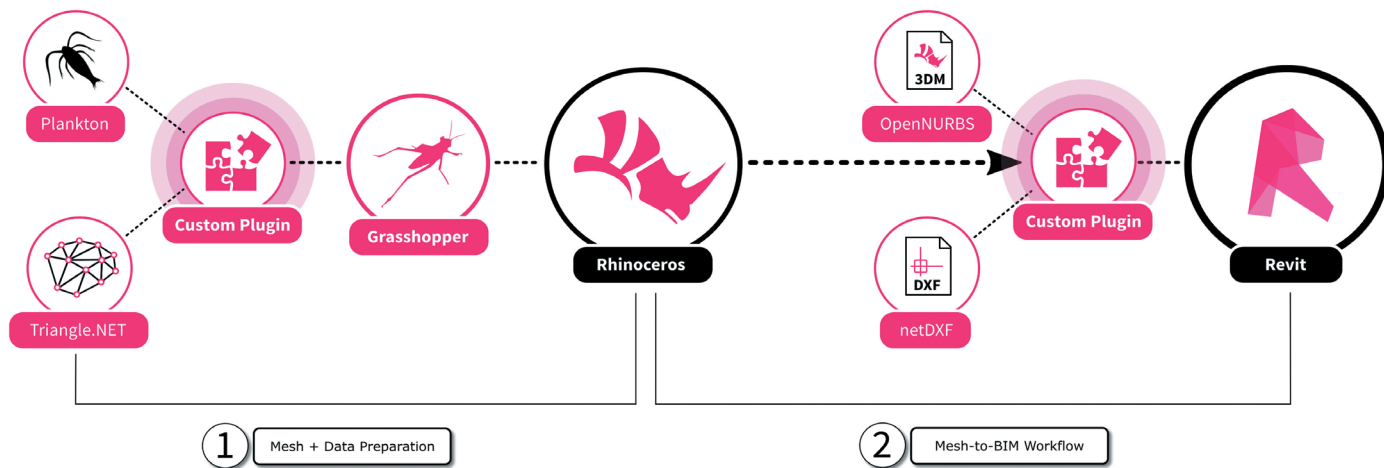


A Novel Mesh-Based Workflow for Complex Geometry in BIM

Nathan Miller
Proving Ground
David Stasiuk
Proving Ground



1

ABSTRACT

Various well-established digital modeling software platforms enable architectural design teams to rapidly sculpt and iterate over complex, doubly curved, and organic geometries. However, the software platforms that are used to author such geometries are rarely the same that are used for later-stage project development and delivery. For these phases of project execution, projects of even modest complexity are managed through building information modeling (BIM) software. Yet most BIM solutions are not suitable for natively handling the design of geometrically complex forms, failing to provide lightweight, responsive, or flexible authoring interfaces. A further complication is their inability to readily import or integrate any complex geometric elements or assemblies generated elsewhere. The development of improvements to interoperability between authoring and production software therefore remains an important goal in contemporary architectural practice.

This paper describes a practical methodology that then engages various Application Programming Interfaces (APIs) and open-source programming tools to address the problem of interoperability for complex geometry in BIM. Specifically, it identifies meshes as a well-positioned data structure for use within the context of preparing complex design geometry for BIM production. We describe a novel technique for the efficient interoperability of complex NURBS poly-surface objects from one authoring platform, employing design meshes that cleanly capture not just geometry, but also user and procedurally derived descriptive data elements for advanced representation and analysis within a BIM production environment.

- 1 The Mesh-to-BIM workflow utilizing open source libraries, platform APIs, and custom software development.

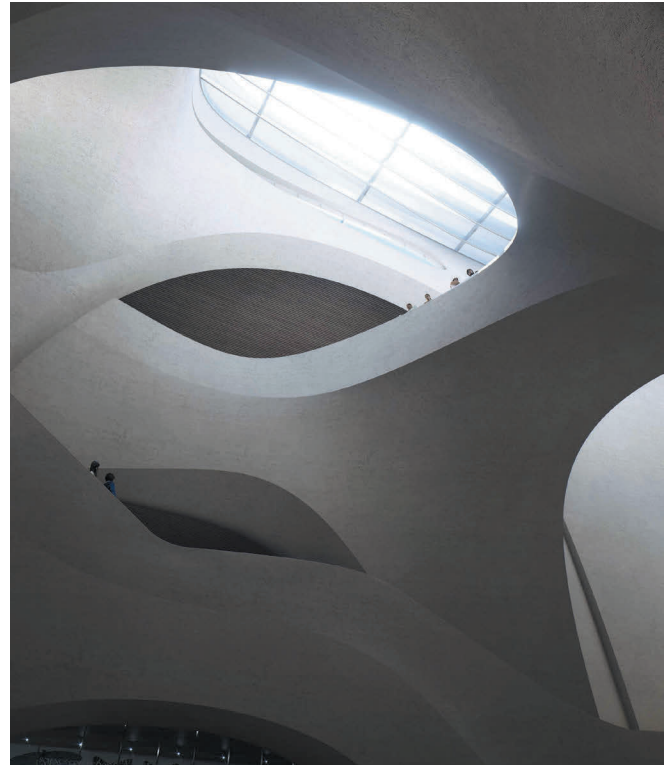
INTRODUCTION

Beyond their fundamental role in geometric representations for 3D computer graphics, polygonal mesh data structures are critical for many digitally led architectural design modeling approaches. Mesh data structures are typically defined by an arbitrary set of points, with associated edge-connectivity and surface-face tables organizing them into a surface representation. Meshes are then fundamentally discretized data structures whose capacity for geometric description and metadata association is both flexible and resolution-dependent.

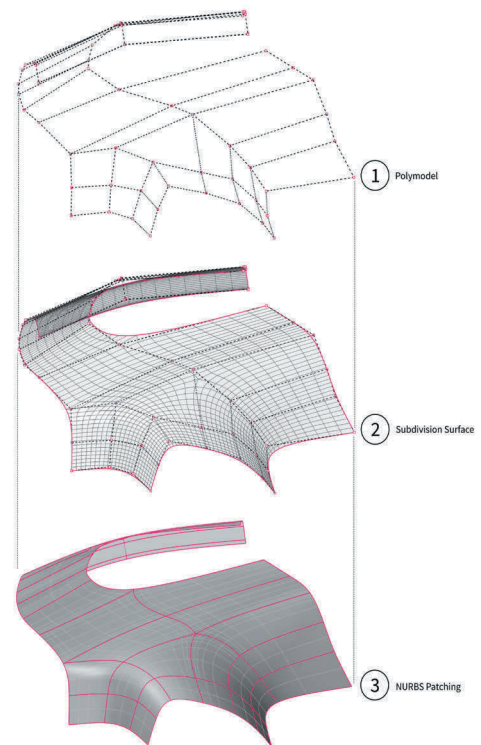
One common application of design meshes is in authoring sculptural or non-regular geometry, where designers rapidly develop and iterate complex, doubly curved, and/or organic geometries. Specialized software platforms enable the direct manipulation of comparatively coarse geometric cages that are then algorithmically refined into more fluid forms, such as through Catmull-Clark or Loop subdivision smoothing.

Alternatively, meshes are essential for many computational analysis techniques applicable to data-driven building design approaches. These approaches are traditionally more evaluative than generative and are used to verify assumptions after the design process is mostly complete rather than as dynamic, real-time informers of early-stage design decisions. However, through advances in computational design software within a continuously maturing open-source culture, computational designers have recently enjoyed increased access to many of these techniques during the generative stages of design, such as for understanding structural behavior through finite element analyses or solar behavior through insolation and daylighting simulations. This feedback empowers architects to better integrate performance-related concerns into the design process and also to effectively engage with engineers.

Meanwhile, most contemporary building information modeling (BIM) authoring and production software packages are managed through constructive solid geometry (CSG) modeling techniques. This framework has evolved over thirty years from early BIM authoring tools such as RUCAPS, TriCad, Calma, and GDS, and is also consistent with contemporary feature-based solid modeling software packages, like Solidworks and Inventor (Weisberg 2008). This is related to the prioritization of decision trees and boolean operations as key drivers in geometry authoring, and in representing and resolving the relationships between multiple, interdependent building systems in the same model, such as for windows or doors in walls (Eastman et al. 2011). Contemporary authoring platforms such as Revit frequently will also employ NURBS modeling techniques for expressing surface curvature, typically transforming these surfaces into solids when placing



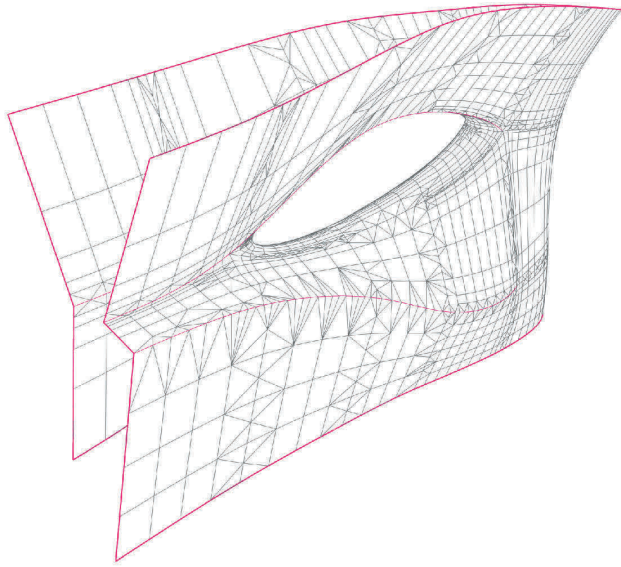
2



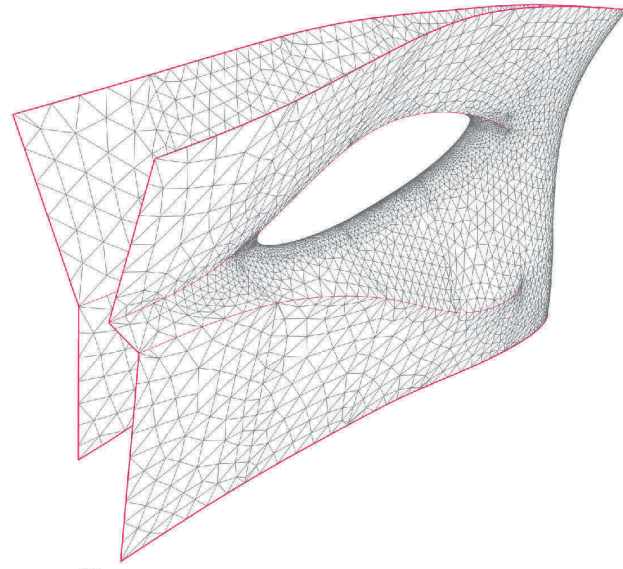
3

2 The design of the American Museum of Natural History Gilder Center by Studio Gang Architects utilized a form of mesh modeling with T-Splines.

3 Diagram of a design process that utilizes coarse mesh wireframes to NURBS object through subdivision modeling.



1 Rhino Render Mesh



2 Curvature Adaptive Mesh

4

assemblies or design systems into models. While meshes may be understood as discrete, resolution-specific data structures, NURBS and solid-models are then continuous, mathematical, and resolution independent.

Interestingly, for model viewing in digital displays, NURBS and solid models are invariably translated into preview meshes within BIM production software. Yet despite this internal process of managing multiple geometric representations in parallel, the explicit creation, derivation, and/or manipulation of polygonal meshes in BIM is often poorly supported or entirely absent. The importing and exporting of mesh-based formats is found to be similarly lacking among leading BIM software.

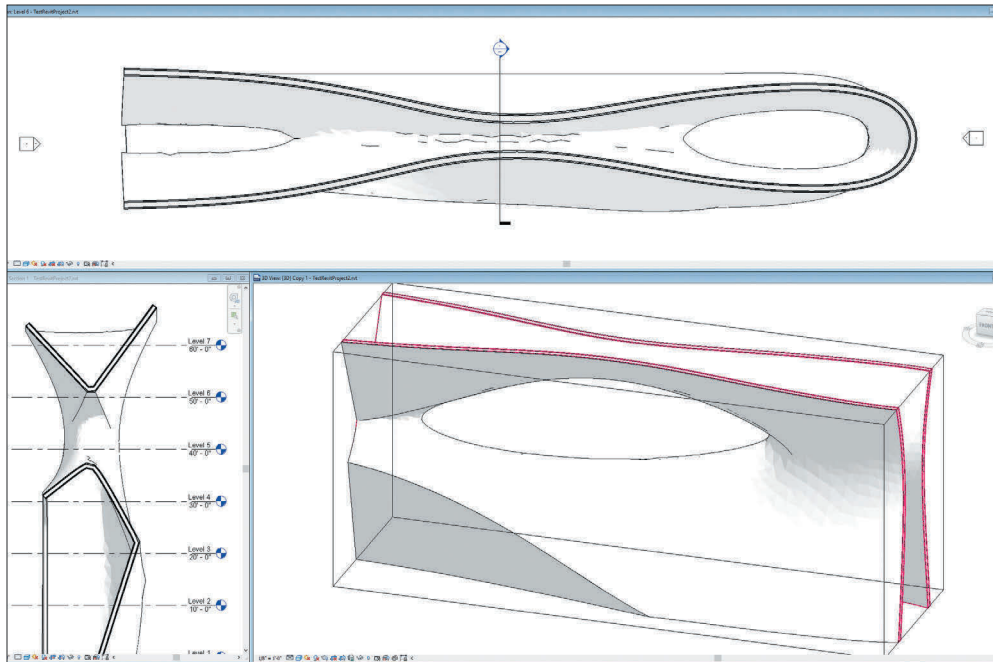
In this context, there is a clear disconnect between 1) mesh-based modeling approaches vital to contemporary design practices and 2) the building industry's continued drive toward BIM-based production and delivery. For projects that aim to exploit mesh-based workflows during design, BIM's inadequacy in supporting these data structures produces both friction and waste during production, including painful remodeling processes and incomplete or low-fidelity documentation for complex geometries. This includes the need for production staff to "mask" or "hand-draw" various objects on a per-drawing basis, a time-consuming process that directly contradicts BIM's fundamental utility for project delivery: to simultaneously manage design geometry across multiple, interdependent, multi-media and on-demand representations.

We address these challenges through a mesh-based workflow that enables improved integration between design conceptualization, iteration, analysis, and production. We present custom tools that have been developed, tested and deployed for multiple, large-scale architectural projects. These customize McNeel's Rhinoceros 3D modeling platform for design authoring, and Autodesk's Revit for BIM project delivery. For this paper, we will first present an overview of this method, and then subsequently outline each of its key constituent phases in further detail. Finally, we will review real-world results from its implementation in two significant building projects and discuss future applications and opportunities for mesh-based workflows in BIM.

METHOD OVERVIEW

The workflow presented here enables designers to successfully transfer complex mesh geometry from design software to a BIM production platform (referred to as mesh-to-BIM). It leverages mesh data structures for capturing geometry and representing critical element properties during project development, building upon current research in the use of adaptive meshing techniques and recent implementations of practical interoperability workflows.

To focus our efforts, we developed functional tools to support the transfer of meshes stored in Rhinoceros 3D into Revit. Rhinoceros has been selected because of its popularity as a geometry authoring and editing platform among both academic and professional practitioners, and for its broad support of mesh



- 4 Native NURBS Brep to mesh conversion for Rhino (left) and curvature-adaptive NURBS Brep to mesh conversion outlined in this paper (right).
- 5 Imported mesh objects with hidden edge visualization and clean graphic representation for 3D and 2D documentation inside of Autodesk Revit.

5

file formats. Revit has been selected as the leading BIM production platform in North America. Crucially, both platforms support programmatic development through APIs that use Microsoft's .NET ("dot net") software development framework.

This workflow follows two general steps:

1. Preparation: We define several geometry preparation steps that allow users to control mesh resolution and apply important features, such as material thickness and sharp creases, before translation into the BIM production environment. This paper focuses on an approach that derives curvature-adaptive meshes directly from complex BREP polysurface geometry, through custom tools developed for Rhino and Grasshopper.
2. Transfer and data assignment: We then discuss a custom software application for translating meshes into a compatible format for use within Revit. The critical component within this part of the workflow involves exploiting DXF as a file transfer method. This file format appears to be the singular method by which external users can access mesh edge settings that, in Revit, enable visibility control for the types of clean representations that are essential for high-quality project delivery. The workflow additionally automates parameter creation and assignment through custom API connections. Here, users may assign important object attributes as they transfer meshes for use in the Revit.

PREPARATION: CURVATURE ADAPTIVE MESHING

The term *adaptive meshing* applies to mesh production methods that vary edge and face sizes according to local conditions. These methods most commonly manage a mesh's fidelity to the geometric basis from which it is derived, with higher levels of refinement (i.e., smaller edges) employed in areas of greater detail or extreme curvature. One such approach is deployed here for the effective representation of complex geometry in BIM production environments.

Integration of references and source material

This method focuses specifically on transforming complex NURBS-based BREPs into curvature-adaptive meshes and relies on the original BREP for both topological and geometric parameters for initial mesh creation and refinement. It has been developed using three key programming instruments:

1. BREPs in RhinoCommon: RhinoCommon is a programming library that exposes object classes for Rhino within the .NET framework. NURBS surfaces here are structured to convert two-dimensional UV parameter spaces into continuous 3D geometric representations. Boundary representations (BREPs) extend them by allowing multiple surfaces to connect along shared edges, and for holes or perimeter trims to apply splits or breaks in the continuous logic of the underlying NURBS geometry. RhinoCommon enables BREP traversal through a comparatively complex data structure that relies on a hierarchy of faces, edge loops, trim curves, edges and vertices.
2. Triangle.NET: Delaunay triangulation is a well-established

methodology (Lee & Schachter 1980) for generating good quality triangulated meshes in a planar UV space, such that for a set of triangulated points, no single point within one triangle lies within the circumcircle of any other, resulting in triangles that have as large a minimum angle as possible. Triangle.NET (Woltering 2013) is an open-source .NET port of the C++ library Triangle (Shewchuk 1996) developed for producing Delaunay triangulations that are fast, robust, and flexible in their parameterization. Of chief importance here is its ability to create Delaunay triangulations that respect non-convex boundary conditions and interior holes.

3. Plankton: Plankton is an open-source .NET library (Piker & Pearson 2013) developed for creating and managing manifold meshes using a Halfedge data structure. Halfedge mesh data structures are particularly well-suited for performing the types of topology transforming operations that are necessary for local decimation (reduction in resolution) or refinement (increases in resolution)

Additionally, this method implements two documented algorithms for mesh refinement: one for edge length specification for curvature adaptation, and the other for executing the topology changes to the mesh for compliance with these targeted values. These are described in the following two subsections.

Curvature-adaptive edge length calculations

The calculation of targeted edge lengths for locally adaptive mesh elements are derived from the approach described in "Adaptive Remeshing for Real-Time Mesh Deformation" (Dunyach et al. 2013). Edge lengths are specified to more closely approximate the curvature of the underlying geometry such that an error tolerance parameter ϵ is achieved; this error tolerance describes for all mesh edges "the maximum allowed geometric deviation of the triangle mesh from the underlying smooth surface geometry." First, for each vertex location x the discrete maximum absolute curvature κ is calculated. This is based on both the mean curvature H and Gaussian curvature K :

$$\kappa_i = H_i + \sqrt{H_i^2 - K_i}$$

From these inputs, a target edge length $L(x_i)$ for each vertex can be derived:

$$L(x_i) = \sqrt{\frac{6\epsilon}{\kappa_i} - 3\epsilon^2}$$

Each individual edge $e = (x_1, x_2)$ is then assigned a target length based conservatively on the minimum edge length calculated for both adjacent vertices, where $L(e) = \min \{ L(x_1), L(x_2) \}$. Edge

lengths can furthermore be clamped for user specified minimum and maximum allowable values, such that $L(e)$ is a set of $[L_{min}, L_{max}]$.

The method presented here differs in a key manner from Dunyach, et al., in that it relies on the UV parameter space and direct curvature querying potential of the original NURBS-based BREP face to calculate κ rather than an underlying mesh geometry. Having the original NURBS-based object available reduces computational expense versus mesh-based curvature calculations.

Local mesh decimation and refinement

Local mesh decimation and refinement are employed to modify mesh topology for approximating target edge lengths. They follow the approach described in "Dynamic Remeshing and Applications" (Vorsatz, Rössl, & Seidel 2003), with the code partly adapted from the open-source MeshMachine library (Piker 2014) which implements this approach. The algorithm modifies mesh topology and vertex geometry through four main operations, performed in nested iterations over the mesh: edge splitting, edge collapsing, edge flipping, and smoothing.

Edge splitting occurs when a given mesh edge has been evaluated as being longer than a target length, and an increase in resolution is desired. Edge collapsing occurs when an edge has been determined to be shorter than a target threshold, and a decrease in resolution is desired. Edge-flipping occurs where it will reduce valence error, creating a better-balanced mesh. Using the halfedge data structure, these topological operations are "inexpensive and local." Smoothing is performed such that each non-boundary vertex is adjusted to the average of all vertices with which it shares an edge.

NURBS parameters and BREP topology

Previous remeshing implementations typically modify existing input mesh geometries, with resulting quality ultimately resting on the resolution of the input mesh. For decimation, this is not a significant issue; however, for refinement, vertex locations must be estimated from coarser representations. Although some sophisticated methods have been developed to identify and enhance feature resolution from these coarser representations (Nealen, Igarashi, Sorkine, & Alexa 2006), vertices are frequently pulled to input meshes – or NURBS surfaces where available – and then subjected to a final smoothing pass. This is both computationally expensive and problematic in areas of tight curvature. We instead leverage NURBS parameters and BREP topological data from the baseline geometry to enable both a higher degree of fidelity to the original geometry and faster processing.

We first calculate vertex arrays along input BREP edges at locations that satisfy target edge lengths according to the calculations described in section 3.2, such that neither refinement nor decimation will be further necessary along input edges. Each resulting vertex is assigned its UV value for its adjacent faces. Because each edge can be solved independently, this operation takes advantage of multi-core processing as it iterates over the BREP.

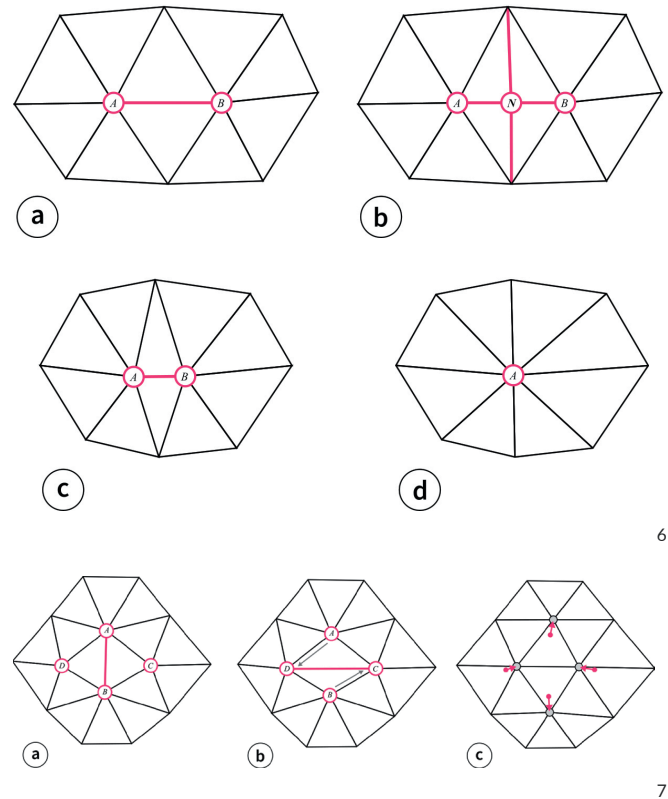
The Delaunay triangulation algorithm exposed through Triangle.NET may be parametrized using contours, which are vertex sequences that define both outer boundaries and holes. These contours ensure that all boundary mesh edges will always be fixed and defined consecutively according to the vertex arrays previously calculated. Delaunay triangulations occur in 2D space. While faces in NURBS-based BREPs define 3D geometry, each face is comprised of a 2D UV parameter space. Therefore, vertices developed through triangulation are constrained to this parameter space, and it is trivial to express their 3D form.

Fixed boundary edges and UV parameter spaces local to each BREP face allows for executing mesh solving and refinement in parallel, again taking advantage of multi-core processing. Furthermore, geometry fidelity is improved and computational expense is saved by relying on the underlying NURBS parameter space for both curvature calculation and vertex location during smoothing, eliminating extensive adjacency tests and pulling operations, respectively.

Edge prioritization for refinement and decimation

Multiple adaptive remeshing iterations are required for the mesh to converge on a stable result that satisfies input parameterization. Existing solutions typically iterate over the mesh edge table and execute changes to the topology for each edge that meets criteria for splitting or collapsing. In order to maintain stability for a given iteration, any vertex that is adjacent to a split or collapsed edge is constrained from further transformation (Figure 8). After each iteration is completed, the mesh is subjected to a compaction, where vertices and edges flagged for removal during collapse are formally extracted from the mesh, and its topology tables are cleaned and renumbered.

In our implementation, a simple evaluation and sorting step attends to this process, improving both speed of convergence and stability. Edges are first scored relative to ideal target lengths, and determined to qualify for either split or collapse. Then edges eligible for splitting are prioritized based on those whose actual to ideal length ratio is highest, and edges eligible for collapsing are prioritized based on those whose actual to ideal length ratio is lowest.



- 6 Mesh topology modification: edge splitting (a) target edge AB is split with (b) a new vertex N inserted at the edge midpoint, resulting in two new faces; edge collapsing (c) target edge AB is removed from the mesh, with the decimation (d) of both adjacent faces and vertex B.
- 7 Mesh topology modification: edge flipping occurs when the vertices shared by faces $\Delta(A, B, C)$ and $\Delta(A, B, D)$ have unbalanced valence (a), such that flipping shared edge AB to DC (b) would reduce valence excess. Laplacian smoothing (c) is applied to adjust the location of vertices in the base Brep UV space to create an even distribution, such that each vertex P is relocated to a weighted average of its neighbors N for each iteration.

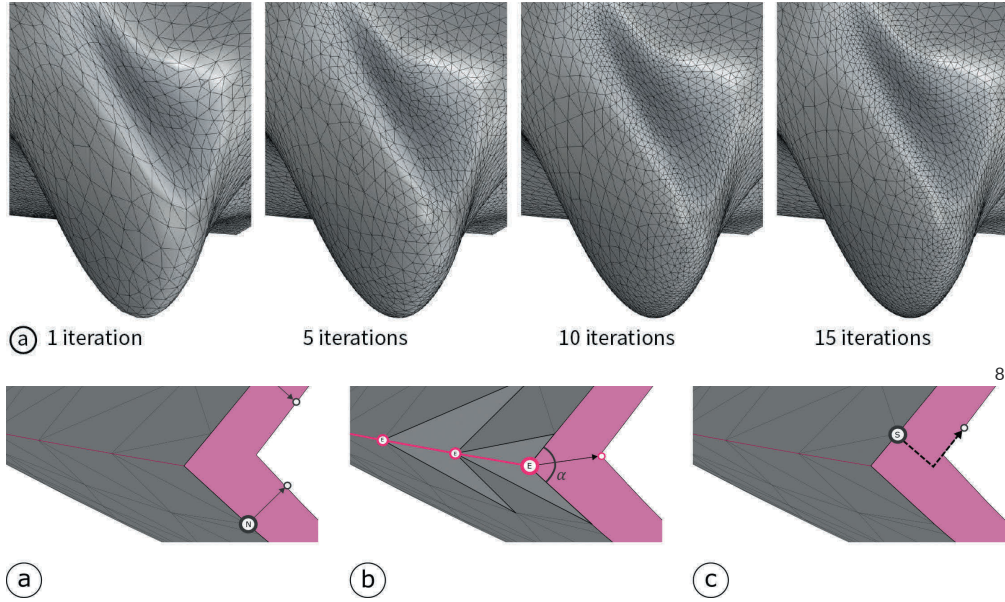
Figure 8 visually illustrates the differences in convergence speed and mesh quality for the same input geometry processed both with and without this prioritization schema. In the presented example, the prioritization-based approach achieves approximate stability at iteration 12, where the average and standard deviation of actual edge length versus ideal does not change substantively beyond this iteration. For the non-prioritized approach, approximate stability is achieved at iteration 20.

Mesh shelling and feature specification

Preparing design meshes for BIM production requires a satisfactory shelling process for proper representation of material assemblies with specified thickness. While more advanced mesh offsetting approaches have been demonstrated as robust and efficient, such as in the development of fabrication level-of-detail

8 Iteration of algorithm applied with sorting edge variances to prioritize splitting and collapsing.

9 Shelling logics: the standard offset for vertices (a) is based on a unitized vertex normal and a target thickness. For vertices on creases (b) the adjacent faces are used to modify the multiplying vector length based on the adjacent face angles in order to create uniform thicknesses. Vertices that would normally be engulfed by the offset thickness around creases or areas of tight curvature (c) are mitigated by smoothing non-feature vertices in the offset mesh.



9

assembly geometries (Ross, Hambleton, and Aish 2016), the shelling process described here is more modest in scope.

Most basic mesh shelling techniques 1) duplicate the base mesh by offsetting vertices using corresponding, unitized vector normal multiplied by a specified thickness, and 2) stitch the original mesh with the offset mesh along any naked edges. This method works well for continuous meshes, but is problematic for meshes that contain areas with high degrees of curvature, have edges that require constraints to other geometries, or contain creases with sharp edges. We therefore extend the standard approach with a few simple steps for managing these unusual conditions.

Vertices associated with geometry-constrained edges are always pulled to the constraining geometry after offsetting, such as where an angled wall may meet a floor or ceiling condition. Constraining geometry may be a BREP, mesh, curve or plane.

Creased edges are offset based on their edge normals rather than their vertex normal. The base edge normal V_e is the average of its adjacent face normal vectors. Its offset multiplier is modified according to the angle α between two adjacent face normals, such that the adjusted unit edge normal $V_{ae} = V_e (1/(\sin(\alpha*0.5)))$.

Finally, offset vertices are smoothed according to a user-defined number of iterations to reduce instances where edges cross each other during shelling due to tight curvature or crease angles.

Deployment

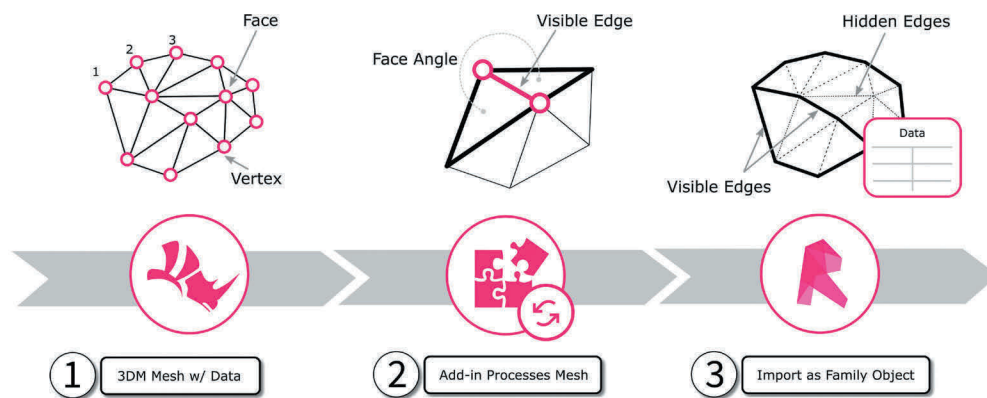
The approaches described above are deployed within the Rhino environment through its visual scripting interface Grasshopper,

which allows for component authoring through the .NET framework. Users are able to integrate these tools directly into their parametric workflows.

TRANSFER: INTEROPERABILITY IMPROVEMENTS FOR MESH GEOMETRIES

The mesh-to-BIM interoperability workflow described in this paper relies on the availability of the following APIs and file format libraries:

1. Revit API: Autodesk Revit provides an extensive API through Microsoft's .NET framework. It provides data access and element creation methods for the Revit file format, enabling data extraction and task automation. It also provides access to Revit's importing and exporting capabilities for supported file formats including DWG, DXF, and SAT. The mesh-to-BIM workflow extends this functionality through a custom 'addin' that has been developed to take advantage of the API's import and task automation classes.
2. OpenNURBS: This is an open-source library supported by McNeel for the reading and authoring of formatted Rhino files. Its geometry classes include meshes, to which users are afforded the ability to assign custom data. It is noteworthy that use of the OpenNURBS library is independent of Rhino licensing, which allows for great flexibility when deploying the mesh-to-BIM tool as a standalone plugin for Revit.
3. NetDXF: The Drawing Exchange Format (DXF) is a critical component to the workflow. Revit exposes a direct import of DXF files as one of the few native means of transferring meshes into Revit. The open source library NetDXF is used then to convert Rhino meshes into a format that can be consumed by Revit. Critically, it has been discovered that visibility settings associated with DXF-based meshes



10 The Mesh-to-BIM importer is a custom Revit Add-in that uses Rhino OpenNURBS to read and process a mesh with desired visibility and data settings in Revit.

10

are a class feature, and that these settings are respected in the Revit environment. When mesh edge visibility is changed to 'hidden' in the DXF mesh, the resulting Revit display recognizes these graphical controls. This capability is essential for the mesh-to-BIM workflow. A custom Revit add-in application is created using these resources, and is exposed as a command within the Revit interface. End-users employ this tool to automate mesh importing by directly accessing geometry contained within a Rhino file. This allows users to both control mesh edge visibility and then parameterize them as Revit families. The algorithm applies the following logics:

1. Rhino mesh geometries are read using the OpenNURBS library. Faces and vertices of each mesh object are accessed along with data related to Rhino layers, materials, and custom-applied user data.
2. The mesh is converted into the DXF file format based on the underlying topology. Using user-specified angle tolerance calculations and naked-edge status, face edges are specified as either displayed or hidden within the DXF mesh.
3. The Revit API is then used to automate the creation of a new 'Family' element into which the DXF is imported, which is then loaded and placed into the Revit project for coordination with other 3D building elements.

DATA ASSIGNMENT: MESHES WITH BIM ATTRIBUTES

Beyond the creation of geometric meshes, the mesh-to-BIM workflow also enables data assignment and classification for transferred mesh elements. The ability to relate data to the imported geometric component allows the user to extend the use of the mesh beyond mere coordination and leverage the geometry as a documentable component within a project. At the most basic level, the mesh-to-BIM process facilitates the classification of elements into known element categories. At a more advanced level, the workflow makes use of custom data properties and translates them into custom parameters.

The use of OpenNURBS as a mesh transfer format also enables the use of "User Data" to assign custom properties to objects. Custom properties may include data about material, manufacturer, or analysis performed outside of the BIM environment. With structured User Data assigned to the object, the mesh-to-BIM reads all associated parameter key-value pairs and automates the creation and assignment of parameters to the newly created Revit Family element allowing it to be scheduled and quantified like a native Revit element.

RESULTS FROM PRACTICE

The mesh-to-BIM method described in this paper has been deployed in two major building projects in the past year, the Gilder Center for the American Museum of Natural History by Studio Gang Architects, and the confidentially titled "Project B". This section briefly reviews the deployment of the mesh-to-BIM workflow described in this paper.

The Gilder Center for the American Museum of Natural History

Studio Gang Architects has deployed the mesh-to-BIM workflow as part of the development and delivery phases of the Gilder Center retrofit and extension for the American Museum of Natural History. A central feature for their design is the complex, organic form of the main atrium space, which has been created using the T-Splines subdivision modeling plug-in for Rhino. Although this model may be directly translated to a mesh or a BREP within Rhino, the resulting geometries each produced graphic difficulties, scalability issues, and undesirable file sizes when imported directly into Revit.

The mesh-to-BIM workflow is then deployed to address these concerns. In this case, special attention is given to adaptive re-meshing for detail control in areas of tight surface curvature while producing coarser mesh topologies in smoother areas, which feeds forward into the shelling processes previously

- 11 The design of the American Museum of Natural History Gilder Center by Studio Gang Architects leveraged the Mesh-to-BIM workflow to translate complex mesh geometry into Revit for documentation.



described that enable representation of thickness in the architectural finished surface. In concert with other custom tools also dependent on the resulting mesh geometries, the design firm has been able to effectively produce drawing sets at the necessary resolution for advanced coordination and project delivery.

Confidential Project B

Project B is a separate museum project with a similar problem set, but with a wider scope that encompasses both complex exterior forms and interior public areas. These geometries have first been developed in Maya, which are then significantly modified in Rhino, with several applied trims and splits exacerbating their complexity. These splits define distinct exterior regions whose individual elements integrate multiple material conditions and various assembly types, including a fiber reinforced polymer (FRP) facade, a green roof, glazing, and photovoltaics. The executive architecture team responsible for project delivery has deployed the mesh-to-BIM workflow through the use of Rhino layer structures for mesh component classification into related material categories.

While the production and documentation of both projects will remain 'in progress' for the next year, the teams have reported that the use of the custom workflows for mesh development and importing has resulted in fast and reliable coordination of complex geometry. The resulting 3D BIM assets have also been found to facilitate BIM production in ways otherwise intractable through conventional workflows. Had either team been limited to out-of-the-box compatibility provided by Revit, achieving

requisite resolution for delivery would have necessitated drafting-based workarounds—including the production of 2D CAD overlays and laborious masking techniques to hide unwanted geometry for visual clarity and/or revised modeling with low fidelity geometry approximation—that would have significantly compromised output quality and put the delivery schedule at risk, undermining the principle efficiencies BIM production software claims as central to its value.

FUTURE WORK: THE CHALLENGES AND OPPORTUNITIES OF DIGITAL PROLIFERATION

This paper has described a workflow focused on interoperability between two popular software platforms, one typically used for geometry authoring and management, and the other for BIM production. It is deployed through custom-built software that provides architects with a streamlined and flexible mesh-to-BIM workflow, with special attention to geometric refinement for production-grade representation and object-level metadata integration, and its utility has been demonstrated through multiple applications in real-world large-scale projects.

The reliance on a wide range of freely available programming libraries and APIs exposes an interesting problem for BIM as it relates to the contemporary computational design community. Even as BIM becomes increasingly important for advanced project delivery, and its advocates position "single-source-of-truth" models as the future of the AEC industries, open-source libraries and tools for design and analysis continue to proliferate. This has a compounding effect, as computational designers

develop alternative approaches to design and representation, share them, and spur further development. In this context Kevin Kelly would suggest that an increasingly heterogeneous set of tools is a more inevitable outcome (Kelly 2010) rather than a single, unified modeling and production environment. Yet Kelly also suggests that one component to effectively curating this inevitability toward value-add results is identifying common meta-structures for development. From a purely programming perspective, it appears that the .NET framework is one such common structure. We suggest that mesh data structures also behave as another type of framework for promoting a productive heterogeneity in the ongoing development of computation design tools for architecture. While their use has been well exploited for both design and analysis, their potentials as instruments for interoperability appear to be less well-explored, and merit continued investigation.

ACKNOWLEDGEMENTS

The research and development of the mesh and BIM workflows have been heavily informed by direct collaborations with architectural design teams. The authors would like to acknowledge the collaboration with Studio Gang Architects and the amazing design work produced for the American Museum of Natural History Gilder Center which pushed the boundaries of the technology outlined in this paper.

REFERENCES

- Dunyach, Marion, David Vanderhaeghe, Loïc Barhe, and Mario Botsch. 2013. "Adaptive Remeshing For Real-Time Mesh Deformation." In *Eurographics Short Papers*. Girona, Spain: Eurographics. <https://doi.org/10.2312/conf/EG2013/short/029-032>
- Eastman, Charles, Paul Teicholz, Rafael Sacks, and Kathleen Liston. 2011. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, 2nd ed. Hoboken: Wiley.
- Kelly, Kevin. 2010. *What Technology Wants*. New York: Penguin.
- Lee, D. T., and B. J. Schachter. 1980. "Two Algorithms For Constructing A Delaunay Triangulation." *International Journal of Computer & Information Sciences* 9 (3): 219–242. <https://doi.org/10.1007/BF00977785>
- Miller, Nathan. 2010. "Makeshift: Information Exchange and Collaborative Design Workflows." In *LIFE in:formation, On Responsive Information and Variations in Architecture: Proceedings of the 30th Annual Conference of the Association for Computer Aided Design in Architecture*, edited by Aaron Sprecher, Shai Yeshayahu, and Pablo Lorenzo-Eiroa, 139–144. New York: ACADIA.
- Nealen, Andrew, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2006. "Laplacian Mesh Optimization." In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia*

and Southeast Asia, 381–89. Kuala Lumpur, Malaysia: GRAPHITE. <https://doi.org/10.1145/1174429.1174494>

Piker, Daniel. 2014. "MeshMachine." GitHub. Retrieved from <https://github.com/meshmash/MeshMachine>

Piker, Daniel, and Will Pearson. 2013. "Plankton." GitHub. Retrieved from <https://github.com/meshmash/Plankton>

Ross, Elissa, Daniel Hambleton, and Robert Aish. 2016. "Face-Offsetting Polygon Meshes With Variable Offset Rates." In *Advances in Architectural Geometry 2016*, edited by Sigrid Adriaenssens, Fabio Gramazio, Matthias Kohler, Achim Menges, and Mark Pauly, 40–61. Zurich: vdf Hochschulverlag AG an der ETH Zurich.

Shewchuk, Jonathan R. 1996. "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator." In *Applied Computational Geometry: Towards Geometric Engineering*, edited by M. C. Lin and D. Manocha, 203–222. Berlin: Springer. <https://doi.org/10.1007/BFb0014497>

Vorsatz, J., C. Rössl, and H.-P. Seidel. 2003. "Dynamic Remeshing and Applications." *Journal of Computing and Information Science in Engineering* 3 (4): 338–44. <https://doi.org/10.1115/1.1631021>

Weisberg, David E. 2008. *The Engineering Design Revolution: The People, Companies and Computer Systems That Changed Forever the Practice of Engineering*. Retrieved April 5, 2017, from <http://cadhistory.net/>

Woltering, C. 2013. "Triangle.NET." CodePlex. Retrieved December 2, 2016, from <http://triangle.codeplex.com/>

IMAGE CREDITS

Figure 2 and 11: © Studio Gang and MIR

All other drawings and images by the authors.

Nathan Miller is the founder of Proving Ground – a new business venture that delivers data-driven solutions for architecture, engineering, and construction businesses. As a consultant, Nathan has worked with some of the most reputable organizations in the building industry to create strategies and technologies for leveraging data in the built environment. Nathan's work in data and computation has impacted the design and delivery of a wide range of built environments including stadiums, skyscrapers, hospitals, and museums.

David Stasiuk is the Director of Applied Research at Proving Ground. He is engaged in the application of data-driven workflows and advanced technology in both academic research and practice. As part of his PhD research at the Royal Danish Academy of Fine Arts, Dave focused on investigating a wide range of topics including parametric workflows, geometry rationalization, interoperability, fabrication, and complex material assemblies. In practice, he has delivered numerous computational design solutions to building professionals including architects, engineers and manufacturers.