A Generic Method for Distinguishing Planar Geometries

1 Introduction

The problem originated from the task of automatically counting distinct tiles for KSA Pavilion (Figure 1). More specifically, although a repetitive pattern with limited tile types was designed, extra "folded" tiles are introduced at the corners, where different angles and folding positions all requires customized fabrication (Figure 2).

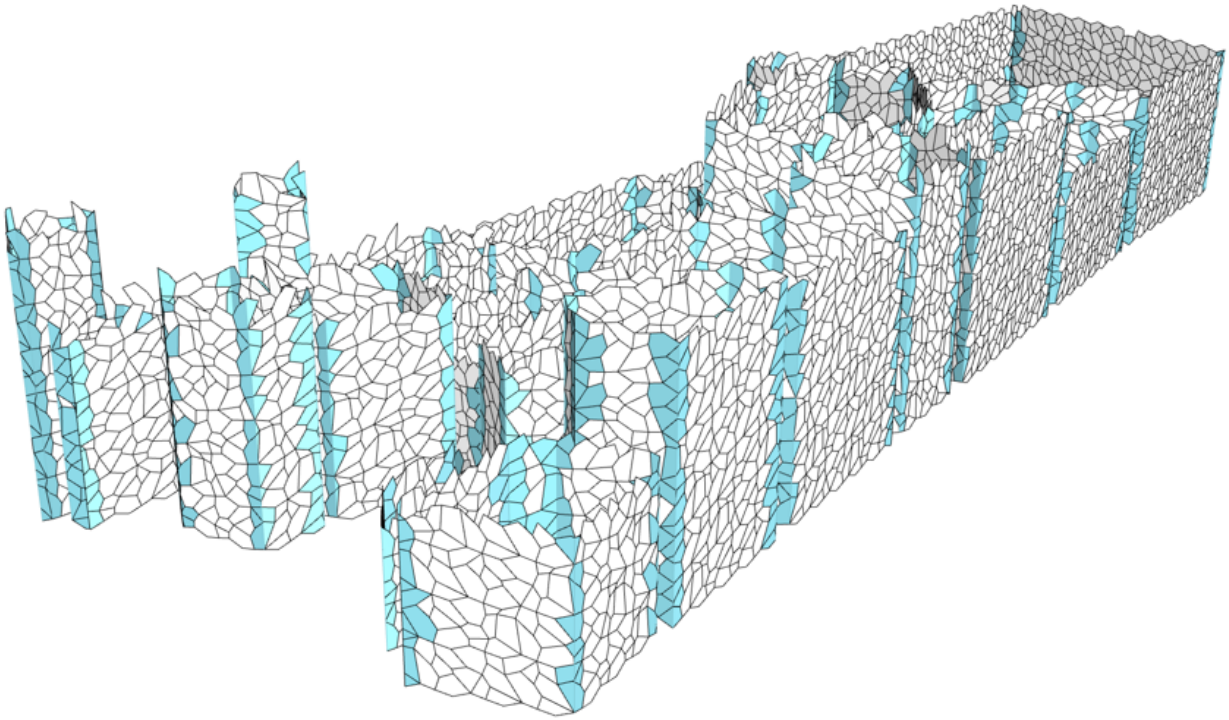

Figure 1: "Voronoi" tiling of KSA Pavilion

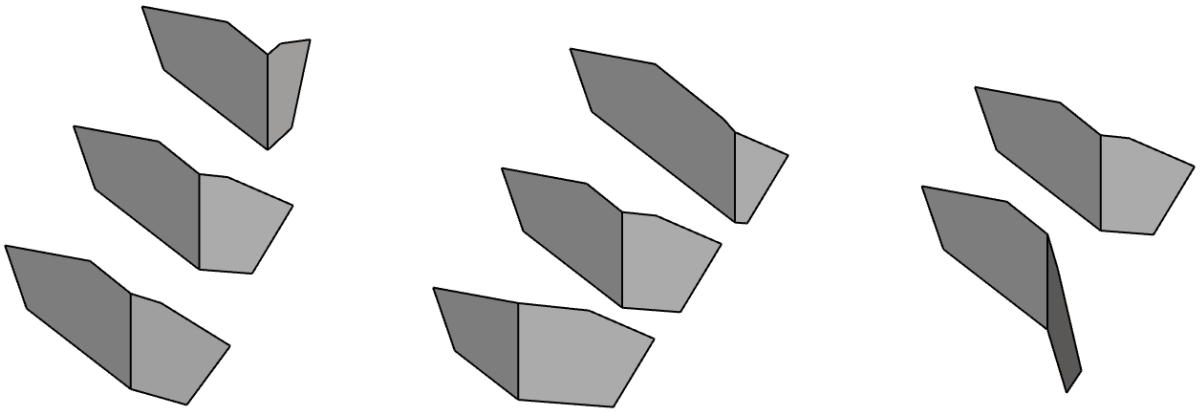

Figure 2: (Left) Different tiles introduced by different angles (Middle) Different tiles introduced by different folding positions (Right) Mirrored tiles are not the same

Through only grasshopper, it is challenging to determine whether two Breps are same. It may involve finding an orienting method and checking the alignments of all vertices, which are slow and complicated especially for categorizing a large number of Breps all consist of multiple surfaces.

Therefore, a fast method is needed where distinct numeric attributes of arbitrary shapes irrelevant to the tiles' orientation are retrieved as unique identifiers. The process to generate identifiers should: **1) Be**

deterministic for the same shape (For example, the process of picking two vertices based on points' distances to other points is not deterministic if applied to a square) **2) Be different for different shapes** (Examples in Section 3.3).

2 Domain

The method introduced in this document are applicable to geometries of following characteristics: 1) Consists of one surface or multiple surfaces with shared edges (A single Rhino BRep type geometry); 2) All component surfaces are planar; 3) All edges are straight.

3 Identifying Arbitrary Shape

3.1 Steps

Let V and E be the collections of vertices and edges extracted by Grasshopper's *Deconstruct Brep* component.

1. Retrieve a collection of points P, i.e. the union of all vertices and midpoints of edges:

$$P = V \cup \{\text{Midpoint}(e) | e \in E\}$$

2. Obtain pairwise distances of P:

$$D = \{\text{Distance}(P_i, P_j) | 1 \leq i < j \leq |P|\}$$
$$(\text{Thus } |D| = \frac{|P|(|P|-1)}{2})$$

3. Concatenate the sorted sequence of D as the identifier of this shape:

$$D_{(1)}D_{(2)}...D_{(n)} \text{ where } D_{(1)} < D_{(2)} < ... < D_{(n)}, n = |D|$$

3.2 Examples

3.2.1 $1 - 2 - \sqrt{5}$ Triangle

15 distances will be obtained from a triangle (Figure 3). Rounding numbers to 3 decimal places, the identifier will be: 500**500**500**1000**1000**1000**1000**1118**1118**1118**1118**1414**2000**2062**2236

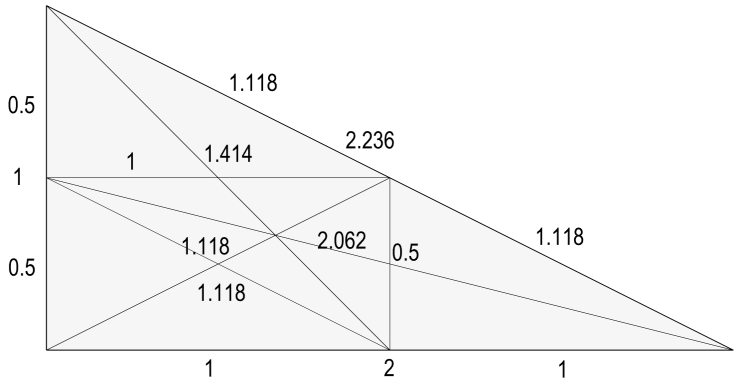

Figure 3: The pairwise distances in an $1 - 2 - \sqrt{5}$ triangle

3.2.2 Folded Shape

55 distances will be obtained from a folded shape consists of one triangle and one quadrilateral (Figure 4). Note that the three dimensional distances across the two components will vary based on the folding angle, which serve as an identifier to the angle.

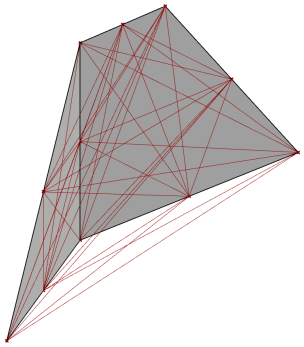

Figure 4: Visualized pairwise distances in a folded shape

3.3 Clarification

3.3.1 The necessity of including edge midpoints

Using $P = V$ fails to distinguish the following pair (Figure 5):

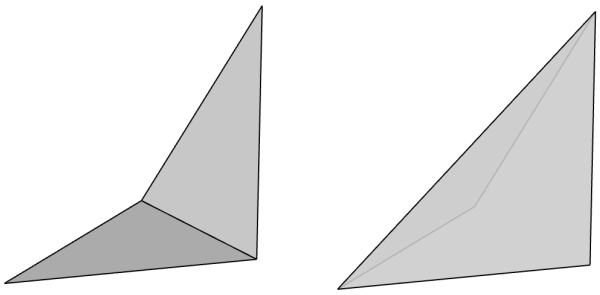

Figure 5: A pair of shapes having the same set of vertices

3.3.2 The necessity of including vertices

Using $P = \{\text{Midpoint}(e) | e \in E\}$ fails to distinguish the following pair (Figure 6):

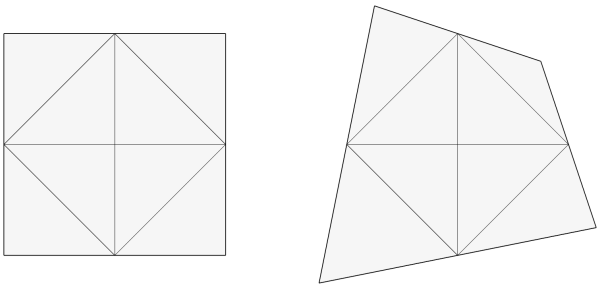

Figure 6: A pair of shapes having the same set of edge midpoints

4 Distinguishing Mirrored Shapes

The method described above works for planar shapes, but relative distances will be the same for mirrored non-planar geometry. Therefore, an extra method is developed for distinguishing mirrored geometries consists of multiple non-planar surfaces.

4.1 Steps

Use the same method from previous section to retrieve P. Let C be the collection of centroids extracted by Grasshopper's *Area* component.

1. Define a function F, serving as a generic method to deterministically pick points from a geometry:

$$F(p) = \sum_{i=1}^{|P|} \text{Distance}(p, P_i)$$

2. Obtain three points P_a, P_b, P_c:

$$P_a = \arg\min_{P_i} F(P_i)$$
$$P_b = \arg\min_{C_i} F(C_i)$$
$$P_c = \arg\max_{C_i} F(C_i)$$

3. Obtain two vectors W_a, W_b:

$$\vec{W_a} = P_b - P_a$$
$$\vec{W_b} = P_c - P_a$$

4. Obtain a position P^*:

$$P^* = P_a + \vec{W_a} \times \vec{W_b}$$

If $\arg\min_{C_i} F(C_i) = \arg\max_{C_i} F(C_i)$, obtain two positions P_a^*, P_b^*:

$$P_a^* = P_a + \vec{W_a} \times \vec{W_b}$$
$$P_b^* = P_a + \vec{W_b} \times \vec{W_a}$$

5. Obtain the distances of all vertices to this position:

$$D' = \{\text{Distance}(P_i, P^*) | 1 \leq i \leq |P|\}$$
$$(\text{Thus } |D| = |P|)$$
$$\text{or}$$
$$D' = \{\text{Distance}(P_i, P_a^*), \text{Distance}(P_i, P_b^*) | 1 \leq i \leq |P|\}$$
$$(\text{Thus } |D| = 2|P|)$$

6. Concatenate the sorted sequence of D' as the identifier of this shape:

$$D'_{(1)} D'_{(2)} ... D'_{(n)} \text{ where } D'_{(1)} < D'_{(2)} < ... < D'_{(n)}, n = |D'|$$

4.2 Example

Below is an example where vectors and distances involved in the process are visualized (Figure 7) for a pair of mirrored geometries. It is clear that their D' are completely different.

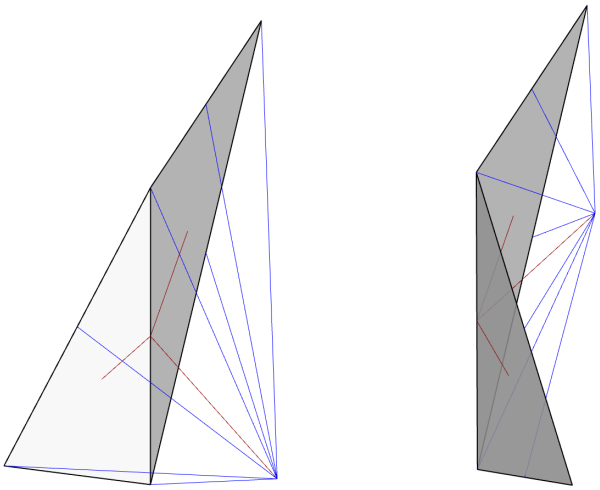

Figure 7: Visualized W_a, W_b, P_aP^* (Red, scaled for demonstration) and D' (Blue) for a pair of mirrored geometries.

4.3 Note

4.3.1 Robustness

Despite of the extra part in step 4, this method may fail if $|\arg\min_{P_i} F(P_i)| > 2$ or $|\arg\min_{C_i} F(C_i)| > 2$ or $|\arg\max_{C_i} F(C_i)| > 2$, i.e. there are multiple equivalent points to pick. See one example below (Figure 8). Therefore, this method is more suitable for counting irregular shapes without a high order of symmetry.

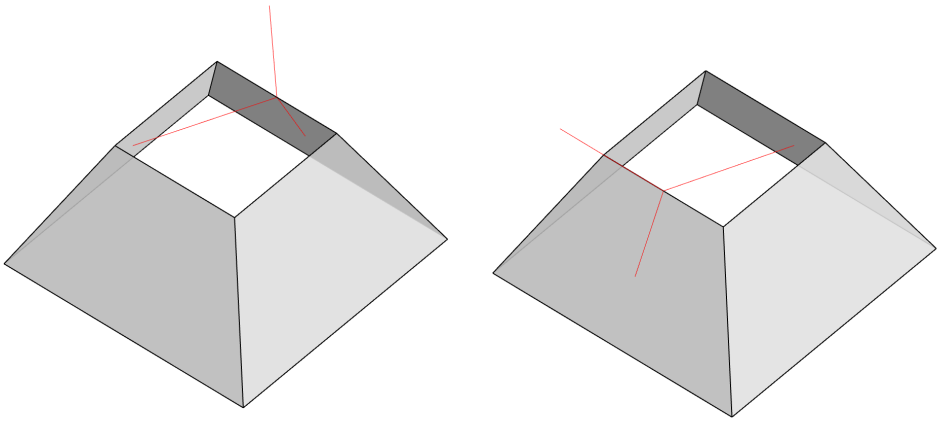

Figure 8: A pair of geometries having multiple equivalent edge midpoints and face centroids, leading to different P^* positions.

4.3.2 Necessity

In current script, this identifier is append to the previous identifier (Section 3). However, it is not clear whether this identifier itself is unique for arbitrary non-planar shapes. A general proof or counter example is welcome.

5 Postprocess

With identifiers, classes from C#, for example HashSet, are convenient to perform categorization, count number of unique tiles, and query each shape. Optionally, SHA256 algorithm can be applied to reduce the length of identifiers. (Figure 9)

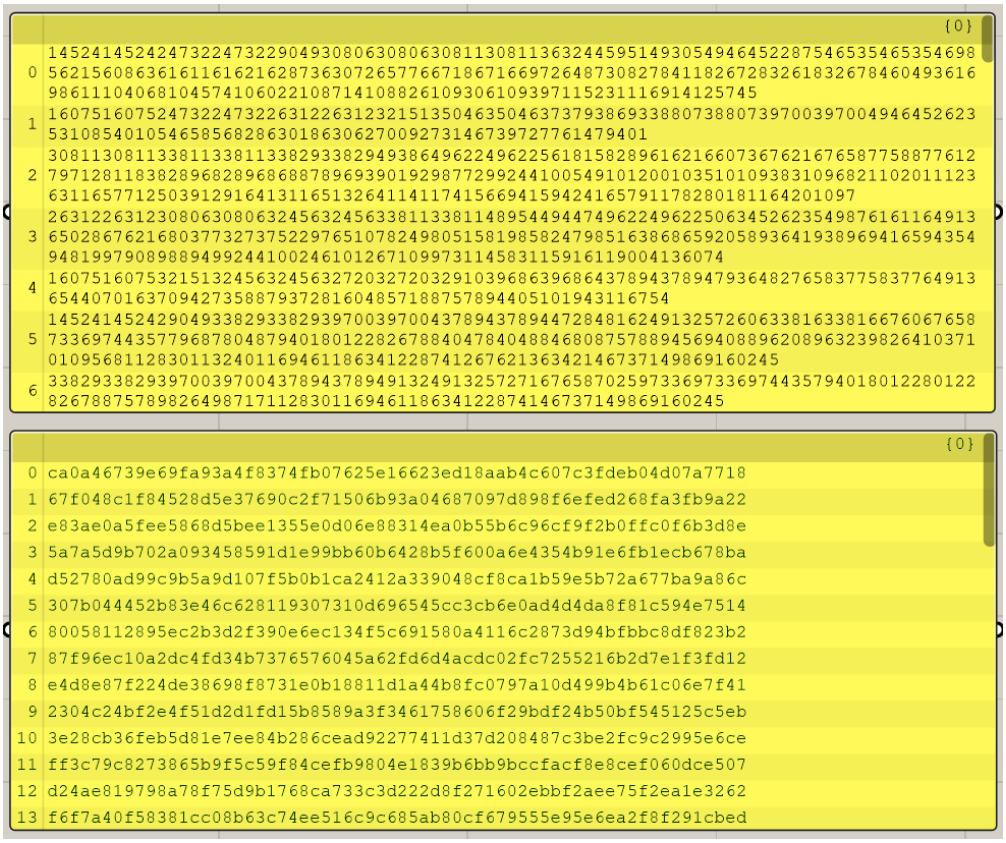

Figure 9: Encoded identifiers of tiles for KSA Pavilion (Top) and their SHA256 encoded results (Bottom).