origami draft 1

Benjamin Wettle

March 2020

1 Introduction

This Algorithm uses box pleating methods to design optimal crease patterns for any given origami design.

2 Background

Origami is the ancient art of folding paper to represent various forms. Traditional artists found step by step instructions to fold the base, and then added details. These designs were found by experimentation, and modification of existing designs. More modern artists such as Robert Lang [1] have developed algorithms to generalize this process to work for arbitrary bases.

2.1 Definitions

In order to design an origami base, we first have to define a notation to describe it. The paper used is assumed to be a rectangle of negligible thickness. Though no such perfect paper exists in reality, origami paper is thin enough to ignore for most designs. Most origami artist use a square paper for designs, but the definitions and algorithm are the same for the more general case of a rectangle. For a box pleated design, the paper is folded into a grid pattern, with equally spaced lines running vertically and horizontally. In a final design, some of these lines will not be folded, as they can clutter the artistic style of the design. They are use full in the description and optimization, as they define the placement and areas of the nodes.

A leaf node is the simplest element of a base. This represents a loose flap of paper, like the example shown in Figure 2. These flaps can be easily folded to represent object like arms, fingers, wings or even ears. This versatility allows wildly different designs to be made from the same simple structure of paper.

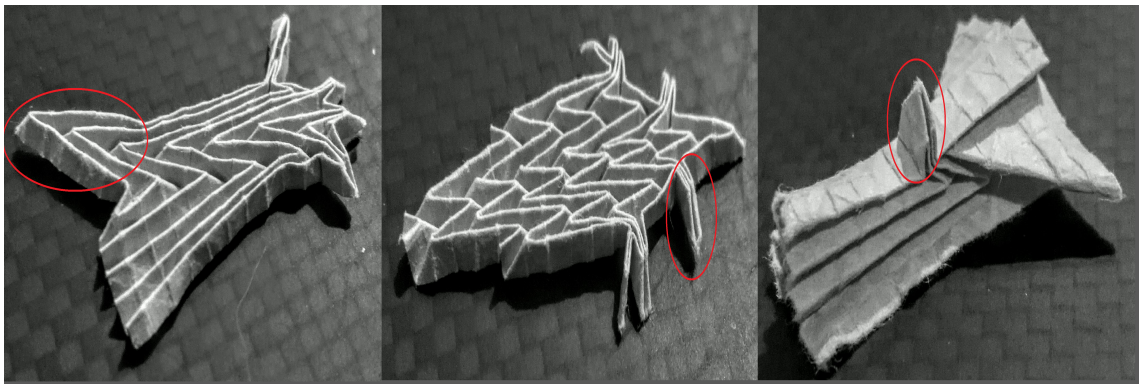

Figure 1: Leaf Node examples, (a) (b) (c)

Figure 2: Figure 3, (a) (b) (c)

Figure 3 shows the same leaf nodes in a plan view. The red shaded area shows the paper used to make each node. The minimum paper required (in a box pleated design) is a square centered on the tip of the leaf node. The square must be twice the size of the flap, on each side. The square can be partially off the paper, as in figure 2(b), but the center must be on the paper to be folded. This area cannot overlap with the area used for other nodes, or the paper would fold into a loose flap.

Figure 4 shows two leaf nodes positioned on the same paper, in folded and unfolded states. From this, we can see a simple rule to find if two nodes overlap. The minimum distance is determined by adding the sizes of the two nodes together. Either the difference in X or Y coordinates must be equal or greater than this minimum distance.

The second kind of node is a river node. these nodes separate two groups of nodes from each other. Figure 5 shows five leaf nodes, with three small nodes on one side of a river node, and two larger leaf nodes on the other side. Figure 5 also shows the area allocated to the river node. this area must surround the

Figure 3: Figure 4, (a) (b)

leaf nodes on one side, and must be of a constant thickness, even when it turns directions. This thickness is the same size as the size of the river node.

2.2 problems with circle packing

These representations are not the true minimum space necessary to fold these nodes. Instead, circles could be used for the leaf nodes. a river could be represented by a curved space, instead of a gird based polygon. these representations are smaller than box pleating, which could reduce the overall wasted space. there are a few reasons why box pleating is a good alternative to circle packing though.

First, the wasted space is only potential. It is highly unlikely that all or even most of the "corner" space would be used in a circle packed design. figure(6) shows a simple origami base. figure 7 shows designs with circle packing and box packing on the right and left respectively. the loss in efficiency is far from the theoretical maximum, and is out weight by concerns outside of space efficacy.

folding time for circle packed designs for many circle packed designs, the locations of creases are arbitrary, and not easy to fold in practice. Tools like Reference finder can find points and lines through simple folds. this adds extra layers of complexity and can clutter designs with excessive creases. with a box packed design, the folds are all at multiples of 45 from each other. this is easily fold able, and requires no extra tools to fold. in addition, the flaps produced are standardized. with a circle packed design, a flap could have a range of possible angles and thicknesses, all or which impact the creative versatility of the flap. with a box pleated design, detailing can be experimented with before the full base is done. this standardization also makes the actual folding process easier and less open to error.

3 Approach

The GUI allows the user to enter in a design, and any constraints then need. The optimization algorithm needs (at it's simplest) two pieces of information. The first is a list of leaf nodes. River node's locations are determined in the crease generation step, so they will not be optimized. The second piece of information is the minimum distances from each leaf node to the others. This is found beforehand, using a simple breadth first search. constraints can also be placed on nodes positions. If two nodes are paired about one axis, then one is added to a list of dependant nodes. when it's partner is optimized, the dependant node is also added. Nodes can also be set on an edge, or on the line of symmetry. This line is set as either the x axis or the y axis.

Additionally, there are several pieces of information that modify the optimization algorithm. A larger buffer can make the algorithm search a larger area.

This slows the speed significantly, so it is set to 0 by default. If the design needs to be optimized to a rectangle of certain proportions, then the algorithm can be set to weight different axis accordingly.

Exhaustive search Then comes the core question of the optimization- where is the best place for these nodes to go? The first node is easy to place, since all distances are relative. It can be placed at any point, such as (0,0). Since optimizing a single node is trivial, we must also account for the rest of the nodes.

The second node could go many places, as long as it meets the minimum required distance. At this point, we don't know were the best spot is, so we will place it in several spots, for different potential designs. A square with sides of $2 * (minimumdistance + buffer)$ is searched for potential locations. This square is searched around each of the nodes already in the design. Initially this buffer is set to zero, in the hope that the node can be found close by, without many gaps in the design. Of the spaces, those overlapping the first node can be automatically eliminated. this gives $4 * ((minimumdistance + buffer) \exp 2 - (minimumdistance - 1) \exp 2)$ placements to check. Then, we check to make sure the new node does not overlap any other nodes. If it does not, we know that is a possible design.

for each of the nodes left, the algorithm checks all possible placements for the next node, and repeats. Once all the nodes are added, the best one is selected, based on the minimum size. This method of search is quite slow, as a huge number of nodes must be searched. For a design with n leaf nodes, each new node added make on the order of n new designs. each must be tested for overlap wit all the other nodes, which takes an order of n as well. since this is is repeated recursively for all nodes, the algorithm takes an order of $(n^2)^n$

This is too slow to handle large numbers of nodes. Fortunately, it can be improved greatly. instead of searching exhaustively, the algorithm searches with a depth first search. By tracking the best final design found, and it size, many incomplete designs can be pruned from the search. Only if the size is smaller should the design be considered at all. Designs of equal size are stored, and can be checked for minor improvements at the end. Additionally, the designs can be scored as they are built. The score is simply the sum of all the gaps in the design. If an incomplete design already has more gaps than the best, it can be pruned for the search. Designs can also be eliminated by size and score before being checked for overlap, to reduce the wasted time. Some designs are simply rotated or mirrored versions of each other, and these can be eliminated early. Once the best design is found, the crease pattern can be generated. if the design has no overlaps, this patterns will collapse into the base. First, a grip of mountain and valley creases is made. the valley creases follow the grid lines of the square. in between these runs mountain folds, to form the box pleated structure. In practice, some of these line are unnecessary. For display purposes,

it is useful to have all of them for guidance on the diagonal creases. for these creases, the areas allocated for each node is found. leaf nodes are found by forming squares, as in the definitions section. river nodes are made by adding borders to the nodes they enclose. Once the areas are complete, there may be some unused space left over. this is saved in a excess node, which is not part of the design. for all the areas, the angle bisectors are found. the bisectors are moved sequentially inwards, until they meet or run out of space.

4 Evaluation

TODO compare time and efficiency of a few designs to tree maker. Ive run into a small problem here, as treemaker is very picky on the desings that it will try to optimize. This is one big advantage of mine- it will find a way for anything at all.

5 Related work

mention robert lang, tree maker, reference finder, and origami design secrets .

6 Conclusion

I'll write the conclusion and abstract last.