Proceedings of the IASS Annual Symposium 2016
"Spatial Structures in the 21st Century"
26–30 September, 2016, Tokyo, Japan
K. Kawaguchi, M. Ohsaki, T. Takeuchi (eds.)

Data management and modelling of complex interfaces in imperfect discrete-element assemblies

Ursula FRICK*, Tom VAN MELE[a], Philippe BLOCK[a]

**ETH Zurich, Institute of Technology in Architecture, Block Research Group
Stefano-Franscini-Platz 5, HIL H 46.2, 8093 Zurich, Switzerland
frick@arch.ethz.ch

[a] ETH Zurich, Institute of Technology in Architecture, Block Research Group

Abstract

Recent advances in digital fabrication and on-site construction have created new possibilities for the (pre-) fabrication of highly customised building components and the realisation of complex, assembled architectural geometries. However, currently there are no structural design methods or tools available that address the specific challenges related to the design and analysis of discrete assemblies. This paper describes an approach for the identification of contact interfaces between the elements of discrete assemblies and presents a data structure for these interactions. Furthermore, the paper presents a method for handling misaligned (e.g. intersecting), and non-coplanar interfaces. The proposed approaches are suitable as computational back-end for equilibrium analysis of discrete-element assemblies, such as the Rigid Block Limit Equilibrium method.

Keywords: Discrete-element modelling, as built geometry, imperfections, interface detection, data management, computational limit equilibrium analysis.

1. Introduction

Discrete-element assemblies are structures formed by individual units. They appear in architectural construction with a wide range of materials and unit sizes. For instance, in masonry, the units can be relatively small, whereas in assemblies of prefabricated housing units, the units are quite large. The presented work is part of a larger research project that is concerned with the structural design of assemblies from given units, and the discretisation of given geometries into structurally feasible assemblies (Frick *et al.* [1]).

This paper presents parts of the digital design framework concerned with imperfections of the assembly geometry and the identification of the resulting interfaces. It also presents an approach for the data management of assemblies that enhances the workflow in a parametric design environment.

1.1. Background and problem statement

In the last few decades, advances in computer-aided design and digital fabrication have facilitated the realisation of new geometries for discrete-element assemblies. Highly customised components can be produced through novel manufacturing processes, such as 5-axis CNC milling (e.g. Rippmann et al. [5]) and large-scale 3d-printing (e.g. Hansemayer et al. [2]). Therefore, the units are no longer limited to standardised or cubical geometries. This opens up new design possibilities, but also creates several

challenges, such as the detection of interfaces between units with non-quadrilateral and non-planar faces.

Additionally, fabrication and assembly tolerances often lead to discrepancies between a digital model and the corresponding physical model. This can lead to incorrect digital simulations of, for example, collapse of masonry vault, as discussed in Van Mele *et al*. [9]. The study suggests that digital models of the as-built geometry could be used to make the simulations more accurate, for example, by mapping the digital units to their measured locations in the physical assembly. However, due to fabrication tolerances and imprecise measurements, these reconstructed digital models often contain overlaps or gaps between interacting units. These interfaces have to be detected properly to use the models for as-built equilibrium calculations.

1.2. Outline and Contributions

The presented research thus focuses on the development of methods for the detection of interfaces of imperfect assemblies. Blocks with non-planar and non-quadrilateral faces, and configurations with penetrating and non-touching block contacts are addressed. Figure 1 illustrates a simple example of such an imperfect configuration.

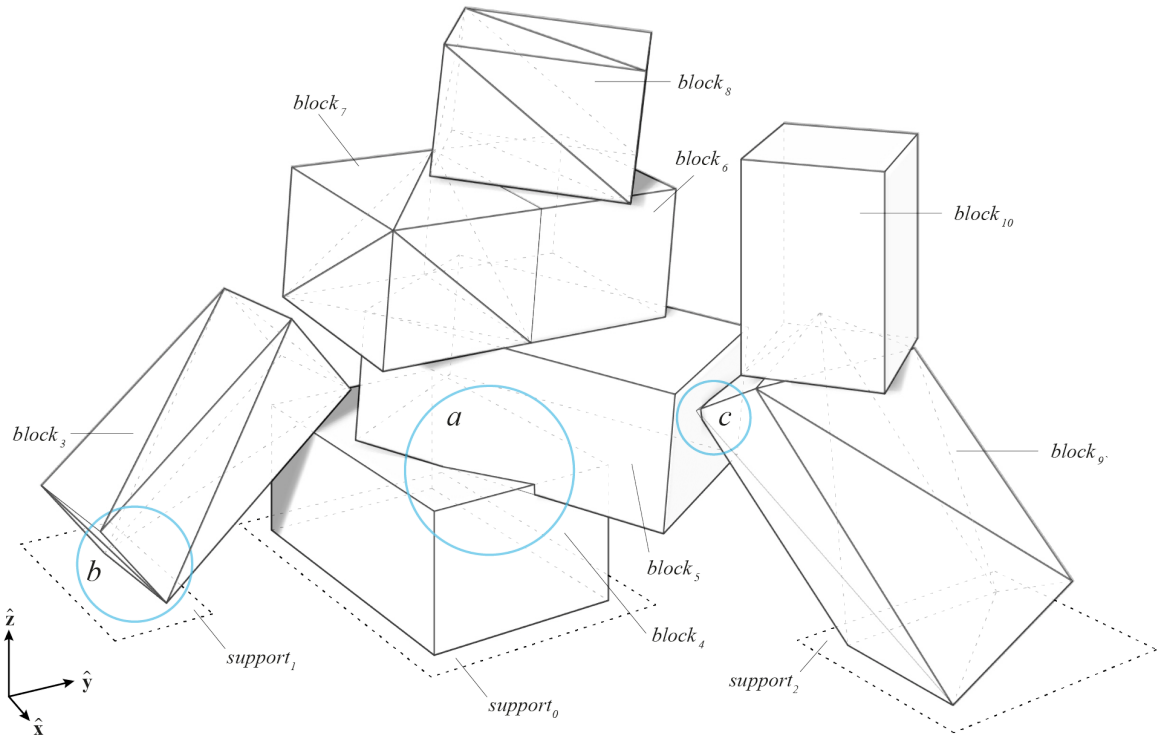

Figure 1: Imperfect block assembly with misaligned face-face (a), face-edge (b), and face-vertex(c) contacts.

Section 2 describes the data structure that will be used to manage the relationships between the components of an assembly. In Section 3, the detection algorithms are described for different interface types. In Section 4, the algorithms are applied to the measured as-built geometry of a small-scale model of a cross vault. The results for different values of detection tolerance are discussed. Finally, in Section 5, we provide an outlook on further research.

2. The block model

In the presented approach, an assembly of blocks, supports, and their interfaces (Figure 2) is represented by a directed graph *G(V,E)* with *V* vertices and *E* edges (Figure 3). This graph is called the block model. Similar approaches, but with undirected graphs, were used to describe 3D models composed of intersecting, planar pieces (Schwartzburg *et al.* [6]) and the connectivity of LEGO brick sculptures (Testuz *et al.* [8]).

A vertex in the graph represents a component of the assembly. Currently, only blocks and supports are considered. An edge represents an interface 1) between a block and a support, 2) between two individual blocks, or 3) between two blocks of a compound block. A compound block exists out of multiple blocks that are joined through "internal" interfaces and structurally act like one unit. The directions of the edges are defined by the orientation of the local interface frames (*uvn*-coordinate systems) and support units are marked as 'fixed' vertices. The block model is used as the computational back-end for equilibrium analysis using the Rigid Block Limit Equilibrium method (Livesley [3] and Whiting [10]). The method is based on static equilibrium equations that represent force and moment interactions between each 'free' block and the interfaces of adjacent blocks. Note that during these equilibrium calculations, positive force values represent compression and negative values tension for blocks corresponding to "start" vertices. For blocks corresponding to "end" vertices, positive force values are tension and negative compression.

All relevant interface data, such as the geometry of the interface, the local frame, and general interface attributes (e.g. material properties like friction coefficients) are stored in attribute dictionaries on the corresponding edge in the block model. After equilibrium analysis, data about the force transfers at the interfaces are stored on the edges as well.

Note that in this study, only convex block geometries are considered. Non-convex blocks can be created with compound blocks, existing out of multiple convex blocks with the edges identifying the interfaces between those blocks marked as "internal". For example, blocks 6 and 7 in Figure 3 could form a compound block, with edge e(6,7) marking the internal interface. Limiting the block geometries to convex geometries simplifies the interface detection. For convex blocks, it is impossible that a line or point-interface exists if a face-interface is already established. This does not exclude that multiple contacts of the same 'type' could exist. An example of such multiple contacts between two blocks can be seen in Figure 3 between block 6 and block 8 (face-interface) or between block 3 and block 4 (line-interface). For instance, all blocks have convex geometries, but the faces of block 8 are triangulated. Therefore, there are two faces of block 8 in contact with block 6. In the block model, this is addressed by storing both contact geometries on e(6,8).

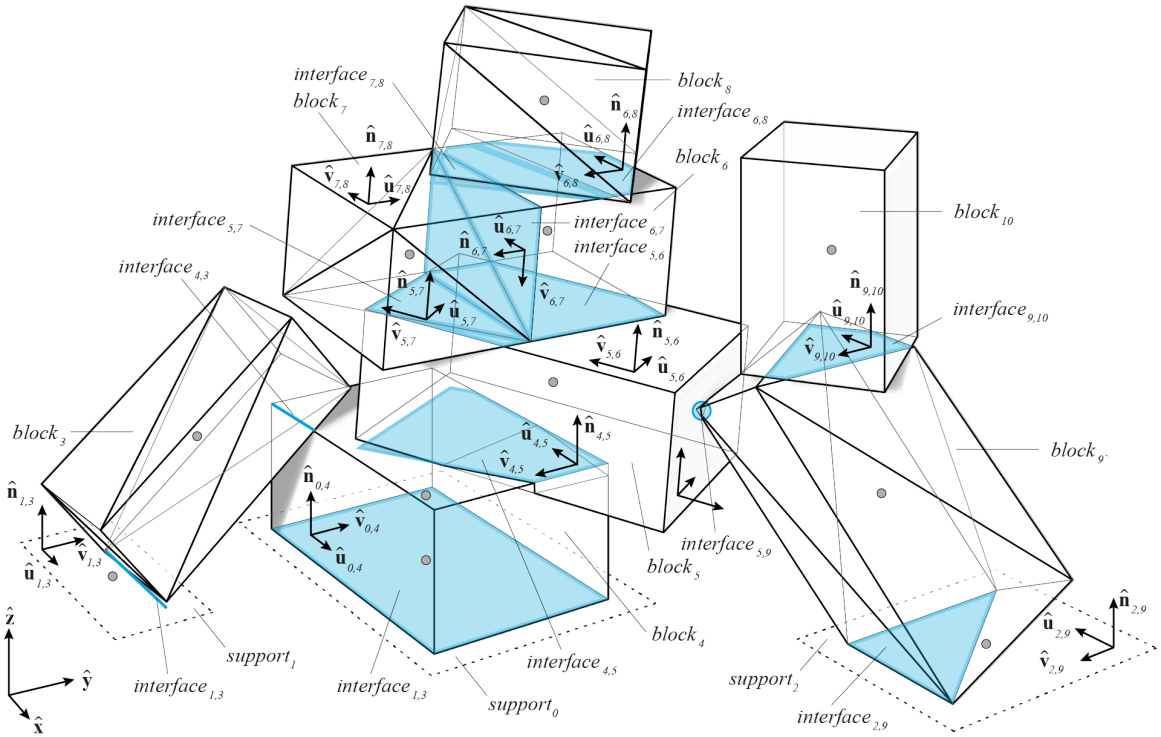

Figure 2: Diagram of an imperfect block assembly illustrating supports, blocks, and the resulting interfaces.

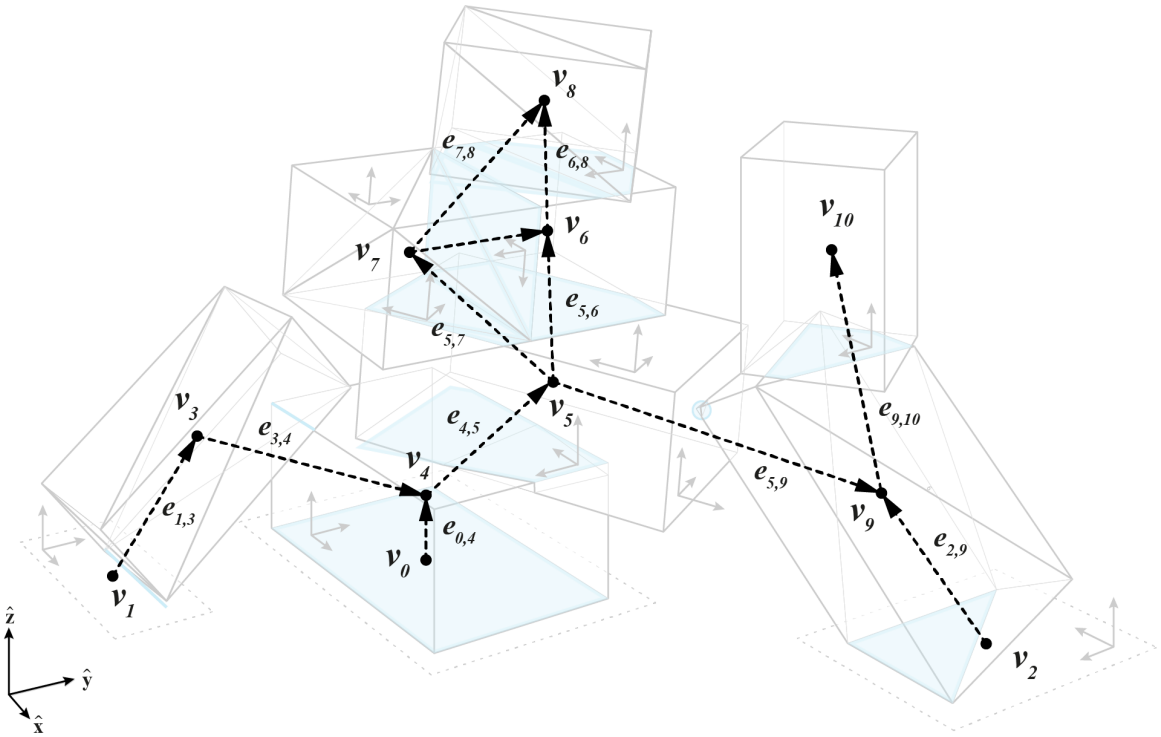

Figure 3: Diagram of the corresponding block-model illustrating the graph *G(V,E)* with *V* vertices and *E* edges.

3. Interface detection

For each block of an assembly, the interfaces with other blocks are detected face per face, treating each face in turn as the *base face*. Three contact types are considered: face-face, face-edge, and face-vertex contacts. The computation of all three contact types is based on the generation of a local *uvn*-coordinate system or *local frame* of the base face. To identify interfaces with a neighbouring block, the vertex coordinates of the block are transformed from global *xyz*-coordinates to local *uvn*-coordinates, with *u* and *v* the local in-plane coordinates and *n* the local out-of-plane coordinate. By comparing the coordinates along the *n*-axis to a user-defined tolerance, contact between blocks can be detected. If a contact is found, an edge is added to the block model, and the properties of the contact interface are stored on the edge.

Note that for every block, the detection of interfaces is limited to those blocks of the assembly that are actually close enough to be neighbours. For fast lookup of nearest neighbours of a block, a kd-tree [7] is created using the locations of the blocks represented by their centre points.

3.1. Face-face contact - face interface

To identify face-face contacts, all *n*-coordinates of the face vertices of a neighbouring block are compared to the user defined tolerance *t* (Figure 4). If all *n*-coordinates of a face of the neighbour are smaller than *t*, this face is considered coplanar with the base face. A polygon of the neighbour's face is then constructed in the 2D space of the local frame. If this polygon intersects with the polygon of the base face, and the area of the intersection is bigger than a user defined minimum area, an edge is added to the block model and the relevant data of the face-interface are stored on the edge.

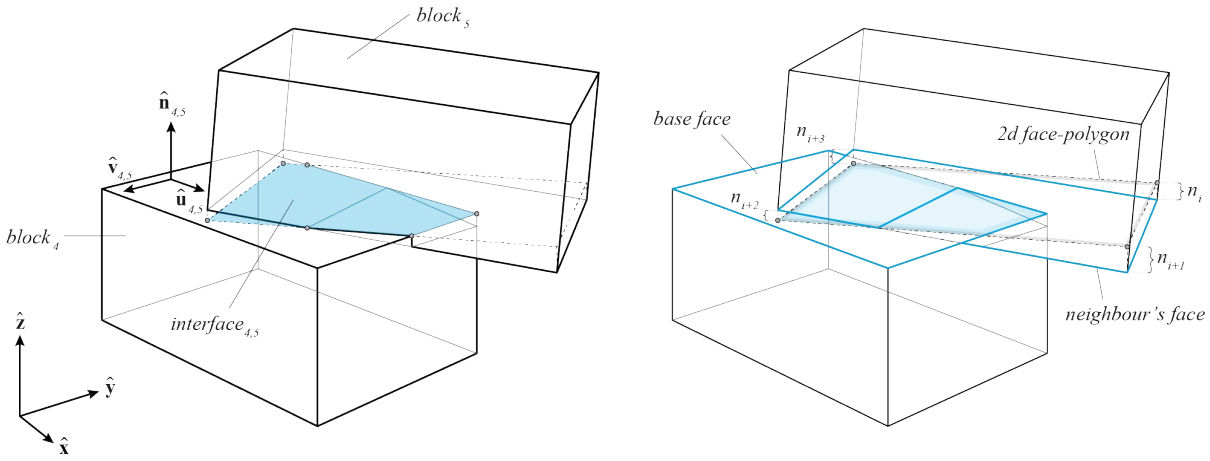

Figure 4: Diagram of a misaligned face-face contact resulting in a face-interface.

3.2. Face-edge contact - line interface

In case of a possible face-edge contact, all *n* values of the neighbour's edge vertices are compared to the tolerance *t* (Figure 5). If all *n* values are smaller than *t*, the neighbour's edge lies in the plane of the base face. From the neighbour's edge, a line (*edge-line*) is constructed in the 2D space of the local frame. The next step is the check for an intersection between the face-polygon of the block and the edge-line of the neighbour. If an intersection can be found and the length of the resulting line-interface

is bigger than a user defined length-tolerance, an edge is added to the block model and the data of the line-interface are stored on the edge.

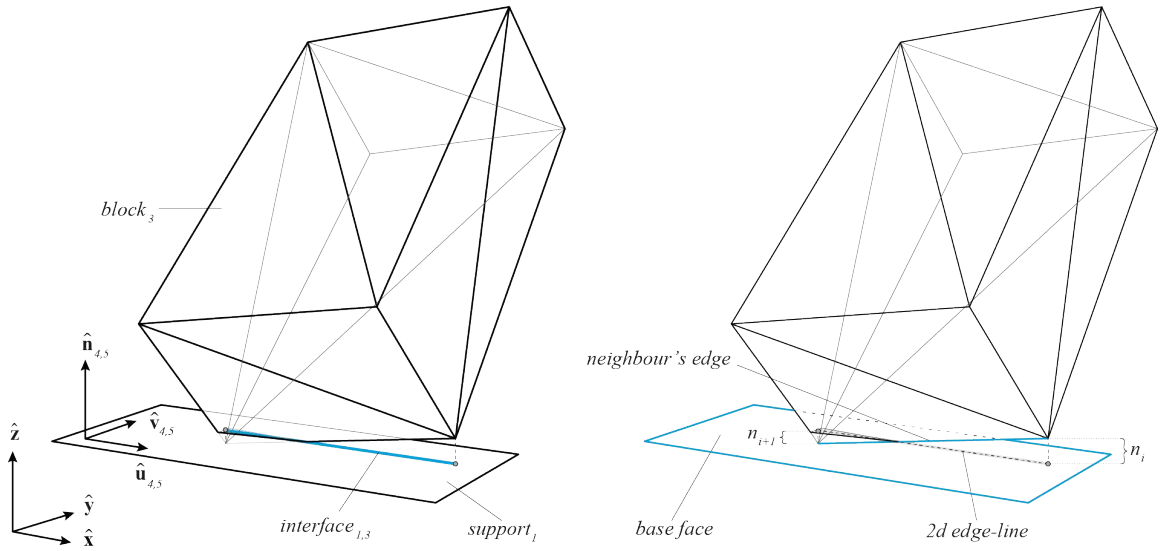

Figure 5: Diagram of a misaligned face-edge contact resulting in a line-interface.

3.3. Face-vertex contact - point interface

In case of a possible face-vertex contact, the *n* value of the neighbour's vertex is compared to the user defined tolerance value *t* (Figure 6). If *n* is smaller than t, the neighbour's vertex lies in the same plane. From the neighbour's vertex, a 2D vertex is constructed in the space of the local frame. If the 2D vertex lies inside the polygon of the base face, an edge is added to the block model and the data of the point-interface are stored on the edge. Note that once a point-interface is detected no further search for other point contacts is necessary, since no additional point-interface can exist for convex geometries.

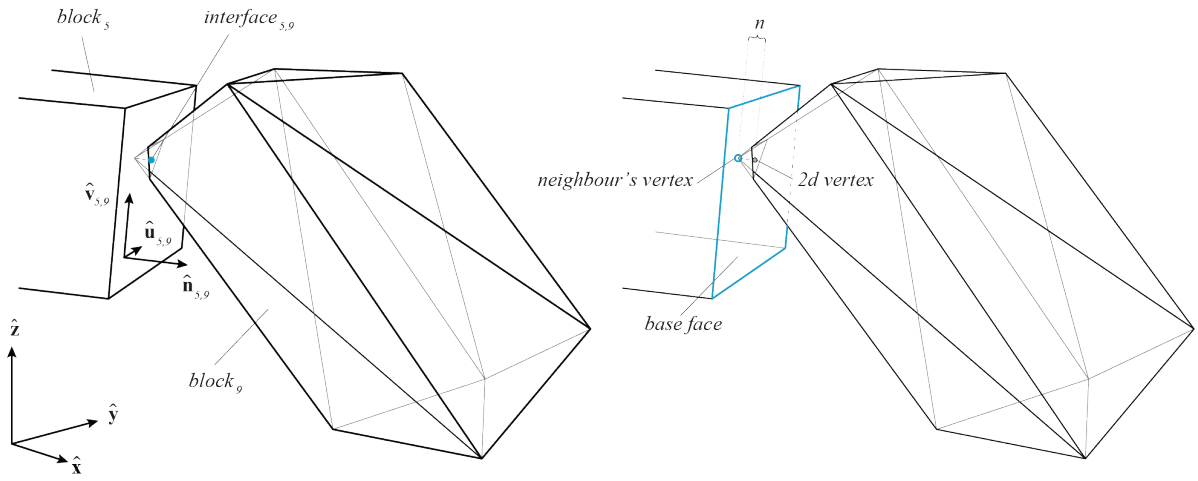

Figure 6: Diagram of a misaligned face-vertex contact resulting in a point-interface.

4. Results

This section presents the results of the algorithm applied to a digital model that was constructed by mapping digital block geometries to their measured locations in a 3D-printed cross vault model (Figure 7, left). For detailed information about the measuring system, see Van Mele *et al.* [9]. Note that only the position of the blocks was changed. The geometry and size of the blocks were kept as digitally designed for fabrication.

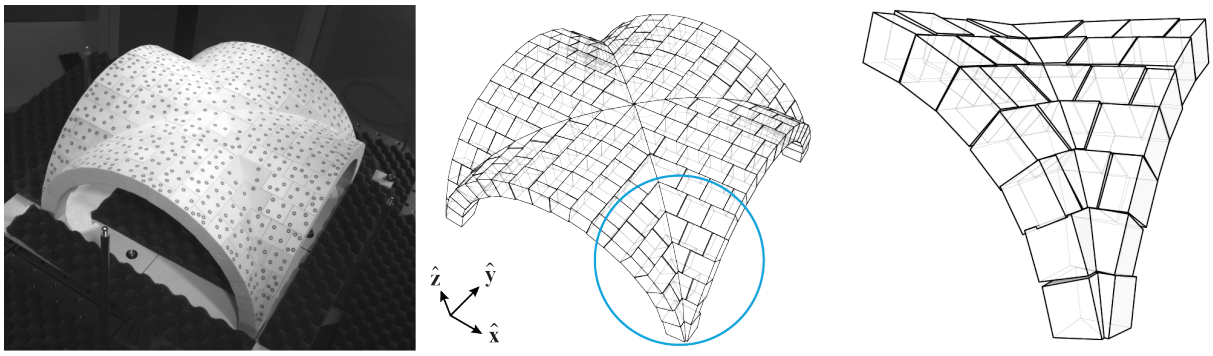

Figure 7: Photograph of the cross vault model (left), the diagrams of the digital model (middle), and a close up of the digital model (right).

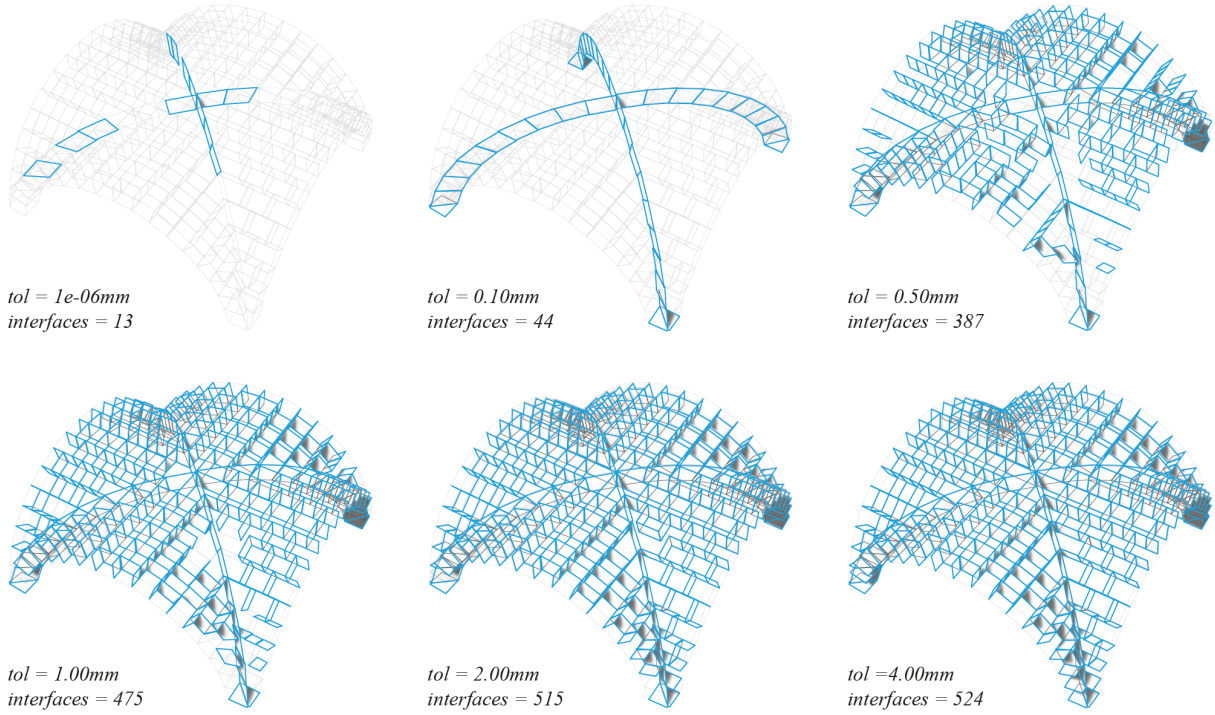

Figure 8: Results of the interface identification for various tolerance values from 1e-06mm to 4.00mm.

In the reconstructed digital model all blocks are slightly misaligned compared to their neighbours (Figure 7, middle and right). The model is composed of 180 blocks (144 convex blocks, 36 compound

blocks on the ribs) with edge lengths from 2.15 to 55.37mm. Different tolerances for coplanarity were used to illustrate their effect on the identification of interfaces. In all cases, the area tolerance was 50mm^2. Line and point-contacts were not considered. This is based on the assumption that in this initial configuration only face contacts exist in the physical model.

The results of the interface identification are illustrated in Figure 8. For a coplanarity tolerance value of 1e-06mm (the default numerical tolerance) only 13 face-interfaces were recognised (Figure 8, top, left). For the tolerance value of 0.10mm, the internal interfaces of all compound blocks and the compound blocks and the supports were detected. For a tolerance value of 4.00m, all 524 considerable face-interfaces were identified. In the original 'perfect' model, 524 face-interfaces were recognised with a tolerance value of 0.10mm.

5. Conclusion and Outlook

This study presented a feasible approach to address imperfections in discrete models and to incorporate them into the digital modelling process. Further investigation is needed to understand the influence of those imperfections on the structural model and the structural analysis. A possible approach might be the comparison of digital simulations with the structural behaviour of the physical models.

To improve readability of the results, different visualisation modes will be implemented. Tolerance values of coplanarity might be visualised with different colours for penetration and offset, the magnitude could be visualised with saturation. Assembly units will be checked for convexity and planarity before the interface identification. Faces will automatically be split into planar parts if the user defined tolerance for planarity is exceeded. Non-convex components will be convexified by discretizing them into convex parts.

The developed methods and libraries will be shared as a standalone, open-source Python [4] library. The research is part of the interdisciplinary research within the Swiss National Centre of Competence in Research (NCCR) Digital Fabrication.

Acknowledgements

The authors would like to thank Cristián Calvo Barentin for providing the digital cross vault model, which is part of his PhD research at the BRG.

The presented research was supported by the NCCR Digital Fabrication, funded by the Swiss National Science Foundation (NCCR Digital Fabrication Agreement # 51NF40-141853).

References

[1] Frick U, Van Mele T and Block P. Decomposing three-dimensional shapes into self-supporting discrete-element assemblies. *Proceedings of the Design Modelling Symposium,* Springer International Publishing Copenhagen, 2015, 187-201.

[2] Hansmeyer M and Dillenburger B. Digital Grotesque – Towards a Micro-Tectonic Architecture. *SAJ Serbian Architectural Journal: Architectural education in the post-digital age* 2013, 5-2013-02: 194-201.

[3] Livesley, R.K. A computational model for the limit analysis of three-dimensional masonry structures. *Meccanica* 1992, 27(3):161–172.

[4] Python Software Foundation. Python 2.7. *https://docs.python.org/2.7.* Last visit: 01. June 2016

[5] Rippmann M, Curry J, Escobedo D, and Block P. Optimising Stone-Cutting Strategies for Freeform Masonry Vaults. *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium* 2013

[6] Schwartzburg Y and Pauly M. Fabrication-aware Design with Intersecting Planar Pieces. *Computer Graphics Forum* 2013, 32:317-326

[7] SciPy-cKDTree. Last visit: 01. June 2016 *http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.spatial.cKDTree.html*

[8] Testuz R, Schwartzburg Y, and Pauly M. Automatic Generation of Constructable Brick Sculptures. *Eurographics* 2013, 81-84

[9] Van Mele T, McInerney J, DeJong M and Block P. Physical and Computational Discrete Modeling of Masonry Vault Collapse, *Proceedings of the 8th International Conference on Structural Analysis of Historical Constructions*, Wroclaw, Poland, 2012, 2552-2560.

[10] Whiting E. Design Of Structurally-Sound Masonry Buildings Using 3D Static Analysis. *PhD thesis,* Massachusetts Institute of Technology, 2011.