing *geometric* and *adjacency* relationships between features of interest extracted from a *detail*.

Those geometric features of interest, coupled with the geometric and adjacency relationships between them, as well as with semantic information, become a *rich descriptor* that we will refer to as *feature collection*. Figure 4.1b depicts the geometric features of interest of each detail aligned with the 3d building skeleton; Figure 4.1c shows the resulting feature collections aligned with the skeleton sides, already in the process of procedural contraction. Each step of the procedural contraction algorithm described in Chapter 8 and Chapter 9 eliminates one line segment of any of those feature collections. The algorithm employs *geometric relationship functions* to help establish an intuitive contraction sequence that leads to the result shown in Figure 4.1d.

It is our software engineering hypothesis that the workflow depicted in Figure 4.1 is both time-saving and generalisable and that it automatically generates feasible 3d geometry.

CHAPTER 5

The Detail Library

In this Chapter we will address the data selection process we outlined in Section 1.4. In addition, we will provide the theoretical underpinnings for the answer to research question **RQ1**: *Which geometric information has to be extracted from a detail, so that it can be consistently applied across the entire conceptual 3d model?*, which we will give in full in Chapter 10. Finally, we present result **R1**: *A modeling guide for the typical 2d construction documentation level detail, in order to reduce the involvement of human agents in the pre-processing effort*, which we listed in Section 1.3.

A typical detail can be seen in Figure 5.1. We will now outline a procedure for translating a detail into a *detail description* - containing a contour comprised of line segments coupled with their user-confirmed geometric properties, the geometric relationships of significance between them, and any user-supplied semantic information. The detail descriptions will be subsequently translated into feature collections in Chapter 7.

5.1 Extracting the Detail Description

Figure 5.3 shows the extraction of the geometric information (or features of interest) from the detail depicted in Figure 5.1. This can be handled separately, according to material, or together as a monolithic shape (see also Figure 5.4), depending on our intent. If a quick shape study is needed to determine the visual impact of the detail on the conceptual design, a monolithic representation will supply sufficient information. If, on the other hand, a bill of quantities for the construction documentation has to be calculated, a layered model will be more appropriate.

The detail description is defined in relation to the *structural axes* of the conceptual 3d model. The structural axes are the intersections of all surface tangent planes with the plane normal to the edge segment of the 3d building skeleton and intersecting it at the site of the detail definition. This site is usually indicated in the corresponding floor plans of

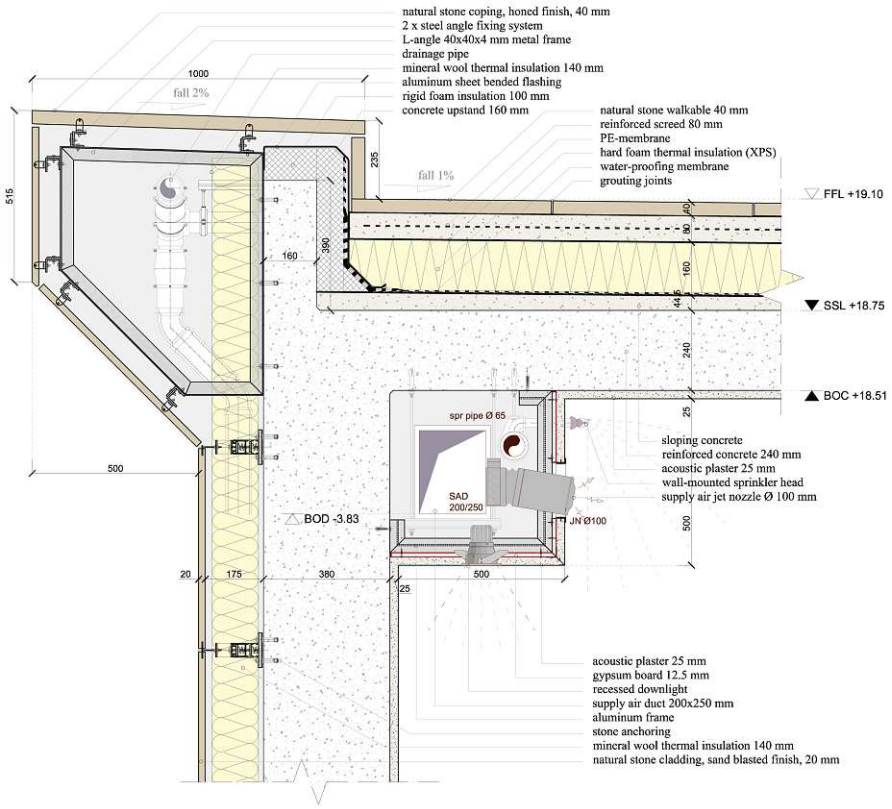


Figure 5.1: A typical 2d architectural detail, designated scale 1 : 5.

the building (see Figure 5.2). In order to produce a detail with a minimum of distortions, the standard procedure is to cut a place of interest (e.g., an edge where two surfaces,

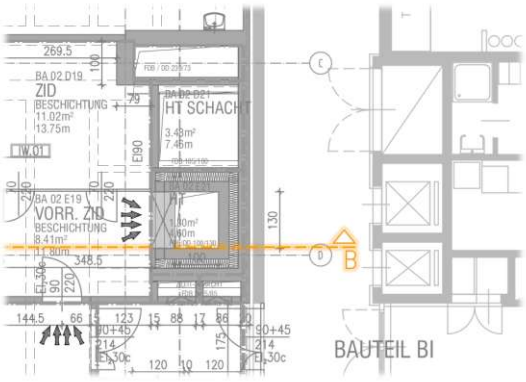


Figure 5.2: The mark of a detail definition site is highlighted in orange: in this case it is a vertical section perpendicular to an edge of interest and annotated as 'B'.

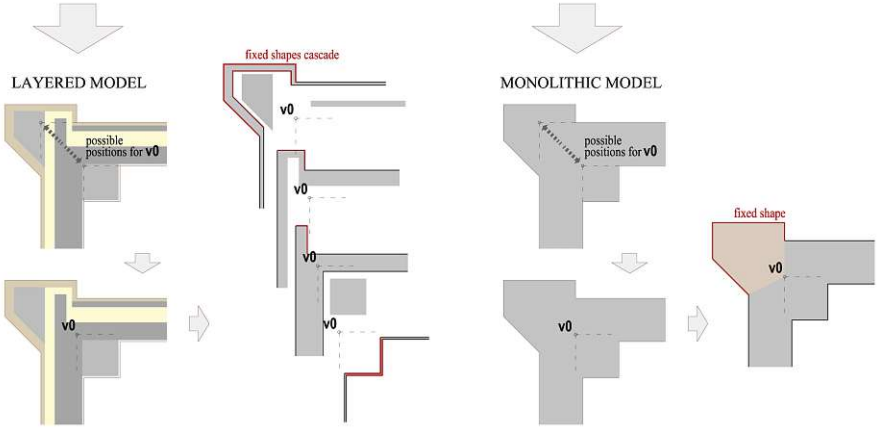


Figure 5.3: Translating a detail into a detail description. At this stage geometrical information is converted into semantic information, as shown in Fig. 5.2 and Tables 5.1, 5.2, and the participation of a human agent (e.g., the designer) is required.

such as a wall and the roof, meet) maintaining a chosen angle (usually 90°) to the feature we want to represent without distortion (e.g., the edge) and to define the thickness and profile of the finished structure in relation to the existing geometry (e.g., the two surfaces and the edge). In Figure 5.1 and Figure 5.3 the cut edge is represented by the vertex **v0** and the cut surfaces (or structural axes) - by the dotted lines intersecting in **v0**. The relationship between the detail description and the conceptual 3d model is subject to only two constraints: that the structural axes lie within the detail description and that each is aligned with at least one significant line contained in it (see Figure 5.4).

The material choice limits the minimal and maximal dimensions of any structure. For us the minimal dimensions are of particular interest since they introduce additional constraints into the detail descriptions. For example, load bearing armed concrete needs to maintain a certain minimal cover for its reinforcement as protection from weather, pollutants and other substances that, although benign to our skin (e.g., red wine or fruit juice), have an abrasive effect on cement and steel [RM10]. Thus any part of the detail with armed concrete as its designated material has to be at least as thick as a predefined minimum, for example, determined by an industry standard such as the European norm CEN - EN 13670 "Execution of concrete structures".

In the specialized literature only the typical detail solution (i.e., for two planes intersecting at right angles) is shown. The task of adapting the chosen typical solution to the geometry of the conceptual 3d model is assigned to the designer. In our work the same task is delegated to an expert-guided translation process that culminates in a detail library (see Figure 5.7).

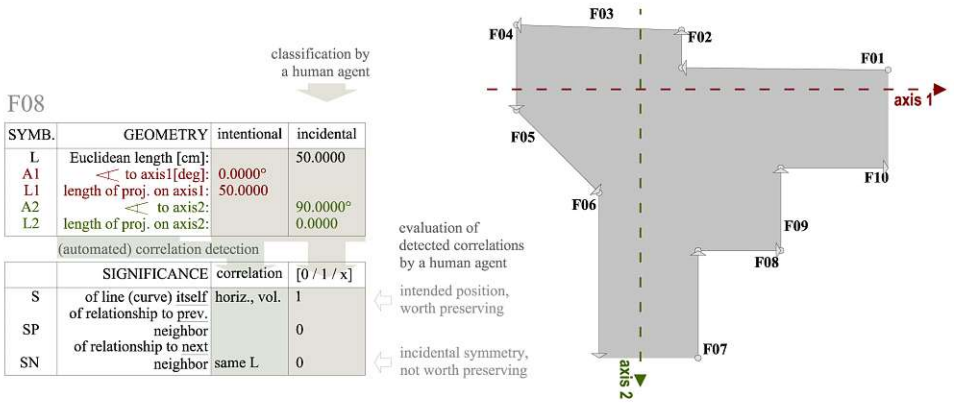


Figure 5.4: Attributes and dependencies for feature 'F08' in the monolithic model of the detail in Figure 5.1. The geometric attributes are objective. Their significance within the detail however is subjective, varies from one application to another, and can be reliably determined only by a human expert.

5.2 Translating Features of Interest into Geometric Relationships

In the translation process from a detail to a detail description ready for alignment with the conceptual 3d model, the first step involves the extraction of the relevant geometry - in the example in Figure 5.4 this is the directed contour of the detail (with each line segment **FN** knowing its previous and its next neighbour). The next step analyses the geometry of the contour and, similar to [Sha08], searches for the following relationships: symmetry (elements of equal length, forming the same angle with a structural axis and at the same distance from it), and elements parallel or perpendicular to a structural axis. These geometric relationships, when declared as significant by the designer, have the purpose of replacing objective metrics and thus circumventing drawing inaccuracies.

We are only interested in extracting attributes and relationships that remain invariant throughout the lifetime of an element, similar to [FCSF17]. Since some of the steps in the *procedural contraction* algorithm involve parallel projection, distances and angles are not preserved. What is preserved, however, are parallel elements, right angles to elements parallel to the projection direction, lengths of elements at right angles to the projection direction, and ratios.

Therefore, in the next step, the geometric attributes for each line segment are noted - its Euclidean length, and its angle to and projected length onto each of the structural axes (see Table 5.1). The fourth and most crucial step involves a categorisation of the attributes by a human agent (e.g., the designer) - into intentional and incidental (see Table 5.2). The second Table also includes the human agent's assessment of the significance of the observed geometric relationship of the line segment with its neighbours

- length ratio and angle. In Figure 5.4 the length equality of **F08** and its next neighbour, for example, are categorised as incidental - hence not worth preserving.

Table 5.1 shows the geometric attributes and their categorisation for all segments of the contour in Figure 5.4. The angle between **F06** and **axis 2**, for example, is not 90° - however, the human agent has described as 'vertical' (indicated as | in the table). Another example for the disparity between objective measurements and user assessment is the deviation of the roof and parapet surfaces depicted in Figure 5.1 from the horizontal **axis 1** (segments **F01** and **F03**). This is a significant (and even annotated) part of the detail as these surfaces are designed to direct the flow of water. In spite of this, the effect of the deviation on the overall design is negligible because the incline doesn't run in the same direction over long distances. Every few meters there is a gully where the incline reverses direction. Thus **F01** and **F03** are assumed parallel to **axis 1** (indicated as –). In all such cases the user intent takes precedence over objective geometric attributes. Some segments (**F03**, **F04**, **F05**, **F08** and **F09**) are described as determining the enclosed volume (indicated as 'vol') - i.e., of particular importance for functional reasons (in Figure 5.1 they enclose the space for the drainage pipe and for the interior bulkhead).

Geometric Properties										
L	Euclidean length									
Ai	angle to Axis i									
Li	projected length on Axis i									
	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10
L	125.01	22.82	100.05	51.56	70.71	100.00	65.00	50.00	50.00	65.00
A1	70.57°	90.00°	1.78°	90.00°	45.00°	0.01°	90.00°	0.00°	90.00°	0.01°
L1	125.00	0.00	100.00	0.00	50.00	50.00	0.00	50.00	0.00	65.01
A2	89.43°	0.00°	88.22°	0.00°	45.00°	89.01°	0.00°	90.00°	0.00°	89.01°
L2	1.25	22.82	3.11	51.56	50.00	100.00	65.00	0.00	50.00	0.01
Classification by a Human Expert										
S	significant characteristics									
SP	significant relationship to previous									
SN	significant relationship to next									
	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10
S	–	|	– vol	| vol	vol	|	|	– vol	| vol	–
SP	x	higher	higher	lower	angle	none	x	none	same L	none
SN	lower	lower	higher	angle	none	x	none	same L	none	x

Table 5.1: The geometric information and significance characteristics of the ten line segments comprising the monolithic model in Figure 5.3 and Figure 5.4.

Table 5.2 shows the invariant (under parallel projection) geometric relationships extracted from Table 5.1 that, together with the contour, become part of the detail description. Each of the structural axes has a *parallel set* - a set of line segments within the contour

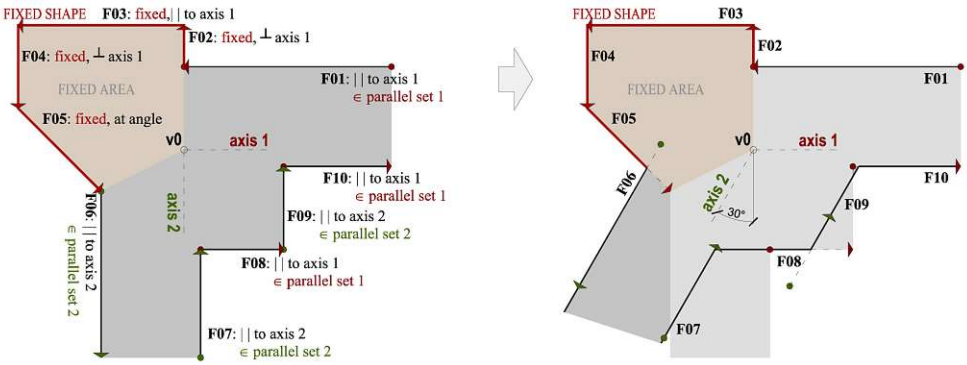


Figure 5.5: Changing the angle between the structural axes has no influence on the semantic relationships established in the previous steps (see Table 5.2).

that are parallel to it; each of those line segments has a fixed distance to the axis. In addition, there is a group of line segments (from **F02** to **F05**) that belong to a *fixed shape* relative to **axis 1** - those are the most important, i.e. most preservation-worthy, geometric attributes (position and angle to the relevant axis) of the detail and retain their high priority in all subsequent stages (those are the red line segments in Figure 5.5).

5.3 From a Detail Description to a Template

The extracted geometric relationships place each of the line segments contained in the detail description into one of three groups with regard to further manipulation. The first one contains all elements in a parallel set (e.g., **F01**, **F06** to **F10** in Figure 5.5). These are the elements easiest to manipulate or discard. The second one contains elements at arbitrary angles to both structural axes. They are the ones whose manipulation influences their neighbors the most (if **F05** was not part of a fixed shape, it would belong in this group). The third group contains the *fixed shapes*. They are the detail elements of greatest importance, which we want to preserve for as long as possible. They are fixed in

	Invariants									
	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10
parallel set 1	•		•					•		•
dist. to Axis 1	50		75					-60		-10
parallel set 2						•	•		•	
dist. to Axis 2						50	-10		60	
fixed shape		•	•	•	•					
pos. to Axis		⊥,d=50	∥,d=75	⊥,d=25	∠,d=-25					

Table 5.2: The invariants extracted from Table 5.1. Changes of the angle between the structural axes have no influence on them (see Figure 5.5).

position both to one of the structural axes (**axis 1** in this case) and to their intersection point **v0** and are handled as one unit for as long as possible. In this Section we prepare the detail description for alignment with the conceptual 3d model by creating templates with instances for axes angles 30°, 60°, 120°, and 150°, and a procedure for interpolation between template instances.

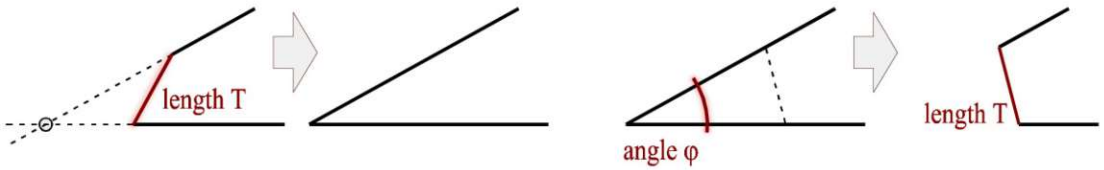


Figure 5.6: Dealing with non-buildable geometry: on the left - the highlighted segment is too short, on the right - the highlighted angle is too small.

Figure 5.5 demonstrates a change in angle between the structural axes. Whereas the left diagram shows the standard angle of 90° between the axes, the right one shows how we arrive at an angle of 120°. Such variations are needed in case we apply the *detail* to an edge of the *3d building skeleton* where the surfaces meet at an angle other than 90°. Notice that the fixed shape retains both its position to **v0** as well as its angle to **axis 1**, to which it is fixed. The change is performed by rotating the second parallel set defined in Table 5.2 along with its axis around **v0**. This results in a change of the Euclidean length **L** of some line segments, once we have intersected the rotated line segments with their neighbors to restore the adjacency relationship in the 90° template instance. The projected length onto the axis, to whose parallel set the line segment belongs, might change; however, the distance of the line segment to its axis does not. The significance of this will become clear when we define a line segment's *preservation score* in Chapter 8. The line segments possessing only one neighbor (e.g., **F01**, **F06**, **F07** and **F10**) can be lengthened or shortened at their free end until their projected length matches the one before the rotation. This is done in order to avoid too short line segments incompatible with the minimal segment length we will set for our model.

As we noted in the beginning of this Chapter, we want to arrive at a *buildable 3d model*. In geometric terms this means avoiding too small line segments and too sharp angles. In order to achieve this, we need to define thresholds and procedures for adapting the detail description once such a threshold has been reached. Figure 5.6 illustrates this idea. The left-hand diagram shows the case when the length of a line segment reaches the threshold **T**. If its two immediate neighbors are not parallel, we can intersect them and eliminate the too short segment. The case when they are parallel involves more than just the three segments and we will demonstrate the procedure involved when we encounter the problem in a specific context (see Chapter 11).

As Figure 5.6 shows, we could easily be trapped in an endless loop of eliminating too short segments only to produce angles that are too sharp and need to be bevelled, thereby producing segments that are too short. As the elimination of sharp angles is performed

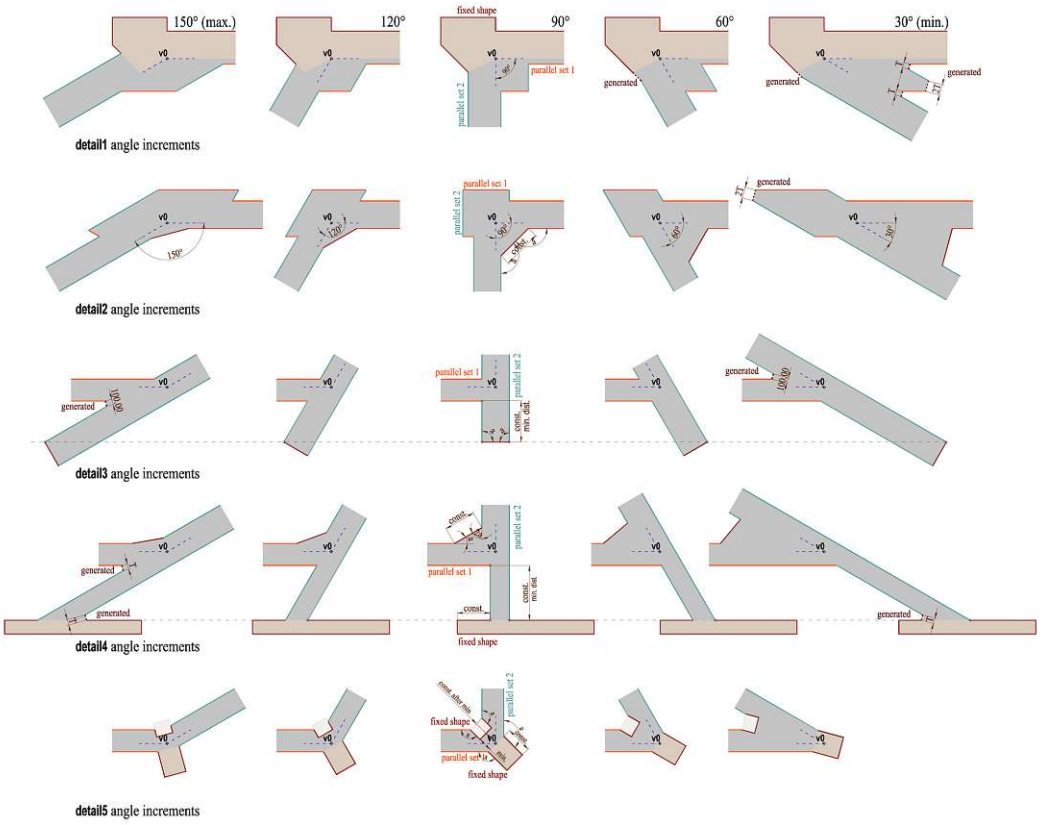
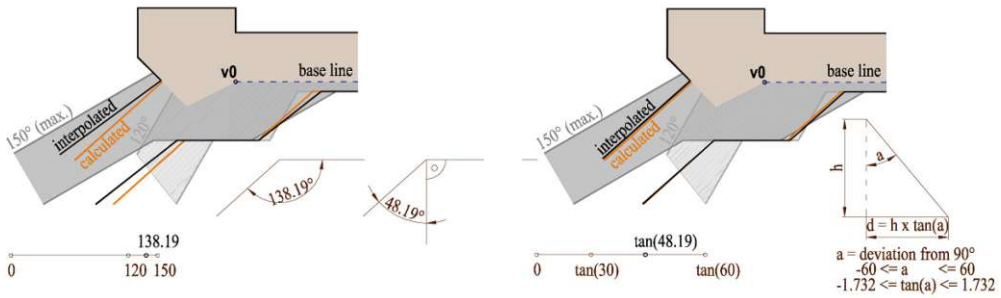


Figure 5.7: The middle column depicts the default instance of each detail template - the structural axes are at a 90° angle. The angles 150°, 120°, 90°, 60°, and 30° were constructed. All others instances are calculated as interpolations of these.



(a) (b)

Figure 5.8: Interpolation between instances of the template for Detail 1 (see Figure 5.7) to obtain the detail instance for 138.19° (in a regular pentagon). Based on (a) the angle: inaccurate; (b) the tangens function of the angle: sufficiently accurate.

by chamfering the two neighbors (i.e., generating a new segment of minimal length **T** between them that forms the same angle with both), it looks like replacing one problem with another. The mechanism for avoiding this will be presented in Chapter 8 and Chapter 9 when we discuss *procedural contraction* for *features* (1d line segments with rich descriptors).

Figure 5.7 shows a small template library of monolithic models of five details with the instances with axes angles of 30°, 60°, 120°, and 150° constructed as shown in Figure 5.5. The results of the procedure for elimination of sharp angles can be seen in the left-most and right-most columns. Figure 5.8 illustrates the procedure for interpolation between templates when the angle between the structural axes deviates from the ones saved in the template. Alternatively, instead of using templates with pre-constructed instances for several fixed angles and an interpolation procedure, we could use the rotation algorithm presented in Figure 5.5 to produce template instances for arbitrary angles on demand.

Now we have a *library* of detail descriptions relative to a pair of *structural axes*, consisting of one or more sequences of line segments attributed with adjacency and geometric relationships. In the subsequent Chapters we will translate these detail descriptions to feature collections aligned with the 3d building skeleton extracted from the conceptual 3d model.

At this stage we can already provide the answer to research question **RQ1**: *Which geometric information has to be extracted from a detail, so that it can be consistently applied across the entire conceptual 3d model?*

ARQ1: *We are interested only in the connected contour of the detail as a whole, or the contours of each of its material layers, and in the contour's position and orientation relative to the structural axes of the conceptual 3d model. The geometric properties of the contour that should be preserved when interacting with its neighbor details need to be confirmed by an expert. They include line segments parallel to a structural axis, line segments at an angle of functional significance to a structural axis, and line segments that delimit spaces of functional significance. Furthermore, there should be a set of transformation rules for adapting the extracted contours to the angle between the structural axes of the conceptual 3d model.*

CHAPTER 6

Preparing the 3d Building Skeleton and Detail Alignment

In this Chapter we address the data selection process we outlined in Section 1.4, this time focused on the conceptual 3d model. In addition, we provide the theoretical underpinnings for the answer to research question **RQ2**: *What conditions have to be fulfilled by the conceptual 3d model, or by the result of its pre-processing, so that the detail information can be applied to it correctly in the context of the architectural domain?*, which we will give in full in Chapter 10. Finally, we present result **R2**: *A modeling guide for the conceptual 3d model to facilitate the extraction of salient features, such as edges and corners*, which we listed in Section 1.3.

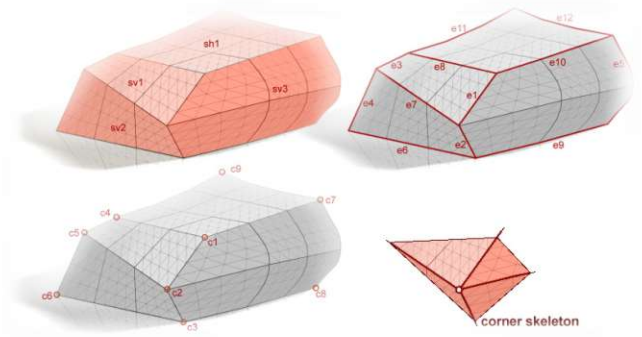


Figure 6.1: Extracting the 3d building skeleton from the conceptual 3d model.

As we outlined in Chapter 4, the *3d building skeleton* is the structure of the *conceptual 3d model* - the tangent planes of the surfaces intersecting at the edges, the edge approximations as polylines, and the corner points (see Figure 6.1). As we showed in Section 5.1, *details* are defined within sections orthogonal to their respective edges. In Section 5.3,

we discussed how a *detail description* derived from such a detail is adapted to an edge with an arbitrary angle between the structural axes. The most critical parts of the 3d building skeleton are the corners where several edges, each of which can require a different detail, intersect. In Chapter 4, we outlined a method for extracting the 3d building skeleton form the conceptual 3d model with particular emphasis on the corners. The bottom right image in Figure 6.1 displays a *corner skeleton* - the corner point and a linear approximation of each of the edges intersecting in it. As the extraction of the corner skeleton is not the focus of this work, we will omit further details here and refer instead to the relevant sections in the prototype presentation in Chapter 9. In this Chapter we will discuss the method for aligning the detail descriptions with their respective edges in the corner skeleton. Based on this alignment, we will build edge compositions, segment compositions and feature collections. In the subsequent Chapters we define the rules for feature interaction within feature collections in order to create a *corner 2-manifold with boundary* (see Figure 4.1d).

6.1 Properties of the Corner Skeleton

The corner skeleton is fully described by a list of $(n+1)$ vertices representing the top of the corner (i.e., the intersection of all edges - T in Figure 6.2) and a distinct vertex along each of its n edges, $n \geq 3$, respectively. We assume that we are so close to the corner of the 3d building skeleton that the representation of the edges as straight line segments is sufficiently accurate (see Chapter 4). For the purpose of the algorithm presented in Chapter 8 each of the n vertices along the edges of the corner skeleton has to be at the same distance from the top T.

6.2 Aligning the Detail Description to the Corner Skeleton

The first step in aligning the detail description derived from the detail as described in Section 5.3 is to determine the angle θ between the structural axes (line segments LQ_h and MQ_h at edge TQ in Figure 6.2 and Figure 6.3). The general rule for calculating it is the following (notation as in Figure 6.2):

$$\Delta LMT : d^2 = \frac{1}{\cos^2\lambda} + \frac{1}{\cos^2\mu} - \frac{2\cos\varphi}{\cos\lambda\cos\mu}, \tag{6.1}$$

where $\lambda, \mu, \eta, \sigma$ are the angles between each two neighbouring edges at the corner, and φ is the angle opposite the diagonal d: $\measuredangle LTM$ in triangle ΔLMT

$$\Delta LMQ_h : d^2 = \tan^2\lambda + \tan^2\mu - 2\tan\lambda\tan\mu\cos\theta$$

$$d^2 = \frac{\sin^2\lambda}{\cos^2\lambda} + \frac{\sin^2\mu}{\cos^2\mu} - 2\frac{\sin\lambda\sin\mu}{\cos\lambda\cos\mu}\cos\theta \tag{6.2}$$

From equations 6.2 and 6.2 follows:

$$\frac{1}{\cos^2\lambda} + \frac{1}{\cos^2\mu} - \frac{2\cos\varphi}{\cos\lambda\cos\mu} = \frac{\sin^2\lambda}{\cos^2\lambda} + \frac{\sin^2\mu}{\cos^2\mu} - 2\frac{\sin\lambda\sin\mu}{\cos\lambda\cos\mu}\cos\theta$$

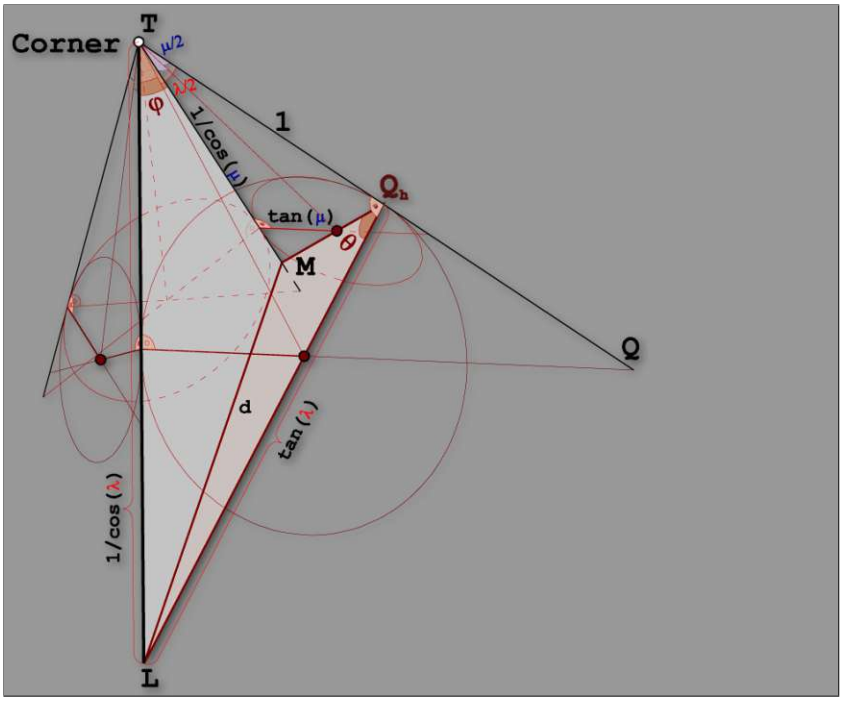


Figure 6.2: A corner skeleton: calculating the structural axes angle θ before aligning the detail description with an edge to from a part of the edge composition.

$$\frac{1-\sin^2\lambda}{\cos^2\lambda}+\frac{1-\sin^2\mu}{\cos^2\mu}-\frac{2\cos\varphi}{\cos\lambda\cos\mu}=-2\frac{\sin\lambda\sin\mu}{\cos\lambda\cos\mu}\cos\theta$$

$$2-\frac{2\cos\varphi}{\cos\lambda\cos\mu}=-2\frac{\sin\lambda\sin\mu}{\cos\lambda\cos\mu}\cos\theta$$

$$\cos\lambda\cos\mu-\cos\varphi=-\sin\lambda\sin\mu\cos\theta$$

$$\boxed{\cos\theta=\frac{\cos\varphi-\cos\lambda\cos\mu}{\sin\lambda\sin\mu}} \tag{6.3}$$

The next step is transferring the adapted *detail description* (i.e., the template instance of angle θ) to the plane LQ_hM perpendicular to the edge TQ (see the red plane in Figure 6.4 and 6.5). The exact procedure is described in Section 9.2. Here, we will only point out that the *structural axes* of the detail description both lie in their respective corner faces LQT and MQT. The collection of all detail descriptions transferred to their respective edge of the corner skeleton is the *edge composition*.

The disadvantage of the edge composition is that it introduces joints between the transferred edge-based detail description within the faces of the corner skeleton, that are, in essence, artificial edges. Figure 6.6 shows the placement of the edge composition planes in red for three different scenarios - a cube corner, a regular pentagon pyramid corner and an arbitrary corner. The joint between the detail description of edge A and

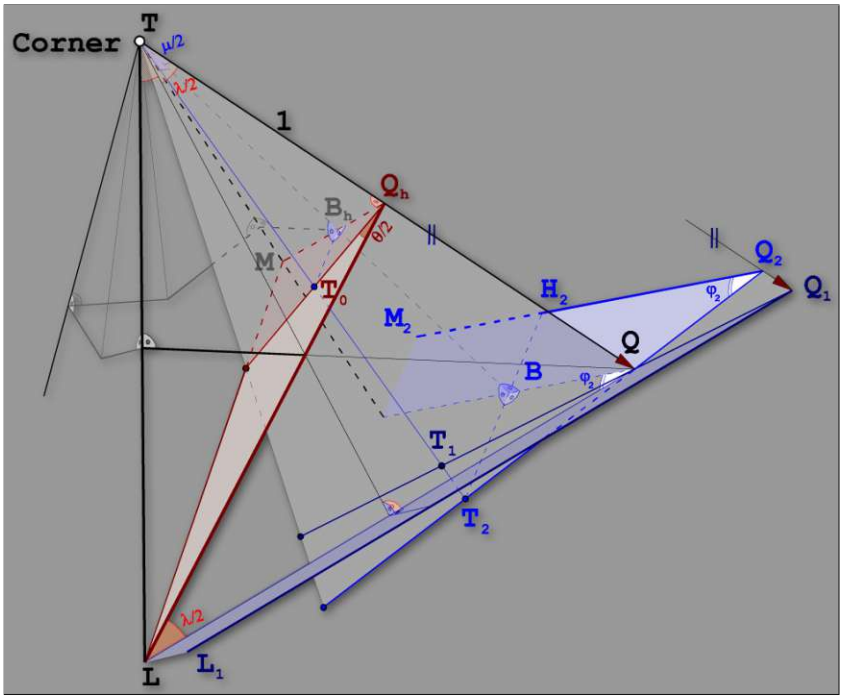


Figure 6.3: Calculating the segment angles φ_1 and φ_2 before projecting the detail description along an edge TQ as part of the segment composition.

edge B, for instance, is marked in red as $jointAB$. In order to move the joints back to a more natural position - the edges - we transform the edge composition into a *segment composition* by projecting each edge-based detail description along its respective edge onto the two planes normal to the corner faces, incident with that edge, and intersecting the edge and its respective neighbor at the same distance from the top (planes QQ_1M_1 and QQ_2L_2 in Figure 6.3). The result can be seen in Figure 6.4 for a square equilateral pyramid corner and in Figure 6.5 for an arbitrary quadrilateral pyramid corner.

We can determine the angle φ (e.g., φ_2 in Figure 6.3) of the segment at each edge joint by performing the projection, or by using the following calculation:

$$\Delta TB_hQ_h : \|B_hQ_h\| = \tan(\mu/2), \|TB_h\| = \frac{1}{\cos(\mu/2)}$$

$$\Delta T_0B_hQ_h : \|T_0B_h\| = \tan(\mu/2)\tan(\theta/2)$$

$$\Delta TLQ_h : \|TL\| = \frac{1}{\cos\lambda} \Longrightarrow \|TQ\| = \frac{1}{\cos\lambda}$$

$$\Delta TQB : \|TB\| = \frac{\cos(\mu/2)}{\cos\lambda}, \|BQ\| = \frac{\sin(\mu/2)}{\cos\lambda}$$

$$\Delta TT_0B_h, \Delta TT_2B : \frac{\|T_0B_h\|}{\|T_2B\|} = \frac{\|TB_h\|}{\|TB\|}$$

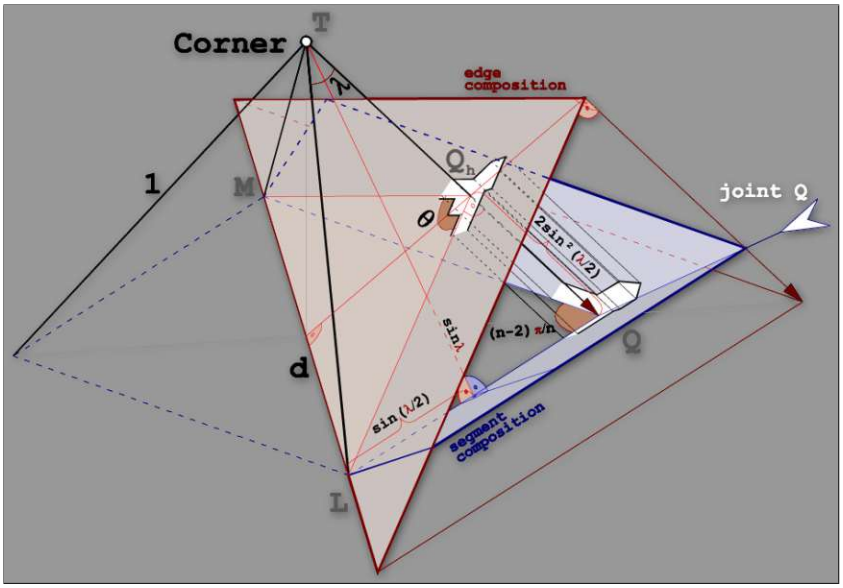


Figure 6.4: Converting an edge composition (in the red plane) into a segment composition (in the blue planes) through projection parallel to the edge TQ. The red plane is perpendicular to the edge TQ, the blue planes are perpendicular the faces TQL and TQM. In this case the corner skeleton is a square equilateral pyramid.

$$\frac{\tan(\mu/2)\tan(\theta/2)}{\|T_2B\|} = \frac{\cos\lambda}{\cos^2(\mu/2)} \Longrightarrow \|T_2B\| = \frac{\tan(\theta/2)\sin(\mu/2)\cos(\mu/2)}{\cos\lambda}$$

$$\Delta T_2QB : \tan\varphi_2 = \frac{\|T_2B\|}{\|BQ\|} \Longrightarrow \tan\varphi_2 = \frac{\tan(\theta/2)\sin(\mu/2)\cos(\mu/2)}{\cos\lambda}\frac{\cos\lambda}{\sin(\mu/2)}$$

$$\boxed{\tan\varphi_2 = \tan(\theta/2)\cos(\mu/2)} \tag{6.4}$$

Figure 6.6 shows the placement of the segment composition planes in blue for three different scenarios. The change in the Euclidean length of line segments from the edge-based to the face-based detail descriptions is shown in Figure 6.5: all vertical distances to the structural axes remain unchanged (marked in white as b), all horizontal distances to the joint(marked in white as a and $a/\cos(\lambda/2)$) are divided by $\cos(\lambda/2)$ where λ is the angle at the top of the corner skeleton in the relevant corner face.

6.3 Building the Segment Composition

After the alignment of the detail description to the segment composition planes, we need to also align the free ends of the line segments possessing only one neighbor in both detail descriptions in order to form a contiguous *segment composition* (see Figure 6.7). The segment composition consists of two polylines (here marked as black and white),

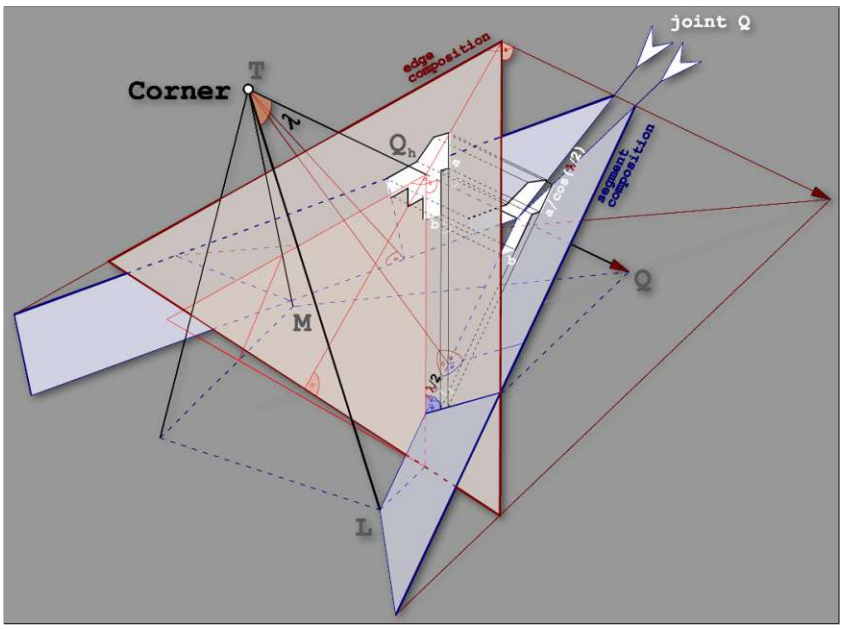
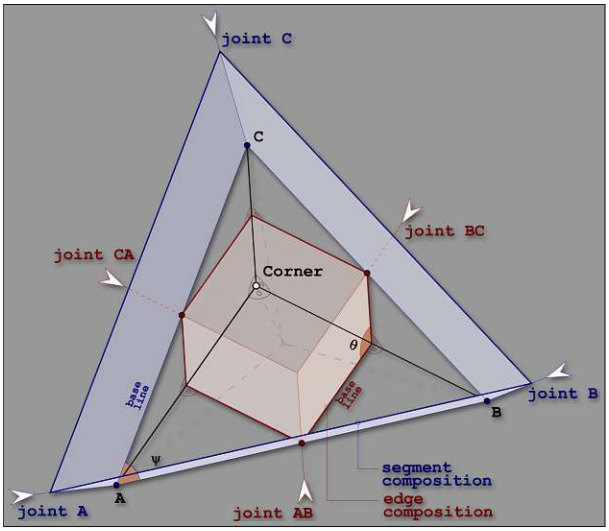


Figure 6.5: Converting an edge composition (in the red plane) into a segment composition (in the blue planes) through projection parallel to the edge TQ. The red plane is perpendicular to the edge TQ, the blue planes are perpendicular the faces TQL and TQM, respectively. In this case the corner skeleton is an arbitrary quadrilateral pyramid.

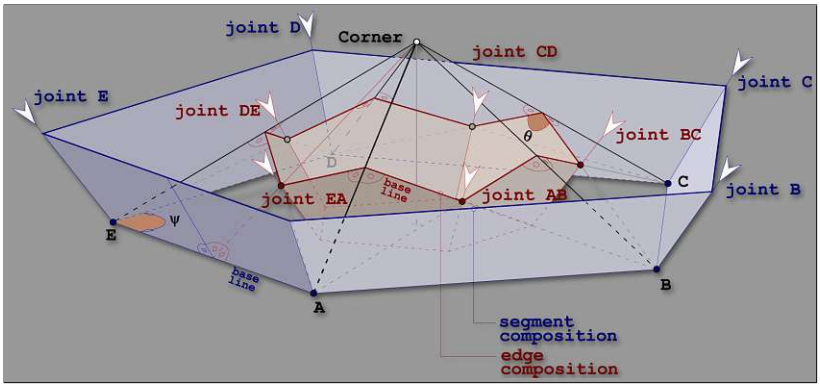
each starting at the joint of the corner skeleton to which the respective detail description was applied. From now on we will refer to this edge as the *owner* of the polyline. The segment of the structural axes that lies in the segment plane is denoted as *base line*. In the subsequent Chapter we build a ruled surface over each face of the corner skeleton by shortening this line in each segment, according to pre-defined rules, and shifting it towards the top of the corner skeleton until it reaches length zero.

Since the details under discussion in this work concern the edges of the building and not the surfaces between them, the last segment of each polyline is typically parallel to the structural axis, or base line, (cases (a) and (b) in Figure 6.7). An example of a deviation from this it shown in cases (c) and (d), where the last segment of polyline A could intersect the base line if extended beyond the limit of the detail description (the hatched area), which would result in a non-manifold, hence not buildable, corner.

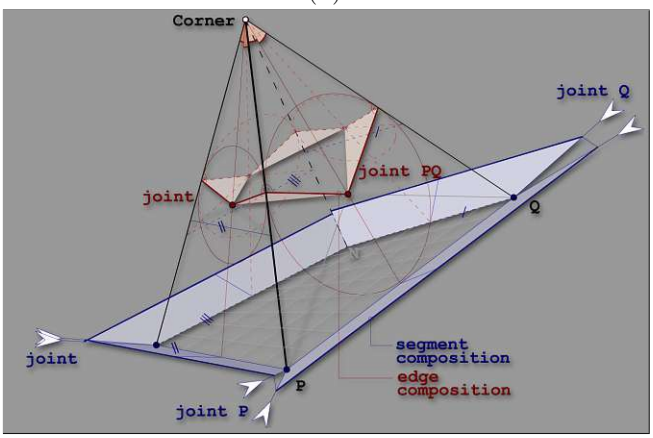
The goal is to leave the surfaces between the edges (marked as $jointA$ and $jointB$ in Figure 6.7) free from modifications to enable a possible subsequent design step concentrating on surface fine-tuning. Thus, when the difference in the distance to the base line is under a certain pre-defined threshold value T we can remove the step as shown in Figure 6.7(a) by adopting an uniform distance to the base line in both polyline segments. This threshold stems from the requirement discussed in Section 5.3 that the modified polyline cannot have segments with length less than a certain material-dependent minimum. However,



(a)



(b)



(c)

Figure 6.6: Edge composition planes (red) and segment composition planes (blue) in the skeleton of (a) a cube corner, (b) a regular pentagon corner, (c) an irregular corner.

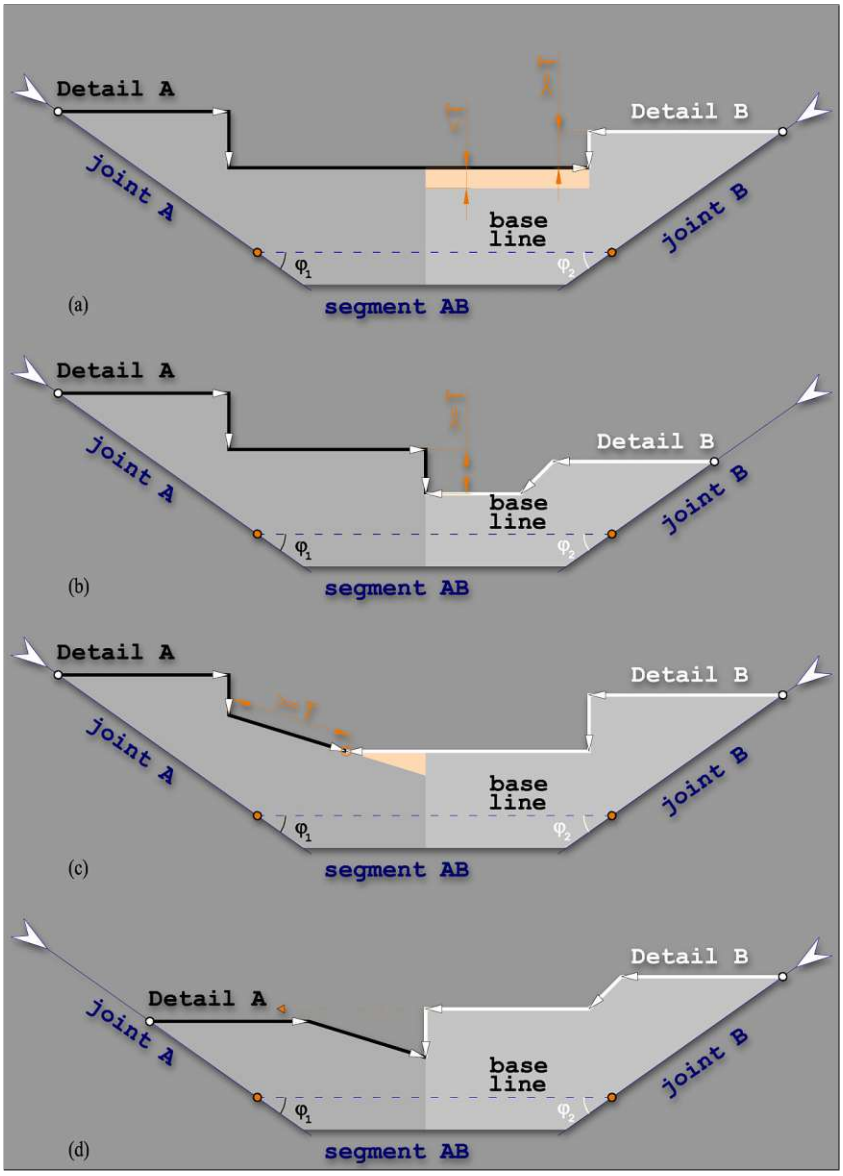


Figure 6.7: Building the segment composition from the aligned detail descriptions.

when the difference is larger than T or the resulting polylines would contain segments shorter than T, a step cannot be avoided (Figure 6.7(b) and (d)). The additional polyline segment is appended to the polyline whose last segment is at a larger distance from the base line.

The polylines with designated *owner* edges in the resulting contiguous segment composition now receive the *geometric relationship* information derived from their respective details in Section 5.2 and Section 5.3 and convert each line segment into a 1d *feature* with a *rich descriptor*. It contains the owner edge as well as geometric characteristics: whether

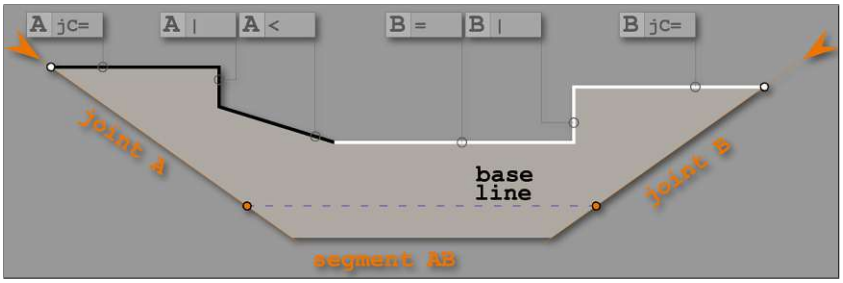


Figure 6.8: Building the feature collection for the segment shown in Figure 6.7(c) from the polylines with designated *owner* edges and *topological descriptors*.

the polyline segment is horizontal (=), vertical (|), or at an angle $\alpha, 0 < |\alpha| < \pi/2$ (<) relative to the *base line*. It also contains relationships: whether the polyline segment continues in the neighboring segment (**jC** - joint continuity), whether it is a part of a fixed shape (**f**), or whether a pair of segments are in a parent-child relationship and need to be activated together (**p** and **k**). The rich descriptor has states and can transition from one to another during the *procedural contraction*: **g** for procedurally generated polyline segments, **a** for currently active segments that need to remain active in the next step. All of these characteristics will be presented in detail in the next Chapter.

The set of features with rich descriptors together with the base line and joints build the *feature collection* for the segment. Figure 6.8 shows the conversion from a contiguous *segment composition* in Figure 6.7(c) to a feature collection.

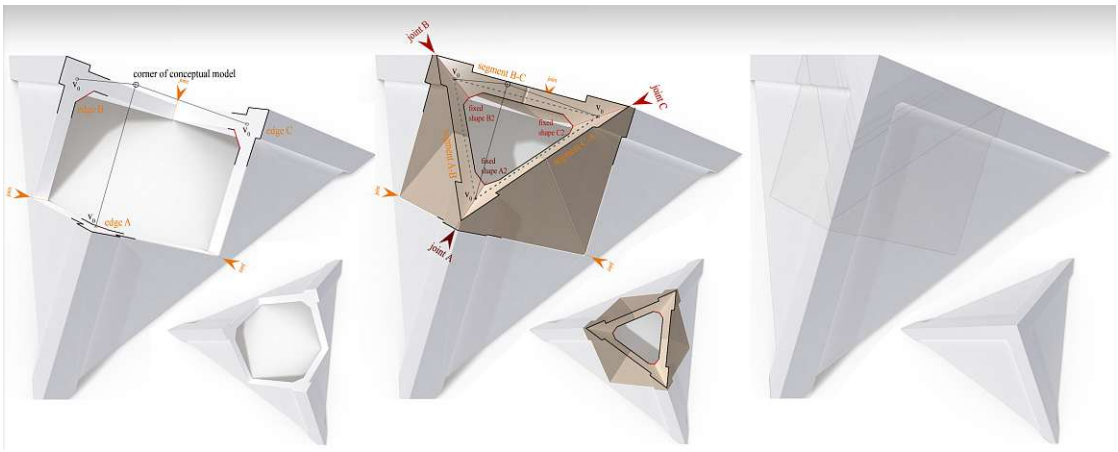


Figure 6.9: Edge composition, feature collection and corner 2-manifold with boundary for a cube corner with detail 2 applied to each edge (see also Section 5.3).

The sequence of transitions from an edge composition to a feature collection and resulting corner 2-manifold with boundary is shown in Figure 6.9 for a cube corner with detail 2 from the Detail Library in Section 5.3 applied to each of its edges. Figure 6.10 shows the same for a cube corner with details 1, 2 and 5 applied to its edges and Figure 6.11 shows

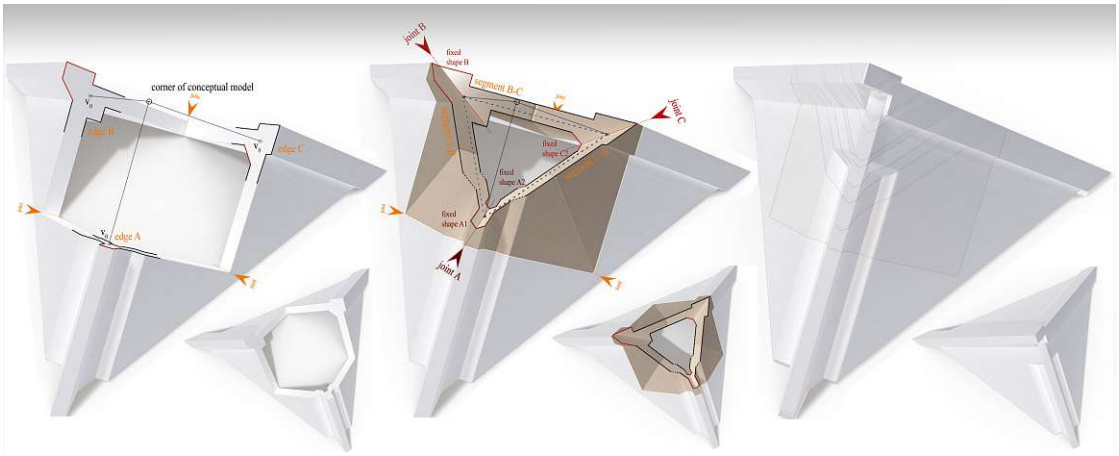


Figure 6.10: Edge composition, feature collection and corner 2-manifold with boundary for a cube corner with details 1, 2 and 5 applied to each edge (see also Section 5.3).

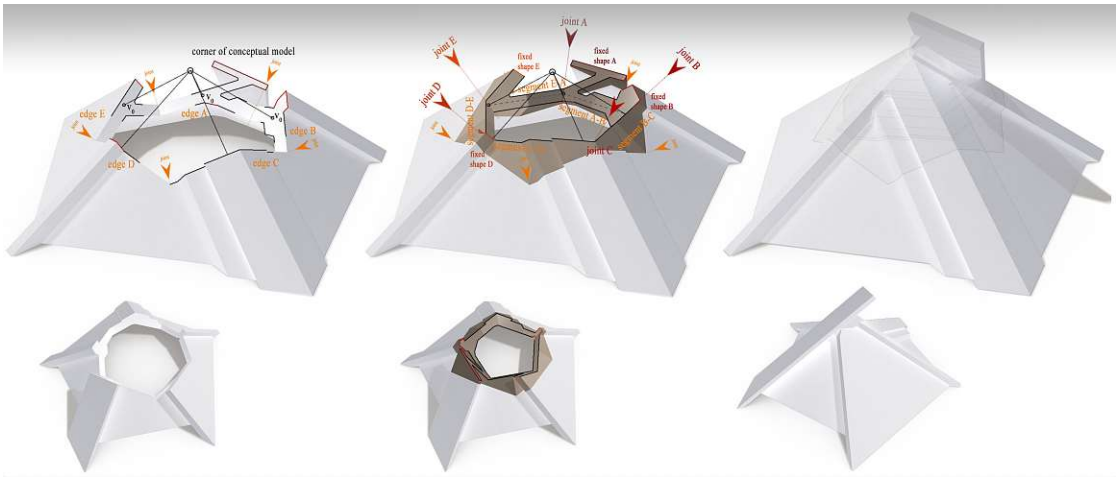


Figure 6.11: Edge composition, feature collection and corner 2-manifold with boundary for a regular pentagon pyramid corner with details 1 to 5, applied to each edge (see also Section 5.3).

it for a regular pentagon pyramid corner with details 1 to 5 applied to its edges.

Before we proceed with the characteristics of the feature collection in the next Chapter, we can deliver the answer to research question **RQ2**: *What conditions have to be fulfilled by the conceptual 3d model, or by the result of its pre-processing, so that the detail information can be applied to it correctly in the context of the architectural domain?*

ARQ2: *For the purposes of this algorithm, the conceptual 3d model has to have edges and corners identifiable by one of the methods we presented in Section 3.4. This excludes surfaces with too little variance in their curvature, e.g. an ellipsoid, or surfaces with too large variance in their curvature, e.g. a porcupine. Furthermore, the edges of the model*

should be able to be approximated by straight lines at a distance to the corner allowing the application of details and there should be no less than three edges intersecting at each corner.

This Chapter can be regarded as result **R2**: *A modeling guide for the conceptual 3d model to facilitate the extraction of salient features, such as edges and corners.*

CHAPTER 7

The Feature Collection

This Chapter and Chapter 8 provide the answer to research question **RQ3**: *After we apply the corresponding details to adjacent model edges, the algorithm produces the buildable 3d model, the so-called as-designed model. Form the point of view of the AEC industries, what degree of feasibility has this model?* We demonstrate that the feasibility of the model is given by construction, i.e., the rules we outline for the assembly and contraction of the feature collection guarantee that the result is a buildable 2-manifold with boundary.

In the previous Chapter we built the feature collection of a corner skeleton face relative to a base line. Here we will present the formal definitions of base line and of feature:

Definition 7.1 The *base line* represents the structural axis of a feature collection. It is obtained by intersecting the face of the corner skeleton with a plane normal to it and intersecting the face's adjacent edges at equal distances from the top of the corner (see the blue dashed line in Figure 7.1). It is limited at both ends by the two joint lines that represent the common boundary with the neighbour base line (see $jointA$ and $jointB$ in Figure 7.1).

Definition 7.2 A *feature* is a 1d line segment of a polyline and has the following attributes: Its local coordinate system has its origin on the base line, its x-axis is collinear with it, and its y-axis points outward. It is owned by exactly one joint. It has a predecessor neighbour feature and a successor neighbour feature, part of the same polyline, both of which can be null. It has a rich descriptor containing geometric relationship information (see key in Section 7.1 on the right), a contraction length (always measured parallel to the base line), and a restriction on its contraction length due to the potentials (see Section 7.2) of the feature collection. It also has a preservation score.

The goal of the *procedural contraction* is to reduce the number of features in the feature collection with each consecutive step. This results in the feature collection advancing towards the top of the corner skeleton, while spanning a ruled surface between its current state and its state from the previous step. When contraction is no longer possible, the ruled surfaces for each face of the corner skeleton are stitched together to produce a *buildable corner 2-manifold with boundary*. This strategy is similar to the one demonstrated in [OS88] concerning the propagation of fronts governed by equations of motion, which are subject to topological breaking and merging.

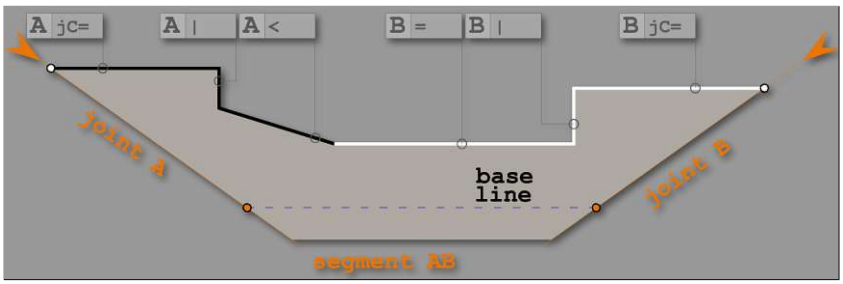


Figure 7.1: Building the feature collection for the segment shown in Figure 6.7(c) from the polylines with designated *owner* edges and *topological descriptors*.

In the subsequent Sections we will be discussing the segment polygon, for which we present the following:

Definition 7.3 The boundary of the area enclosed by the feature collection, the base line and the joint lines is the *segment polygon*.

7.1 Feature Descriptors

The *feature collection* has the advantage of being purely two-dimensional. In this way we can construct the corner manifold by performing transformations only in one plane. We describe these transformations as *contraction*: shortening or contraction of the active feature parallel to the base line leads to shortening of the base line along with it. Our goal is to contract the base line down to a point. The feature descriptors carry the information necessary for the selection of the next feature to contract.

7.1.1 Contraction Length

For a feature with *topological descriptor* $=$ (i.e., horizontal relative to the base line) the *contraction length* is its Euclidean length. For a feature with topological descriptor $<$ (i.e., slanted or at anlge $\alpha, 0 < |\alpha| < \pi/2$ to the base line) or $|$ (i.e., vertical relative to the base line) the contraction length is calculated as follows: if there is a line parallel to the base line through the higher end point of the feature, that intersects another feature without crossing the inside of the *segment polygon*, or a line parallel to the base

line through the lower end point of the feature that intersects another feature without crossing the outside of the *segment polygon*, the length of this line is considered the contraction length of the feature (see feature **a** in Figure 7.2); if there is no such line, the feature has infinite contraction length (see feature **b** in Figure 7.2). Features with infinite contraction length can be eliminated only as a side effect of the elimination of other features or potentials (see Section 7.2).

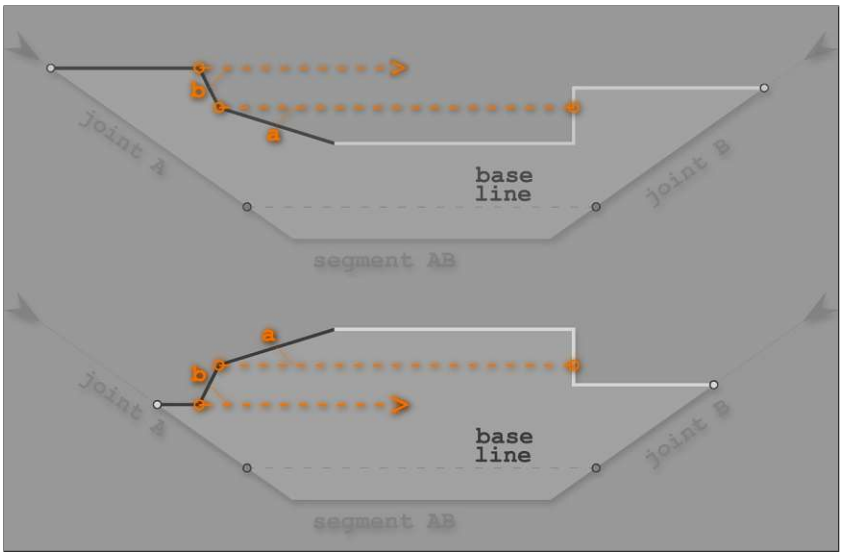
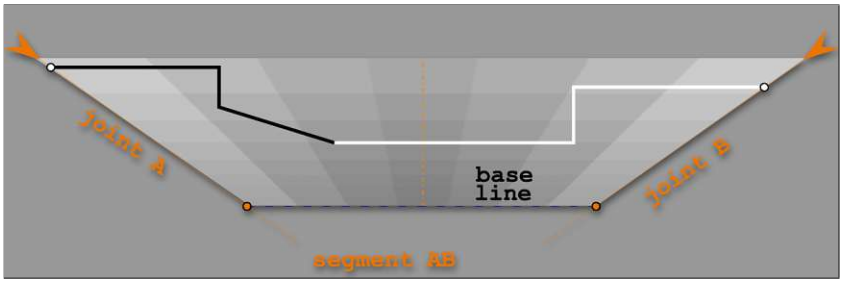


Figure 7.2: The contraction length of a feature with descriptor <. Feature **a** has a finite contraction length; feature **b**, on the other hand, has infinite contraction length.

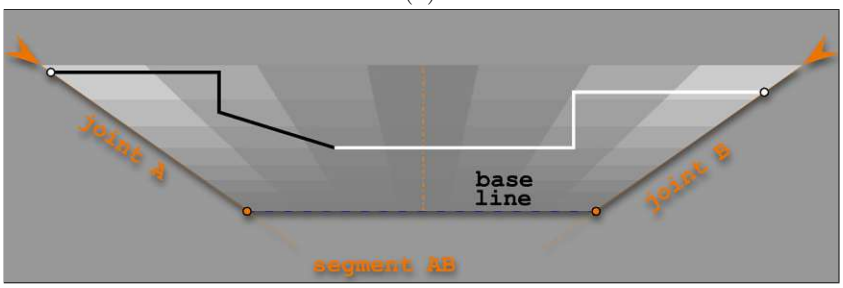
7.1.2 Preservation Score

The process of contracting a feature ends with its reduction to a point, i.e. its elimination. In this way, in each contraction step we lose detail information. The *preservation score* tells us how preservation-worthy each feature is. It is calculated based on a *score field*. Figure 7.3 shows a discrete 7×8 score field. The score is highest in areas closest to the joints and furthest from the base line as these areas have the highest influence on the appearance of the edges. Areas closest to the base line and closest to the symmetry axis of the segment have the highest influence on the appearance of the surface connecting the edges, which we want to leave free from modifications for possible subsequent design steps. The preservation score of a feature is calculated by integrating the score field values over the length of its line segment.

The *topological descriptor* of a feature influences its preservation score in the following manner: if the feature is marked as **jC** (joint continuity), its score is doubled; if the feature is marked as **g** (generated procedurally), its score is set to zero as it does not carry any information concerning the original detail description; if the feature is marked as **f** (part of a *fixed shape*), its score is the sum of the scores of all features with this descriptor in the same fixed shape.



(a)



(b)

Figure 7.3: The score field based entirely on the relative positions of the base line and the joints to neighbour segments. (a) shows a 7×8 score field using the function $x^2 + y^2$, (b) - a score field using the function xy^2.

7.1.3 Restrictions on Contraction

In order to avoid situations where the contraction of a feature leads to self-intersections or to overstepping the limits of the base line and joints, we introduce the concept of *noses* and *necks*. As shown in the top image in Figure 7.4, if we contract feature **a** fully, the resulting polyline would occupy both sides of the joint with the vertex marked as **neck 1** lying on the wrong side of the joint. Similarly, contracting feature **b** would produce a self-intersecting polyline due to the angle at the vertex marked as **neck 2**. This results in the following:

Definition 7.4 Let **v** be a vertex in the polyline underlying the feature collection and let L_v be a line through **v** parallel to the *base line* that intersects side **S** of the *segment polygon* before leaving the inside of this polygon for the first time (dashed blue lines in the top image in Figure 7.4). Let S_v be a line through **v** parallel to **S** (dashed orange lines in the top image in Figure 7.4). If the two features incident to **v** and the polygon side **S** lie on different sides of S_v, then **v** is a *neck*.

In the bottom image in Figure 7.4 the contraction of feature **a** would result in the vertex marked as **nose ext** lying on the wrong side of the joint. Similarly, contracting feature **b** would produce a self-intersecting polyline due to the angle at the vertex marked as **nose int**. This results in the following:

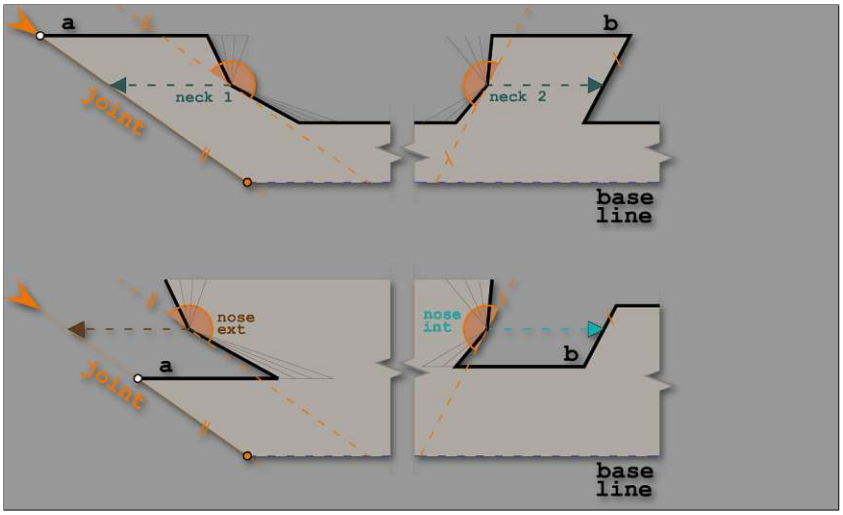


Figure 7.4: Noses and necks in the feature collection.

Definition 7.5 Let **v** be a vertex in the polyline underlying the feature collection, and let L_v be a line through **v** parallel to the *base line* that intersects side **S** of the *segment polygon* or one of the joint lines before entering the inside of this polygon for the first time. Let S_v be a line through **v** parallel to **S**. If the two features incident to **v** and the polygon side **S** lie on different sides of S_v, then **v** is a *nose*. Noses, for which **S** is one of the joints, are external; all others are internal.

Necks and noses are the starting points of *potential* lines (see Figure 7.5) that restrict the contraction of features.

7.2 Potentials

The *potentials* have the role of maintaining the integrity of the segment polygon throughout the contraction of the feature collection by imposing constraints on contraction lengths.

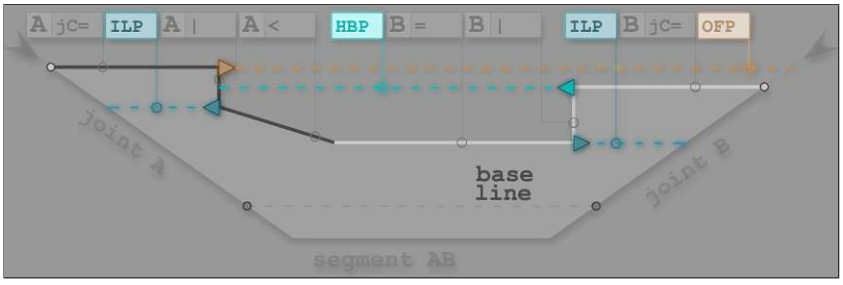


Figure 7.5: The potentials resulting from the feature collection shown in Figure 7.1 (see the key in Figure 7.6).

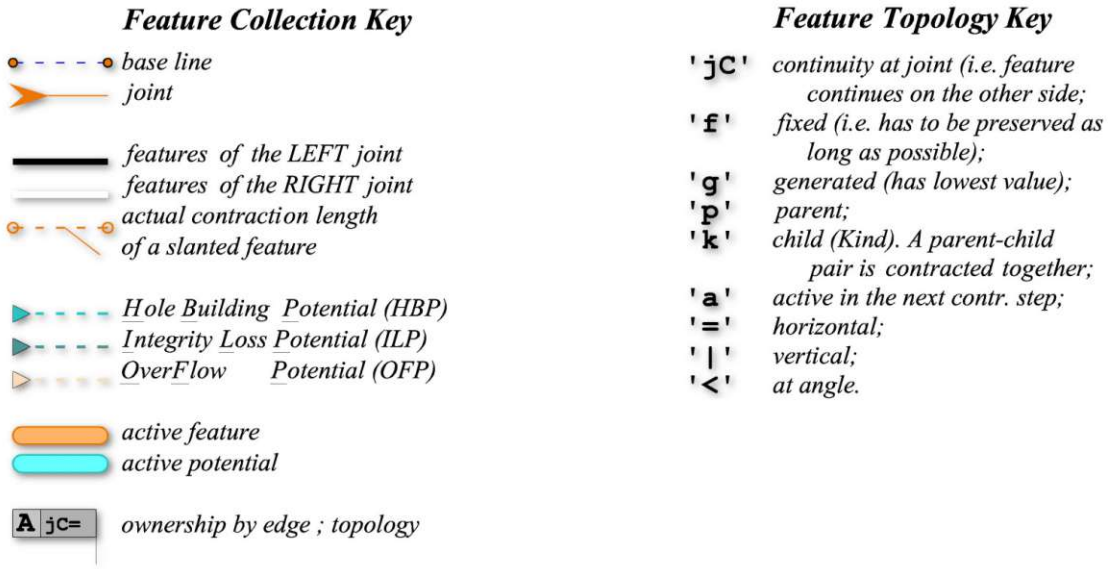


Figure 7.6: The feature collection and potential keys. They apply to all figures in this Chapter and the next.

Potentials are defined as follows:

Definition 7.6 The vector parallel to the base line and with starting point at an internal nose and endpoint at its next intersection with the segment polygon is a *Hole Building Potential (HBP)*. All HBPs lie outside of the segment polygon. The vector parallel to the base line and with starting point at an external nose and endpoint at its next intersection with one of the joints is an *OverFlow Potential (OFP)*. All OFPs lie outside of the segment polygon and either their starting point or their endpoint lies on one of the segment joints. The vector parallel to the base line and with starting point at a neck and endpoint at its next intersection with the segment polygon is an *Integrity Loss Potential (ILP)*. All ILPs lie inside the segment polygon.

The potential types defined above can be seen in Figure 7.5. Each potential has a length and a *reach* - the collection of features within its influence. A feature within its reach can be contracted only until the length of the potential reaches a predefined threshold T. Each potential vector also has a *preservation score*, calculated using the method for calculating feature preservation scores described in Section 7.1 - by integrating the score field values over the length of the vector.

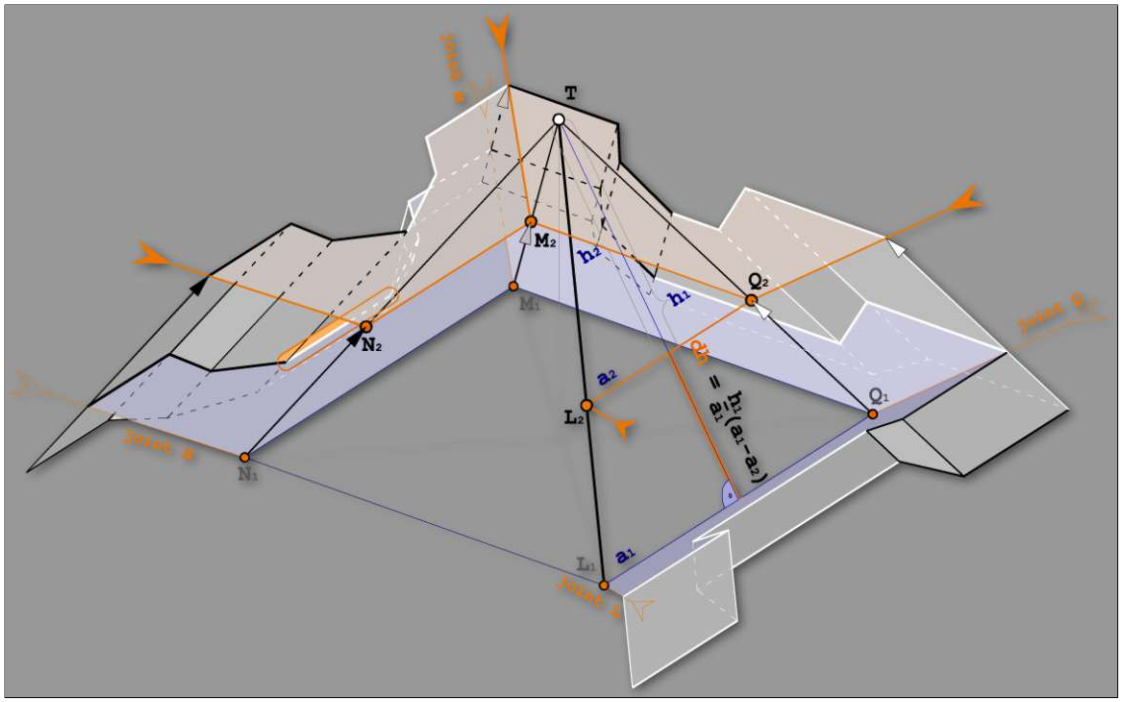


Figure 7.7: One contraction step: the marked feature (the orange highlight on the left, corresponding to corner face $\Delta N_1 M_1 T$) is contracted fully. Consequently the base line in segment LQ changes its length from a_1 to a_2 and the corner solution advances towards the corner top by ∂h.

7.3 Step by Step Construction of the Corner 2-Manifold by Contracting the Feature Collection

The contraction of the *feature collection* and, consequently, of the *base line* has the effect of advancing the corner 2-manifold, or *corner solution*, towards the corner top (see Figure 7.7). If the base line length measures a_1 before the current contraction step and a_2 afterwards, the movement towards the top measured along the bisector of the angle at the top T of the relevant corner side (see line segment marked ∂h between L_1Q_1 and L_2Q_2 in Figure 7.7) can be expressed as:

$$\partial h = \frac{h_1}{a_1}(a_1 - a_2)$$

or

$$\partial h = \frac{h_1}{a_1}\partial a$$

where h_1 is the distance of the base line from the top before the current contraction step. Each feature is automatically moved along its owner edge to its new position in 3D. The

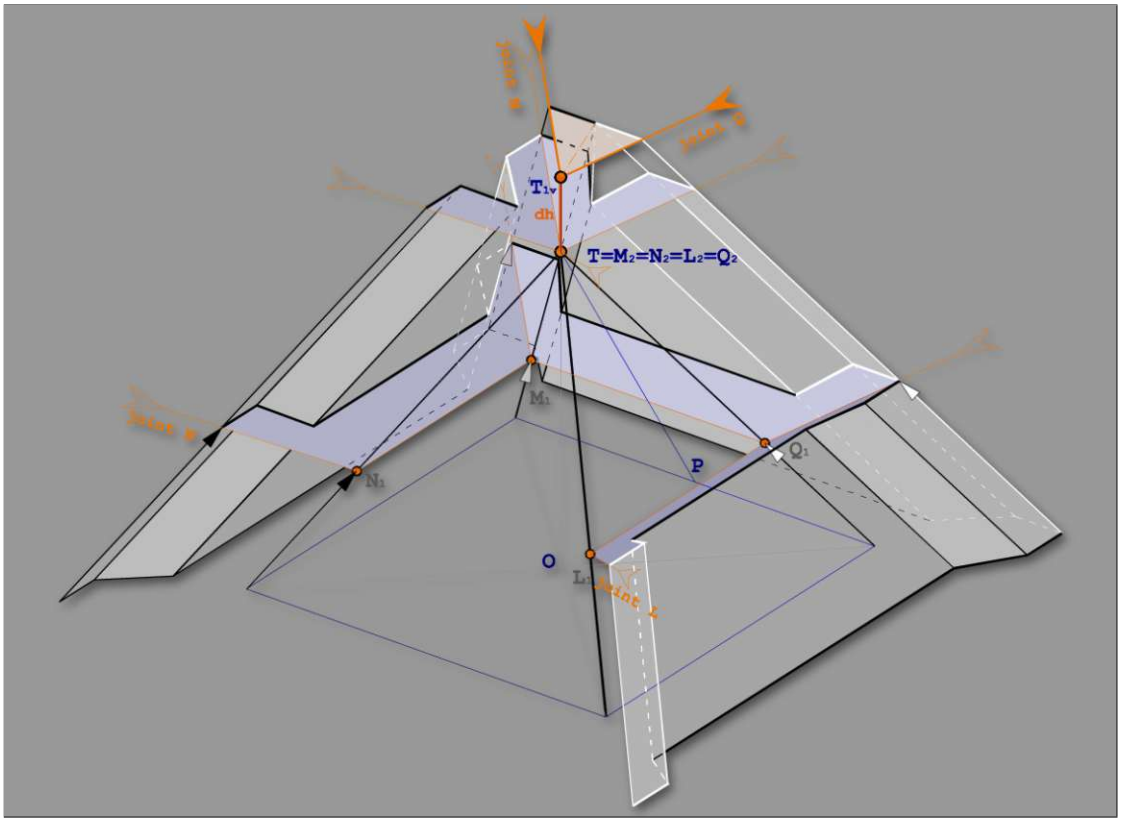


Figure 7.8: Contraction after the base line length has reached zero: the intersection point of the joints M and Q moves along the intersection line of the bisector planes at edges TM and TQ.

positions of not yet contracted features do not change relative to their respective owner joint in the base line plane.

As long as the length of the base line is positive, each contraction step causes its shortening. As the corner solution approaches the corner top, the base line becomes shorter, until it reaches zero at the top. Up to this point, the base line has the role of an interface between the feature collection for its segment and the corner skeleton, whereas the joints are the interfaces to the neighbour segments. Once the corner top is reached, the interface to the corner skeleton is no longer necessary (see Figure 7.8).

Figure 7.9 shows this situation in the context of the *feature collection*. The intersection point of the edge joints traverses the intersection line TT_{1v} of the bisector planes of their respective edges (edge Q and M in Figure 7.8, 7.10 and 7.12). Its normal projection in the feature collection plane is a line passing through the middle of the *base line* (see the orange dashed line passing through O in Figure 7.9(a)). The advancement ∂h_v of the corner solution is measured along this line (see also Figure 7.11 and Figure 7.12) as:

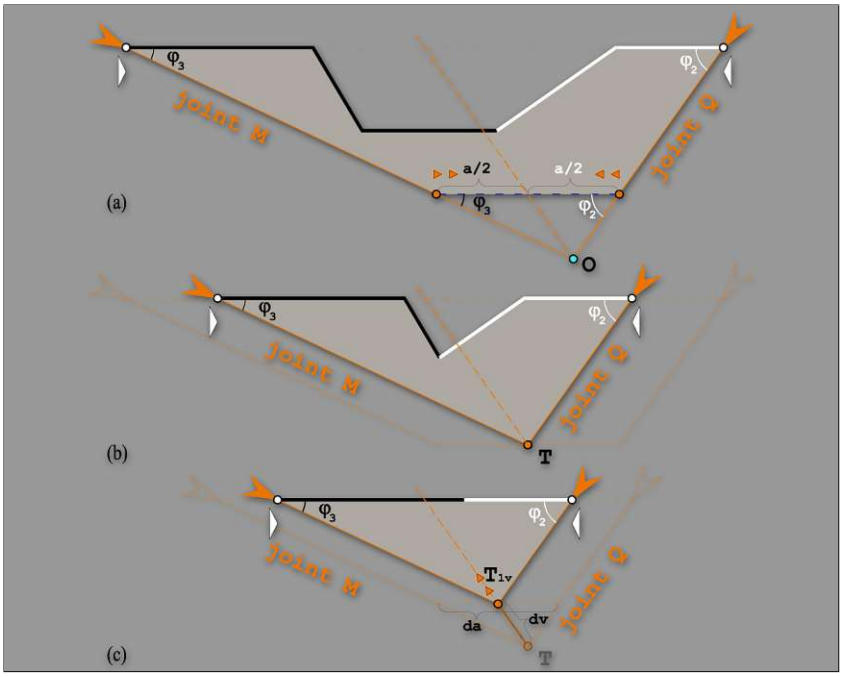


Figure 7.9: Contraction of the feature collection beyond length zero. (a) shows a situation somewhere along the corner side; (b) shows the situation when the base line reaches the top **T** and becomes zero (i.e. a point); (c) shows the next contraction step: the effect is that the intersection point of the joint lines moves away from the point in (b). Figure 7.10 shows the symmetrical case and Figure 7.12 - the asymmetrical case in 3D.

$$\partial h_v = \frac{\partial v}{b} h, \partial v = \partial a \sqrt{\frac{\sin^2(\varphi_2)}{2\sin^2(\varphi_2+\varphi_3)} + \frac{\sin^2(\varphi_3)}{2\sin^2(\varphi_2+\varphi_3)} - \frac{1}{4}}$$

where ∂a is the size change of the feature collection measured parallel to the (former) base line from joint to joint, ∂v the advancement of the intersection point of the joints along the normal projection of the line TT_{1v} in the plane defined by the joints (and base line), and φ_2 and φ_3 the angles between each joint and the base line (see Figure 7.9).

In cases where the corner skeleton is a regular pyramid, the intersection lines of the bisector planes of each pair of neighbour edges are identical (see line TO in Figure 7.10). In the general case, however, they differ from each other (see Figure 7.11 and lines TT_2 and TT_3 in Figure 7.12). The consequence of this is the *partial overlapping* of the feature collections of each corner side, which poses additional restrictions on the contraction length of individual features. However, it does not affect the contraction procedures discussed in the following Chapter.

Both in the regular and in the general case, the last step of building the *corner solution*, after the contraction length limits of all remaining features have been reached, involves

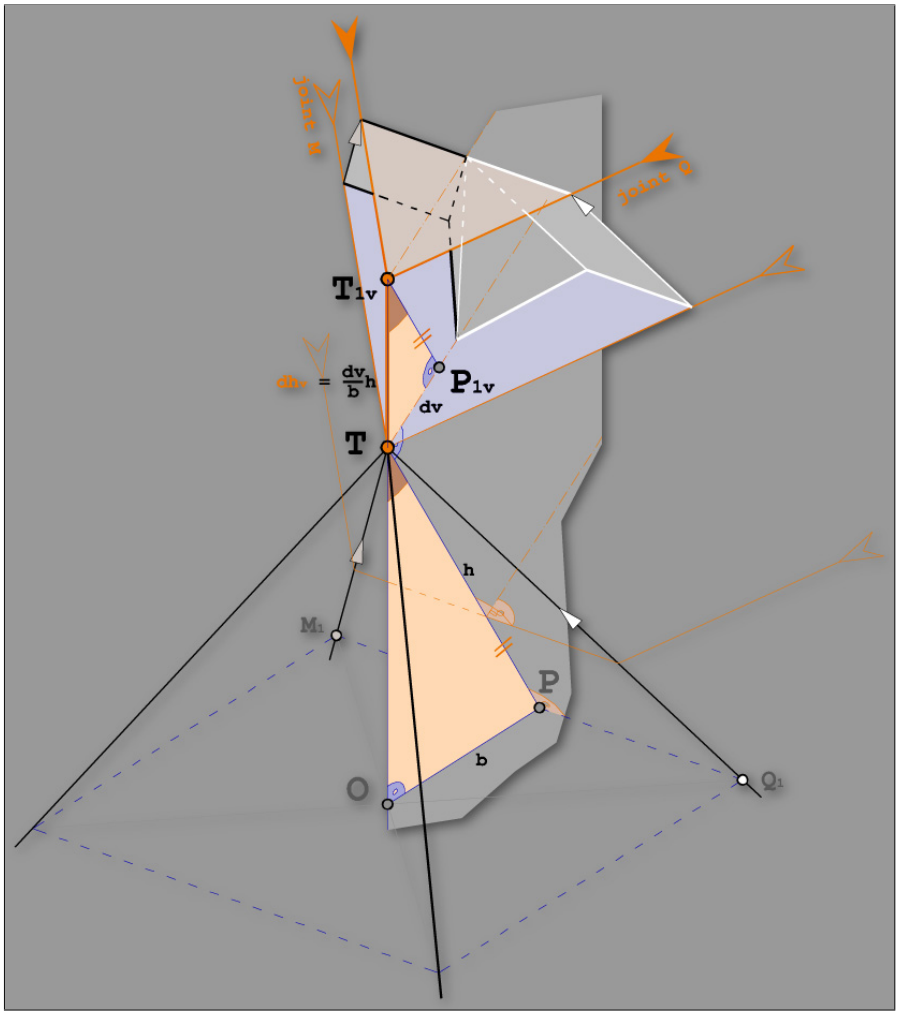


Figure 7.10: Contraction of the feature collection in Figure 7.8 beyond length zero in 3D. Calculation of the movement ∂h_v of the feature collection along the intersection of the *bisector planes* of edges TQ and TM.

the construction of a *cap* 2-manifold where the involvement of the designer is again possible, and even desirable (see Chapter 9).

After assembling the feature collection (see the examples in Figure 7.13) and preparing for transferring the results of its contraction to the corner skeleton, with the effect of advancing the corner solution in the direction of the corner top, in the next Chapter define the *contraction procedures*.

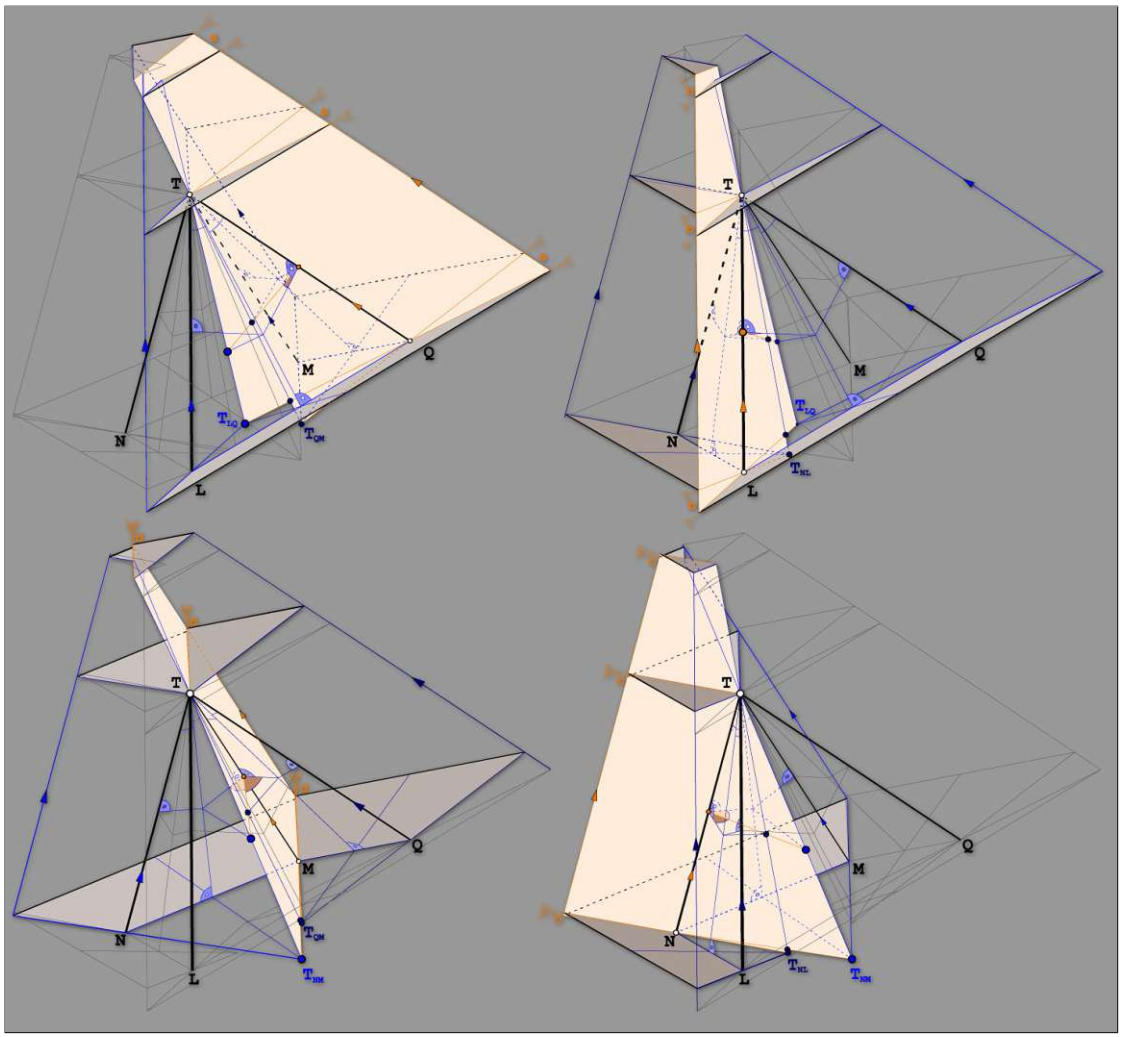


Figure 7.11: The bisector planes for each edge in a corner skeleton contain the segment joints - general case.

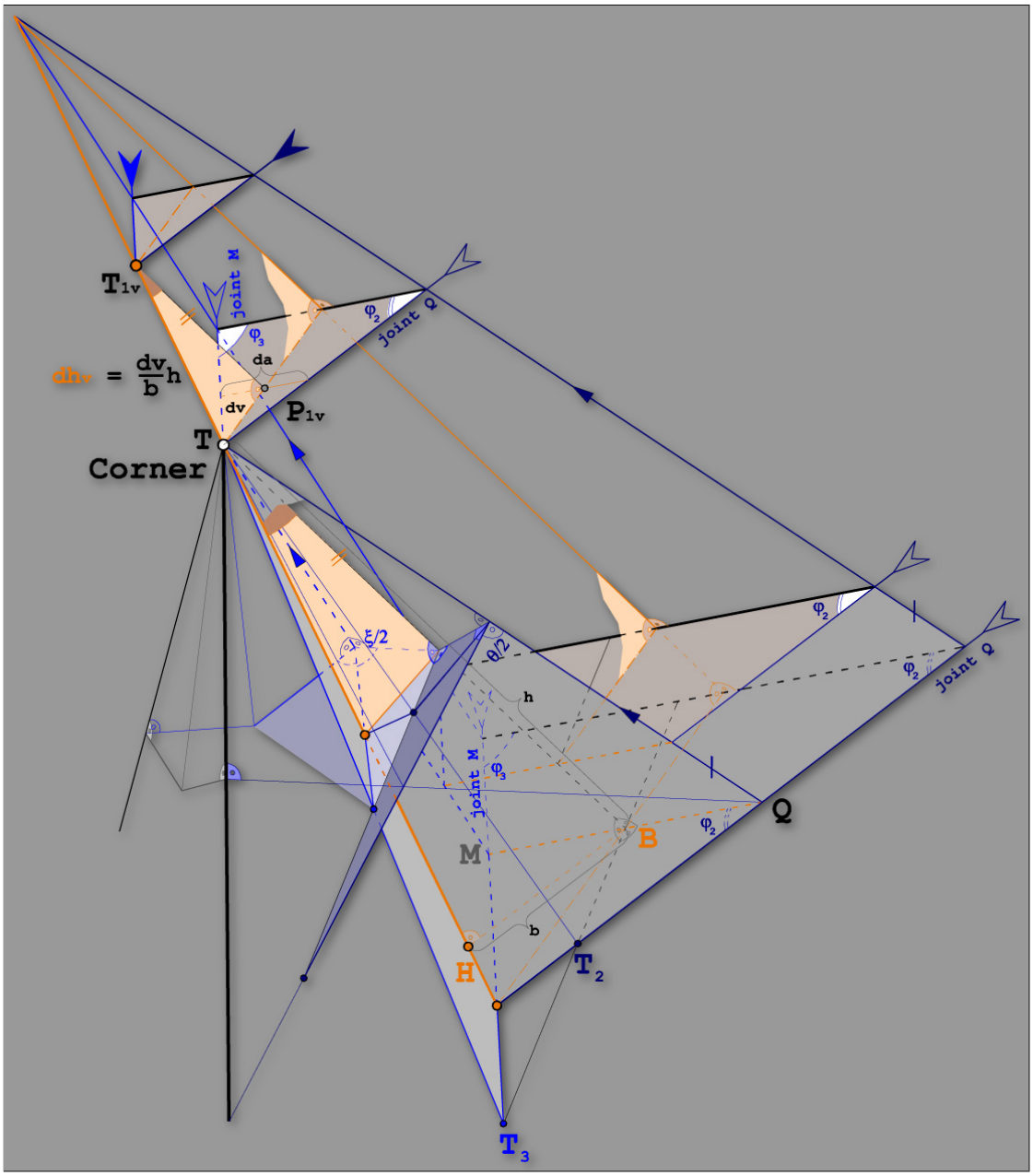
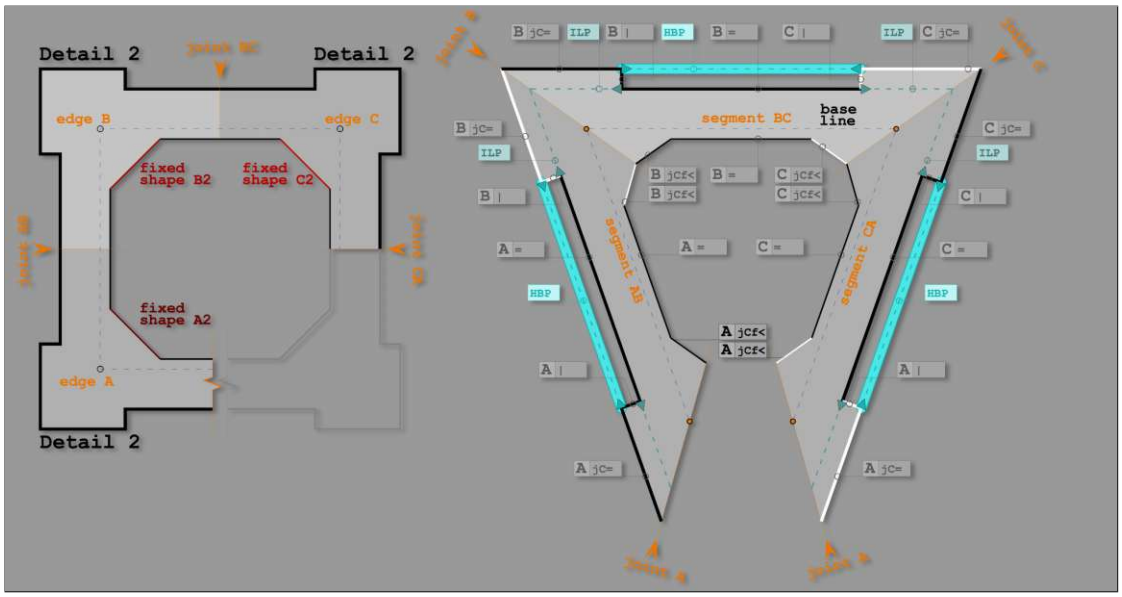
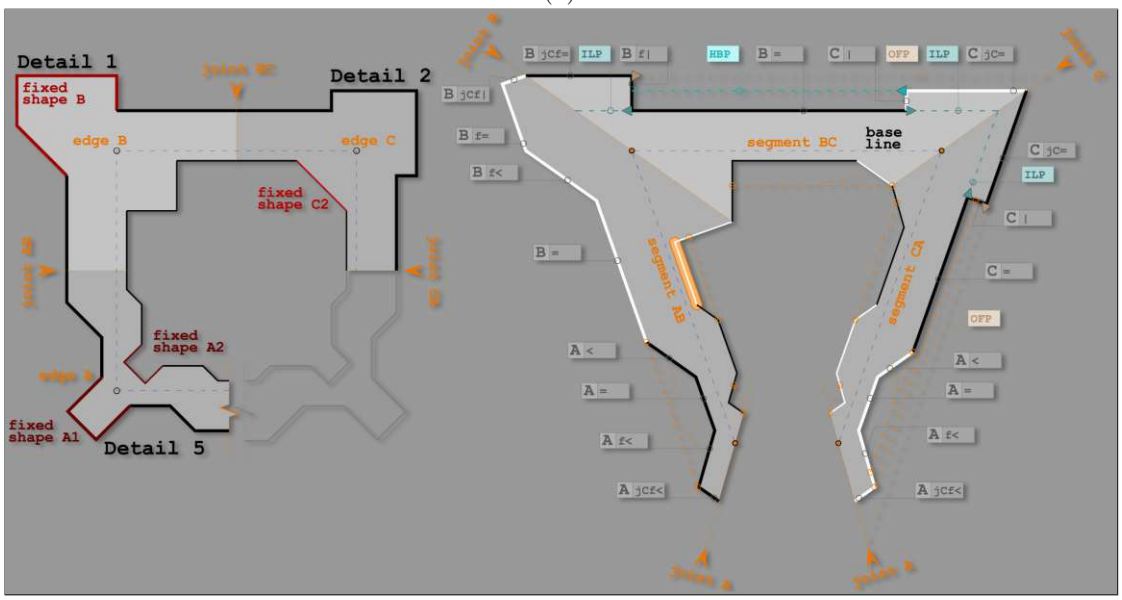


Figure 7.12: Contraction in 3D - general case. The orange plane ε passes through the intersection line TH of the bisector planes of edges TM and TQ and through the middle of MQ, marked as B. The points M and Q are at equal distance from T. Once the base line has been contracted to length zero, the contraction can continue along line TH (see point T_{1v}).



(a)



(b)

Figure 7.13: The *edge composition* and corresponding *feature collection* for the corner solution shown in Figure 6.9(in (a)) and Figure 6.10(in (b)) respectively.

CHAPTER 8

Procedural Shape Contraction

In this Chapter, we present result **R3**: *The abstract outline of the Procedural Shape Contraction algorithm*, while outlining the rules guaranteeing the feasibility of the resulting 3d model (see research question **RQ3**).

Before each contraction step the following updates are performed for the feature collection of each segment of the corner skeleton: The score field is re-evaluated for the new length of the base line, the preservation score of each feature and its contraction length are updated, and all potentials are calculated anew. The restriction on the contraction length of each feature is determined based on the potentials in whose reach (area of influence) it lies. If the base line length is zero, the score field is a linear function of the distance of the feature from the intersection point of the joints measured along the (former) base line normal.

Subsequently the Hole Building Potential (HBP) with lowest score or, if none exists, the feature with lowest score over all feature collections (in all segments of the corner skeleton) is chosen for contraction. If there is more than one potential or more than one feature with the same preservation score, the one at shortest distance to its base line, or to the intersection point of its joints, is chosen. If the distances are also equal, the contraction of these potentials or features proceeds simultaneously.

8.1 Contraction of a Feature

The contraction of a feature parallel to the base line is shown in Figure 8.1. The marked feature has the lowest preservation score, its contraction length is its Euclidean length and none of the potentials has a restricting influence on it. Consequently, we can contract the feature fully. The base line is shortened by the feature's contraction length and the position of the rest of the features, relative to their owner joints, remains unchanged.

The contraction of a slanted feature (topological descriptor <) is shown in Figure 8.2. The marked feature again has the lowest score, its contraction length, however, is represented by a line parallel to the base line and intersecting the neighbour to the side of the vertex closer to the base line (on the left). In this case, the contraction has an effect on the above-mentioned neighbour. The second image in Figure 8.2 shows the contraction of the next feature with lowest score, which is again slanted. Its neighbor to the left is shortened even further. As the third image in Figure 8.2 demonstrates, a feature with infinite contraction length (the one with topological descriptor <, see also Section 7.1) is effectively contracted as a side effect of the contraction of vertical or slanted neighbors. This, however, can produce a feature with Euclidean length under the pre-defined threshold T.

The sequence of full contractions in Figure 8.2 is possible because none of the angles between the new neighbor pairs is smaller than the pre-defined threshold α. In cases where this condition is not fulfilled (see Figure 8.3), full contraction is prevented. Instead, we contract the feature until it reaches the pre-defined minimal length T, and then perform a *hole closing procedure*.

The purpose of the hole closing procedure is to eliminate one of the neighbors causing the problem. We take the neighbor with finite contraction length (in this case, the one to the right of **a** in the top image in Figure 8.3). After the partial contraction of the initially chosen feature **a**, we convert the contraction length representation of the chosen neighbor into a new generated feature (topological descriptor **g**) and eliminate or shorten all features within its reach (see bottom image in Figure 8.3). As the preservation score of a generated feature is invariably zero (see Section 7.1), the next feature to be contracted is the newly generated one, which completes the contraction of feature **a**.

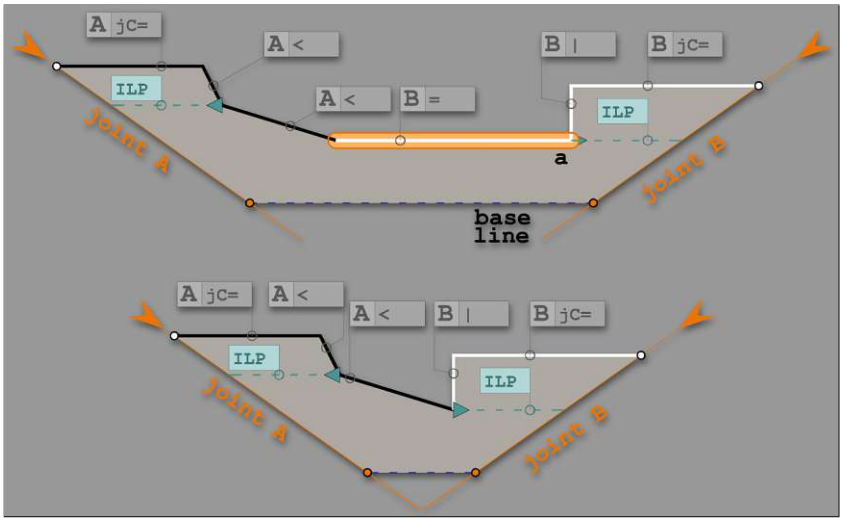


Figure 8.1: Full contraction of a horizontal feature (see the key in Figure 7.6). The highlighted feature **a** has the lowest score based on the current score field. There are no restrictions on its contraction length (in this case its Euclidean length).

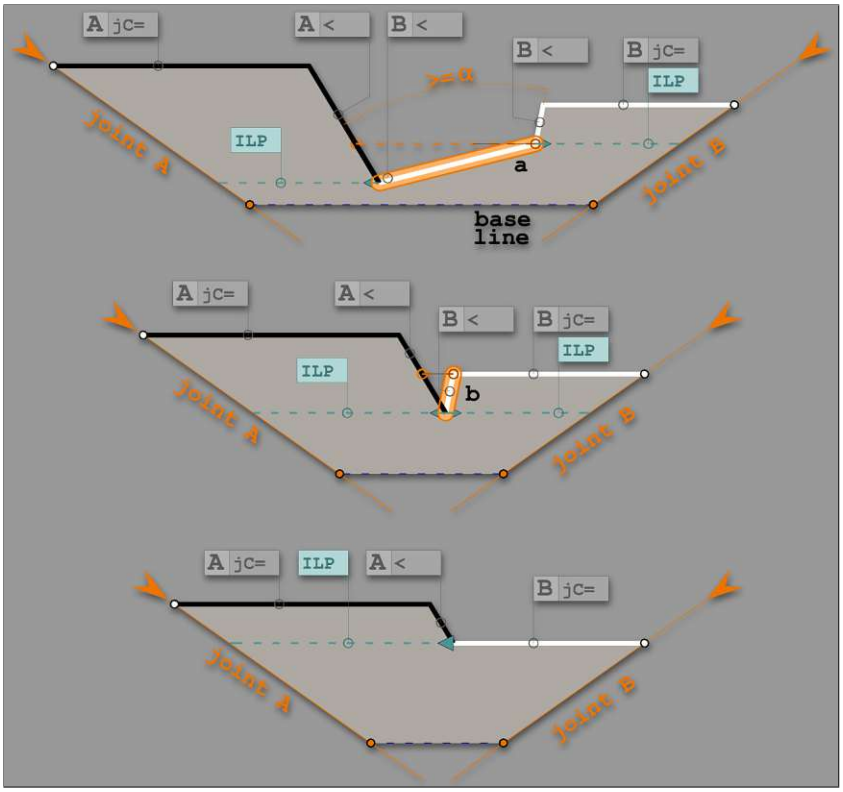


Figure 8.2: Full contraction of a slanted feature (topological descriptor <, see the key in Figure 7.6) when the angle between its neighbors is larger or equal to a pre-defined minimum α. Its contraction length is shown as a dashed line.

The feature collection in the bottom image of Figure 8.2 represents a configuration that cannot be further simplified through feature elimination if we want to maintain the joint continuity (topological descriptor **jC**). We can only perform partial feature contraction, first until the joint adjacent feature with lower preservation score (closer to the base line or intersection point of the joints) reaches the pre-defined minimal length T, and then until the other joint adjacent feature does the same. After that, it is time for the designer to determine with what type of *cap* to complete the corner solution.

8.2 The Influence of Potentials on Contraction

The potentials of a feature collection have a restricting effect on the contraction of features. No feature within the reach of a potential vector can be fully contracted, if that results in a shortening of the potential vector to a length below the pre-defined threshold T.

Figure 8.4 shows a feature collection containing a Hole Building Potential (HBP). As the name indicates, disregarding the restrictions emanating from it results in an uncontrolled

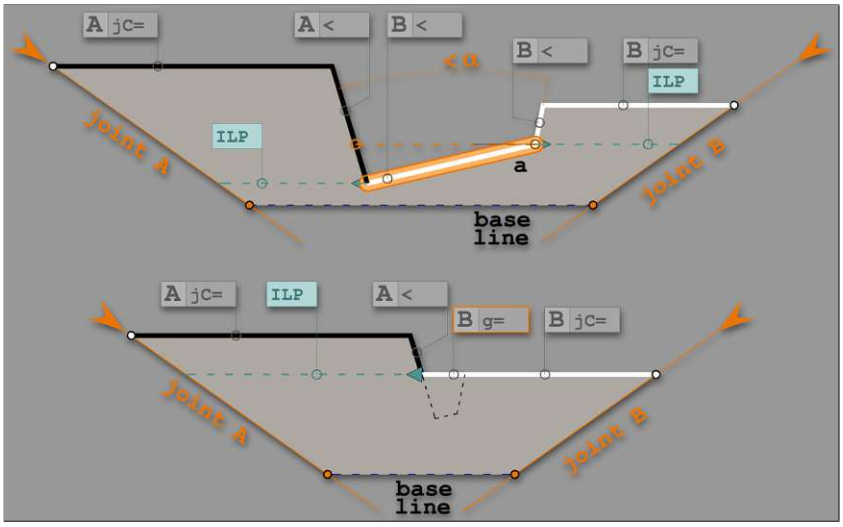


Figure 8.3: Full contraction of a slanted feature (topological descriptor <, see the key in Figure 7.6) when the angle between its neighbors is smaller than a pre-defined minimum α. Instead of performing a full contraction, we contract the feature to the admissible minimum length T and then perform a *hole closing procedure*, which produces the generated feature marked with topological descriptor **g**.

intersection of features producing a *hole* in the corner solution. Therefore, if the feature with lowest preservation score (**a** in this case) is within the reach of a HBP vector, and its contraction length is larger than $L_{hbp} - T$, where L_{hbp} is the length of the HBP vector and T - the pre-defined minimal admissible line length, we perform *HBP contraction*. This consists of the partial contraction of feature **a** until the length of the HBP vector reaches T, followed by the *hole closing procedure* described in the previous Section. Finally, we convert the shortened HBP vector into a generated feature (see bottom image in Figure 8.4).

The Integrity Loss Potential (ILP) represents the reverse situation - disregarding the restrictions emanating from this type of potential results in an uncontrolled separation of features into groups producing a split (or loss of integrity) in the corner solution. Figure 8.5 shows a feature collection containing multiple ILP vectors. Form the point of view of architectural design, this is an atypical situation with the building volume receding at the corners. Nevertheless, it is worth examining for applications outside the domain of architecture.

If the feature with lowest preservation score (**a** in the top image in Figure 8.5) is within the reach of an ILP vector, and its contraction length is larger than the length of $L_{ilp} - T$, where L_{ilp} is the length of the ILP vector and T - the pre-defined minimal admissible line length, we perform *ILP contraction*. This consists of the partial contraction of feature **a** until the length of the ILP vector reaches T, followed by *shape extraction*. In the case presented in Figure 8.5, feature **a** is within the reach of ILP 1 to 4; consequently, the

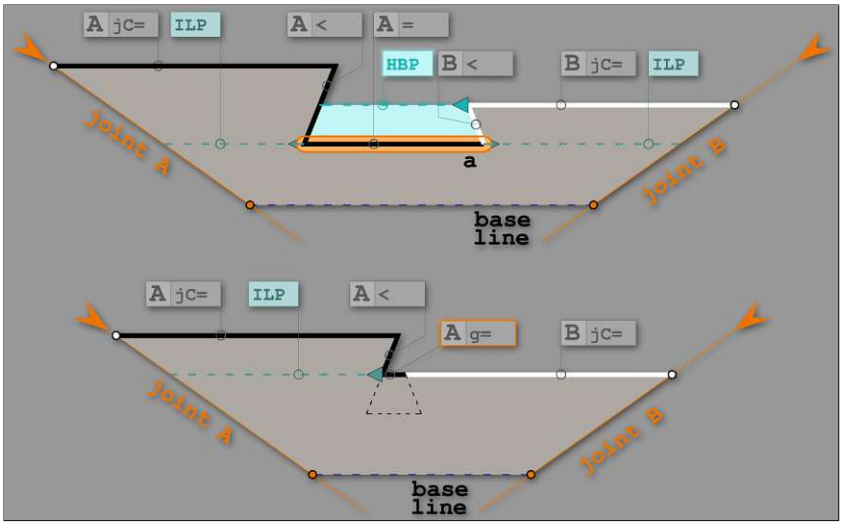


Figure 8.4: Contraction of a Hole Building Potential vector (see the key in Figure 7.6). The shortened HBP vector is subsequently converted into a generated feature.

shortest ILP vectors (2 and 3) are the ones determining the restriction on the contraction of **a**. The second and third images in Figure 8.5 show the shape extraction procedure: we convert the potential vector with length T and lowest preservation score into a generated feature, split the feature collection along it, and calculate a *capping polygon* from the group of features within the reach of the just converted ILP vector.

The OverFlow Potential (OFP) serves to prevent intrusion (or overflowing) of one feature collection into one of its neighbour feature collections. The designated interface between feature collections is the joint; hence, no feature without the topological descriptor **jC** is allowed to have common points with a joint.

An example of this is presented in Figure 8.6. The feature with lowest preservation score, **a**, is within the reach of the OFP vector. The effect of this is that **a** can be contracted only until the length of the OFP vector reaches the threshold T. The second image in Figure 8.6 shows that no further contraction on the left is possible, as all features belonging to the left joint are within the reach of the OFP vector. In this case, in order to advance the corner solution, we employ the *parent-child contraction procedure*, as we have a feature **b1** with topological descriptor **p** (parent) and a feature **b2** with a matching topological descriptor **b2** (child, or Kind). The child feature (with contraction length less or equal to that of the parent) is contracted fully, whereas the parent is contracted either fully or partially, and all features between these two change position relative to their joint (see bottom image in Figure 8.6).

As OFPs are caused by features further away from the base line than most others, they have high preservation scores and cannot be eliminated by contracting their neighbors alone. That is why shape extraction or child-parent contraction procedures are necessary, in order to advance the corner solution. The description of the algorithm prototype in

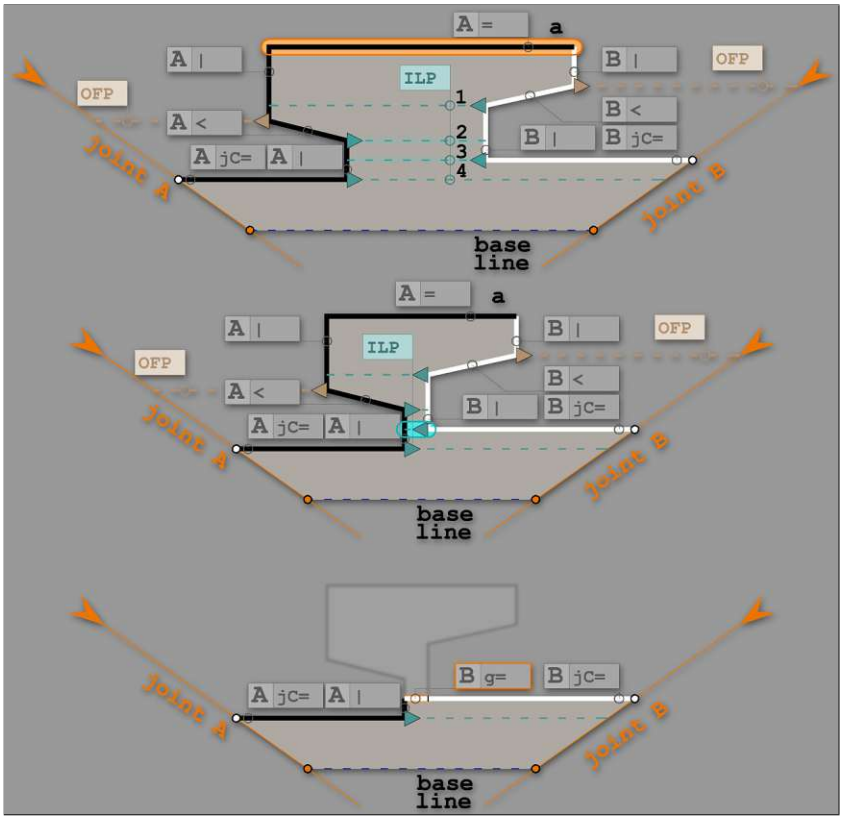


Figure 8.5: Feature contraction restricted by an Integrity Loss Potential vector (see the key Figure 7.6). After further contraction of the feature with lowest score (**a** in this case) is no longer possible due to ILP restrictions, we perform *shape extraction*.

the next Chapter offers additional details.

In conclusion, in this Chapter and Chapter 7, we provided the answer to research question **RQ3**: *After we apply the corresponding details to adjacent model edges, the algorithm produces the buildable 3d model, the so-called as-designed model. Form the point of view of the AEC industries, what degree of feasibility has this model?* We demonstrated that the feasibility of the model is given by construction, i.e., the contraction rules defined on the feature collection guarantee that the result is a buildable 2-manifold with boundary. The next Chapter provides the evaluation of this claim by running the Procedural Shape Contraction algorithm on multiple test cases.

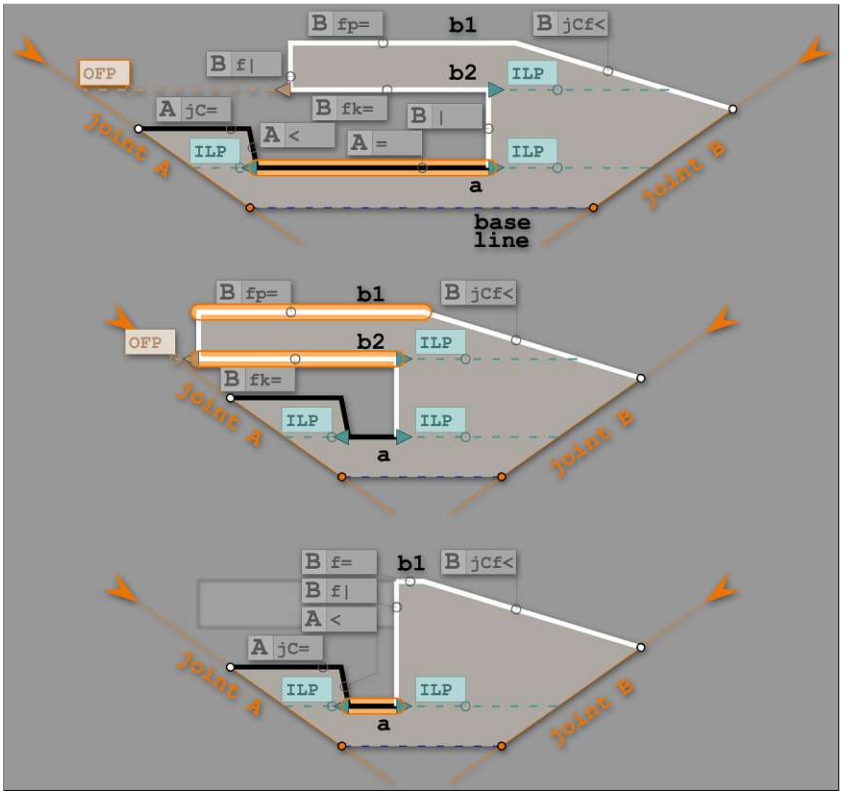


Figure 8.6: Feature contraction restricted by an OverFlow Potential vector (see the key in Figure 7.6).

CHAPTER 9

Algorithm Prototype

In this Chapter we present result **R4**: *The prototypical implementation of the fully automated Procedural Shape Contraction algorithm and a set of test cases.* This prototype demonstrates the application of a *detail* to a *conceptual 3d model*. We will walk step-by-step through the algorithm when applied to a *corner skeleton* where several surfaces in the conceptual 3d model intersect.

9.1 From a 3d Polygon Soup to a Corner Skeleton

In this Section, we demonstrate the method we utilize to extract the corner skeleton from a partially connected surface model, typical of models exported from CAD tools.

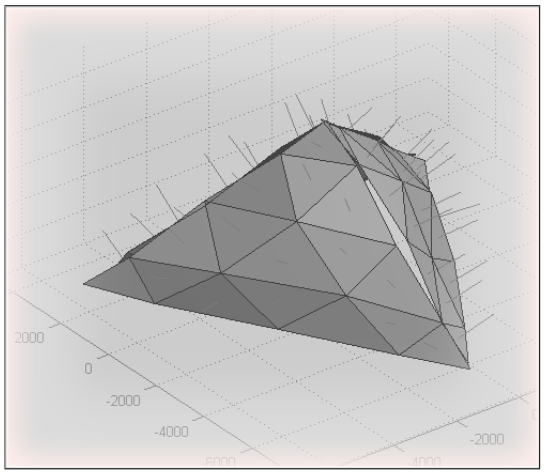


Figure 9.1: Import indexed geometry from file.

The geometry of the conceptual 3d model is imported from an ***.obj**-File as multi-indexed geometry. We assume that it consists of several meshes, each representing a (part of a) wall, that intersect in one point. First, we want to find the mathematical definition of this point - the *corner skeleton* described in Section 6.1. We calculate the face pivots and normals and visualize the geometry in flat-shaded mode.

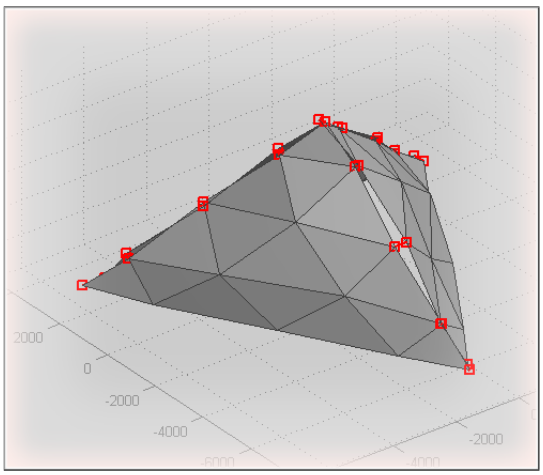


Figure 9.2: Find the vertices where the surfaces (almost) intersect.

We look for the vertex pairs, belonging to different meshes, that are within a predefined threshold of each other. We assume that those belong to the same edge of the corner skeleton. In the image to the left these disjoint vertex pairs are marked as red squares. We use them in the next step to determine the neighbouring faces.

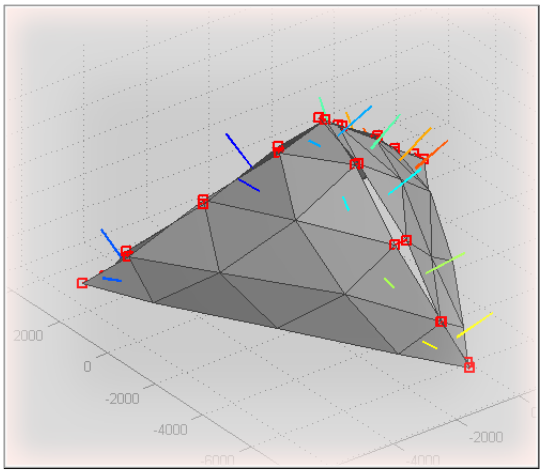


Figure 9.3: Find the neighbouring faces.

We look for neighbouring faces - both those sharing a common edge, and those whose closest edges are sufficiently close (with vertices among those found in the previous step). We further test if the cosine of the angle between their respective face normals is below a threshold T. In essence, we look for neighbouring faces at sufficiently sharp angles to each other to detect the edges of the corner skeleton.

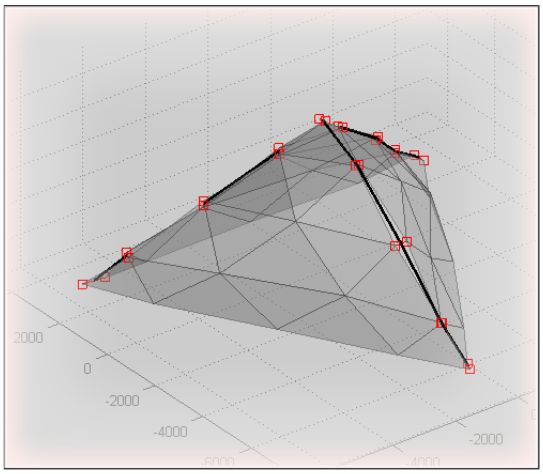


Figure 9.4: Find the edges of the corner skeleton.

We retrieve the *edge chains* - sequences of single vertices (for the single edges, belonging to the same mesh) and of double vertices (for the double edges, belonging to different meshes). We convert the double edge chains into single edge chains through bilinear interpolation.

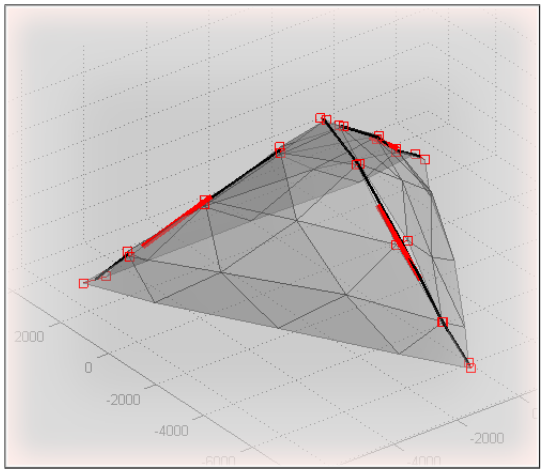


Figure 9.5: Find a mathematical model for the edges of the corner skeleton.

We apply *Principal Component Analysis* to calculate the main axis and pivot of each edge chain. In the image the axes are drawn as thick red line segments centered at each pivot. We subsequently prune the edge chains by calculating a new vector (through averaging) for each sequence of vectors where the angle between any 2 vectors is less than a certain threshold (in this case 25°). We assume that the new vectors represent the true edges of the corner skeleton.

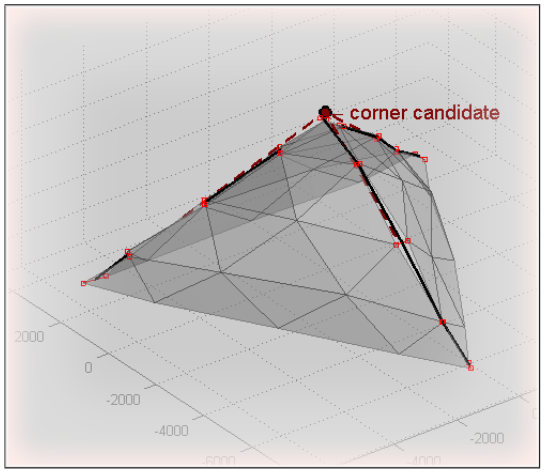


Figure 9.6: Find a corner point candidate.

We traverse the edges of the corner skeleton incrementally and determine the points where they come closest. The corner point candidate is the average of the three points closest to each other. Alternatively, we can calculate the shortest distance between each pair of edges. Bilinear interpolation between the coordinates of the points along the lines, between which this shortest distance was calculated, also gives a good approximation of the corner point.

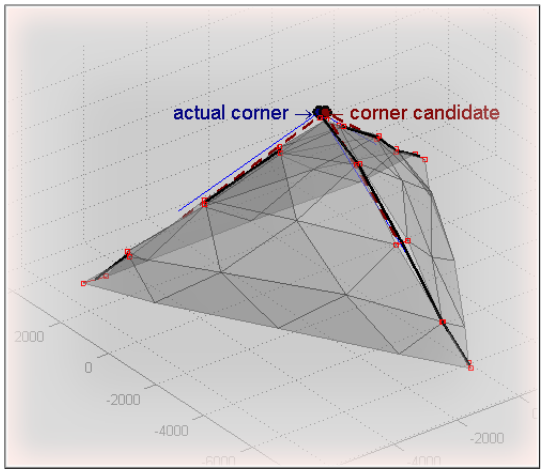


Figure 9.7: Fit a mathematical model to the corner.

We take a number of mathematical models of pyramids (i.e. a tetrahedron, a square pyramid, a hexagonal pyramid, etc.) and look for the best fit to the previously calculated edge vectors. In this example we have 3 edge vectors and therefore we pick a tetrahedron as a mathematical representation. Since the face angles at the corner are within a predefined threshold of the 90° angle, we can pick the representation of a cube corner (see Figure 9.8).

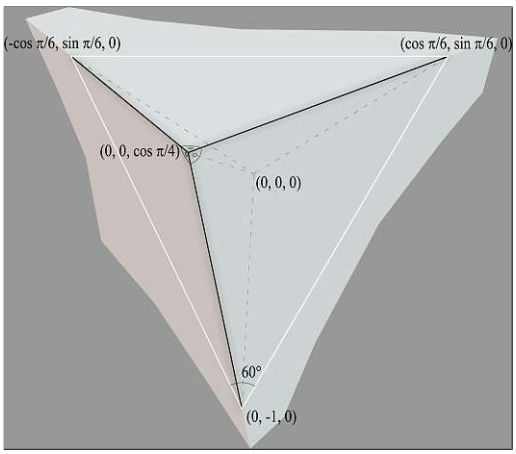


Figure 9.8: A mathematical model of a cube corner.

We perform the fitting by looking for the *affine transformation* of the tetrahedron model, represented by its 4 corner points, that produces the closest approximation of the 3 edges and 1 corner we calculated in the previous steps. The shear-component is removed from the transformation matrix by the use of polar decomposition.

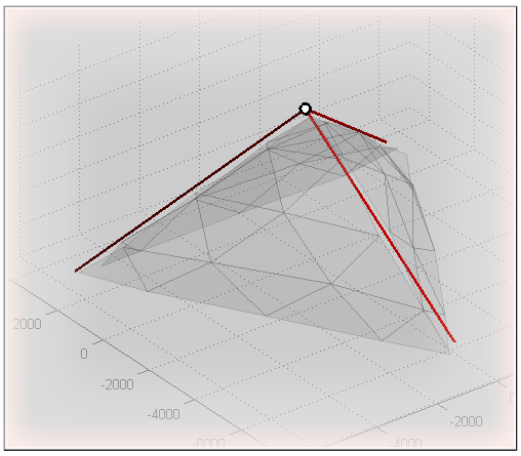


Figure 9.9: Result.

This new model serves in the second part of the algorithm as the *corner skeleton* to which the *detail* is aligned. From now on the corner skeleton is represented by a 4x3 matrix. Each row represents one point lying on an edge at a fixed distance from the corner; the last entry is the corner itself.

9.2 Applying a Detail to the Corner Skeleton of the Conceptual 3d Model

In this Section, we demonstrate the method for aligning an element of the detail library with the extracted corner skeleton by matching and aligning the structural axes. This involves the transformation of the edge-aligned detail composition to the face-aligned segment composition.

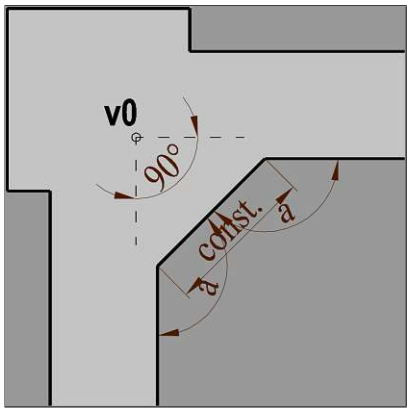


Figure 9.10: Import detail geometry from file.

The detail is imported from an ***.obj**-File containing named sequences of vertex coordinates and vertex attributes. The detail in this example consists of two polylines, drawn in black in the image to the left, with a feature descriptor derived from the vertex attributes associated with each polyline segment. The dashed line in the image represents the local coordinate system that corresponds to the structural axes described in Section 5.1.

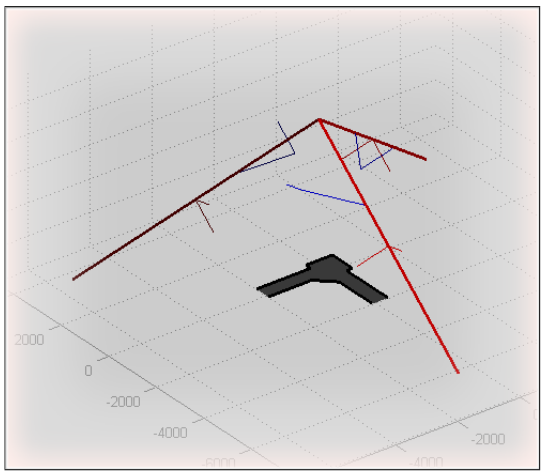


Figure 9.11: Process the corner skeleton from Section 9.1.

We calculate, for each edge segment of the corner skeleton, the plane that is normal to it and passing through its middle. It is represented by the normalized edge vector pointing towards the corner, and both the normalized vectors at right angles to the edge and lying in the faces incident to that edge in the corner skeleton (red in the images to the left). It serves as the edge coordinate system, or local *structural axes*.

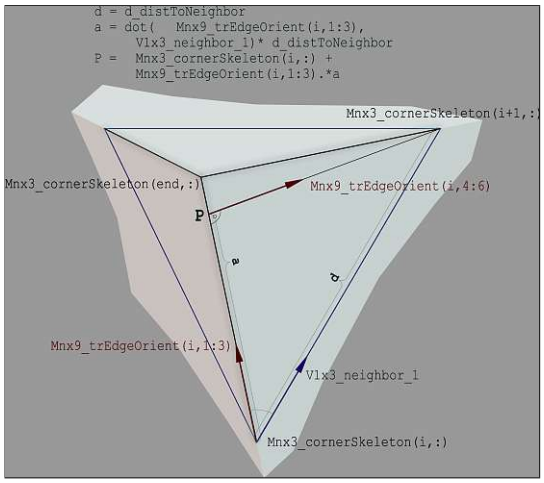


Figure 9.12: Function sketch.

We need to define several coordinate systems in order to transfer the *detail description* from the edges to the faces, in order to construct the *feature collections*. The image to the left shows the construction of the edge coordinate system.

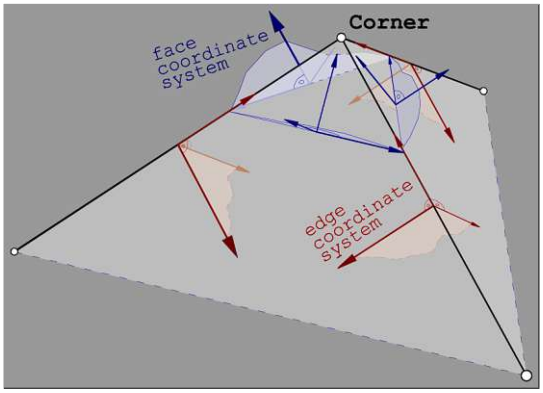


Figure 9.13: Coordinate systems.

We calculate the coordinate system of each face. It is defined by a point on the bisector of the angle between the two corner edges as the *origin*, a point at the intersection between one of the edges and a line through the origin and perpendicular to the bisector (lying on the *x-axis*), and, finally, a point produced by moving the origin a pre-defined distance along the face normal (lying on the *y-axis*). The implicit *z-axis* is defined by the origin and the top of the corner skeleton. The resulting coordinate system is drawn in blue.

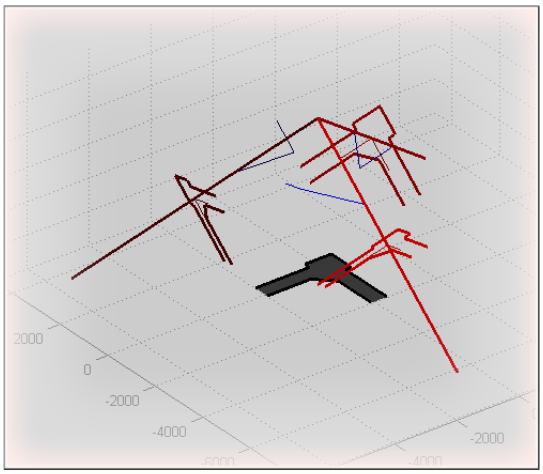


Figure 9.14: Mapping the detail description onto each edge.

In this step we transform the *detail description* from the detail's local coordinate system [0 0 0; 0 0 1; -1 0 0; 0 -1 0], where the first entry is the origin, to the local coordinate system of each edge of the corner skeleton saved in the previous step. The transformation consists only of translation and rotation.

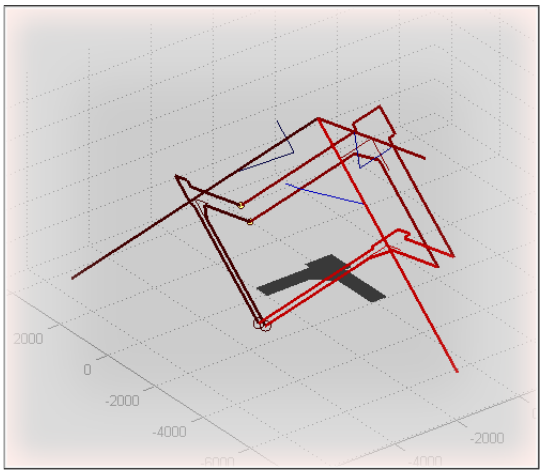


Figure 9.15: Establishing continuity.

We determine which polyline ends, belonging to separate copies of the detail description, are closest and, therefore, candidates for extension. Subsequently we extend those until they intersect. The result of this step is shown in the image to the left.

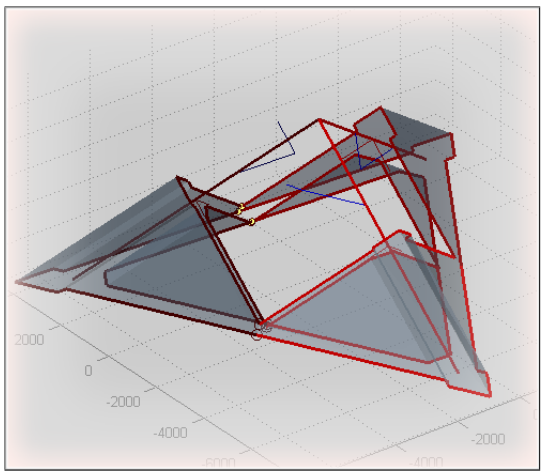


Figure 9.16: Projecting the detail description onto the ground plane.

The contour, resulting from the projection, is the interface for combining the finished 3d model with other geometry. The *ruled surfaces*, spanned between this contour and the edge-aligned detail descriptions, is the first step in building the corner 2-manifold, or *corner solution*.

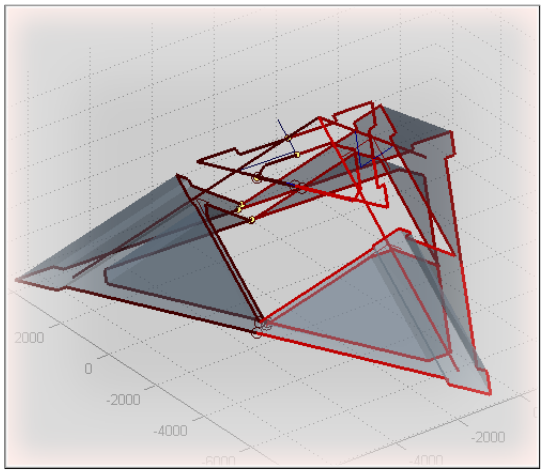


Figure 9.17: Transforming the edge-aligned detail description copies to face-aligned attributed polylines.

We transfer the joints between the detail description copies from the faces to the edges, where there are fewer constraints on them (see Section 6.2). Each detail description copy is projected along its edge twice: on each of the *xy-planes* of the coordinate systems of the faces incident to the edge. Each new copy is expanded from 3 to 5 dimensions by adding the feature descriptor from Figure 9.10 as well as the index of the owner edge to the 3 spatial coordinates of each polyline vertex.

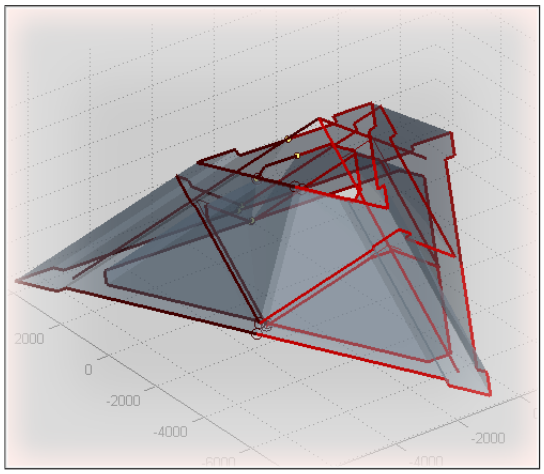


Figure 9.18: Adjusting the descriptor.

The clipping of each projected copy is performed by the bisector plane of the angle between the two faces incident to the detail description's owner edge and the *yz-planes* of the faces' coordinate systems (see Figure 9.13). After the clipping, the polyline segment attributes (*feature descriptor* and *owner*) are adjusted.

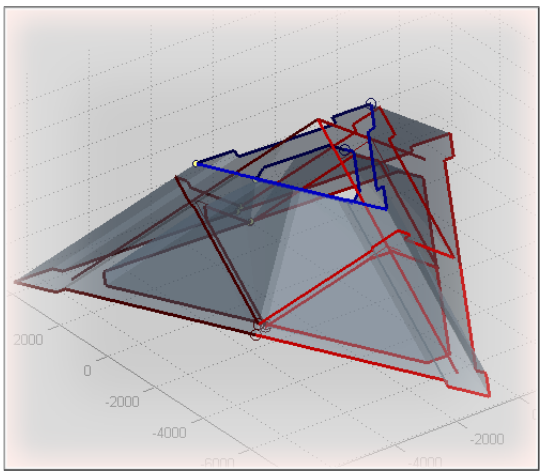


Figure 9.19: Joining the face-based attributed polylines.

The two attributed polylines per face are combined into a single 5-dimensional attributed polyline sequence. Collinear consecutive polyline segments are merged into one. The polylines are then ordered according to the maximal distance of any of their vertices from the top of the corner skeleton. We recalculate the attributes and reattach them to the polyline sequence.

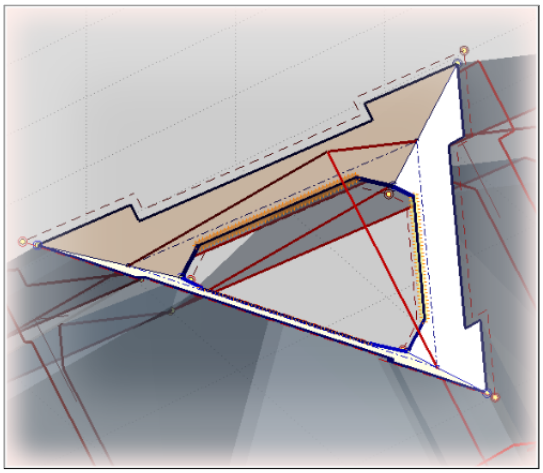


Figure 9.20: Transforming the attributed polyline sequences into feature collections.

From this step on, the attributed polyline sequences are regarded as *feature collections* relative to a *base line* (see Section 7.1). The base line for each face of the corner skeleton is obtained by intersecting that face with the plane, parallel to the *xy-plane* of the face's coordinate system (see Figure 9.13), on which the current face-aligned attributed polyline sequence resides.

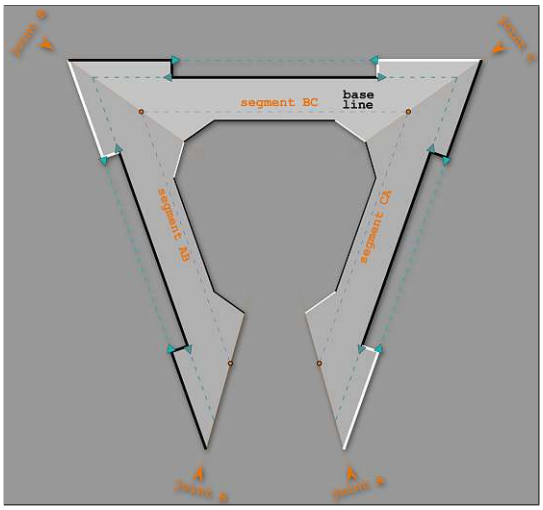


Figure 9.21: Result

All potentials are calculated. By applying *procedural shape contraction* (see Section 7.3) to each feature collection, the base line shortens and advances the corner 2-manifold towards the top of the corner skeleton. As each feature collection contracts, its descriptors are updated.

9.3 Procedural Shape Contraction Leads to the Corner Solution

Here we demonstrate the functionality of the prototype by performing the Procedural Shape Contraction algorithm on two of 170 pre-defined feature collections. In the first step, we import the geometry of a pre-defined corner skeleton from an ***.obj**-File. We apply the algorithm to the feature collection after aligning it with one of the corner skeleton's faces.

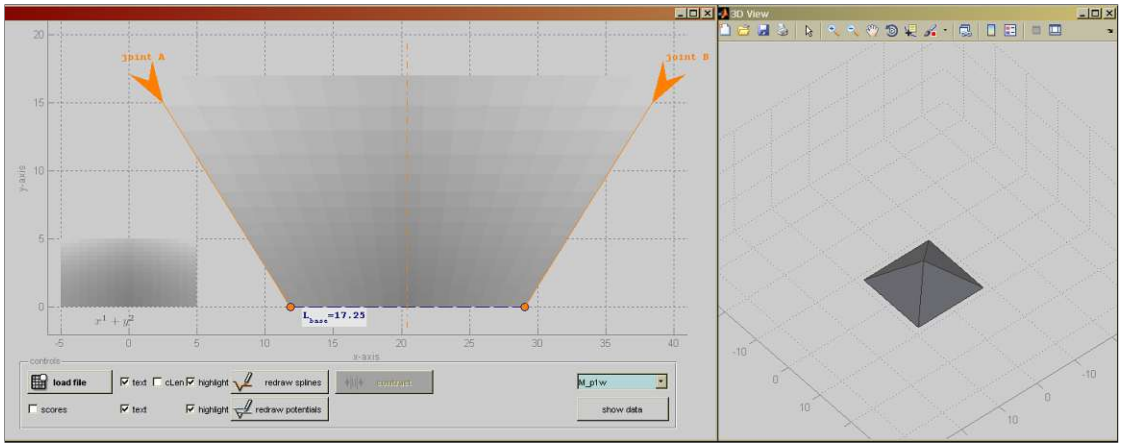


Figure 9.22: Importing the corner skeleton.

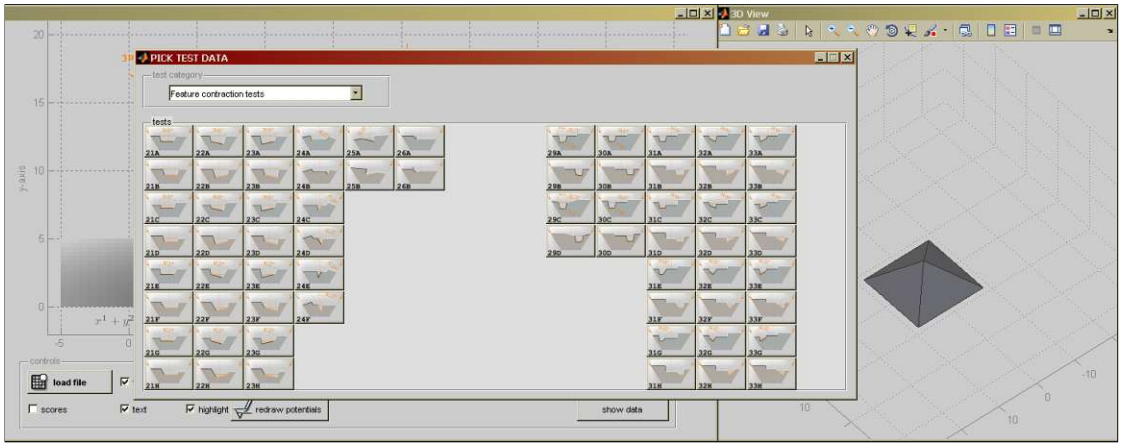


Figure 9.23: Importing a feature collection.

Figure 9.22 shows the user interface for testing the algorithm. The 2d view is on the left, the 3d view - on the right. The geometry of the *base line* (the blue dashed line) and the *joints* (the orange lines) to neighbouring faces are calculated as described in the previous Section and a *score field* is built using the function $x + y^2$. In this score field, the score increases quadratically with the distance from the base line and linearly with the distance from the medial axis of the field. Thus, features highest above the base line and closest to the joints have the highest preservation score and are, consequently, most

preservation-worthy (see Section 7.1).

In the step shown in Figure 9.23 we select one of 170 pre-defined feature collections (see Figure 9.24) representing various facets of the algorithm - the contraction of horizontal, slanted or vertical features, contraction under the constraints of Hole Building Potential (HBP), Integrity Loss Potential (ILP) and OverFlow Potential (OFP). The feature collection is subsequently aligned with one of the faces of the corner skeleton and ready for contraction.

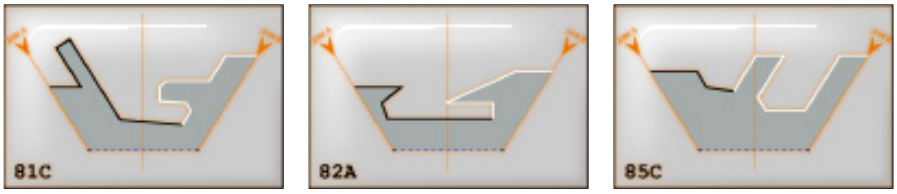


Figure 9.24: Three of the 170 test cases.

9.3.1 Feature Contraction

Test 21C, shown in Figure 9.25, demonstrates the contraction of features without the influence of potentials (see Section 8.1). In the 2d-view the highlighted feature (framed in orange) is the next to be contracted. Its contraction length is represented by the dotted orange line. Its descriptor (see the annotation) contains the topology (<), the preservation score (13.46), and the maximal contraction allowed by influencing potentials (Inf, i.e. infinite). The active feature is not the one with the lowest preservation score; that is its neighbour to the right. Its contraction, however, would result in the angle between the new neighbours falling below a user-defined threshold - therefore the feature with the next lowest score is selected for contraction.

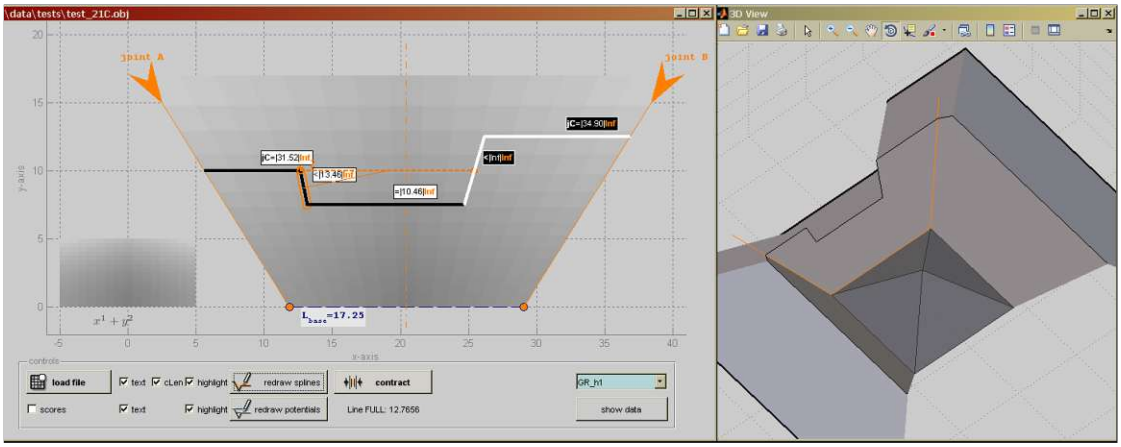


Figure 9.25: Initial configuration of test **21C**.

A feature, similarly to a potential, has a *reach* - a collection of features affected by its contraction. For the currently selected, or active, feature this collection contains the two features to its right. The consequence of this is that, even though the angle between the neighbours of this feature would be above the user-defined threshold after its contraction, it still cannot be fully contracted, because its immediate neighbour to the right is within its reach and its contraction length is smaller.

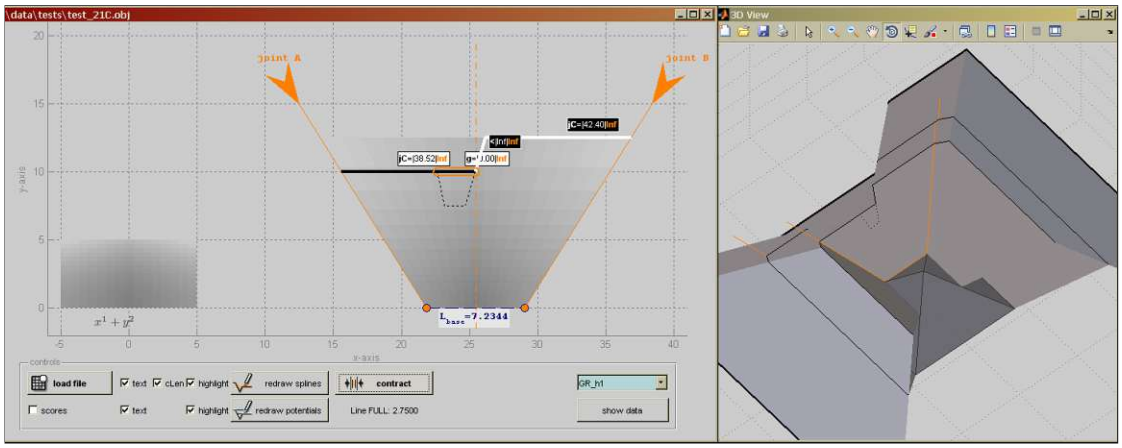


Figure 9.26: Feature contraction followed by a hole closing procedure.

Therefore, the contraction is performed similarly to the one described in Figure 8.3 in Section 8.1. The active feature is contracted until one of the features within its reach - the one to its immediate right in this case, reaches the admissible minimal length. Subsequently, a *hole closing procedure* is performed. The result is a new face in the corner 2-manifold (the polygon with a dashed border in the 2d-view in Figure 9.26) and a new generated feature (topological descriptor **g**). Such features have a preservation score of zero - therefore the new feature becomes immediately active and is contracted in the very next step, depicted in Figure 9.27.

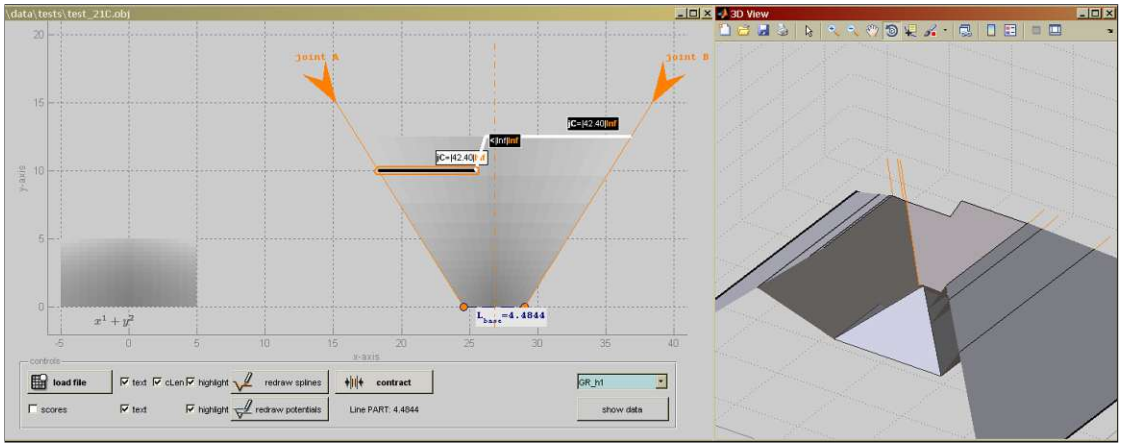


Figure 9.27: Contracting the base line to length zero.

Figure 9.27 shows a typical configuration of the feature collection with two joint-adjacent features that can be fully contracted and a slanted feature that depicts the height difference of these two features. This slanted feature has infinite contraction length and, therefore, an infinite preservation score (see the descriptor in the 2d-view) and cannot be eliminated by contraction. The active feature has the same preservation score as the right-most feature, but it is the lower of the two and is therefore selected (see Section 8.1). This leads to the configuration in Figure 9.28.

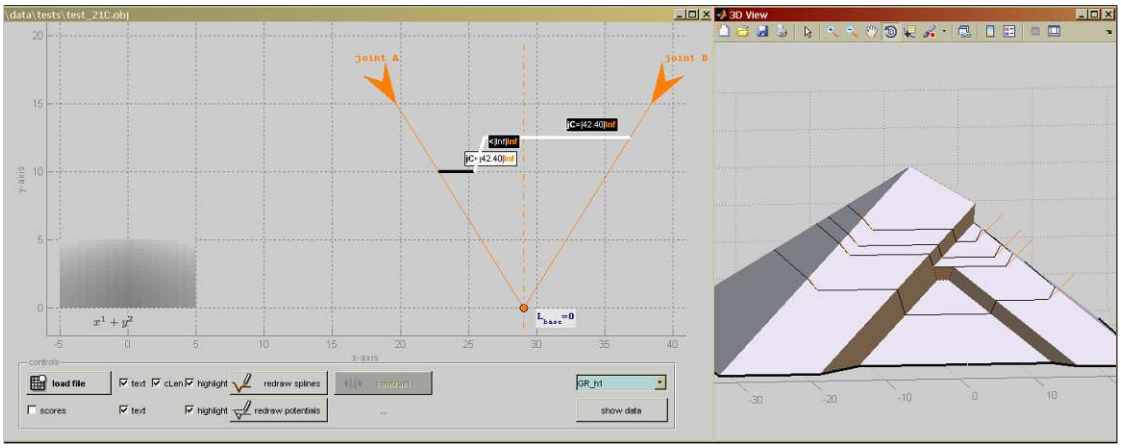


Figure 9.28: Finalising the corner 2-manifold.

Now that the base line is contracted to a point, the contraction proceeds as described in Figure 7.9 in Chapter 7. The 3d-view shows the final result: the lower joint-adjacent feature was contracted fully, and the higher feature was extended to cap the corner 2-manifold.

9.3.2 Feature Contraction within the Reach of a Potential

Test 31H, whose initial set-up is shown in Figure 9.29, demonstrates the contraction of features under the influence of potentials (see Section 8.2).

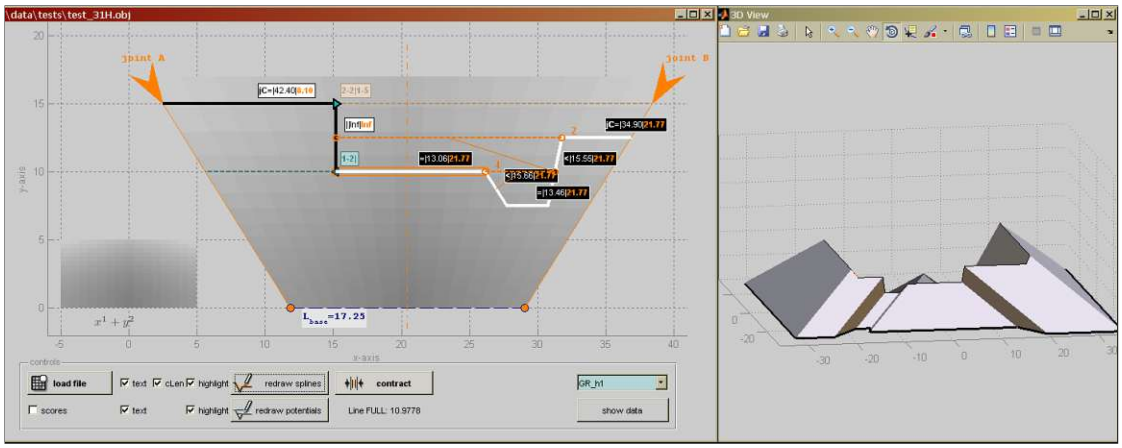


Figure 9.29: Initial configuration of test **31H**.

The descriptor of the feature with the lowest preservation score (active, framed in orange) contains an upper limit to its contraction length resulting from the fact that it is within the reach of the *OverFlow Potential* in the direction of joint B (see the third entry in the annotation in the 2d-view in Figure 9.29: 21.77). However, since the contraction length of the active feature is smaller, it is fully contracted, resulting in the configuration in Figure 9.30.

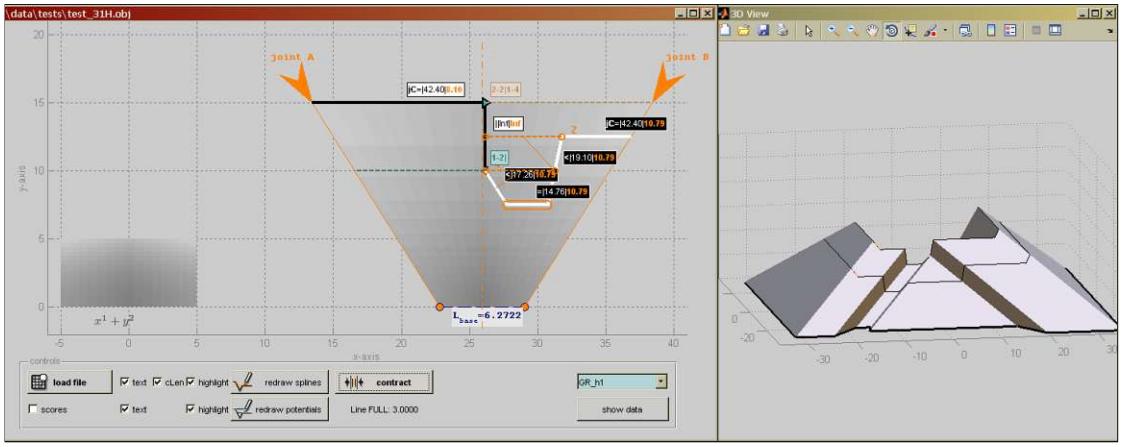


Figure 9.30: Contraction while considering the resulting angle between new neighbours.

Here, the angle between the neighbours of the active feature would be above the user-defined threshold after its contraction. In addition, its contraction length is smaller than the upper limit determined by the OverFlow Potential. Therefore, a full contraction can be performed.

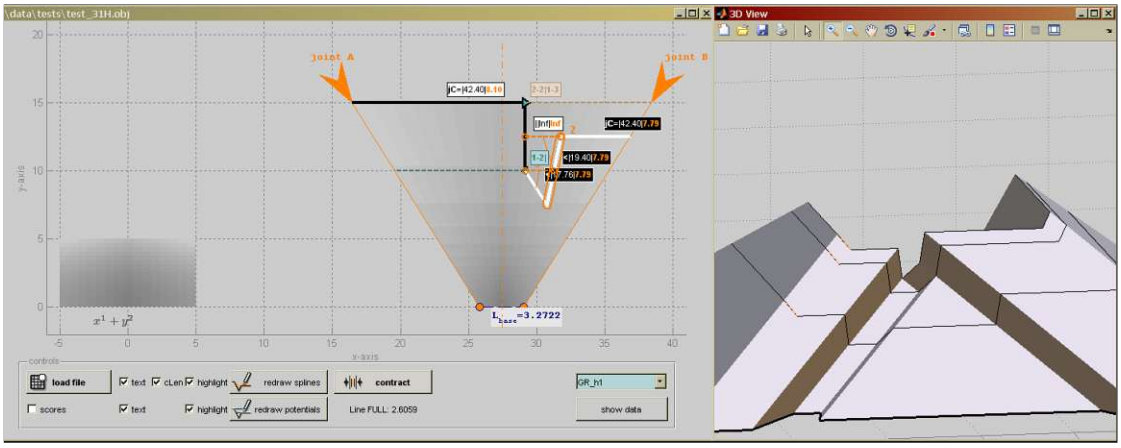


Figure 9.31: Contracting a slanted feature with a non-empty reach.

The result can be seen in Figure 9.31. The configuration is now similar to the one in Figure 9.25. The active feature has a non-empty *reach* which contains a feature with a smaller contraction length (the white slanted feature to its immediate left). This prevents a full contraction. Instead, the active feature is again contracted until the feature with the smallest contraction length within its reach reaches the admissible threshold. A *hole closing procedure* completes this step, resulting in the configuration in Figure 9.32.

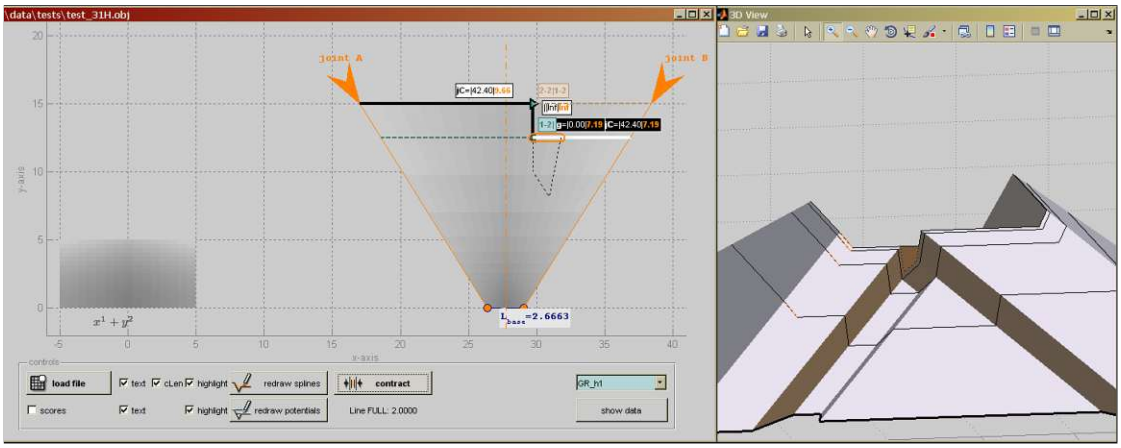


Figure 9.32: Feature contraction followed by a hole closing procedure.

The contraction of the resulting generated feature (highlighted in Figure 9.32) is performed similarly to the one described in Figure 8.3 in Section 8.1 and in Figure 9.26 above, which leads us to Figure 9.33.

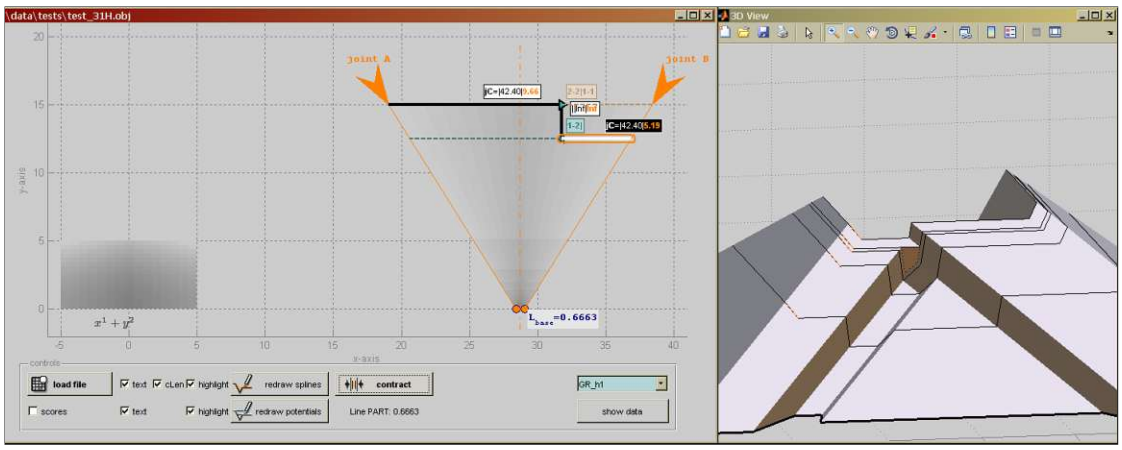


Figure 9.33: Contracting the base line to length zero.

The configuration in Figure 9.33 is similar to the one in Figure 9.27 above. The vertical feature has infinite contraction length and, therefore, an infinite preservation score (see the descriptor in the 2d-view) and cannot be eliminated by contraction. The active feature has the same preservation score as the left-most feature, but it is the lower of the two and is therefore selected (see Chapter 8). The two potentials place an upper limit to the contraction, but in this case it still allows a full contraction of the active feature, resulting in the configuration in Figure 9.34.

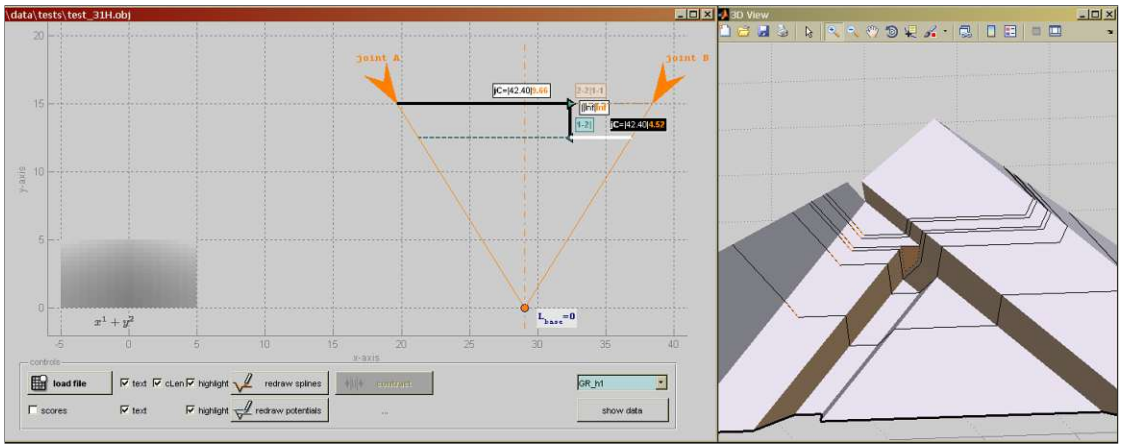


Figure 9.34: Finalising the corner 2-manifold.

This configuration is similar to the one in Figure 9.28 above. Once the base line is contracted to a point, the contraction proceeds as described in Figure 7.9 in Chapter 7. The 3d-view shows the final result: the lower joint-adjacent feature was contracted fully, and the higher feature was extended to cap the corner 2-manifold before the *Integrity Loss Potential* (the thick blue dashed line in the 2d-view) could limit the contraction.

The 170 test cases were added in batches to the test pool during the development and evaluation of the algorithm. Our **Data Collection** strategy (see Section 1.4) consisted of

the manual composition of the 2d feature collections, the collection of the automatically generated 3d models, the time the prototype needed to create those models, and, finally, several manual attempts to build the corner solution performed by the author.

CHAPTER 10

Evaluation of the Results

In this Chapter we give a summary of our results and present the evaluation of the software engineering hypothesis we described in detail in Chapter 4 by answering the research questions we posed in Section 1.3. After analysing the data we gathered during the evaluation phase described in the previous Chapter, we present result **R5**: *The evaluation both of the feasibility and of the time-saving aspect of the proposed method.*
RQ1: Which geometric information has to be extracted from a *detail*, so that it can be consistently applied across the entire *conceptual 3d model*?
ARQ1: We are interested only in the connected contour of the detail as a whole, or the contours of each of its material layers, and in the contour's position and orientation relative to the structural axes of the conceptual 3d model. The geometric properties of the contour that should be preserved when interacting with its neighbor details need to be confirmed by an expert. They include line segments parallel to a structural axis, line segments at an angle of functional significance to a structural axis, and line segments that delimit spaces of functional significance. Furthermore, there should be a set of transformation rules for adapting the extracted contours to the angle between the structural axes of the conceptual 3d model.

RQ2: What conditions have to be fulfilled by the *conceptual 3d model*, or by the result of its pre-processing, so that the detail information can be applied to it correctly in the context of the architectural domain?
ARQ2: For the purposes of this algorithm, the conceptual 3d model has to have edges and corners identifiable by one of the methods we presented in Section 3.4. This excludes surfaces with too little variance in their curvature, e.g. an ellipsoid, or surfaces with too large variance in their curvature, e.g. a porcupine. Furthermore, the edges of the model should be able to be approximated by straight lines at a distance to the corner allowing the application of details and there should be no less than three edges intersecting at each corner.

RQ3: After we apply the corresponding details to adjacent model edges, the algorithm produces the *buildable 3d model*, the so-called as-designed model. Form the point of view of the AEC industries, what degree of feasibility has this model?
ARQ3: We demonstrated that the feasibility of the model is given by construction, i.e., the feature collection assembly rules and the contraction rules subsequently defined on it guarantee that the result is a buildable 2-manifold with boundary.

RQ4: How much time is saved, on average, by applying our method as compared to the manual modeling of the as-designed model?
ARQ4: The evaluation we performed in Chapter 9 showed that, given a fully assembled feature collection, the time it takes the prototype to produce the corner solution is under 1 minute in all 170 test cases. In comparison, the manual attempts to build the corner solution by hand performed by the author invariably took a minimum of 4 hours, with some taking several days and producing an inferior result to the one achieved by the prototype. Therefore, we estimate that the method we presented in this work saves the designer form several hours to several days per corner solution.

CHAPTER 11

Conclusion and Future Work

This work contributes to user-guided domain-specific geometry generation. The presented method supports the creativity of the architect by enabling the automated transfer of construction documentation level detail onto the conceptual 3d model of earlier project phases.

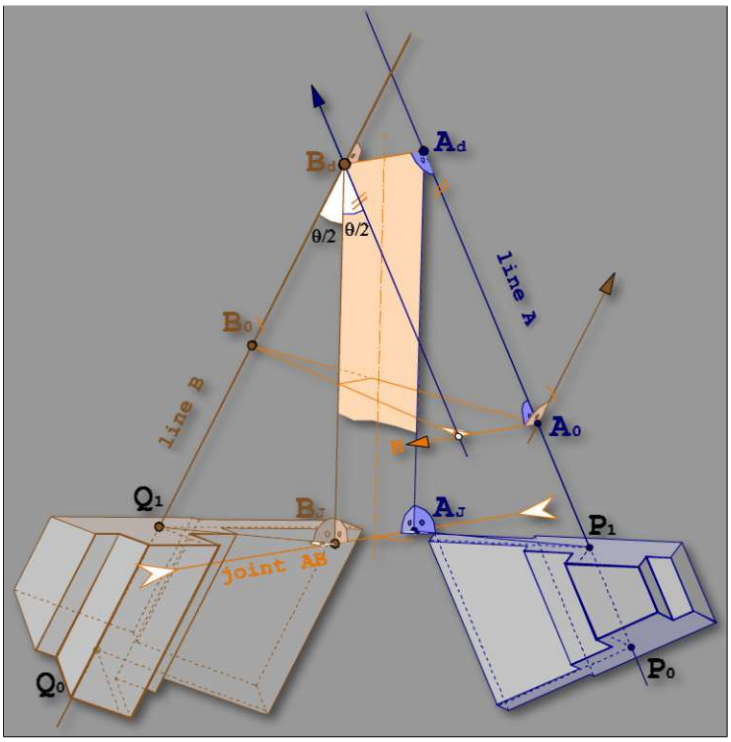


Figure 11.1: Intersecting sweep surfaces - initial configuration.

This allows the evaluation of multiple shape configurations without any modelling overhead by employing procedural modeling techniques where appropriate and thus enabling a true collaboration between a human creative agent and computer technology. Unlike most of the research in the area of geometry generation presented in Chapter 3, our method is not based on machine-learning techniques and does not require the user to formulate rules. Instead, it derives its components from the typical human-guided workflow in the domain.

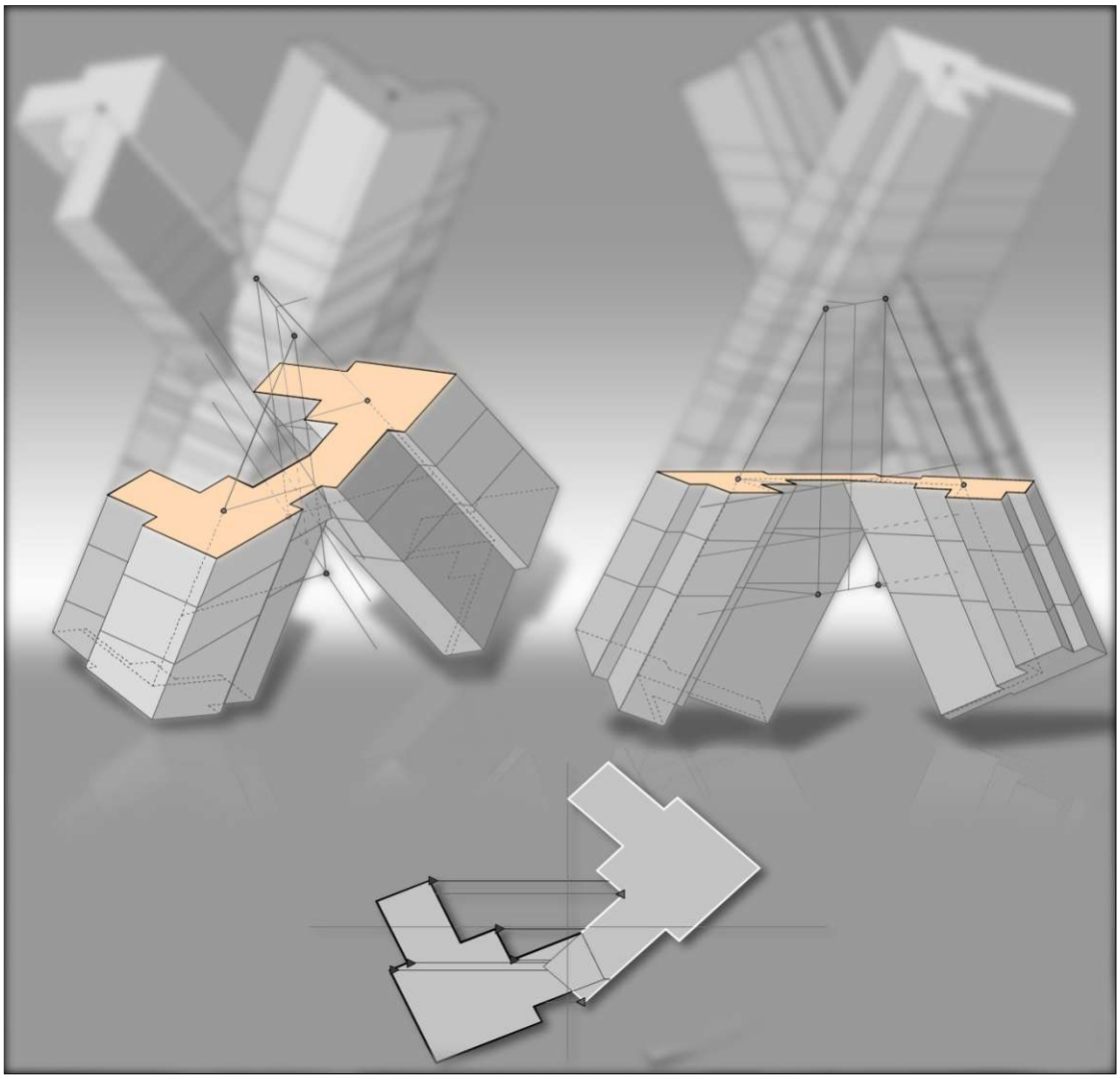


Figure 11.2: Intersecting sweep surfaces - an intermediate step.

However, in our future work, we intend to investigate the potential of neural networks to create detail libraries by supplying a training set of detail descriptions evaluated by domain experts.
The algorithm presented here is developed with the field of architecture in focus. However,

it can be applied in all areas where geometry needs to be intersected according to a set of user-defined rules. In Figure 11.1 and Figure 11.2 we have a pair of skew lines being used as guide curves for two different 2d shapes, in order to produce two intersecting sweep surfaces. The intersection process can be controlled through procedural shape contraction. The algorithm can be upgraded to support polylines, and subsequently splines, as guide curves for the contraction in general position. The set of constraints on the choice of features to contract, that here consists only of a minimal admissible length and a minimal admissible angle, can be expanded to include any other geometric characteristics.

List of Figures

List of Tables

Index

Glossary

3d building skeleton Information extracted from any 3d surface model: the edge curves, the dihedral angles between the tangent planes of the surfaces intersecting at each edge, and the corners where the edge curves intersect. 13, 14, 21–23, 31, 33, 35, 36, 91

B-rep A geometric representation of an object, using its limits or boundaries. In the case of a 3d solid, its B-rep consists of boundary surfaces. 2

base line The segment of the structural axes that lies in the plane of the segment composition. 40, 43, 47, 50, 51, 53, 54, 61, 63, 76, 77, 80, 83

BIM Building Information Modeling is both a data model for storing geometric data coupled with non-geometric information and a method for interactive building development involving multiple stakeholders with different fields of expertise. 1, 11

buildable 3d model A 3d surface model (and 2-manifold) that contains only geometry that can be built in the designated materials and using the appropriate building technologies. 5, 22, 31

conceptual 3d model An abstract and easy to edit 3d representation of an architectural concept during the conceptual design phase, consisting of (possibly parametric) curves and surfaces. 3, 4, 6, 11, 13, 17, 18, 22, 25, 27, 28, 31, 33, 35, 69, 91

corner skeleton Part of a 3d building skeleton: one corner point and a linear approximation of each of the edges intersecting in it. 36, 37, 40, 47, 54, 61, 69, 70, 72, 73, 77, 91

corner solution The 2-manifold enveloping a corner skeleton, resulting from the successful contraction of all feature collections at the corner, and the stitching together of the produced ruled surfaces. It is part of the buildable 3d model. 53–55, 63–65, 75

CSG Constructive Solid Geometry. A method for generating geometric shapes by applying Boolean operations to solid primitives, such as boxes or cylinders, or other CSG. 2

detail 2d architectural detail drawing that is incorporated in the construction documentation. It may be in scale ranging from 1:50 to 5:1, contains descriptions of materials and building techniques, and measurements. 4, 6, 10, 11, 13, 17, 18, 21–23, 25, 27, 28, 30, 31, 35, 40, 69, 72, 91

detail description Information extracted from a detail, containing a contour comprised of line segments coupled with their user-confirmed geometric properties, the geometric relationships of significance between them, and any user-supplied semantic information. 25, 27–31, 33, 36–39, 42, 49, 73, 74, 91, 92

edge composition The collection of all detail descriptions transferred to their respective edge of the corner skeleton. 37

feature A 1d line segment of a polyline with a rich descriptor, containing geometric relationship and topological information, contraction length, and a preservation score. It is part of a feature collection. 47, 61–63

feature collection Extracted from a detail: geometric features of interest, coupled with the geometric and adjacency relationships between them, as well as with semantic information. It is the subject of procedural contraction. 23, 25, 33, 43, 47, 48, 53, 54, 59, 61, 63–65, 73, 76, 77, 80, 92, 93

plus-energy building A building that, on average, generates more energy than it consumes. 2

procedural contraction Intersecting surfaces ruled by 1d shapes by discarding segments of those 1d shapes sequentially while managing adjacency and geometric relationships between them. 6, 11, 23, 28, 43, 48

segment composition The transformed edge composition: by projecting each edge-based detail description along its respective edge onto the two planes normal to the corner faces, incident with that edge, and intersecting the edge and its respective neighbor at the same distance from the top. 38, 39, 41–43, 92

segment polygon The boundary of the area enclosed by the feature collection, the base line and the joint lines. 48, 50, 51

structural axes the intersections of all surface tangent planes with the plane, normal to the edge segment of the 3d building skeleton and intersecting it at the site of the detail definition. 25, 33, 37, 73

Bibliography

[ADBW16] Daniel G. Aliaga, undefinedlke Demir, Bedrich Benes, and Michael Wand. Inverse procedural modeling of 3d models for virtual worlds. SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery.

[AKM+06] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal. Mesh segmentation - a comparative study. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 7–19, June 2006.

[ALM19] Gerasimos Arvanitis, Aris S. Lalos, and Konstantinos Moustakas. Saliency mapping for processing 3d meshes in industrial modeling applications. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 683–686, 2019.

[ARSF09] Marco Attene, Francesco Robbiano, Michela Spagnuolo, and Bianca Falcidieno. Characterization of 3d shape parts for semantic annotation. *Computer-Aided Design*, 41(10):756 – 763, 2009. Selected Papers from the 2007 New Advances in Shape Analysis and Geometric Modeling Workshop.

[ATC+08] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Trans. Graph.*, 27(3):1–10, aug 2008.

[BBL+17] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[BDK98] G. Barequet, C. A. Duncan, and S. Kumar. Rsvp: a geometric toolkit for controlled repair of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):162–177, April 1998.

[BK97] G. Barequet and S. Kumar. Repairing cad models. In *Proceedings. Visualization '97 (Cat. No. 97CB36155)*, pages 363–370, Oct 1997.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[Bui13] BuildingSMART. Ifc4 release, 2013.

[BVGP09] Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. Semantic deformation transfer. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 36:1–36:6, New York, NY, USA, 2009. ACM.

[BZA+16] Mohcine Bouksim, Fatima Rafii Zakani, Khadija Arhid, Taoufiq Gadi, and Mohamed Aboulfatah. Evaluation of 3d mesh segmentation using a weighted version of the ochiai index. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–7, 2016.

[CBK20] Jerome Charton, Stephen Baek, and Youngjun Kim. Mesh repairing using topology graphs. *Journal of Computational Design and Engineering*, 8(1):251–267, 12 2020.

[CC08] C. Chen and K. Cheng. A sharpness-dependent filter for recovering sharp features in repaired 3d mesh models. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):200–212, Jan 2008.

[CCZ+15] Jianjun Chen, Bingwan Cao, Yao Zheng, Lijun Xie, Chenfeng Li, and Zhoufang Xiao. Automatic surface repairing, defeaturing and meshing algorithms based on an extended b-rep. *Advances in Engineering Software*, 86:55–69, 2015.

[Cer11] Tomo Cerovsek. A review and outlook for a 'building information model' (BIM): A multi-standpoint framework for technological development. *Advanced Engineering Informatics*, 25(2):pp. 224–244, 2011.

[Cor01] C. A. Cory. Utilization of 2d, 3d, or 4d cad in construction communication documentation. In *Proceedings Fifth International Conference on Information Visualisation*, pages 219–224, July 2001.

[CTJL10] C. Chiang, K. Ting, B. Jong, and T. Lin. Global mesh optimization with automatic surface structure preservation. In *2010 International Computer Symposium (ICS2010)*, pages 396–401, Dec 2010.

[CTO+10] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su. Point cloud skeletons via laplacian based contraction. In *2010 Shape Modeling International Conference*, pages 187–197, June 2010.

[DM13] C. Dore and M. Murphy. Semi-automatic techniques for as-built bim façade modeling of historic buildings. In *2013 Digital Heritage International Congress (DigitalHeritage)*, volume 1, pages 473–480, Oct 2013.

[dSIR17] Wallas H. S. dos Santos, Paulo Ivson, and Alberto Barbosa Raposo. Cad shape grammar: Procedural generation for massive cad model. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 31–38, 2017.

[Dum17] Jérémie Dumas. *Controllable shape synthesis for digital fabrication*. Theses, Université de Lorraine, February 2017.

[DZXL09] F. Dong, R. Zhang, R. Xiao, and B. Lei. Mesh simplification with global contour feature preservation. In *2009 WRI World Congress on Computer Science and Information Engineering*, volume 2, pages 679–685, March 2009.

[EHSF16] Johannes Edelsbrunner, Sven Havemann, Alexei Sourin, and Dieter W. Fellner. Procedural modeling of round building geometry. In *2016 International Conference on Cyberworlds (CW)*, pages 81–88, 2016.

[EPE18] Sara Eloy, Pieter Pauwels, and Athanassios Economou. Ai edam special issue: advances in implemented shape grammars: solutions and applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 32(2):131–137, 2018.

[ESV98] J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):133–144, April 1998.

[FCSF17] Qiang Fu, Xiaowu Chen, Xiaoyu Su, and Hongbo Fu. Pose-inspired shape synthesis and functional hybrid. *IEEE Transactions on Visualization and Computer Graphics*, 23(12):2574–2585, 2017.

[FE20] Jonas Freiknecht and Wolfgang Effelsberg. Procedural generation of multistory buildings with interior. *IEEE Transactions on Games*, 12(3):323–336, 2020.

[FP14] Foteini Fotopoulou and Emmanouil Z. Psarakis. A visibility graph based shape decomposition technique. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 1, pages 515–522, 2014.

[FYK10] D. Fletcher, Y. Yue, and M. A. Kader. Challenges and perspectives of procedural modelling and effects. In *2010 14th International Conference Information Visualisation*, pages 543–550, July 2010.

[GE18] Thomas Grasl and Athanassios Economou. From shapes to topologies and back: an introduction to a general parametric shape grammar interpreter. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 32(2):208–224, 2018.

[GJWW14] Paul Guerrero, Stefan Jeschke, Michael Wimmer, and Peter Wonka. Edit propagation using geometric relationship functions. *ACM Trans. Graph.*, 33(2):15:1–15:15, April 2014.

[GSMCO09] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph.*, 28(3):33:1–33:10, July 2009.

[HE22] Tzu-Chieh Kurt Hong and Athanassios Economou. What shape grammars do that cad should: the 14 cases of shape embedding. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 36:e4, 2022.

[HKYM17] Haibin Huang, Evangelos Kalogerakis, Ersin Yumer, and Radomir Mech. Shape synthesis from sketches via procedural models and convolutional networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(8):2003–2013, 2017.

[HPW05] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. Smooth feature lines on surface meshes. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, 2005. Eurographics Association.

[HZK16] Meha Hachani, Azza Ouled Zaid, and Raoua Khliwi. Segmentation of 3d articulated meshes using shape diameter function and curvature information: Anonymous icme submission. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–5, 2016.

[IFPW10] Martin Ilčík, Stefan Fiedler, Werner Purgathofer, and Michael Wimmer. Procedural skeletons: Kinematic extensions to cga-shape grammars. In *Proceedings of the 26th Spring Conference on Computer Graphics*, SCCG '10, pages 157–164, New York, NY, USA, 2010. ACM.

[IMAW15] Martin Ilčík, Przemyslaw Musialski, Thomas Auzinger, and Michael Wimmer. Layer-based procedural design of façades. *Comput. Graph. Forum*, 34(2):205–216, May 2015.

[Jan95] Klaus P. Jantke. Introducing a two-level grammar concept for design. In *5. Theorietag, Automaten und Formale Sprachen*, pages 105–131, 1995.

[JBX+20] R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. *ACM Trans. Graph.*, 39(6), nov 2020.

[JTT+15] Caigui Jiang, Chengcheng Tang, Marko Tomičí, Johannes Wallner, and Helmut Pottmann. Interactive modeling of architectural freeform structures: Combining geometry with fabrication and statics. In Philippe Block, Jan Knippers, Niloy J. Mitra, and Wenping Wang, editors, *Advances in Architectural Geometry 2014*, pages 95–108, Cham, 2015. Springer International Publishing.

[KBB16] Nilanjana Karmakar, Arindam Biswas, and Partha Bhowmick. Reeb graph based segmentation of articulated components of 3d digital objects. *Theoretical Computer Science*, 624:25–40, 2016. Advances in Discrete Geometry for Computer Imagery.

[KCKK12] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, 31(4):55:1–55:11, Jul 2012.

[Kit04] B. Kitchenham. Procedure for performing systematic reviews. Technical Report TR/SE-0401, Computer Science Department, Keele University, Keele, Staffs ST5 5BG, UK, 07 2004.

[KLMK19] Javor Kalojanov, Isaak Lim, Niloy Mitra, and Leif Kobbelt. String-based synthesis of structured shapes. *Computer Graphics Forum*, 38(2):27–36, 2019.

[KMG+21] Vojtěch Krs, Radomír Měch, Mathieu Gaillard, Nathan Carr, and Bedrich Benes. Pico: Procedural iterative constrained optimizer for geometric modeling. *IEEE Transactions on Visualization and Computer Graphics*, 27(10):3968–3981, 2021.

[KST+09] P. Koutsourakis, L. Simon, O. Teboul, G. Tziritas, and N. Paragios. Single view reconstruction using shape grammars for urban environments. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1795–1802, Sept 2009.

[LHL20] Shu Liu, Jia-Li He, and Sheng-Hui Liao. Automatic detection of anatomical landmarks on geometric mesh data using deep semantic segmentation. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2020.

[LLZL21] Andrew Luo, Tianqin Li, Wen-Hao Zhang, and Tai Sing Lee. Surfgen: Adversarial 3d shape synthesis with explicit surface discriminators.

In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16218–16228, 2021.

[LMV01] J. Llados, E. Marti, and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1137–1143, Oct 2001.

[LS20] Xufeng Lang and Zhengxing Sun. Structure-aware shape correspondence network for 3d shape synthesis. *Computer Aided Geometric Design*, 79:101857, 2020.

[LSLCO05] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 479–487, New York, NY, USA, 2005. ACM.

[LWW08] Markus Lipp, Peter Wonka, and Michael Wimmer. Interactive visual editing of grammars for procedural architecture. In *SIGGRAPH'08: International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 2008 Papers 2008*, pages 1 – 10, 2008.

[LZH+07] Y. Lai, Q. Zhou, S. Hu, J. Wallner, and H. Pottmann. Robust feature classification and editing. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):34–45, Jan 2007.

[MBRM+16] Eva Millán, María-Victoria Belmonte, Manuela Ruiz-Montiel, Juan Gavilanes, and José-Luis Pérez-de-la Cruz. Bh-shade: A software tool that assists architecture students in the ill-structured task of housing design. *IEEE Transactions on Learning Technologies*, 9(3):244–257, 2016.

[MdlCRGRM20] Lawrence Mandow, José-Luis Pérez de-la Cruz, Ana Belén Rodríguez-Gavilán, and Manuela Ruiz-Montiel. Architectural planning with shape grammars and reinforcement learning: Habitability and energy efficiency. *Engineering Applications of Artificial Intelligence*, 96:103909, 2020.

[MG13] A. Martinovic and L. Van Gool. Bayesian grammar learning for inverse procedural modeling. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 201–208, June 2013.

[MM11] P. Merrell and D. Manocha. Model synthesis: A general procedural modeling algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):715–728, June 2011.

[MW09] Antonija Mitrovic and Amali Weerasinghe. Revisiting ill-definedness and the consequences for itss. In *Proceedings of the 2009 Conference*

on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling, page 375–382, NLD, 2009. IOS Press.

[NALM19] Stavros Nousias, Gerasimos Arvanitis, Aris S. Lalos, and Konstantinos Moustakas. Fast mesh denoising with data driven normal filtering using deep autoencoders. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 260–263, 2019.

[NT03] F. S. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):191–205, April 2003.

[OBS04] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.*, 23(3):609–612, Aug 2004.

[Oh19] Sahuck Oh. A new triangular mesh repairing method using a mesh distortion energy minimization-based mesh flattening method. *Advances in Engineering Software*, 131:48–59, 2019.

[OS88] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[OS09] Mine Özkar and George Stiny. Shape grammars. In *ACM SIGGRAPH 2009 Courses*, SIGGRAPH '09, pages 22:1–22:176, New York, NY, USA, 2009. ACM.

[OT06] Aude Oliva and Antonio Torralba. Chapter 2 building the gist of a scene: the role of global image features in recognition. In S. Martinez-Conde, S.L. Macknik, L.M. Martinez, J.-M. Alonso, and P.U. Tse, editors, *Visual Perception*, volume 155 of *Progress in Brain Research*, pages 23–36. Elsevier, 2006.

[Pat12] G. Patow. User-friendly graph editing for procedural modeling of buildings. *IEEE Computer Graphics and Applications*, 32(2):66–75, March 2012.

[PLH+05] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, and W. Wang. Industrial geometry: recent advances and applications in cad. *Computer-Aided Design*, 37(7):751 – 766, 2005.

[PMAS11] P. Pauwels, R. D. Meyer, M. Audenaert, and K. Samyn. The role of game rules in architectural design environments. In *2011 Third International Conference on Games and Virtual Worlds for Serious Applications*, pages 184–185, May 2011.

[Pot08] Helmut Pottmann. Geometry of architectural freeform structures. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, SPM '08, pages 9–9, New York, NY, USA, 2008. ACM.

[QYSG17] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NIPS'17, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc.

[RH09] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, 14(2):131–164, apr 2009.

[RM10] Christof Riccabona and Karl Mezera. *Baukonstruktionslehre: Rohbauarbeiten.* Manz Verlag Schulbuch, Wien, 2010.

[SAG+13] Alex Shtof, Alexander Agathos, Yotam Gingold, Ariel Shamir, and Daniel Cohen-Or. Geosemantic snapping for sketch-based modeling. *Computer Graphics Forum*, 32(2):245–253, 2013. Proceedings of Eurographics 2013.

[SCOIT05] O. Sorkine, D. Cohen-Or, D. Irony, and S. Toledo. Geometry-aware bases for shape approximation. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):171–180, 2005.

[SE09] Christopher Summerfield and Tobias Egner. Expectation (and attention) in visual cognition. *Trends in cognitive sciences*, 13(9):403–409, 2009.

[SGL+18] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5515–5523, 2018.

[SGY+21] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6087–6101, 2021.

[Sha08] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[SS22] Çağlar Seylan and Yusuf Sahillioğlu. 3d shape deformation using stick figures. *Computer-Aided Design*, 151:103352, 2022.

[SSCO08] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24:249–259, 2008.

[SVS18] Kripasindhu Sarkar, Kiran Varanasi, and Didier Stricker. 3d shape processing by convolutional denoising autoencoders on local patches. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1925–1934, 2018.

[TBOW21] Claudio Tortorici, Stefano Berretti, Ahmad Obeid, and Naoufel Werghi. Convolution operations for relief-pattern retrieval, segmentation and classification on mesh manifolds. *Pattern Recognition Letters*, 142:32–38, 2021.

[TMW02] R. F. Tobler, S. Maierhofer, and A. Wilkie. A multiresolution mesh generation approach for procedural definition of complex geometry. In *Proceedings SMI. Shape Modeling International 2002*, pages 35–271, May 2002.

[TPT17] Panagiotis Theologou, Ioannis Pratikakis, and Theoharis Theoharis. Unsupervised spectral mesh segmentation driven by heterogeneous graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):397–410, 2017.

[TYPC20] Weihua Tong, Xiankang Yang, Maodong Pan, and Falai Chen. Spectral mesh segmentation via ℓ_0 gradient minimization. *IEEE Transactions on Visualization and Computer Graphics*, 26(4):1807–1820, 2020.

[WGJC18] Ning Wei, Kaiyuan Gao, Rongrong Ji, and Peng Chen. Surface saliency detection based on curvature co-occurrence histograms. *IEEE Access*, 6:54536–54541, 2018.

[WW93] L. Weitzman and K. Wittenburg. Relational grammars for interactive design. In *Proceedings 1993 IEEE Symposium on Visual Languages*, pages 4–11, Aug 1993.

[WWSR03] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 669–677, New York, NY, USA, 2003. ACM.

[WZY+10] Yamei Wen, Hui Zhang, Zhongmian Yu, Jiaguang Sun, and Jean-Claude Paul. Reconstructing 3d objects from 2d sectional views of engineering drawings using volume-based method. In *2010 Shape Modeling International Conference*, pages 13–24, 2010.

[XABR21] Xiao Xiao, Pierre Alliez, Laurent Busé, and L. Rineau. Delaunay meshing and repairing of NURBS models. *Comput. Graph. Forum*, 40(5):125–142, 2021.

[XC09] Yaoda Xu and Marvin M. Chun. Selecting and perceiving multiple visual objects. *Trends in cognitive sciences*, 13(4):167–174, 2009.

[YW18] Jianbin Yang and Cong Wang. A developed fuzzy c-means algorithm for mesh segmentation. In *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5, 2018.

[YWR09] X. Yin, P. Wonka, and A. Razdan. Generating 3d building models from architectural drawings: A survey. *IEEE Computer Graphics and Applications*, 29(1):20–30, Jan 2009.

[ZFCO+11] Youyi Zheng, Hongbo Fu, Daniel Cohen-Or, Oscar Kin-Chung Au, and Chiew-Lan Tai. Component-wise controllers for structure-preserving shape manipulation. *Computer Graphics Forum*, 30(2):563–572, 2011.

[ZGX+22] Long Zhang, Jianwei Guo, Jun Xiao, Xiaopeng Zhang, and Dong-Ming Yan. Blending surface segmentation and editing for 3d models. *IEEE Transactions on Visualization and Computer Graphics*, 28(8):2879–2894, 2022.

[ZXK+19] Yiran Zhu, Shu Xu, Jiaqi Kang, Yanping Xue, Chenle Lv, Dan Zhang, Xingce Wang, and Zhongke Wu. Skeleton-based 3d model descriptor and its application in non-rigid shape retrieval. In *2019 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 50–58, 2019.