Warping Pictures Nicely

David Swart
Christie Digital Systems
Waterloo, Ontario
david.swart@christiedigital.com

Abstract

We present a new algorithm, using only simple geometry, to warp imagery from an arbitrarily shaped source region to an arbitrarily shaped target region. Mathematically speaking, the algorithm outputs harmonic maps (every point is at the average position of its neighbors) using new boundary conditions to curb excessive non-uniform stretching and shearing in order to appear more conformal. The algorithm has typical running times measured in seconds. We give some artistic examples to demonstrate how the results can be used in digital photography and other graphical work.

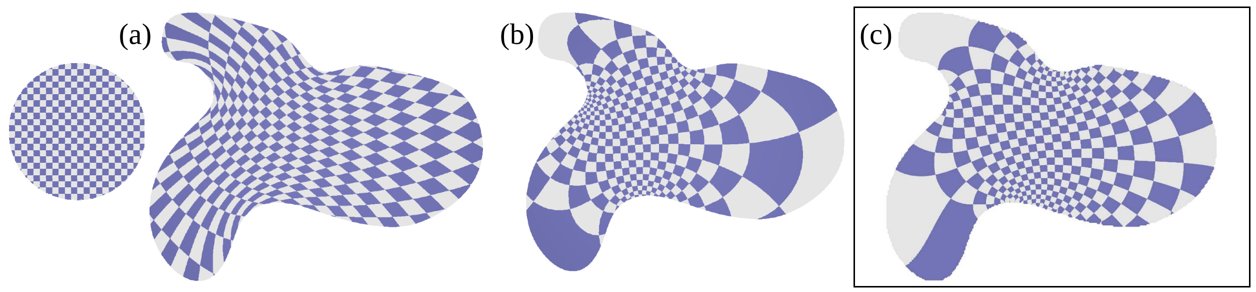


Figure 1: *(a) A harmonic map with Dirichlet (fixed position) boundary conditions from [7]. (b) A discrete conformal map, also from [7]. (c) A harmonic map using our new boundary conditions.*

1. Introduction

Photo editing software has a lot of functionality (filters, plug-ins, actions, etc.) devoted to warping and distorting imagery for artistic effect. In the mathematics, *harmonic* maps create warps that are smooth and even without creases or bumps, *conformal* maps are maps that create warps that do not stretch, compress, or skew the image. (Stretching by the same amount in every direction is allowed.) Let us review some points about harmonic and conformal maps before describing the contributions of this paper.

1.1 Harmonic Maps. Harmonic maps are functions that satisfy the *Laplace equation*: $\Delta f=0$. In the case of harmonic functions of the form, $f(x,y) = z$ (i.e., f: $\mathbb{R}^2 \to \mathbb{R}$), what the equation $\Delta f(x,y)=0$ means is that (x,y,z) is a saddle point where the upward curvature is the same as the downward curvature. However, to warp an image from one 2D region to another, we use functions of the form, $f(x,y) = (u,v)$, or f: $\mathbb{R}^2 \to \mathbb{R}^2$. In this case, f is harmonic if $f_u(x,y) = u$, and $f_v(x,y) = v$ are.

Scientists and engineers use harmonic functions to model various phenomena. Usually there is something known about the boundary conditions of the domain (it is called a Dirichlet boundary condition if the position is fixed). The problem is to find a harmonic function in the interior. *A simple way to discretely calculate harmonic functions is to define a mesh that covers the domain, fix the positions of the boundary, and then iteratively set each vertex to the average position of its neighbors.* This algorithm relies on the property that each point in a harmonic function's domain is the average of its neighbors.

We can use these harmonic maps to warp an image by first setting the boundaries of a mesh to the perimeter of a target shape, solving for the harmonic function and then afterwards using the harmonic solution to do a piecewise interpolation between quadrilaterals. Figure 1(a) shows a harmonic mapping. Here, the boundary is set such that the arc-length distance is the same as on the disc: the disc is a rubber sheet glued to a wire frame and then distorted into the blob shape.

1.2 Conformal Maps. Conformal maps are maps that preserve angles at a local level. For instance, if two curves meet at, say, a 45° angle in the original, then their images under a conformal mapping will also meet at 45°. See Figure 1(b). For $\mathbb{R}^2 \to \mathbb{R}^2$ functions, conformal maps are harmonic. Qualitatively speaking, due to preserving angles, a conformal map does not induce any non-uniform squishing, stretching or skewing. Small shapes still retain their shapes. Although straight lines can become curved, they will intersect the region's boundaries (and each other of course) at the same angles they did in the original. We can typically expect conformal maps to induce some variations in scale and orientation. But often there are hardly any overt signs that an image manipulation has taken place.

With a keen understanding of conformal mathematics, some very fascinating art has been produced (see Bulatov [1]). And certainly conformal art has been realized by artists with a more intuitive sense of the qualities of conformal mappings: de Smit and Lenstra [3] describe the mathematics of M. C. Escher's *Print Gallery* which was produced without explicit knowledge of conformal geometry.

Like harmonic functions, conformal geometry has a vast and rich history with many useful applications in many fields such as physics, engineering, and cartography. The Riemann mapping theorem proved the existence of a conformal mapping between two arbitrary simply connected regions (that are not the entire plane) and the Schwarz-Christoffel formula is the standard tool for achieving these maps (see [4]). However, Schwarz-Christoffel maps can be difficult to implement involving sophisticated mathematics and numerical methods that are susceptible to thorny floating-point arithmetic problems. In computer graphics, there has been a great deal of recent work on discrete conformal approximations that avoid these numerical issues. See [2], [7] and [8] and the many references therein. A typical application of these methods would take a triangular mesh in three dimensions, and then determine a conformal mapping to a 2D mesh in the plane, allowing a convenient tool to apply a texture map to a 3D model. Our algorithm focuses on mapping an image directly to a specified 2D shape. One of the main benefits it has over existing conformal methods is its ease of implementation.

1.3 Contributions. In this paper, we provide a way to generate harmonic mappings from a given region to another region with a prescribed shape that appears conformal in many common situations (i.e., it reduces the amount of shearing and stretching we typically see in harmonic functions with fixed boundaries). See Figure 1(c). The algorithm itself is a modified version of the harmonic function algorithm mentioned in Section 1.1. However, we use a different boundary condition: instead of fixing the boundary points, they are free to slide along the perimeter of the target shape, shifting in whichever direction results in a more conformal edge.

With a simple iterative step, quick convergence, and results that are pleasing enough for digital photography and graphical applications, we hope that this algorithm will be of use to programmers, mathematicians, and artists who may have been previously too shy to take on some of the current conformal techniques found in mathematics.

In Section 2 we explore some popular conformal manipulations to get a sense of the appeal of conformal maps. We discuss the details of the algorithm itself in Section 3. In Section 4 we show some results as well as discuss some limitations of this method. Section 5 shows some artistic examples using some new mappings. Finally, Section 6 discusses some future direction this work might go.

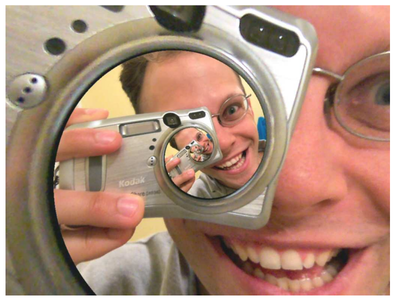
2. Existing Conformal Mappings

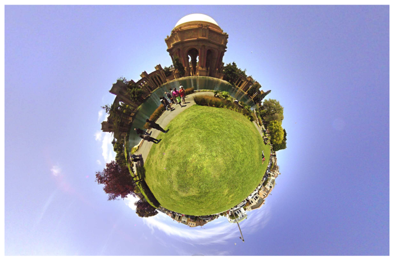
To appreciate the qualities of conformal mappings in digital photography, we compare some popular conformal maps (Figure 2), with some non-conformal counterparts (Figure 3).

Figure 2(a) shows an example of a class of conformal maps known as the Droste effect (see [3]). One of the striking features about this effect is that as your eye moves around the spiral, there is no seam between the apparently infinite copies of each element in the scene.

Germán et al. [5] discuss how conformal mappings play an important role when displaying full panoramas or *visible spheres* on a plane. Figure 2(b) shows a *little planet*: a popular method of displaying a visible sphere using a (conformal) stereographic projection. By tilting the visible sphere, the projection's changes of scale are used as a compositional tool to emphasize the subject.

If we wanted to change something’s rotational symmetry, we accomplish this in polar coordinates (θ, r) conformally by scaling θ by the desired factor x, and then raising r to the xth power. So for instance, if we wanted to change the rotational symmetry of a snowflake from six to two, we would scale θ by three and then cube r. Figure 2(c) shows twelve examples.

(a)


(b)


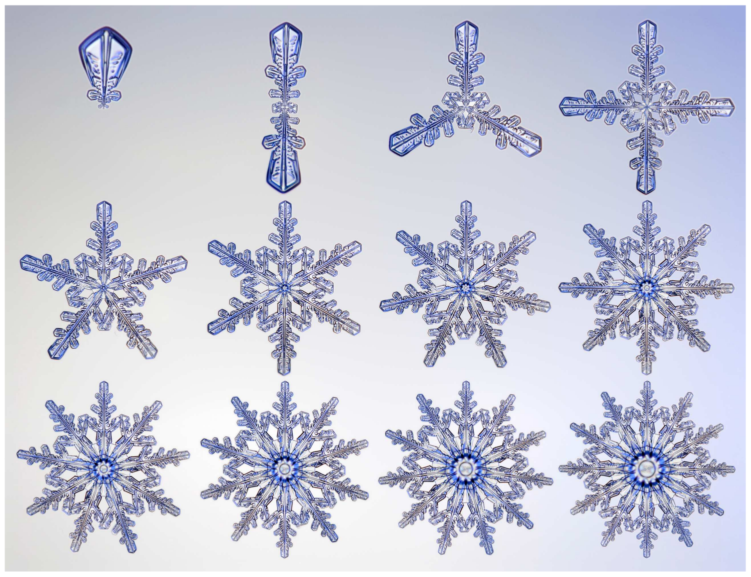
(c)


Figure 2: *(a) The Droste effect. (b) A conformal little planet taken at the Palace of the Fine Arts in San Francisco. (c) Conformally mapping a snowflake to orders of rotational symmetry 1 through 12.*

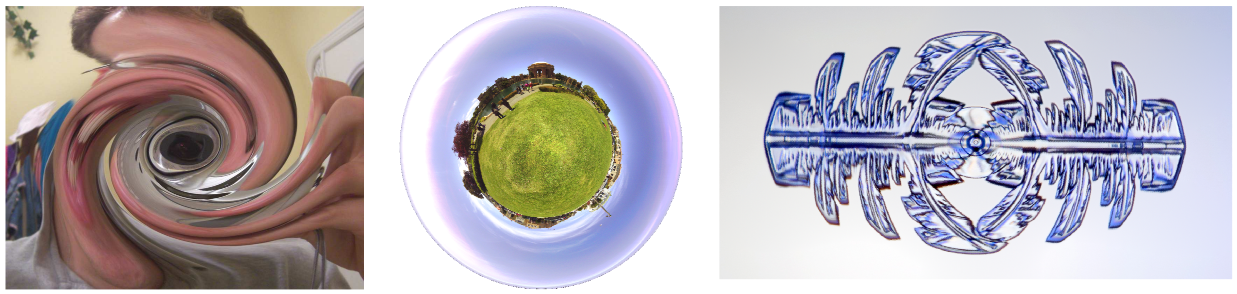
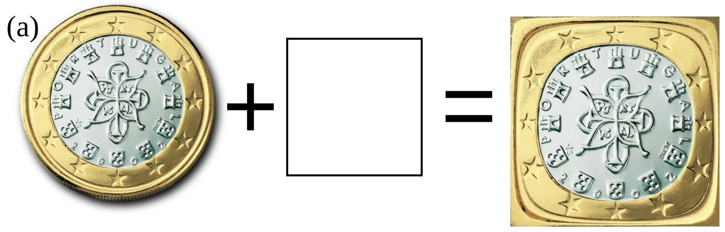


Figure 3: *Non-conformal attempts at the examples in Figure 2.*

3. The Algorithm

Our ultimate goal is to take an image of an arbitrary shape and to warp it nicely to another target shape. See Figure 4(a). If we overlay a sufficiently fine mesh onto the target image and have a corresponding mesh in the original, then we can map points from the quadrilateral faces of one mesh to the other's by using a convenient interpolation scheme of our choosing. The algorithm's role is to output the positions of the original mesh's vertices in the target region.



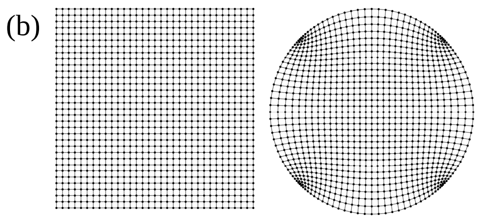


Figure 4: *(a) Given an image (of a euro) and a target shape (square), map the image of the euro to the square. (b) An inverse mapping that accomplishes this.*

Typically when we render an image, we have the location of each pixel in the target and we want the location (and color) of its corresponding pixel in the original. Therefore in practice, we are always concerned with the *inverse* mapping. So for example, when we want to map the circular euro to a square, we need the inverse mapping of a square to a circle shown in Figure 4(b).

The initial mesh that we simulate is a square lattice corresponding to the shape of the original imagery. Next, we associate the boundary points of the mesh to the target shape's perimeter. The boundary points will be constrained to this perimeter. By *constraining a point* we simply mean that if a point is not already on its associated boundary, then we move its position to the closest point of that boundary. The current implementation supports boundaries made up of arc and/or line segment primitives. However, there is no reason why we could not use other curves.

For any vertex v, we associate a position p, an orientation θ, and a size r. Now imagine a plus-sign, centered at p, with orientation θ, and with arms length r: we can impose a conformal condition by minimizing the distance between the endpoints of the plus-sign arms, and the center position of the corresponding neighbor.

Our algorithm iteratively selects a random vertex v in the mesh and tweaks its parameters (p, θ, r) until the mapping has converged. The pseudocode is listed in Table 1.

To tweak an *interior* point, (i.e., the randomly chosen vertex v has all four neighbors), p is simply set to the average of its neighbors. We calculate θ and r in case they are needed later, but they do not impact p directly: θ is calculated such that the plus-sign arms approximate the direction to v's neighbors and r is set to the average distance to v's neighbors. See Figure 5.

```
given vertex $v = \{p, \theta, r\}$, and $v$'s neighbors $N, E, S, W$
$p' \leftarrow (0,0)$, $\theta' \leftarrow 0$, $r' \leftarrow 0$
if both $N$ and $S$ exist, then:                    //$p$ is in interior
    $p' \leftarrow p' + N_p + S_p$
    $\theta' \leftarrow \theta' + 2 \times$ (dir. of $(N_p - S_p) - \pi/2$)  //$\theta$ is $-\pi/2$ from $N$-$S$
    $r' \leftarrow r' + |N_p - p| + |S_p - p|$
else if $N$ exists but not $S$ then:               //$p$ is on boundary
    $p' \leftarrow p'$ + the endpoint of south arm of $N$
    $\theta' \leftarrow \theta'$ + direction of $(N_p - p) - \pi/2$    //$\theta$ is $-\pi/2$ from $N$
    $r' \leftarrow r' + |N_p - p|$
else if $S$ exists but not $N$ then:               //$p$ is on boundary
    $p' \leftarrow p'$ + the endpoint of north arm of $S$
    $\theta' \leftarrow \theta'$ + direction of $(S_p - p) + \pi/2$    //$\theta$ is $\pi/2$ from $S$
    $r' \leftarrow r' + |S_p - p|$
do the above if statement analogously for $E$ and $W$
divide $p'$, $\theta'$, and $r'$ all by the number of neighbors of $v$
if $v$ is constrained then move $p'$ to its boundary's closest pt.
$v \leftarrow \{p', \theta', r'\}$
```

Table 1: *Pseudocode to tweak a vertex v*

If *v* is missing a neighbor, then we consider it on the *boundary* of the target shape. To impose the conformal boundary condition, we use the endpoint of the plus-sign arm of the neighbor opposite the missing neighbor. After *v*'s parameters are tweaked, *v* is constrained by moving *p* to the nearest point on the target shape's perimeter. See Figure 6.

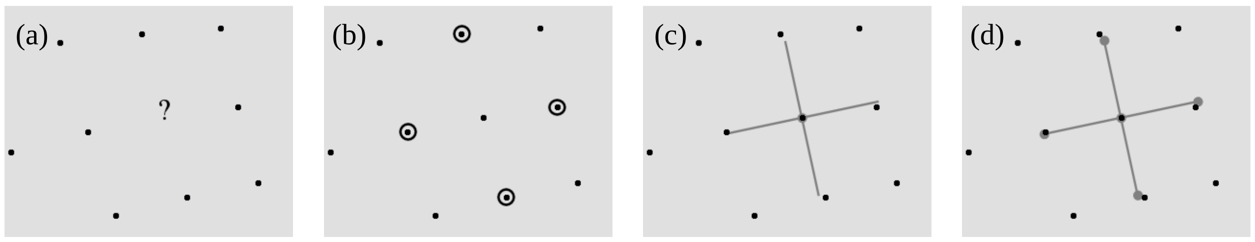


Figure 5: *One step of our algorithm (a) choose a random vertex v. (b) Position p is set to the average position of v's neighbors (circled). (c) The best-fit orientation θ is determined. (d) Finally, the size r of the plus-sign arms is set to the average distance to v's neighbors.*

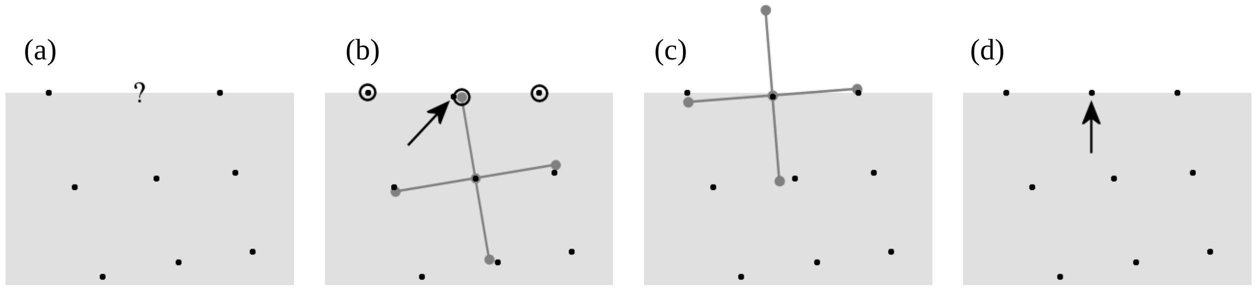


Figure 6: *(a) A boundary condition. v is missing its northern neighbor. (b) The north endpoint of the southern neighbor (circled) is included to calculate p (arrow). (c) θ and r are calculated as before, omitting the missing neighbor. (d) v is moved to the nearest position on the edge*

To decide when to halt the process, we define a metric of conformality *C* as the sum of the distances from each plus-sign arm's endpoint to its corresponding neighbor's position (normalizing these distances by dividing by the arm's size). A simple test for convergence stops the program when the change in *C* after 10,000 iterations is less than a threshold (we chose 1.0). Run times are typically less than ten seconds for a 32x32 mesh and less than a minute for a 64x64 mesh. Once the iterations have stopped, the mesh can be output, and the final step to map imagery from the one mesh to the other completes the task.

4. Results

With this tool at our disposal, we can start creating maps between shapes. The meshes for the original imagery can approximate almost any shape we choose, even taking in a bitmap mask to select which mesh points to keep. Figure 7 shows four examples.

For different *destination* shapes, our final meshes are mapped to regions bounded by line segments and arcs. Figure 7 shows the circle and the square as target shapes and Figure 8 explores some other possibilities. Note that the resulting harmonic maps are not always perfectly conformal: we see that with some shapes such as Figures 8(a) and 8(c) we can see decidedly non-square rectangles in the resulting meshes.

The heart in Figure 8(c) was created using *piecewise constraints*. That is, the boundary points on two of the sides were constrained to the top bumps and not allowed to slide along down the sides. The effect of these artificial piecewise constraints can be significant! Figure 9 shows five parallelograms mapped to a square checkerboard test pattern. Figures 9(a) and (b) are harmonic functions using different Dirichlet conditions. Figures 9(c) and (d) show our approach using piecewise constraints. Figure 9(e) shows our method with no piecewise constraints. That is, boundary points are free to slip around the target shape's corner.

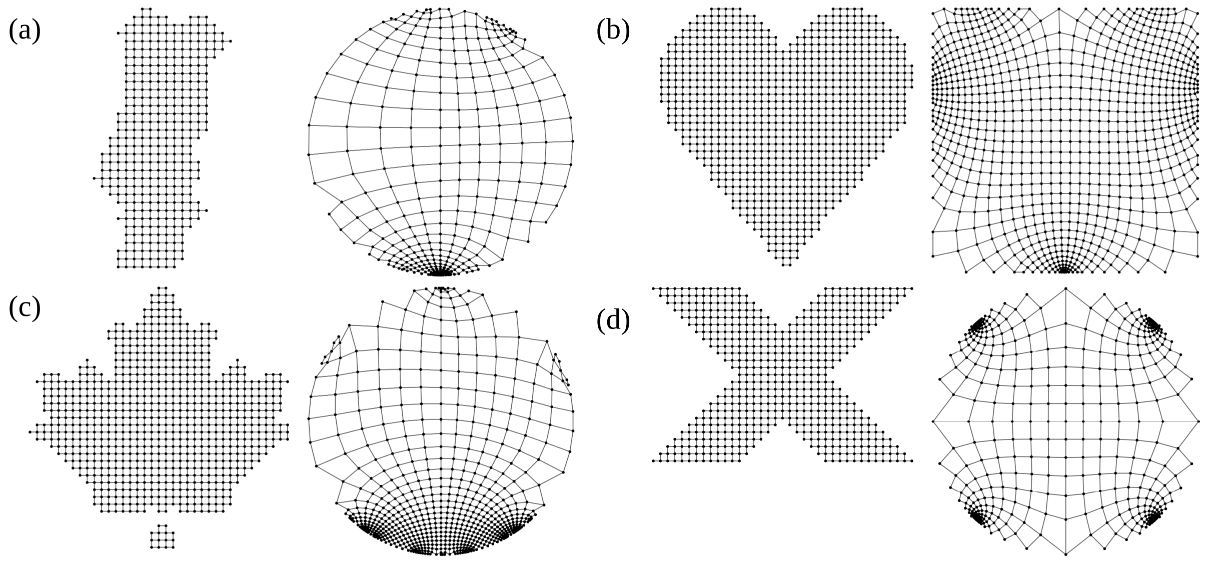


Figure 7: *Mapping (a) a rough outline of Portugal to a circle, (b) a heart to square, (c) a maple leaf to a circle and (d) an x to a circle. In practice, typical meshes have a higher resolution.*

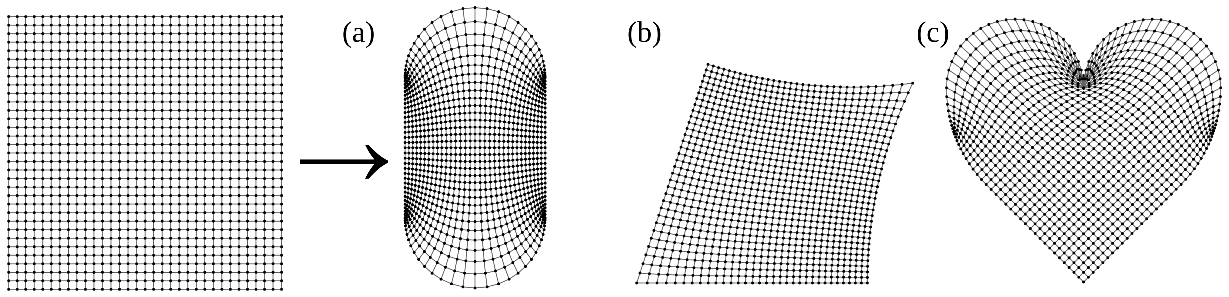


Figure 8: *Mapping a square to (a) an oval (b) a hyperbolic fundamental region as in [9] (c) a heart.*

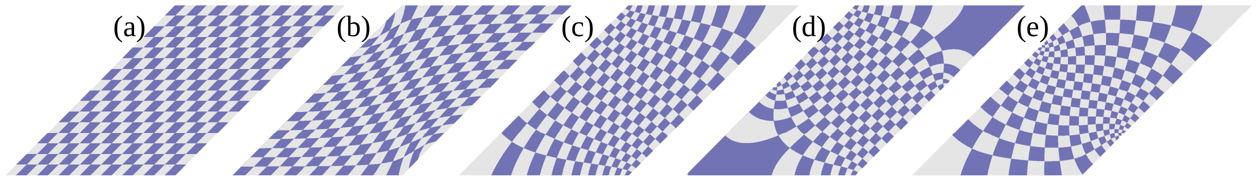


Figure 9: *(a) A harmonic map with a simple fixed boundary and (b) with a fixed boundary such that the boundary lengths are equal to the original. (c) Results of our mapping with all four corners fixed, (d) with just the obtuse corners fixed, (e) and where the boundary points are free to slip over any corner.*

5. Artwork

The original motivation to undertake this project was to put an entire visible sphere panorama onto a tangram set where the edge of each tan (piece) matches another edge smoothly and continuously. To encourage continuous rearrangement, the tans are never meant to be arranged into the familiar square tangram configuration with their edges matching. The Guyou projection, which maps a sphere onto a tileable 1x2 rectangle, fits the bill. We approximate the Guyou projection by first mapping each hemisphere to a circle via a stereographic projection and then mapping that to a square using the square-to-circle mapping from Figure 4(b). Next, abutting the two hemispheres-as-squares gives us an approximation of a Guyou projection from which we cut the tans. See Figure 10.

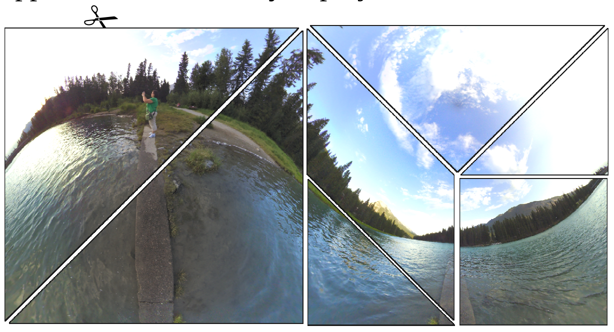


Figure 10: *Panorama Tangram -orama*

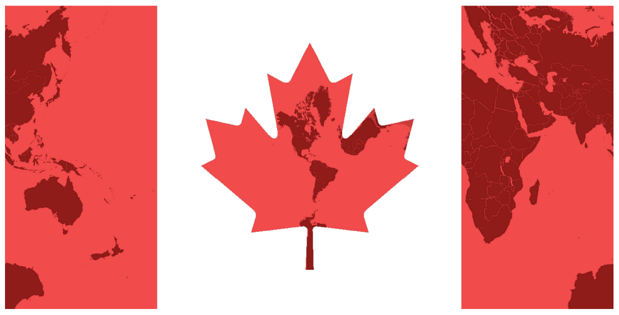


Figure 11: *Canadian Flag Projection*

For fun, if we were to remove one of the square hemispheres from the tangram example and use the maple-leaf-to-circle mapping from Figure 7(c) instead, we would have a world projection, interrupted in three pieces, in the shape of the Canadian flag seen in Figure 11.

Von Gagern and Richter-Gebert [9] map patterns from the plane to hyperbolic space by conformally mapping a pattern's fundamental region to its equivalent on the Poincaré disk. We have created one such mapping in Figure 8(b). In the same vein, let us work with a p6m wallpaper pattern recreated from the well known one in Owen Jones' *Grammar of Ornament* [6]. Figure 12 demonstrates how to convert this p6m pattern to a p4m one by generating a map between two right angle triangles. A simple affine transformation is included for comparison.

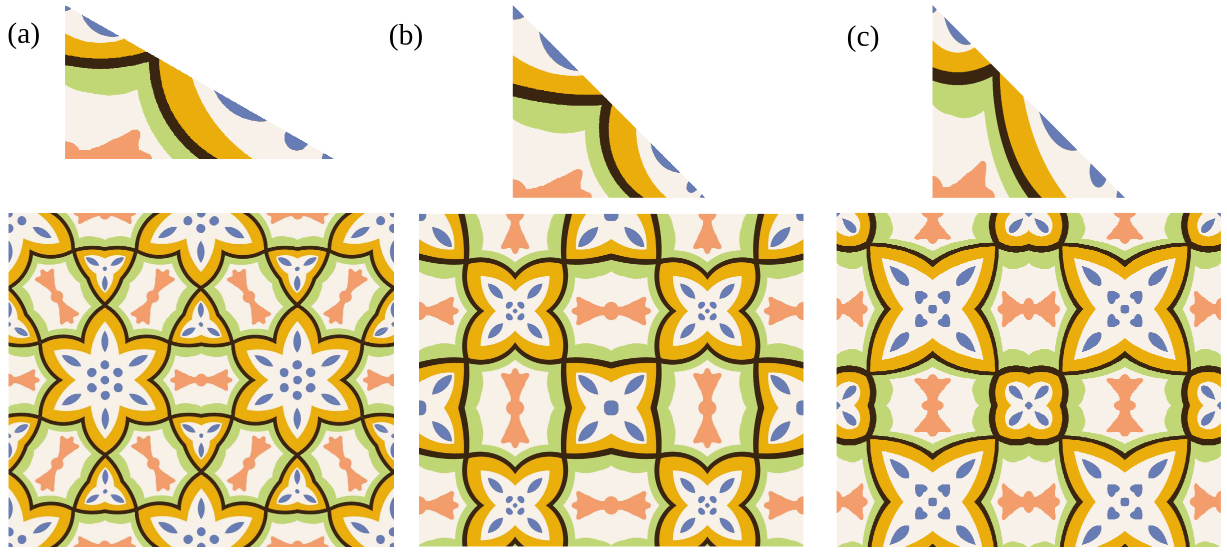


Figure 12: *Changing wallpaper groups from p6m to p4m. (a) The original p6m tiling with the 30-60-90 prototile. (b) Using our method to map the original prototile to a 45-45-90 triangle. (c) A prototile created from a simple (affine) scaling: this preserves area, but has discontinuities (sharp corners that were not in the original) along the prototile's diagonal.*

We finish with one final example showing the results of mapping the well known Sierpinski Triangle and Sierpinski Carpet into each other's shapes. See Figure 13.

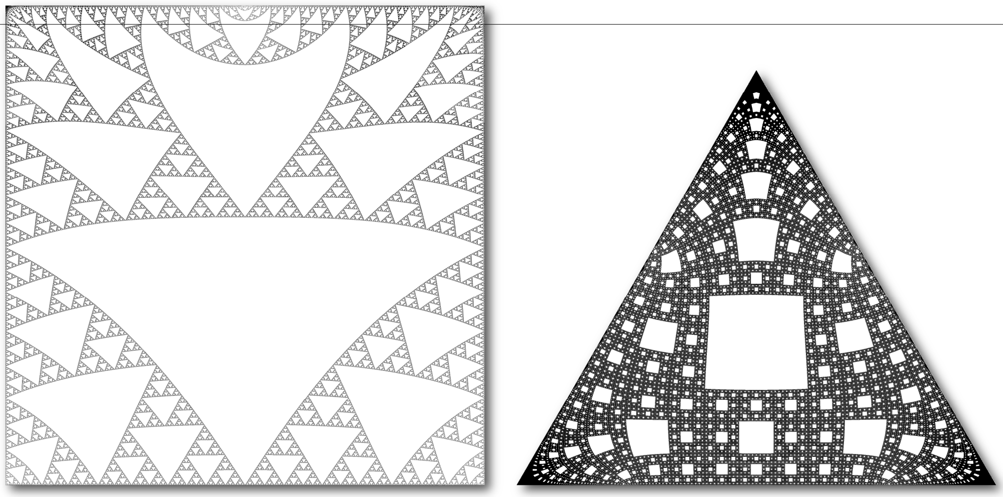


Figure 13: *The Sierpinski Triangle as a Square and the Sierpinski Carpet as a Triangle.*

6. Future Work

This algorithm is well suited for speed improvements. Meshes larger than 100x100 often took more than five minutes. We drastically improved the runtimes for higher resolution meshes by using a multi-grid scheme: we run the algorithm at lower resolutions to determine the global behavior, and then increase the resolution in stages to refine the answer until the desired resolution is achieved.

As mentioned in Section 4, there are limits to which shapes our algorithm does well on and it would be beneficial to understand what they are and why. Perhaps a new parameter tweaking step that uses θ, and r can be found that enforces conformality in the interior; so far, current attempts do not converge very well.

7. Acknowledgements

The author would like to thank Christie Digital for supporting this research, photographer Pam Eveleigh for the use of her snowflake photograph in Figure 2(c) ©2010, and especially the reviewers for their useful feedback.

References

[1] Vladimir Bulatov. Conformal Models of the Hyperbolic Geometry. Presented at MAA-AMS Joint Mathematics Meeting, San Francisco 2010. http://bulatov.org/math/1001/. Accessed February 1, 2011.

[2] Chuck Collins and Kenneth Stephenson. A Circle Packing Algorithm. *Computational Geometry: Theory and Applications*, vol. 25, pages 233–256, 2003.

[3] Bart de Smit and Hendrik W. Lenstra Jr. The Mathematical Structure of Escher’s Print Gallery. In *Notices of the AMS 50*, no 4, pages 446–451, 2003.

[4] Tobin A. Driscoll and Lloyd N. Trefethen. *Schwarz-Christoffel Mapping*. Cambridge Monographs on Applied and Computational Mathematics, 2002.

[5] Daniel M. Germán, Lloyd Burchill, Alexandre Duret-Lutz, Sébastien Pérez-Duarte, Emmanuel Pérez-Duarte, Josh Sommers. Flattening the Viewable Sphere. *Computational Aesthetics 2007*, pages 23–28. 2007.

[6] Owen Jones. *The Grammar of Ornament*. Folio edition, Bernard Quaritch, 1910.

[7] Liliya Kharevych, Boris Springborn and Peter Schröder. Discrete Conformal Mappings via Circle Patterns. In *ACM Transactions on Graphics* 25(2), pages 412-438, 2006

[8] Boris Springborn, P. Schröder, and Ulrich Pinkall. Conformal Equivalence of Triangle Meshes. In *ACM Transactions on Graphics*, 27(3), article 77, 2008.

[9] Martin von Gagern and Jürgen Richter-Gebert. Hyperbolization of Euclidean Ornaments. *The Electronic Journal of Combinatorics*, 16(2), 2009.