ISBN 978-86-6022-055-6

BESPOKE GEOMETRIC GLAZING DESIGN FOR BUILDING ENERGY PERFORMANCE ANALYSIS

Nicholas Mario Wardhana[1], Aikaterini Chatzivasileiadi[2], Wassim Jabi[3], Robert Aish[4], Simon Lannon[5]

[1]Cardiff University (UNITED KINGDOM), WardhanaN@cardiff.ac.uk
[2]Cardiff University (UNITED KINGDOM), ChatzivasileiadiA@cardiff.ac.uk
[3]Cardiff University (UNITED KINGDOM), JabiW@cardiff.ac.uk
[4]University College London (UNITED KINGDOM), robert.aish@ucl.ac.uk
[5]Cardiff University (UNITED KINGDOM), lannon@cardiff.ac.uk

Abstract

Geometry has been found to be one of the main architectural design issues that architects are concerned with when they assess the energy performance of their models through the use of Building Performance Simulation tools. Accurate geometry, coupled with correct energy analysis settings, can avoid errors and provide more accurate results which are among the highest priorities for architects and engineers. Current energy analysis tools commonly use the window-to-wall ratio method to provide a fast and automatic way of modelling glazing, including on multiple surfaces instantaneously. However, this method is prone to geometrical inaccuracies in terms of the size and the location of the glazing, which can in turn contribute to the energy performance gap between modelled and monitored buildings. In addition, simultaneous glazing on multiple surfaces might not be entirely supported in existing applications for complex models. To alleviate these challenges, this paper presents a mechanism for creating a bespoke glazing design on curved surfaces based on the concept of UV-mapping. The glazing can be designed on a 2D planar vector drawing as a set of interconnected curves, either as straight lines or B-Splines. These curves are then mapped unto the UV space of the subdivided and planarised input wall as the glazing. It is hypothesised that a building model with a bespoke glazing design, while more time consuming, allows a more aesthetically representative and geometrically accurate glazing design than one with glazing based on the window-to-wall ratio method, thus minimising the energy performance gap.

Keywords: architectural design, 3D modelling, parametric surfaces, glazing design, energy analysis.

1 INTRODUCTION

Geometry has been found to be one of the main architectural design issues that architects are concerned with when they assess the energy performance of their models through the use of Building Performance Simulation (BPS) tools [1]. While complex and curved geometry is increasingly used in modern architectural practice, energy simulation experts struggle to convert that geometry into analytical models appropriate for BPS tools. Two main geometrical issues plague this process. Firstly, BPS tools commonly require planar geometry and thus curved geometry requires segmentation and planarisation procedures before export. While it is beyond the scope of this paper, in a prior publication, the authors identified and analysed geometrical and topological constraints imposed by the energy analysis tools and detailed several points of failure [2].

Secondly, the modelling of bespoke glazing design is very difficult to translate accurately to analytical models and thus workarounds are invented. One of the most common workarounds for representing glazing design is to omit it from the analytical model altogether and replace it with a simple *window-to-wall ratio*, also known as the *glazing ratio*, to be used in the energy analysis calculations [3]. However, this method is prone to geometrical inaccuracies in terms of the size and the location of the glazing, which can in turn undermine confidence in model predictions, contributing to the energy performance gap between modelled and monitored buildings [4]. In addition, although the use of the glazing ratio method might be a convenient way to assign glazing on multiple surfaces instantaneously, this option might not be entirely supported in existing applications for complex models. For example, OpenStudio [5], which uses the EnergyPlus energy analysis engine [6], is not capable of applying glazing ratio to complex models including sloped and curved surfaces. Thus, other tools would need to be used (e.g.

multi face offset SketchUp plugin), which–according to past experience–can present several shortcomings including distorted geometry and stability issues.

To alleviate the challenges surrounding the current tools' inflexibility as well as the performance gap, it is crucial to have a more geometrically representative glazing design in building models. This paper therefore presents a method to create a bespoke glazing design and apply it on curved surfaces. At the core of this mechanism is the geometric mapping of a glazing design to the wall surface in their parametric spaces, coupled with surface planarisation and subdivision. Because the mechanism is not based on projection, it is not necessary to have a restriction on the location and orientation of the glazing design plane. In addition, it is possible to reach concealed portions of the wall, for example those on a concave wall partially hidden by other wall parts, which are otherwise inaccessible using the projection method. The presented mechanism can be used to create a building model that abides to EnergyPlus' geometric and topological constraints.

The rest of this paper is organised as follows. Section 2 presents a summary of geometric and topological constraints of an EnergyPlus-compliant building model, based on the authors' earlier work. Section 3 introduces Non-Manifold Topology (NMT) terminologies and data structures which are used in this paper, whereas Section 4 presents a workflow to create a building model with bespoke glazing. Section 5 presents a study case of applying a glazing design to a building model, and how this model compares to a conventional glazing design process. Finally, Section 6 discusses the advantages and limitations of the presented bespoke glazing design and mentions a few directions towards future improvements.

2 AN ENERGYPLUS-COMPLIANT MODEL

As compiled in the authors' earlier work [2], EnergyPlus requires that a building model satisfies 14 geometric and topological constraints. Efforts must therefore be taken to ensure that the final result of the mechanism adheres to these constraints. These constraints are listed in Table 1. All constraints except G6 will be addressed in this paper, and the IDs will be used as a cross-reference in the algorithm description whenever a constraint is satisfied.

Table 1. Geometric and topological constraints imposed to a building model by EnergyPlus

ID	Constraint Type	Description
G1	Geometric	Walls must not have holes.
G2	Geometric	The glazing should be an extra topology.
G3	Geometric	The glazing should be either a triangle or rectangle.
G4	Geometric	Each space and surface is preferably convex.
G5	Geometric	Curves are to be avoided.
G6	Geometric	The normal of a roof overhang should point downward.
T1	Topological	The glazing must be attached to a wall.
T2	Topological	The glazing must be co-planar to the wall.
T3	Topological	Every glazing must not touch each other.
T4	Topological	Every glazing must not share two edges with walls, floors, or roofs.
T5	Topological	There must not be a wall that is entirely covered by glazing.
T6	Topological	Glazing must not be located inside another surface.
T7	Topological	Surfaces from adjacent spaces must not overlap.
T8	Topological	Heat transfer between spaces is not computed if there is not a shared surface.

3 TERMINOLOGIES: NON-MANIFOLD TOPOLOGY

The glazing design system uses the Non-Manifold Topology (NMT) data structures and algorithms, which have been found to be suitable for the integration of energy analysis tasks in early building design stages [7]. Based on a review by the authors on existing literatures and software libraries [8], the NMT framework used in this paper comprises of the following elements.

- *Vertex:* a topological equivalent of a point.
- *Edge:* a topological equivalent of a curve in the 3D space, which can be straight (a line) or curved.
- *Wire:* a collection of interconnected edges, either closed or open.
- *Face:* a topological representation of a surface, either flat or undulating, in the 3D space. For each face, its 2D parametric space called the *UV space* is defined. Without loss of generality, it is assumed that the UV space of a face is normalised between the parametric boundaries of [0, 1] x [0, 1]. A face is bounded by one external wire, and may be bounded by internal wires, implying holes.
- *Shell:* a set of interconnected faces touching at their edges.
- *Cell:* a topological representation of an enclosed space. A cell is bounded by one external shell, and may be bounded by internal shells, implying holes.
- *CellComplex:* a group of cells connected at their faces.
- *Cluster:* a group of heterogeneous entities which may be disjoint.

In this paper, internal and external building walls, roofs, and floors are represented as faces. Similarly, the glazing pieces are modelled as faces that are coplanar to and attached to the parent wall. As shown in Fig. 1 [9], the NMT elements are inter-related in a hierarchy such that a topological element may contain one or more its immediate lower-dimensional elements. NMT allows an edge to border more than two faces, which means a face can also bound more than one enclosed internal spaces. This topological relationship and connectivity among the various elements of a topological structure enable a consistent and systematic building modelling paradigm. In addition, as will be demonstrated in Section 4, the glazing framework can also be integrated with various NMT operations, such as the slice operation. Performing this operation in the NMT preserves the adjacency information of various building storeys, which can be represented as cells in a cell complex. In addition to the 3D topological elements, it is also helpful to define some of their 2D counterparts. Specifically, this paper uses VertexUV, EdgeUV, WireUV, and FaceUV, which respectively are the equivalents of Vertex, Edge, Wire, and Face in the UV space.

4 CREATING A BUILDING MODEL WITH BESPOKE GEOMETRIC GLAZING DESIGN

The presented bespoke geometric glazing design will be explained with a demonstration to create a building model. This section introduces a workflow to design an idealised and simplified model of the London City Hall building, which was designed by Foster and Partners, as it exemplifies a complex building with curved walls. It should be noted that while the general dimensions of the model are similar to that of the real building, the idealised model lacks the geometric details of its real-world counterpart, including the same glazing.. The glazing framework is implemented upon an NMT kernel called Topologic, which the authors are currently developing as a plugin to parametric modelling software applications. One of these applications is Autodesk Dynamo, which here is used as an interface to create the building model. Topologic itself is powered by data structures and operations provided by the Open CASCADE Technology (OCCT) library [10].

Algorithm 1 and Fig. 2 present a mechanism to create the London City Hall model with bespoke geometric glazing design. This mechanism consists of five main steps.

1. The user creates a *target wall* and a *glazing design*. These structures will be the primary inputs to the bespoke glazing design mechanism (Fig. 2(a), line 1 in Algorithm 1, will be discussed in Section 4.1).

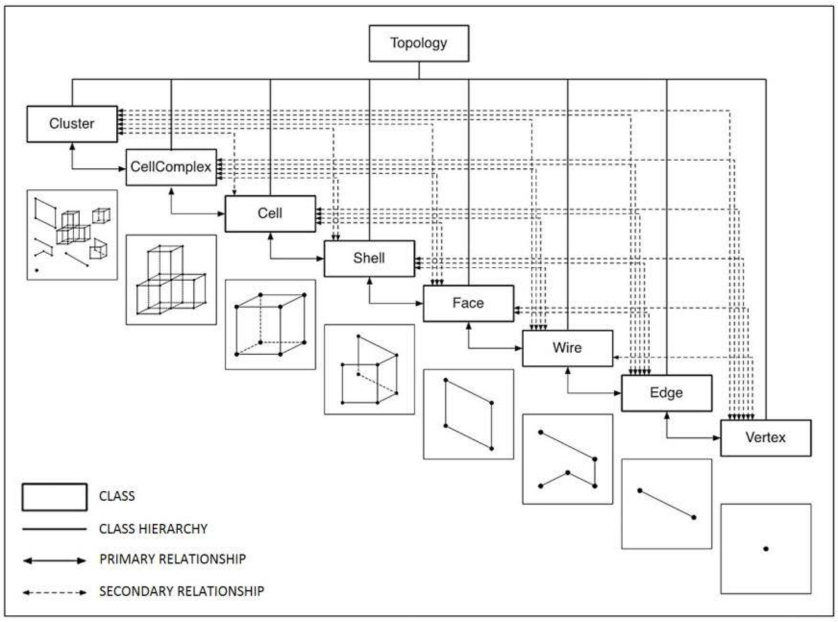

Fig. 1. The NMT elements hierarchy [9]

```
CellComplex CreateBuildingWithGlazing(Face targetWall, Face glazingDesign) {
    Face targetWallWithGlazing = ApplyGlazing(targetWall, glazingDesign);
    Face[] wallPanelsWithGlazing = SubdivideAndPlanarise(targetWallWithGlazing);
    CellComplex building = CreateABuilding(wallPanelsWithGlazing, ...);
    return building; // send to OpenStudio/EnergyPlus
}
```

Algorithm 1. The general pseudocode to create a building model with bespoke geometric glazing design. Only the primary arguments are shown for conciseness. Additional arguments will be shown in the individual algorithms.

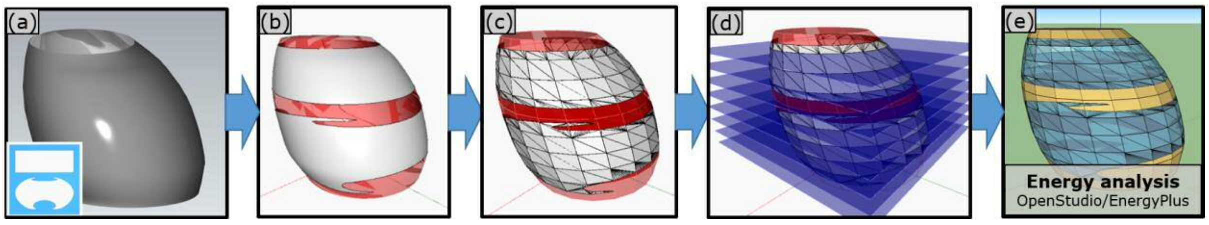

Fig. 2. A workflow to create the idealised London City Hall model with bespoke glazing for energy analysis. (a) The input curved target wall and glazing design. (b) The glazing design is applied to the target wall. (c) The wall and its glazing are subdivided and planarised into a set of wall panels. The glazing is mapped to the corresponding wall panel, triangulated, and scaled down by a tiny factor against the panel's boundary. (d) The wall is capped at the bottom and the top to create a closed shape and sliced into multiple stories. In (b), (c), and (d), the building is slightly scaled down for visualisation purposes to avoid rendering otherwise co-planar surfaces. (e) The building model is sent to OpenStudio/EnergyPlus for energy analysis.

2. The glazing design is mapped to the target wall (Fig. 2(b), line 2 in Algorithm 1, will be discussed in Section 4.2).
3. The wall is subdivided and planarised into a number of flat wall panels. The glazing undergoes the same subdivision and planarisation, and each piece of the subdivided glazing is subsequently mapped to the containing panel. The mapped glazing is then triangulated and scaled down by a very small factor (Fig. 2(c), line 3 in Algorithm 1, will be discussed in Section 4.3).
4. Other modelling operations can be applied to the wall panels. In this case, the planarised wall is capped at the top and the bottom to close the building model, which is subsequently sliced into multiple stories (Fig. 2(d), line 4 in Algorithm 1, will be discussed in Section 4.4).

5. The resulting planarised wall can then be used to create the model of the whole building, which can be passed to the EnergyPlus energy analysis toolkit (Fig. 2(e), line 5 in Algorithm 1).

4.1 The target wall and the glazing design

The presented bespoke glazing design workflow have two primary inputs. The first input is a target wall, which is represented as a parametric face, and can be flat or undulating. In addition, the wall may also be closed along one or both dimensions on its parametric space. Fig. 2(a) shows the curved wall of the model, which was created by performing a loft through a set of vertically stacked circles. This wall is closed on the horizontal direction but open vertically, with holes at the top and the bottom of the wall.

The second input is a glazing design, an example of which is depicted in Fig. 3. A glazing design is a 2D face with by a rectangular outer boundary wire, which corresponds to the target wall border. It is suggested that the dimension of this rectangle is proportional to the (unwrapped, if closed in any direction) target surface's aspect ratio for intuitiveness. The glazing design also contains inner boundary wires, which will be mapped to the target wall as its glazing. The inner boundaries are a closed collection of edges which are represented as parametric curves, which can represent straight and curved lines. The edges in each inner boundary must be non-intersecting, in a way that the inner boundary itself encloses a simple polygon with a continuous area. This glazing design can be constructed inside a parametric modelling software application such as Dynamo, as presented in this paper, or can be created in and imported from vector graphics application as a vector drawing.

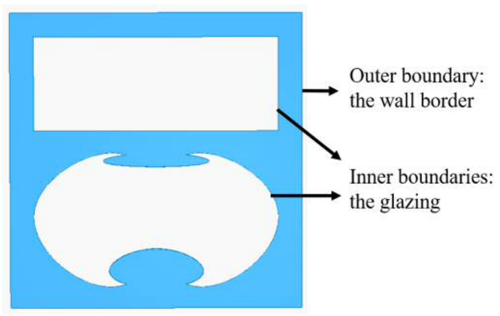

Fig. 3. An example of the user-created glazing design with two inner boundaries. The upper inner boundary consists of only straight edges, whereas the lower inner boundary consists of only curved edges.

4.2 Applying a glazing design to the target wall

The application of the glazing design to a target wall is based on the concept of *UV-mapping* [11], which maps points in a 2D space to a 3D surface. This technique is widely used in the field of Computer Graphics, including in the computer games and animation industries, to map a 2D rasterised texture image from its local 2D coordinate system to a surface in the 3D space. Such mechanism facilitates texture designers to easily and intuitively author a texture design in the 2D space, rather than making such design directly on the 3D surface. The glazing application step uses the same principle to benefit from the same design intuitiveness. As opposed to the original method, however, the difference lies in the application to the inner boundaries of the glazing design, which are parametrically represented.

Algorithm 2 shows the glazing application algorithm. Given a target wall and a glazing design, the latter's inner boundaries are retrieved (line 2). For every curve in the boundary (lines 3-6), a number of points are regularly sampled along the curve (line 7). For each sample point, its normalised UV coordinate on the glazing design is calculated (line 8; here a UV object is simply a collection of two floating points). The UV coordinates of the sample points are subsequently mapped to the target wall to create its sample points counterpart (line 9). From these sample points, a curve can be created by interpolation (lines 10-11). Once all the curves from the same inner boundary are mapped, a surface can be created by clipping the target wall against this mapped boundary (lines 13-14). It should be noted that these steps do not create a hole in the target wall, rather simply extracting a portion of the wall bounded by the aforementioned boundary. Finally, this clipped face will be the glazing and attached to the target wall (line 15). Fig. 2(b) shows the result of applying the glazing design in Fig. 3 to the surface in Fig. 2(a). At this point, both the target wall and glazing may be curved and do not yet comply with EnergyPlus' constraints.

```
1.  Face ApplyGlazing(Face targetWall, Face glazingDesign, int numOfSamples) {
2.     Wire[] glazingDesignInnerWires = glazingDesign->InnerWires();
3.     foreach(Wire glazingDesignInnerWire in glazingDesignInnerWires) {
4.        Edge[] glazingDesignInnerWire = glazingDesignInnerWire->Edges();
5.        Edge[] mappedGlazingEdges = {};
6.        foreach(Edge glazingDesignEdge in glazingDesignEdges) {
7.           Vertex[] sampleVertices = SamplePoints(glazingDesignEdge, numOfSamples);
8.                     VertexUV[] uvSampleVertices = glazingDesign->
               ParameterAtVertices(sampleVertices);
9.                     Vertex[] sampleVerticesOnWall = targetWall->
               PointsAtParameters(uvSampleVertices);
10.          Edge mappedGlazingEdge = Interpolate(sampleVerticesOnWall);
11.          mappedGlazingEdges->Add(mappedGlazingEdge);
12.       }
13.       Wire mappedGlazingWire = CreateWireByEdges(mappedGlazingEdges);
14.       Face mappedGlazing = CreateFaceByClipping(targetWall, mappedGlazingEdges);
15.       targetWall->AddGlazing(mappedGlazing);
16.    }
17.
18.    return targetWall;
19. }
```

Algorithm 2. Applying a glazing design to a target wall

4.3 Wall and glazing subdivision and planarisation

The procedure presented in this section is crucial in making the wall as well as its glazing comply with EnergyPlus' restrictions. Specifically, the final wall must entirely contain convex planar panels, with the glazing modelled as a set of non-touching triangles attached to their parent faces. Algorithm 3 shows the steps to realise such a model.

Given a target wall with glazing (line 1), which is the output of the procedure discussed in Section 5.2, a grid is created in the UV space with the grid vertices provided by pairing the input u and v values (line 2). This grid represents the subdivision of the target wall in its UV space. The vertices are stored in a 2D array, indexed by the positions of the u and v values in the respective lists. If the surface is closed in one direction, the vertices at the final end (u or v equals 1) are not duplicated as these would be the same as the vertices at the other end (u or v equals 0).

The corresponding points on the surface are subsequently retrieved (line 3) and planarised (line 4). The planarisation step uses the ShapeOp library [12], which is based on an iterative local and global constraint optimisation process. The planarisation strategy used in this paper involves attaching two kinds of constraints to various subsets of the grid vertices. Firstly, the closedness constraint is applied to all the grid vertices to prevent excessive deviation. A higher weight is given to vertices at the boundary of the target wall to allow more displacement flexibility to vertices elsewhere. Secondly, planarity constraint is applied to every set of 4 nearby vertices which bound an enclosed rectangular area in the UV space, referred to as a *wall panel*. This planarisation results in a set of convex quadrilateral wall panels, each bounded by 4 straight edges. If enough iterations were performed, the wall panels will be approximately flat with a minor planarity error. The actual creation of the wall panels is deferred to line 18 so that they can be correctly mapped to the glazing.

Once the wall panels are created, the subdivision and planarisation processes are then applied to the glazing. Lines 7-14 iterate through all the glazing in the target wall. Vertices along the glazing boundary are sampled (line 11) and mapped to the target wall's UV space (line 12). The algorithm then systematically iterates through the wall panels by their u and v indices and constructs the glazing for each panel (lines 16-28). At this point it is necessary to map the UV coordinates of the glazing vertices, which are normalised and valid within the original target wall's UV space, to the wall panel's UV space, which may be different. This can be done by performing a bilinear mapping between the glazing vertices to the wall panel, taking into account the UV coordinates of the panel's corners in the original target wall (line 19).

```
1.  Face[] SubdivideAndPlanarise(Face targetWallWithGlazing, double[] uValues,
    double[] vValues, int numOfIteration, int numOfSamples) {
2.     VertexUV[][] uvGridVertices = CreateUVGrid(uValues, vValues);
3.     Vertex[][] gridVertices = targetWallWithGlazing->PointAtParameters(uvGridPoints);
4.     Vertex[][] planarisedGridVertices = Planarise(vertices, numOfIteration);
5.     Face[] wallPanels = {};
6.
7.     VertexUV[][] uvAllGlazingVertices = {};
8.     Face[] glazingFaces = targetWallWithGlazing()->Glazing();
9.     foreach(Face glazingFace in glazingFaces) {
10.       Wire glazingWire = glazingFace->OuterWire();
11.       Vertex[] glazingVertices = SamplePointsOnEachEdge(glazingWire, numOfSamples);
12.       VertexUV[] uvGlazingVertices = targetWallWithGlazing->ParameterAtVertices
    (glazingVertices);
13.       uvAllGlazingVertices->Add(uvGlazingVertices);
14.    }
15.
16.    for(int i = 1 to uValues->Size() - 1) {
17.       for(int j = 1 to vValues->Size() – 1) {
18.          Face wallPanel = CreateFaceByVertices(planarisedGridVertices[i][j],
                                                  planarisedGridVertices[i+1][j],
                                                  planarisedGridVertices[i+1][j+1],
                                                  planarisedGridVertices[i][j+1]);
19.          VertexUV[][] uvMappedGlazingVertices = BilinearMapping(wallPanel,
                                                                 uvAllGlazingVertices);
20.          FaceUV[] uvMappedGlazing = CreateFaceUVByVertexUVs(uvMappedGlazingVertices);
21.          FaceUV[] uvClippedGlazing = Clip(uvMappedGlazing, uvGlazingVerticesAllPanels);
22.          FaceUV[] uvTriangulatedGlazing = Triangulate(uvClippedGlazing);
23.          FaceUV[] uvScaledDownGlazing = ScaleDown(uvTriangulatedGlazing);
24.          Face[] glazing = CreateFaceByUVClipping(wallPanel, uvScaledDownGlazing);
25.          wallPanel->AddGlazing(glazing);
26.          wallPanels->Add(wallPanel);
27.       }
28.    }
29.
30.    return wallPanels;
31. }
```

Algorithm 3. Subdividing and planarising a wall and its glazing

To find the portions of the mapped glazing inside the wall panel, a polygon clipping technique called Vatti clipping algorithm is used [13] (line 21). Its implementation is provided by the General Polygon Clipping (GPC) library [14]. It is possible that one glazing piece is clipped into more than one pieces if it exits and re-enters a wall panel. The left figure in Fig. 4 shows the result of clipping the glazing against the wall panels. At this point the glazing may be in the form of a polygon more complex than a triangle or a rectangle, as EnergyPlus requires. Therefore each glazing piece is then triangulated using the Ear Clipping triangulation [15], with implementation from Mapbox's Earcut.hpp [16] (line 22). The triangulated glazing pieces are then scaled down by a very small factor to prevent them from touching each other as well as the boundaries of the wall panels (line 23). From these glazing pieces in the UV space, the 3D glazing can be created as a set of faces by clipping the wall panel against the UV coordinates. This triangulated glazing is finally applied to the wall panel. The figure on the right in Fig. 4 shows that the scaling process creates tiny gaps between the triangular glazing faces.

The result of this algorithm, as depicted in Fig. 2(c), is a set of wall panels as well as the applied glazing with the following characteristics.

- The wall panels are approximately planar, convex quadrilateral without holes, and bounded by straight lines (satisfying constraints G1, G4 (partially), and G5).
- The glazing is formed by a separate geometry independent from, but co-planar and attached to the wall panel (satisfying constraints G2, T1, and T2).
- Because of the scaling down step, each glazing face is a triangle which neither touches other glazing faces nor the boundary of the wall panel (satisfying constraints G3, T3, T4, and T5).
- Each glazing face does not contain another glazing face (satisfying constraint T6).

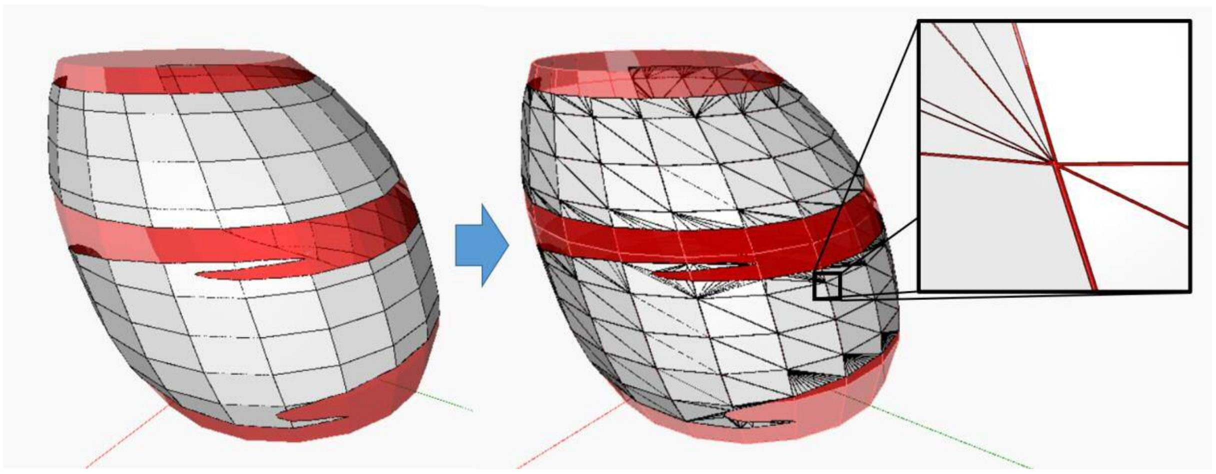

Fig. 4. Left: The result of subdividing and planarising the wall and the glazing. Right: The glazing triangulation and scaling down steps result in triangular glazing faces shows the tiny gaps between the faces.

4.4 Other modelling operations

To create the final model, the wall panels constructed at the previous section were capped to close the holes at the top and the bottom to create a cell. This cell is then sliced by a cluster of eight planes. Because the NMT framework is used, the slice operation does not divide the cell into multiple disjoint pieces. Instead, it results in a cellcomplex, with the portions from the slicing panels being introduced to the building as its floors. As these operations are performed, the glazing information is carried across with proper mapping from the panels of the old model to the new one. This is enabled by OCCT's historical information that keeps track of the generated, modified, or deleted sub-entities after the operation. Within the cellcomplex, adjacent cells (e.g. building stories) are bounded by their shared faces (e.g. floors and ceilings), therefore satisfying constraints T7 and T8. Connection to OpenStudio is provided via the DSOS toolkit [7], which converts the Topologic cellcomplex-representation of a building into an OpenStudio model. The wall panels and glazing faces are respectively converted to OpenStudio's surfaces and subsurfaces.

5 COMPARISON OF THE WINDOW-TO-WALL RATIO METHOD AND THE BESPOKE GLAZING DESIGN

It is beyond the scope of this paper to analyse and comment on the energy performance of using a more accurate representation of glazing design rather than the simplified wall-to-glazing ratio. In addition, a rigorous analysis would require measured data on a real building to which one can compare the results of various analytical models. Yet, one can derive some conclusions from the geometric results of the two approaches.

Fig. 5 and Table 2 show a comparison between two London City Hall models. The model at the left-hand side of the image shows a London City Hall model with the glazing designed using a window-to-wall ratio. There is no unique way to geometrically interpret this ratio, however here the glazing was created by scaling down the parametrically rectangular wall panels to smaller rectangles with an area equal to the window-to-wall ratio. This small rectangle was subsequently triangulated (i.e. converted into two triangles) and applied as a subsurface. With this method, all wall panels would have parametrically identical glazing without much geometric variation. The right-hand side model, on the other hand, was designed using the workflow presented in Section 4. As shown in Table 2, both models have the same total glazing ratio, which is roughly 54.14%. It can be clearly seen that the presented bespoke geometric glazing method introduces a more geometrically representative glazing design. The different glazing modelling paradigms are also apparent: whereas the window-to-wall method assumes the glazing geometry from a given glazing ratio, the bespoke glazing mechanism calculates the glazing ratio from the input glazing design. The latter approach, therefore, provides a framework for architects to experimentally and creatively explore various glazing designs in the early design stage. In terms of simulation time, the model with bespoke glazing has more subsurfaces (674) than those on the model with the window-to-wall ratio (360). This accounts for a longer simulation time of 149.34 seconds, against 67.96 seconds with the other model.

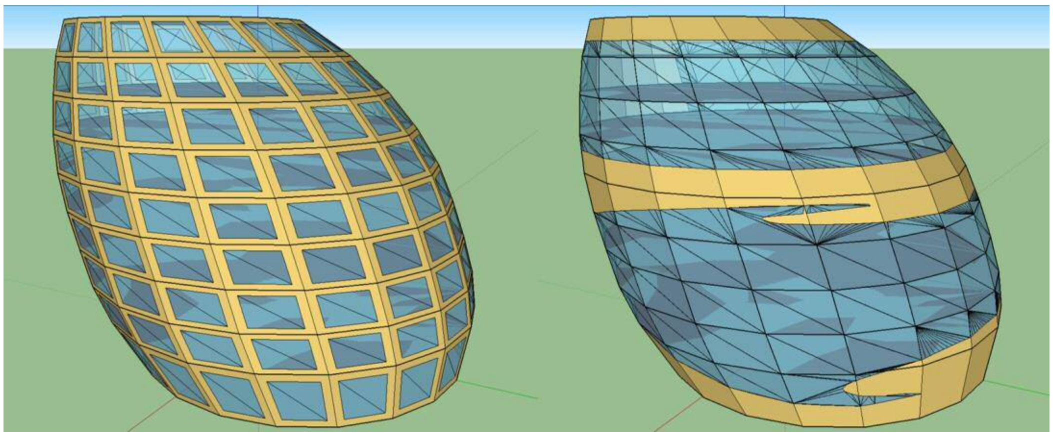

Fig. 5. A comparison between the London City Hall model with the glazing defined by (left) window-to-wall ratio and (right) the bespoke design.

Table 2. Statistics of the London City Hall models with glazing designed with the two methods

Comparison	**Glazing design method**	
	Window-to-wall ratio	Bespoke geometric glazing
Glazing ratio	54.14%	54.14%
Number of wall surfaces	180	180
Number of subsurfaces (glazing faces)	360	674
EnergyPlus simulation time (in seconds)	67.96	149.34

6 FUTURE WORK

A few directions can be considered to improve the current framework. Because the glazing mapping is based on vertex sampling on glazing edges, the mapping resolution and accuracy depend on the number of sampling. A small number of samples will result in a non-representative mapping, whereas a large number of samples will create an equally large number of glazing pieces. It may be worthy to investigate if an exact procedure to create a B-Spline curve on a curved surface [17] will help mitigate the need for a trade-off. Apart from that, the current triangulation method creates a large number of slivers (thin triangles) with small areas (under 0.1 m^2), which, as shown in Fig. 6, have to be removed due to OpenStudio's requirements. Out of the 1096 triangular glazing faces, 422 slivers were removed, which amounted to roughly 38.5% of the original number of glazing faces. If the triangulation result is dominated by slivers, the energy simulation error may accumulate. To alleviate this, it may be useful to device a constrained triangulation strategy which introduces points inside the original glazing to minimise the creation of slivers and restrict the minimum face area to be 0.1 m^2. In addition, a post-processing step to union the glazing faces may also be employed as long as the result is still a triangle or a rectangle. This will reduce the sliver occurrences as well the total number of the glazing faces.

The current implementation does not support slicing glazing faces, which will be an essential feature in the improved framework. Performing this requires not only slicing the geometry of the glazing, but also remapping the portions of the glazing to the correct wall panels, which are also sliced. Finally, it is interesting to include a capability to handle walls with holes, which imposes a challenge in the construction of an Open Studio-compatible model.

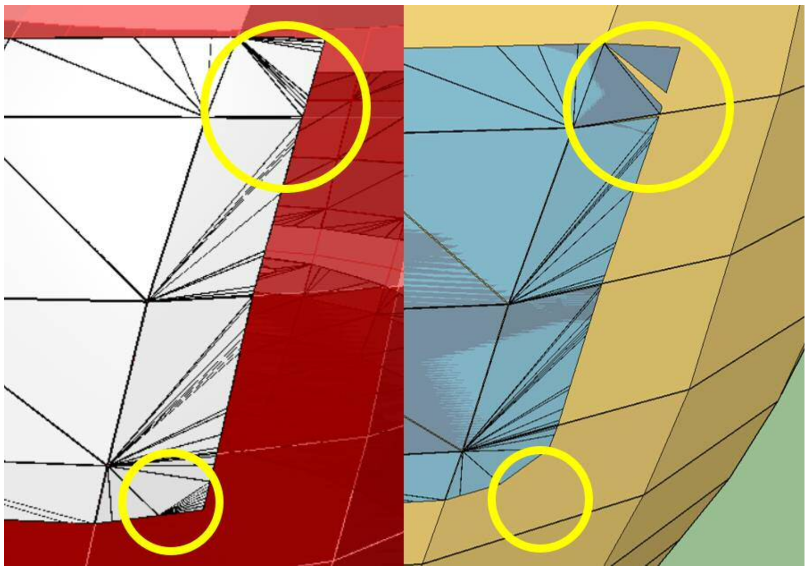

Fig. 6. Some slivers (inside the yellow circles) created by the triangulation procedure performed in Dynamo (left) had to be removed when the building was converted to an OpenStudio model (right, visualisation in Sketchup).

Once these improvements are made, further studies involving real building data will be useful to evaluate the accuracy of the presented glazing design method over the conventional glazing design by window-to-wall ratio. In these studies, the glazing as well as the whole building model will be designed to match the actual building as precise as possible. A comparison will then be done and an analysis will be performed between the energy analysis results of the models with glazing ratio, the presented bespoke glazing mechanism, as well as the monitored data from the actual building.

ACKNOWLEDGMENTS

This project is funded by a Leverhulme Trust Research Project Grant (Grant No. RPG-2016-016).

REFERENCES

[1] Attia, S., Hensen, J. L. M., Beltrán, L., De Herde, A. (2012). Selection criteria for building performance simulation tools: contrasting architects' and engineers' needs. Journal of Building Performance Simulation 5(3), pp. 155–169.

[2] Chatzivasileiadi, A., Lannon, S., Jabi, W., Wardhana, N. M., Aish, R. (2018), Addressing pathways to energy modelling through non-manifold topology. In SIMAUD 2018: Symposium for Architecture and Urban Design, edited by Daniel Macumber, Forrest Meggers, and Siobhan Rockcastle. Delft, the Netherlands: Society for Modeling & Simulation International (SCS).

[3] Smith, L., Bernhardt, K., Jezyk, M. (2011), Automated Energy Model Creation for Conceptual Design, in Symposium on Simulation for Architecture and Urban Design, pp. 53–60.

[4] van Dronkelaar, C., Dowson, M., Spataru, C., Mumovic, D. (2016). A Review of the Regulatory Energy Performance Gap and Its Underlying Causes in Non-domestic Buildings. Frontiers in Mechanical Engineering 1, pp. 1–14.

[5] Guglielmetti, R., Macumber, D., Long, N. (2011). OpenStudio: An Open Source Integrated Analysis Platform, in Building Simulation 2011: 12th Conference of International Building Performance Simulation Association, pp. 442–449.

[6] Crawley, D. B., Lawrie, L. K., Army, U. S., Champaign, C., Curtis, I., Pedersen, O., Winkelmann, F. C. (2000). EnergyPlus: Energy Simulation Program. ASHRAE Journal 42(4), pp. 49–56.

[7] Jabi, W. (2014). Parametric spatial models for energy analysis in the early design stages, in Proceedings of the Symposium on Simulation for Architecture & Urban Design, pp. 17–24.

[8] Chatzivasileiadi, A., Wardhana, N. M., Jabi, W., Aish, R., Lannon, S. A Review of 3D Solid Modeling Software Libraries for Non-Manifold Modeling. To be presented in CAD'18.

[9] Jabi, W., Soe, S., Theobald, P., Aish, R., Lannon, S. (2017). Enhancing parametric design through /non-manifold topology. Design Studies 52, pp. 96–114.

[10] Open Cascade SAS. (2018). Open CASCADE Technology. [Online]. Available: https://www.opencascade.com/.

[11] Catmull, E. E. (1974). A subdivision algorithm for computer display of curved surfaces. PhD Thesis. University of Utah.

[12] Deuss, M., Deleuran, A. H., Bouaziz, S., Deng, B., Piker, D., Pauly, M. (2015). ShapeOp - A Robust and Extensible Geometric Modelling Paradigm. Design Modelling Symposium.

[13] Vatti, B. R. (1992). A generic solution to polygon clipping. Communications of the ACM 35(7), pp. 56–63.

[14] Murta, A. (2014). GPC - General Polygon Clipper library. [Online]. Available: http://www.cs.man.ac.uk/~toby/gpc/.

[15] Held, M. (2001). FIST: Fast Industrial-Strength Triangulation of Polygons. Algorithmica 30(4), pp. 563–596.

[16] Mapbox. (2018). Earcut.hpp. [Online]. Available: https://github.com/mapbox/earcut.hpp.

[17] Renner, G., Weiß, V. (2004). Exact and approximate computation of B-spline curves on surfaces. Computer-Aided Design 36(4), pp. 351–362.