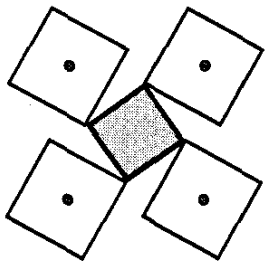

Computer Tools and Algorithms for Origami Tessellation Design

Alex Bateman

1 Introduction

Origami tessellations are a genre of origami that started in the late 1960s with models such as Momatani's stretch wall and later works by Shuzo Fujimoto in his book *Seizo Soru Origami Asobi no Shotai (Creative Invitation to Paper Play)*. More recently, Chris Palmer has brought the art to new heights. The mathematics of the topic have been developed by a number of people including Toshikazu Kawasaki, Helena Verrill, Thomas Hull, and Robert Lang.

To date, there are very few tools available specifically for the design of origami models. The most notable example is TreeMaker by Robert Lang, which has been used to design complex origami bases for constructing insects, animals, and even an Allosaurus skeleton. In this paper, a new program called Tess that can be used to design origami tessellations is presented.

Before the discussion of the program itself, it is necessary to introduce the concepts of crease pattern, folded pattern, and light pattern, A crease pattern is the set of all creases that can be used to fold a piece of paper into a flat-foldable origami tessellation. An example crease pattern is shown in Figure 1(a). A folded pattern shows the folded origami tessellation, where the paper is completely transparent and only the crease lines are shown. This represents an approximation of what is seen when holding the folded crease pattern up to a bright light. This representation is a computational construction that is useful because one does not need to consider the layers of the paper; see Figure 1(b) for

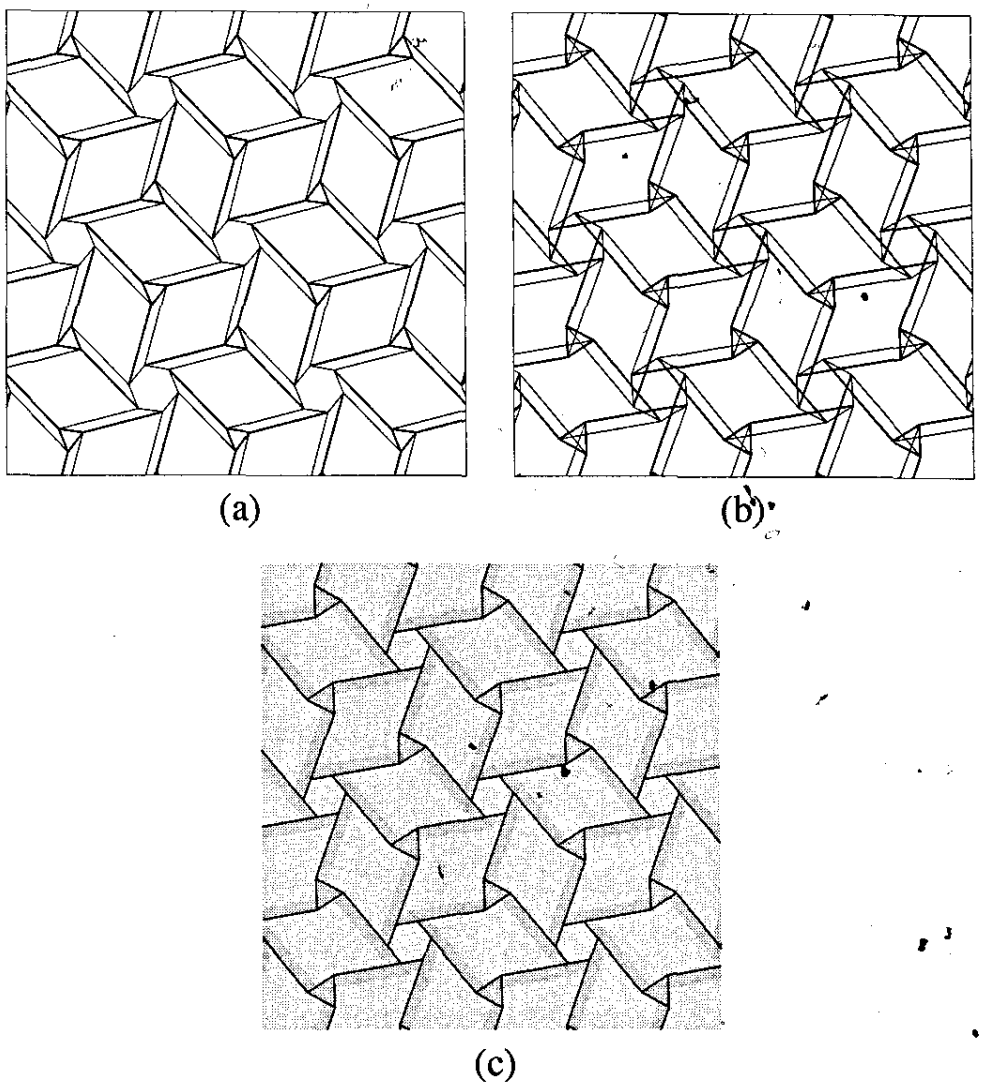

Figure 1. (a) An origami crease pattern. (b) A folded pattern. (c) A light pattern.

an example. The final pattern that we are considering is the light pattern, which shows what an origami tessellation made from semitransparent paper looks like when held up to light. For an example of a light pattern, see Figure 1(c).

2 Algorithms

The key algorithm used in origami tessellations transforms a tiling of the plane into a flat-foldable crease pattern. This algorithm has been developed and used by people in the field such as Paulo Barreto and Chris Palmer, and the algorithm presented here is based on their work. I will describe this algorithm including the parameters used to enumerate the possible crease patterns. This algorithm and the correct choice of parameters leads to the novel result that it can also generate a representation of the folded pattern. Given the folded representation, I present an algorithm that can generate the light pattern of the origami tessellation.

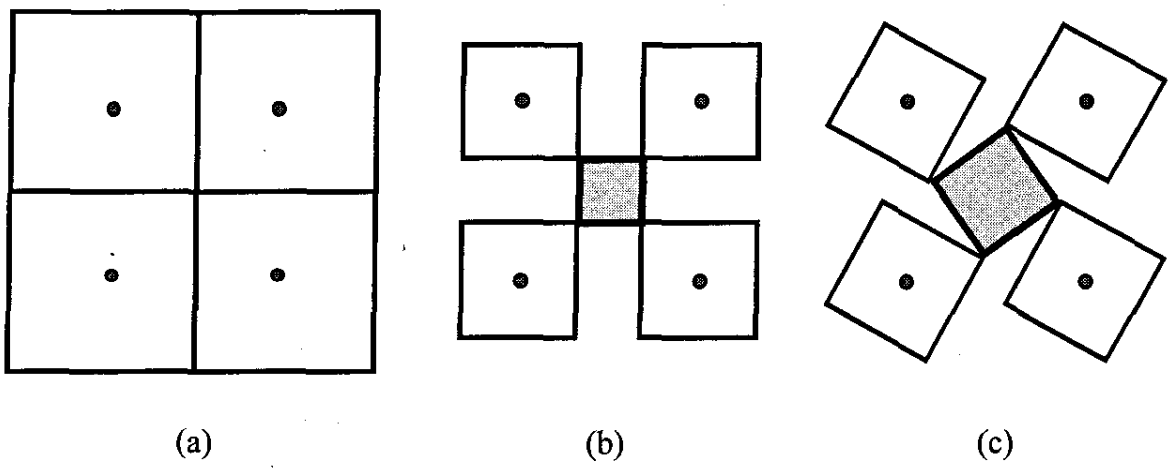

Figure 2. The three steps used to generate an origami crease pattern from a tiling. (a) Generate tiling; the filled dots are the origin of scaling (derived from the vertices of the orthogonal dual of the tiling) and rotation in Steps 2 and 3. The filled dots are linked for a dual of the original tiling. (b) Scale polygons of tiling, and generate a new "baby" polygon, shown shaded. (c) Rotate polygons to generate final pattern. The "baby" polygon vertices do not need to be explicitly calculated as they coincide with the vertices of the original tiling.

2.1 Crease Pattern Generation

Origami tessellations are generated from a tiling of the plane. This tiling must have a dual such that every edge in the tiling is orthogonal to its corresponding edge in the dual. This dual tiling is called the orthogonal dual. Figure 2 shows the steps involved in creating an origami tessellation crease pattern from a tiling for the archimedean tiling 4^4. Given a tiling, each tile must be scaled and rotated, and the origin of scaling and rotation for each tile is defined by the location of the corresponding vertex of the orthogonal dual. In the algorithm presented here, all tiles are rotated and scaled by a uniform amount; however, this is not a requirement to make an origami tessellation. Different tiles in the tiling can be scaled and rotated by different amounts, but this makes the mathematics more complex. During the scaling step, new baby polygons are produced. These polygons are formed by joining the vertices of tiles that are adjacent in the original tiling. The natural choice of parameters would seem to be the scale factor (α) and rotation angle of each tile (θ). However, I use the ratio of the lengths of the tile compared with the length of the edge of the baby tile (α/γ) and the pleat angle (ϕ) (see Figure 3). There is a simple relationship between these two parameter sets, shown in the equations below:

$$\theta = \arctan\left(\frac{1}{\tan\phi + \left(\frac{\alpha/\gamma}{x\cos\phi}\right)}\right);$$

$$x = \frac{2\sin(\pi/n_1)\sin(\pi/n_2)}{\sin(\pi/n_1 + \pi/n_2)};$$

$$\alpha = \frac{1}{\cos\theta + \sin\theta\tan(\theta + \phi)}.$$

Here n_1 and n_2 are the number of sides of the polygons at each end of the pleat being described by the parameters under discussion. So, for the tiling in Figure 2, n_1 and n_2 equal 4.

2.2 Folded Pattern Generation

The same algorithm that is used to generate the crease pattern of a tiling can be used to generate the folded pattern. The choice of parameters discussed above allows for a simple relationship between the parameters for the crease pattern and the folded pattern. To generate the folded pattern, we simply negate the pleat angle. Thus, for a crease pattern with a pleat ratio of 25 degrees, we can generate the folded pattern by using a value of -25 degrees. This result is rather surprising, but it allows us to view origami tessellations as they will look once folded. This is a great aid in design of these tessellation patterns.

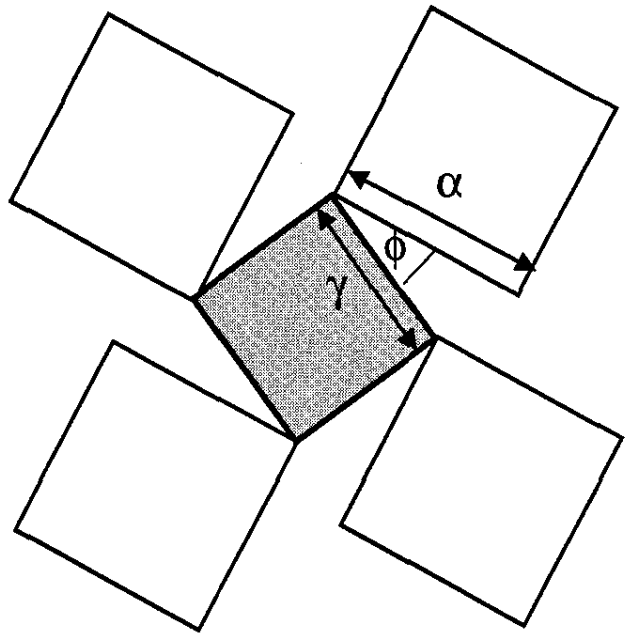

Figure 3. The parameters used to specify origami tessellations. Once a tiling has been chosen, two parameters can be used to describe the simplest type of origami tessellation. The parameters used are the pleat angle, called ϕ, and the pleat ratio (α/γ). α is also the scale factor by which the original tile has been scaled. Thus, an α of 0.5 means that is half the size of that in the original tiling.

2.3 Light Pattern Generation

The conceptually simplest algorithm to generate a light pattern is to consider each pixel in the folded representation and count the number of polygons in which it lies. (This is equivalent to finding the number of layers of paper at any point.) This value is then used to assign a grayscale value to the pixel. For pixels with one layer, a light gray is assigned, and darker grays for thicker regions. This algorithm requires a comparison for every pixel in the pattern. If n is the number of pixels in one dimension, then the complexity of this algorithm is $O(n^2)$. This means that doubling the resolution causes the time needed to calculate the light pattern to quadruple. In contrast, I have implemented an algorithm that requires only $O(n)$ computations. This algorithm considers one horizontal line in the pattern at a time. For polygons that intersect this line, we know that at the leftmost intersection, we need to increment the number of layers, and at the rightmost intersection point, we should decrement the number of layers. This algorithm will calculate a high-resolution light pattern in less than a minute.

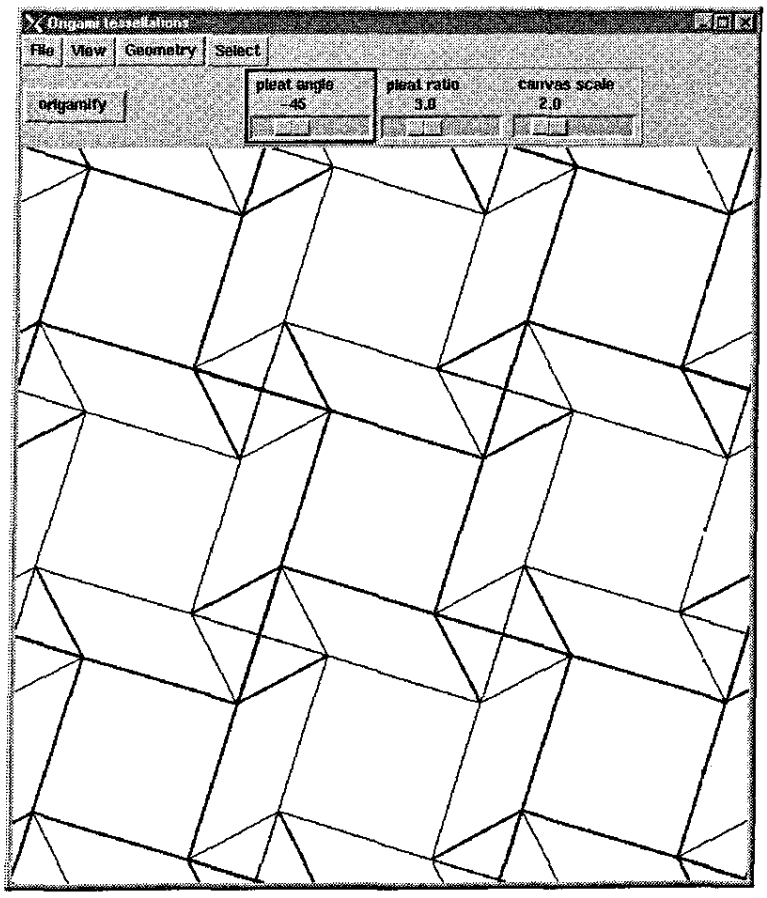

Figure 4. A screen shot from the GUI version of the Tess program. Note that the pleat angle is -45, and therefore shows a folded pattern.

3 The Tess Program

The algorithms described in the previous section have been implemented in a computer program called Tess, which has been written using object-based PERL. The software is freely available and can be downloaded on the web at: http://www.sanger.ac.uk/Users/agb/Origami/.

The Tess program can be used in two different ways: A command line interface can be used, or, if PerlTK is installed, a Graphical User Interface (GUI)-driven version can be used (see Figure 4). The object model used in the program is represented in Figure 5.

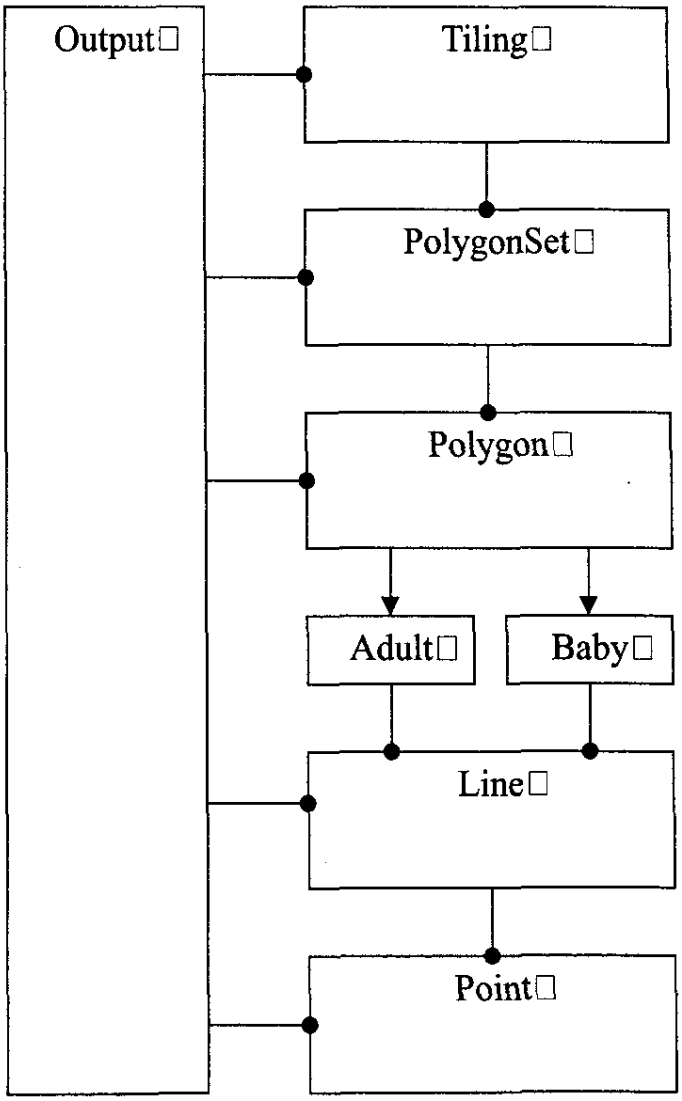

Figure 5. The object model of the Tess program. Container objects are joined to other objects by a line with a circle at its end. Inheritence relationships are shown with arrows. A tiling object contains a collection of polygonset objects, for example. Adult and baby objects are subclasses of the polygon object. The output object is a list of objects to be rendered.

4 Conclusions

I have shown that an algorithm developed by others can also be used to generate representations of the folded origami. This can either show the crease lines or the light pattern of the tessellation. The use of these algorithms in the Tess program allows the design of complex new origami tessellations in silicon, moving the origami practitioner a step closer to the paperless office.