A Pattern Tracing System for Generating Paper Sliceform Artwork

Yongquan Lu and Erik D. Demaine
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar St., Cambridge, MA 02139, USA
yqlu@mit.edu, edemaine@mit.edu

Abstract

Sliceform paper art is an art form where long strips of paper are cut, folded and slotted together to form intricate geometric configurations. We present a system that streamlines the design process for such artwork, which are conventionally highly time-consuming to assemble. This system takes in a polygonal tile-based geometric configuration and traces patterns across tile boundaries to produce a strip-based representation, which can be used to generate the physical paper strips for easy assembly. We exhibit some representative results generated by our system.

Introduction

Sliceform paper artwork is an artform where long strips of paper are cut, folded and slotting together to form geometric configurations. Historically, paper sliceforms have omitted the folding step and focused on 3D shapes; the technique specifically discussed in this paper were first pioneered by artists such as Chris Palmer and Jeff Rutzky [4], and have since been popularized by others such as Christiane Bettens [1]. Fig. 1 shows a representative example. While the physical results are beautiful, assembling a single piece can takes 8 hours or more, if tasks like calculating strip dimensions are done manually. To streamline the design process and promote the art form, we set out to develop a system that automates away these manual tasks.

Since this medium involves weaving strips in intricate ways, it is natural to build on existing visualization techniques for designing and composing Islamic star patterns. Much literature has been written on the subject; in particular, our process is based on Kaplan's approach [2], where geometrical configurations are formed by inscribing polygonal tilings with geometric motifs. While much of the existing artwork, like in Fig. 1, is based on traditional Islamic motifs, in theory there is no such restriction.

Such an approach is good for flat renderings of geometric star patterns. However, it is by itself insufficient for designing sliceform paper artwork. Inscribing motifs in tiles creates a visual illusion of long strips that weave in and out of each other, but the system has no notion of a contiguous strip (see Fig. 2).

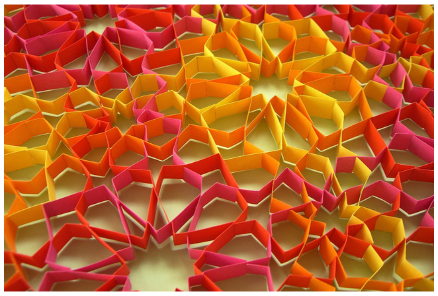


Figure 1: Festival (2010), by the first author.

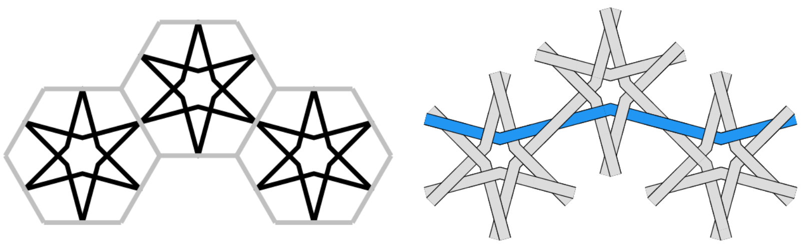


Figure 2: Here, polygonal tiles with inscribed motifs are composed via existing techniques (left) and rendered as an interwoven design (right). Our eyes see contiguous line segments, but the system cannot yet identify, for example, the highlighted strip as a single entity.

This paper presents an algorithm that traces patterns across polygonal boundaries to determine relative segment lengths and intersection positions, so that these patterns can be recreated in physical paper artwork. We have built a web application called Wallpaper [3] that implements this algorithm and enables users to rapidly and flexibly compose their own designs.

A pattern tracing algorithm

A typical geometric design consists of polygonal tiles inscribed with patterns joined together along their edges. Here the term *pattern* has a very specific meaning: a piecewise continuous line segment confined to the interior of the polygonal tile that starts and ends on the edge of the tile. Typically, tiles will have more than one pattern arranged in some symmetric manner, and our objective is to trace the orbit of patterns across polygonal boundaries.

In our implementation, each pattern stores relevant metadata about its geometry: the edges it starts and ends on, as well as its incident angle and relative position on these edges. It also keeps track of its internal geometric structure as a list of segments, where each segment is in turn a list of lengths between joints or intersections. See Fig. 3 for an example.

The first step in tracing patterns across polygonal boundaries is deciding which patterns at the boundary are contiguous and should be treated as part of the same strip. Our matching procedure works as follows: for each pattern p incident on an edge, the algorithm filters out the patterns from the other corresponding edge to those with relative position compatible with p. If the filter is empty, the pattern is not matched. If the filter returns a unique pattern, they are matched.

If there are multiple patterns originating from the same position, the algorithm picks the one with incident angle closest to that of p. It also checks that this is mutual, i.e., that p has incident angle closest to the other edge as well. If this is true, the two patterns are matched; otherwise, p is unmatched.

Note that our checks ensure that contiguity is a symmetric relation, such that even when the user joins two edges with incompatible pattern interfaces, the results match our visual intuition (see Fig. 4).

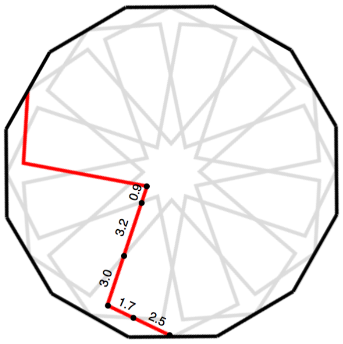


Figure 3: This rosette motif consists of 12 symmetric patterns. The highlighted pattern stores the list of list of lengths between line crossings corresponding to its internal structure from start to end edge, namely `[[2.5,1.7],[3.0,3.2, 0.9],[0.9,3.2,3.0],[2.5,1.7]]`.

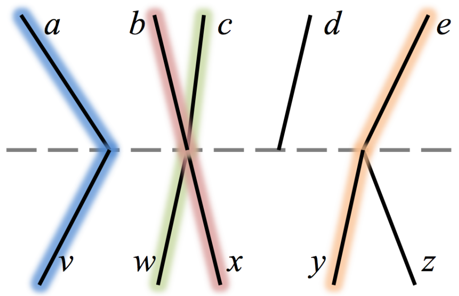


Figure 4: Example of joined edges with incompatible pattern interfaces. Here, consistent with our visual intuition, the matching algorithm returns (a, v), (b, x), (c, w) and (e, y). Patterns d and z are unmatched and their strips terminate at this edge.

Now that patterns are matched up via this criterion, tracing them to construct strips is straightforward. We pick a pattern on some tile arbitrarily, and start tracing across polygonal boundaries first from the entering edge, and then the exiting edge. We can then concatenate the internal list of segments stored by each pattern onto the strip we are building (reversing it if necessary to account for orientation) to build up a full strip. At every step, we also compare against the original pattern that we picked to detect if we enter a loop.

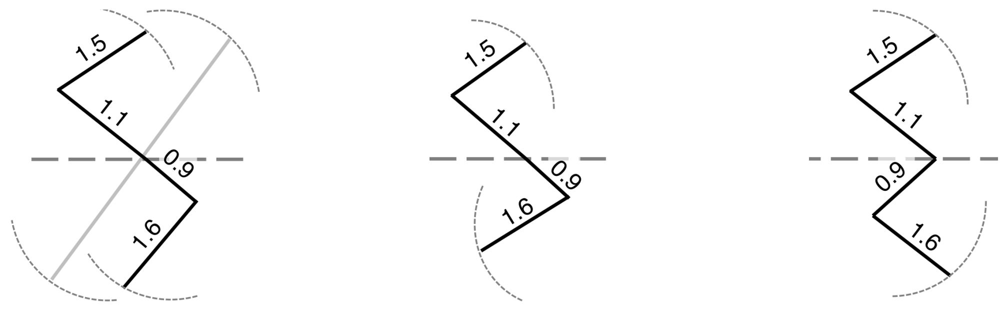
There are some subtleties about the concatenation process. We identify three possibilities and concatenate strips differently based on each case:

(a) if another pattern crosses that edge at the same point, the edge boundary is an intersection and we concatenate the relevant segments together.

(b) if no other pattern originates from that same point and the two incident angles are compatible (within ϵ of each other), the strip visually appears to be a single segment spanning the boundary and we add the lengths together.

(c) if no other pattern originates from that same point but the two incident angles don't match, the strip has a joint at the edge boundary and a simple concatenation of the two lists of segments suffices.

Fig. 5 below illustrates each of these cases.



(a) Output:
`[..., [..., 1.6], [0.9, 1.1], [1.5, ...],...]`

(b) Output:
`[..., [..., 1.6], [2.0], [1.5, ...],...]`

(c) Output:
`[..., [..., 1.6], [0.9], [1.1], [1.5, ...],...]`

Figure 5: Concatenation of patterns is performed differently depending on the edge boundary.

Now by repeating this process until every pattern in a polygonal tile has been assigned to a strip, we may convert a tile-based representation of the geometric configuration to a strip-based one.

Strip rendering in SVG

We have also developed a utility to translate between the data representation of a strip (a list of list of numbers) to a Scalable Vector Graphics (SVG) file representing that strip.

Given the dimensions of a strip, we can render it as a rectangle with vertical slits spanning alternately the top or bottom half.[1] Slotting strips along these slits provides a secure interwoven structure. Joints, or creases, are rendered as full vertical lines in a separate color, so that the user can choose to fold them manually or score them by setting the laser cutter to a different power setting. Fig. 6 shows an example of one such generated strip.

This SVG file can either be printed and cut out by hand, or cut and scored automatically using a die cutter or a laser cutter.

[1]Under relatively weak conditions – all strips terminate on the exterior of the tiling – we can assign crossings at intersections such that each strip always alternates over and under.

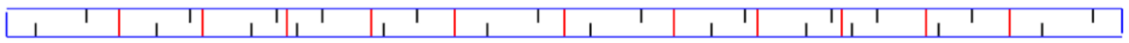


Figure 6: Strips rendered by our system based on the traced dimensions, ready for die-cutting / laser-cutting and physical assembly.

Conclusion

Conventional Islamic star pattern design techniques are very successful in generating symmetrical and aesthetically pleasing geometric configurations. We have demonstrated a system that enables us to physically recreate these geometric configurations with paper strips, by taking a tile-based data representation of such designs and converting it into a strip-based representation.

Brimstone (see Fig. 7), demonstrates the utility of this system — design and strip generation was completed in an hour, when such an intricate configuration would previously have taken much longer to trace and convert by hand.

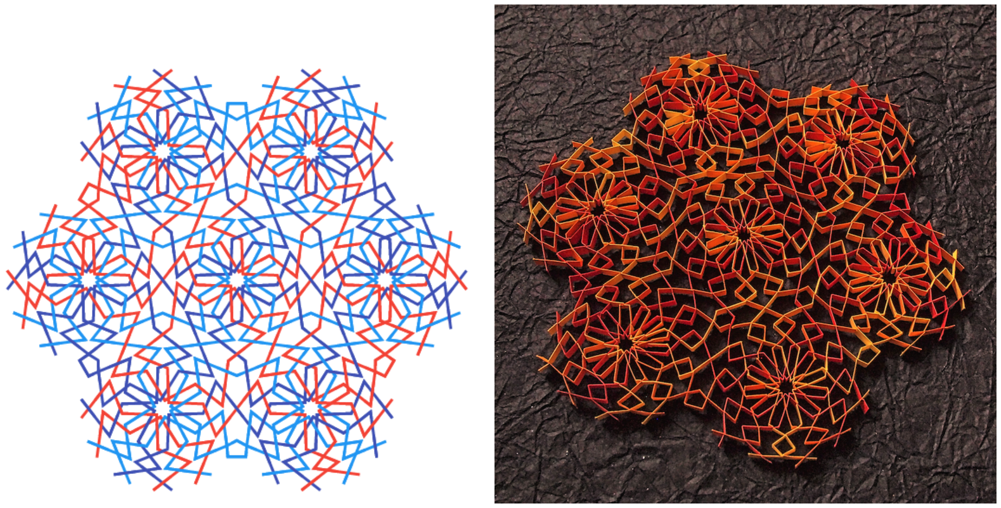


Figure 7: Brimstone (2015). The geometric design (left) was traced to generate paper strips, which was assembled to produce sliceform artwork (right).

References

[1] Christiane Bettens. "Zillij: slice form techniques applied to patterns from arabic art.", 2009. `https://www.flickr.com/photos/melisande-origami/sets/72157613125224450/`, accessed 04-22-2015.

[2] Craig S Kaplan. Computer generated islamic star patterns. In *Proceedings of Bridges 2000*, pages 105–112, 2000.

[3] Yongquan Lu and Erik Demaine. Wallpaper, 2015. `http://yqlu.me/wallpaper/`, accessed 04-22-2015.

[4] Chris Palmer and Jeff Rutzsky. "Zillij", cardstock, 12″, 2009. `https://www.youtube.com/watch?v=2TNUxWVgZTs`, accessed 04-22-2015.