

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).

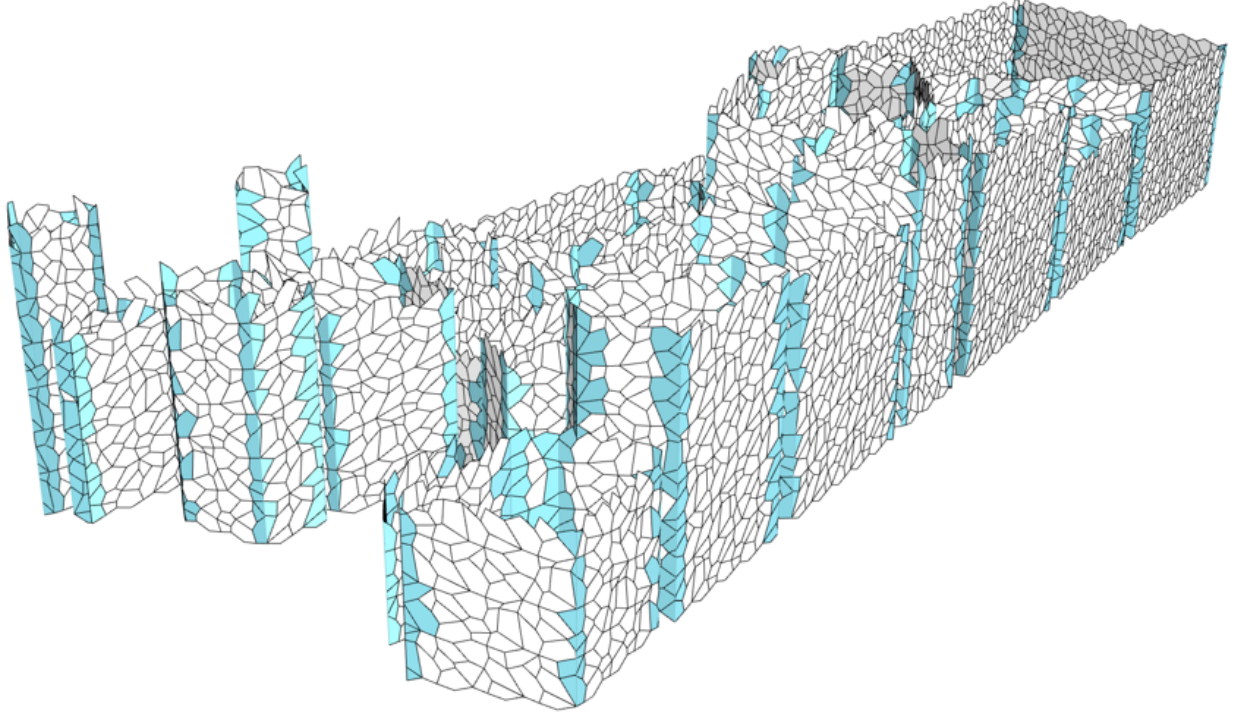


Figure 1: "Voronoi" tiling of KSA Pavilion

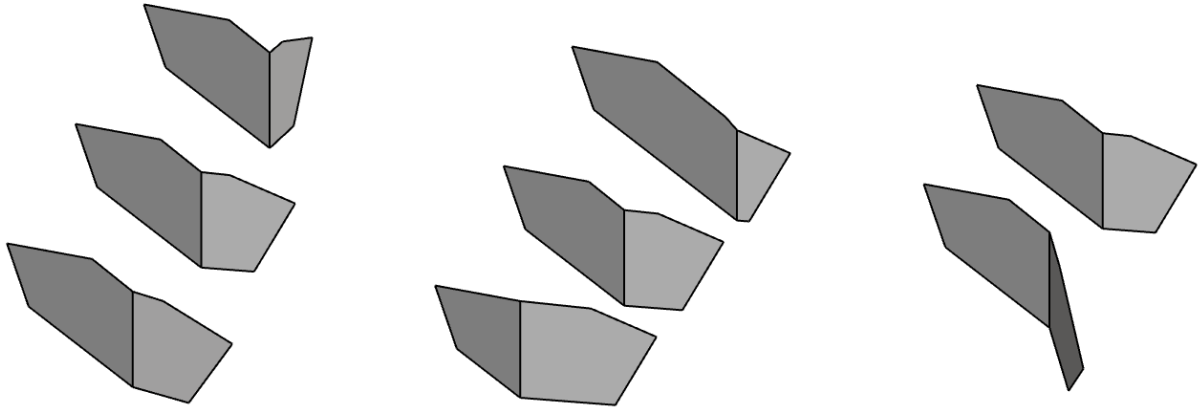


Figure 2: (Left) Different tiles introduced by different angles (Middle) Different tiles introduced by different folding positions (Right) Mirrored tiles are not the same

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 :

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

(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: 50050050010001000100011181118111811181414200020622236

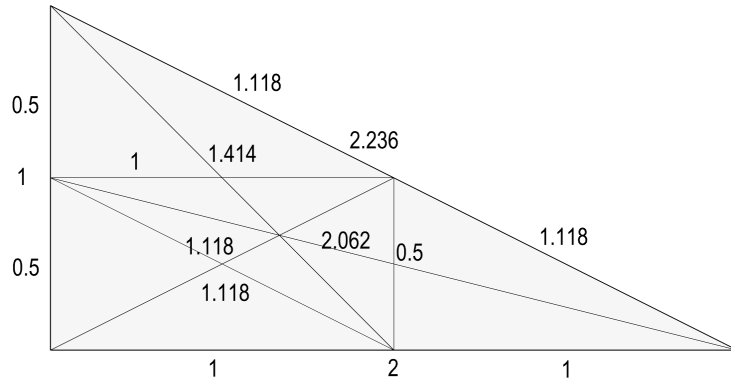


Figure 3: The pairwise distances in an $1 - 2 - \sqrt{5}$ triangle

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.

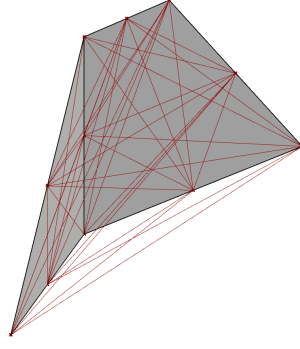


Figure 4: Visualized pairwise distances in a folded shape

3.3 Clarification

3.3.1 The necessity of including edge midpoints

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

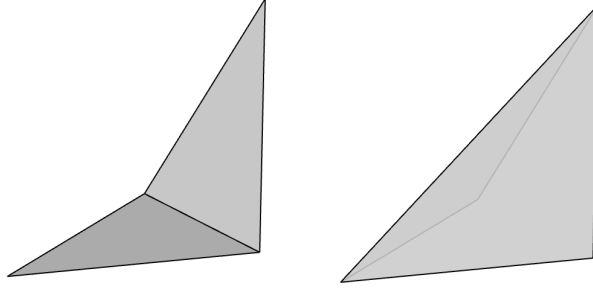


Figure 5: A pair of shapes having the same set of vertices

3.3.2 The necessity of including vertices

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

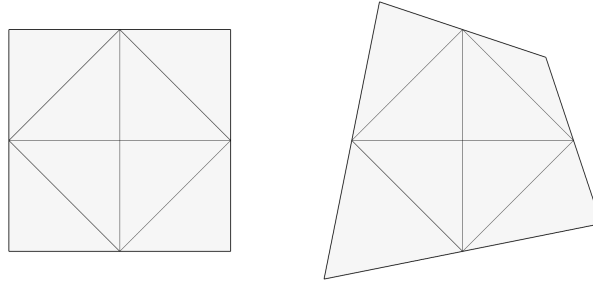


Figure 6: A pair of shapes having the same set of edge midpoints

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:

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

$$(\text{Thus } |D| = |P|)$$

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:

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

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.

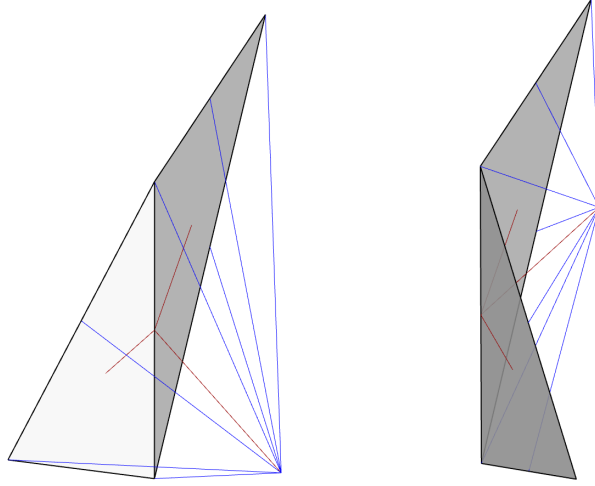


Figure 7: Visualized $W_a, W_b, P_a P^*$ (Red, scaled for demonstration) and D' (Blue) for a pair of mirrored geometries.

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.

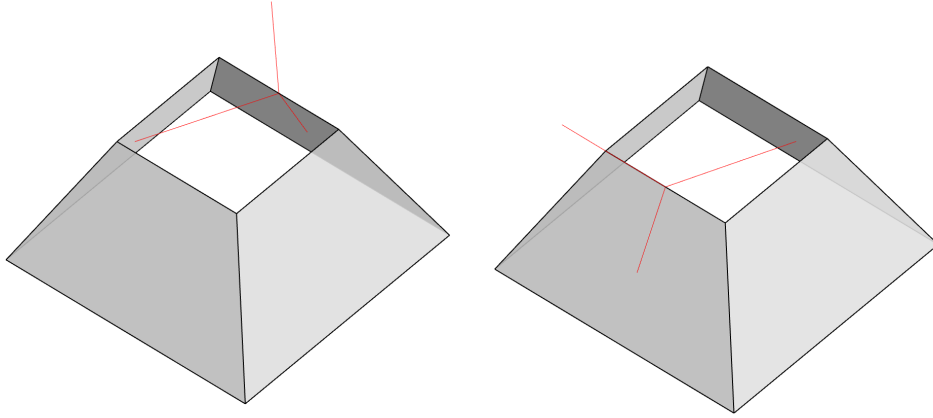


Figure 8: A pair of geometries having multiple equivalent edge midpoints and face centroids, leading to different P^* positions.

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)

		{0}
	1452414524247322473229049308063080630811308113632445951493054946452287546535465354698	
0	5621560863616116162162873630726577667186716697264873082784118267283261832678460493616	
	98611104068104574106022108714108826109306109397115231116914125745	
1	1607516075247322473226312263123215135046350463737938693388073880739700397004946452623	
	5310854010546585682863018630627009273146739727761479401	
2	3081130811338113381133829338294938649622496225618158289616216607367621676587758877612	
	7971281183828968289686887896939019298772992441005491012001035101093831096821102011123	
3	63116577125039129164131165132641141174156694159424165791178280181164201097	
	2631226312308063080632456324563381133811489544944749622496225063452623549876161164913	
4	6502867621680377327375229765107824980515819858247985163868659205893641938969416594354	
	94819979089889499244100246101267109973114583115916119004136074	
5	1607516075321513245632456327203272032910396863968643789437894793648276583775837764913	
	654407016370942735887937281604857188757894405101943116754	
6	1452414524290493382933829397003970043789437894472848162491325726063381633816676067658	
	7336974435779687804879401801228267884047840488468087578894569408896208963239826410371	
	0109568112830113240116946118634122874126762136342146737149869160245	
	3382933829397003970043789437894913249132572716765870259733697336974435794018012280122	
	82678875789826498717112830116946118634122874146737149869160245	

		{0}
0	ca0a46739e69fa93a4f8374fb07625e16623ed18aab4c607c3fdeb04d07a7718	
1	67f048c1f84528d5e37690c2f71506b93a04687097d898f6efed268fa3fb9a22	
2	e83ae0a5fee5868d5bee1355e0d06e88314ea0b55b6c96cf9f2b0ffc0f6b3d8e	
3	5a7a5d9b702a093458591d1e99bb60b6428b5f600a6e4354b91e6fb1ecb678ba	
4	d52780ad99c9b5a9d107f5b0b1ca2412a339048cf8ca1b59e5b72a677ba9a86c	
5	307b044452b83e46c628119307310d696545cc3cb6e0ad4d4da8f81c594e7514	
6	80058112895ec2b3d2f390e6ec134f5c691580a4116c2873d94bfbbcb8df823b2	
7	87f96ec10a2dc4fd34b7376576045a62fd6d4acdc02fc7255216b2d7e1f3fd12	
8	e4d8e87f224de38698f8731e0b18811d1a44b8fc0797a10d499b4b61c06e7f41	
9	2304c24bf2e4f51d2d1fd15b8589a3f3461758606f29bdf24b50bf545125c5eb	
10	3e28cb36feb5d81e7ee84b286cead92277411d37d208487c3be2fc9c2995e6ce	
11	ff3c79c8273865b9f5c59f84cefb9804e1839b6bb9bccfacf8e8cef060dce507	
12	d24ae819798a78f75d9b1768ca733c3d22d8f271602ebbf2aee75f2ea1e3262	
13	f6f7a40f58381cc08b63c74ee516c9c685ab80cf679555e95e6ea2f8f291cbcd	

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