1 Introduction

This report outlines the results obtained by undergraduate researcher Andrew Winslow and faculty advisor Dr. Tom Kennedy during the Spring 2007 semester at the University of Arizona. The research investigated two topic of interest relating to conformal maps in the complex plane: the use of a conformal mapping algorithm in pattern recognition and the limitations of the aforementioned algorithm. In the first topic, the algorithm used was a modified version of the 'Fingerprint' algorithm developed by Mumford and Sharon[2]. As will be shown, the mapping contains ambiguities which prevent similar shapes from being identified as such in a computationally efficient way. The second topic investigates our implementation of the Fingerprint algorithm and its use Zipper algorithm[1], an established algorithm used to compute the conformal map between the interior of a simple closed curve and the unit circle. It will be shown the algorithm fails in computing fingerprints for a large number of shapes.

2 Use of a Conformal Mapping Algorithm in 2D Pattern Recogition

2.1 Algorithm Outline

The entire algorithm can be thought of as a function $\Omega : \mathbb{S} \to Diff(S^1)/PSL_2(\mathbb{R})$, where $\mathbb{S}$ is the set of all simply connected regions in the complex plane, $Diff(S^1)$ is the set of diffeomorphisms on the complex unit circle, and $PSL_2(\mathbb{R})$ is the set of Mobiüs transformations from the unit circle to itself[1]. Let γ, γ' be simple smooth closed curves in the complex plane, where $\gamma = r\gamma' + z, r \in \mathbb{R}, z \in \mathbb{C}$. That is, let γ and γ' differ by exactly a scaling and translation. Then $\Omega(\gamma) = \Omega(\gamma')$. In more general terms, Ω is a function from 2D shapes to a coset space of functions, where shapes belong to the same coset if and only if they differ by exactly a scaling and translation. Thus, the inverse mapping $\Omega^{-1}(Diff(S^1)/PSL_2(\mathbb{R}))$ partitions $\mathbb{S}$, the space of shapes, into sets of 'similar' shapes, where similarity follows intuition by being location and size invariant.

2.2 Why Try Ω?

The motivations behind the investigation of this algorithm in an attempt to classify 2D shapes are numerous. First, it can be used for any shape whose outline is a simply connected curve. While this may seem to initially exclude many common shapes (scenes from Dunkin' Donuts or Discount Tire, for example, could be problematic), it presents a far smaller obstacle than other scheme-induced limitations found in other recognition schemes, such as segment/angle-pair identification found in the LEWIS systems [3]. Next, the mapping has an implicit invariance under scaling and translation. These are two properties of

[1] This is equivalent to the set of Mobiüs transformations of the form $z \to \frac{az+b}{\bar{b}z+\bar{a}}$, a,b $\in \mathbb{C}$

shapes which are only taken into account in very sophisticated recognition systems where context and relative positioning is used to recognize patterns, and frequently present obstacles for simpler recognition schemes. Finally, multiple existing algorithms for computing conformal maps exist[1][4], each of which is extensively developed, making the implementation of a new recognition system involving this algorithm much simpler than creating an entirely new high-level processing algorithm.

2.3 Computing $\Omega(s)$

So after seeing the niceties inherent in Ω, one might hope that distilling the 'shapiness' out of 2D shapes would be straightforward. Unfortunately this turns out to not be true for the algorithm used in this research. Simply examining the codomain (a space of diffeomorphic functions on the complex unit circle) indicates that within the computation of Ω for a shape is an implicit computation of a diffeomorphic, complex-valued function. While Ω may allow for interesting visual analysis of conformal mappings, we will see that Ω, and in particular its codomain, turns out to behave in a way which does not have a simple correspondence to natural intuition of shape similarity.

Let $s \in \mathbb{S}$. Then $\Omega(s)$ is actually a composition of two functions. Each function is a conformal mapping to the complex unit disc, from a simply connected region (the region enclosed by the outline of the shape or the inverse of the outline). We know that such a conformal mapping exists by the Riemann Mapping Theorem. Stated more explicitly, if Γ_- is the region enclosed by a simple closed curve Γ (including the curve) in $\mathbb{C}$ and $\Delta_- = \{z \in \mathbb{C} || z| \leq 1\}$ then we can define $\Phi_- : \Delta_- \to \Gamma$ as a conformal map from the unit disc to the interior of the shape which maps the origin in the shape to the origin in the unit disc. Now let $\Gamma' = \{\frac{1}{z} | z \in \Gamma\}$, and $\Gamma_+ = \Gamma'_-$. Then $\Phi_+ : \Delta_+ \to \Gamma_+$, where $\Delta_+ = \{z \in \mathbb{C} || z| \geq 1\}$, is a conformal map from the exterior of the unit disc to the exterior of the shape mapping the point at infinity to itself. Finally, we have $\Psi = \Phi_+^{-1} \circ \Phi_-$, where the domain and codomain of Ψ are $\{z \in \mathbb{C} || z| = 1\} = S^1$. This function Ψ is a diffeomorphism, and hence $\Psi \in Diff(S^1)$. Note that this function is a member of $Diff(S^1)$ and not $Diff(S^1)/PSL_2(\mathbb{R})$. This is an important difference that will lead to the negation of the possibility of comparing these elements in a computationally feasible way.

2.4 Coset Structure and Mobiüs Transformations

Recall that we have defined Ω as sending each element of $\mathbb{S}$ to a coset in $Diff(S^1)/PSL_2(\mathbb{R})$. However, when describing the algorithm, we computed an explicit function (fingerprint) as the image of a shape. This function is an element of the image coset, $\Omega(s)$. While 'similar' (meaning identical up to a scaling and translation) shapes are sent to the same coset, individual shapes are not neccessarily sent to the same element of the coset. We know the general form of the relationship between any two elements of these cosets, but this knowledge does not trivialize the process of determining if two fingerprints are

located in the same coset. Through direct computation of several 2D shapes and their corresponding image functions, traits of Ω became apparent:

1. Ω is size-invariant. That is, two shapes which are identical up to a scaling have the same image fingerprint. This is natural when considering the conformal maps that make up Ω, and the composition of $\Phi_+^{-1} \circ \Phi_-$ under a scaling of Ω (and hence Δ_- and Δ_- by inverse amounts).

2. Ω is not translation-invariant. Let Γ be a simple closed curve. Then while $\Omega(\Gamma) = \Omega(\Gamma + z)$, the explicit fingerprints which are generated are not the same (they differ by a composition with a Mobiüs transformation). This is a result of the different conformal maps mapping the interior of a shape to the unit disc with different points being mapped to the origin. These conformal maps differ highly in the amount of distortion present as well as the regions within the shape where distortion occurs. Thus the conformal maps computed for $\Omega(\Gamma)$ and $\Omega(\Gamma + z)$ are not related in a simple way as $\Omega(\Gamma)$ and $\Omega(r\Gamma)$ are.

3. There are an infinite number of image elements of Ω in each coset of $Diff(S^1)/PSL_2(\mathbb{R})$. The lack of translation invariance mentioned in trait 2 is over an infinite subset of the coset. Note that this does not imply that Ω is injective (since it is size-invariant).

So in order to compare two fingerprints and determine if they exist in the same coset, there must exist a method for comparing them which does not rely on enumeration of all image elements in the coset. That is, if $f(z), g(z) \in Diff(S^1)$, we require a simple method for determining if $f^{-1} \circ g$ is a Mobiüs transformation of the form $\frac{az+b}{\overline{b}z+\overline{a}}$, a,b $\in \mathbb{C}$. Since f and g are both computed as approximations of the original shape (which is of arbitrary complexity), $f^{-1} \circ g$ is only a collection of points, rather than an explicit formula. Thus if it is to be fitted to a Mobiüs transformation, it must be fitted enumeratively. Since every pair of image fingerprints in a coset are related by a Mobiüs transformation, and each set contains an infinite number of image fingerprints, there are an infinite number of Mobiüs transformations which are possible matches. So this would require enumerating over an infinite set, for which no tractable method exists. So there is no computationally-reasonable way to compare two shapes to determine if they are similar up to a scaling and translation.

2.5 Example: Circle

The problem of trying to determine whether two shapes are within the same coset is illustrated by considering one of the simplest shapes a recognition system could hope to encounter: a circle. In this example, the complex effects that composition with a Mobiüs transformation can have on a fingerprint is seen. The implication of this is to effectively make comparing two of these elements very difficult, due to the vastly different functions contained in the same coset.

The images below show three shapes contained within the same coset preimage (Figure 1), and their respective images within the coset (Figure 2). Notice that each pair of shapes is similar under a scaling and translation. For each fingerprint computed, the origin in the circle is mapped to the origin in the unit disc. An an aside, the fingerprint for Circle 2 can be expressed as the fingerprint for Circle 1 composed on the right with $\frac{z+0.5}{0.5z+1}$ and the fingerprint for Circle 3 can be expressed as the fingerprint for Circle 1 composed on the right with $\frac{z+0.9i}{-0.9iz+1}$.

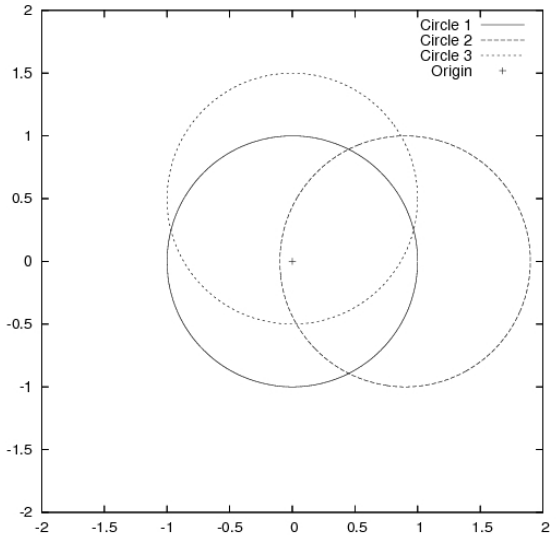

Figure 1: Simple smooth closed curves in $\mathbb{C}$. Circle 1: $\{z \in \mathbb{C} || z| = 1\}$, Circle 2: $\{z \in \mathbb{C} || z - 0.9| = 1\}$, Circle 3: $\{z \in \mathbb{C} || z - 0.5i| = 1\}$.

2.6 Other Issues

It should be noted that our implementation of the fingerprint algorithm makes the basic assumption that there is a known point in the interior of the shape (typically the origin). Without this, the conformal map has no known reference point to map to the origin in the unit disc and thus is no longer unique. Addition of a preprocessing phase where an interior point is computed would add additional computation time, further demonstrating that Ω is unsuited for use in a recognition scheme.

2.7 Summary

As we have shown, the characteristics of Ω imply that computation of whether two shapes are similar is not computationally-feasible using Ω. Due to traits of Ω determined via empirical data, the complexity of the cosets of $Diff(S^1)/PSL_2(\mathbb{R})$,

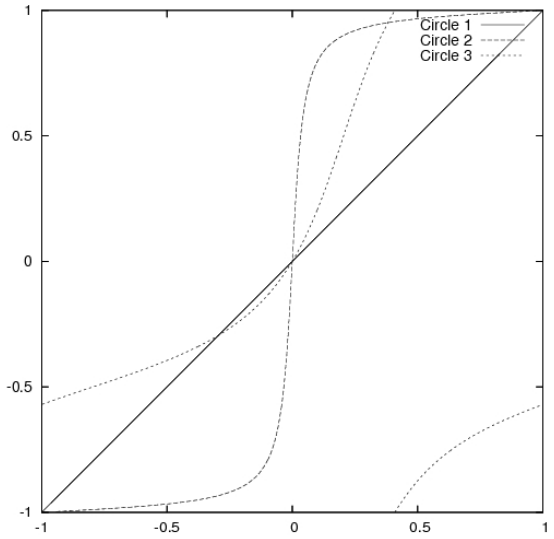

Figure 2: Images under Ω of Circle 1, 2, 3. These are diffeomorphisms on S^1, and hence are periodic (with a normalized period of 2).

and the disparity between coset function theory and implementation, there is no tractable method for determining how the images of two shapes under Ω are related. Because of this, the use of Ω in a pattern recognition system as a method of high-level processing is not possible.

3 Fingerprint Algorithm Difficulties

3.1 Background

During our work with the fingerprint algorithm, it was noted that for certain shapes the algorithm produced invalid results. Within our fingerprint algorithm is an implementation of the Zipper algorithm, which is used to compute the conformal maps from the interior and exterior of a shape to the unit disc. This algorithm computes the conformal map of a shape defined by a sequence of points. Thus, each shape for which a fingerprint is computed is actually a polygon (practically speaking). If we examine the polygons associated with various shapes, we see that if a reasonable number of points ($\approx$300 or more) are used to approximate a 'smooth' shape such as a circle or ellipse, the interior angles of consecutive sides of the polygon are $\approx 180°$. Conversely, if we examine shapes of arbitrary roughness, such as the border generated by the path of a random walk, naturally we find that an interior angle formed by consecutive sides of the polygon can have arbitrary value (between 0 and 360°). Thus, if we wish to examine these shapes within the context of the fingerprint algorithm, we must be able to compute conformal maps for shapes containing interior angles

of any value.

3.2 Motivations

The motivations for development of a fingerprint algorithm which produces valid fingerprints for all polygons is twofold. First, from a philosophical standpoint such an algorithm is 'complete' in the sense that the domain of its input is the set of all simply connected regions. Second, there is an interesting question regarding the fingerprint and its stochastic properties relative to those present in its generating shape.

Consider a random walk in $\mathbb{C}$ representing discretized Brownian motion. We begin the walk at the origin and iterate for a finite number of steps, conditioning it to also end at the origin to form a loop. We then take the boundary of the loop (referred to as Γ) defined by a simple closed curve enclosing the loop and lying entirely on the loop itself. So Γ is the boundary of a simply connected region and so is a valid input for the fingerprint algorithm. An example of a Brownian loop and its corresponding boundary can be seen in Figures 3, 4 below. One might notice that in the example given, the origin lies on (or very nearly on) Γ. Since the walk starts at the origin, it is certainly possible for the origin to be located on Γ itself. In practice this is problem occurs infrequently enough to not inhibit the study of these regions (less than 5% of loops of 20,000 steps or more have the origin located on their boundaries).

So we have a method for converting Brownian loops into a form suitable as input into the fingerprint algorithm. Now we can ask how the output function from the fingerprint algorithm relates to Brownian motion. Is the function explicitly Brownian motion in one dimension? Unfortunately, the study of this question requires a fingerprint algorithm capable of producing valid fingerprints for all simply connected regions.

3.3 Implementation Specifics

The Zipper algorithm uses an iterative approach to compute a conformal map from the interior of a shape to the upper half plane $\mathbb{H}$ (mapping then from $\mathbb{H}$ to the unit disc is simply $f(z) = \frac{z-i}{z+i}$). It does this by conformally mapping the boundary of the shape to a curve in $\mathbb{H}$ which starts at the origin via $i\sqrt{\frac{z-z_1}{z-z_0}}$ where z_0 and z_1 are two adjacent points chosen on the boundary of the shape. Examples of the image of this map for two shapes can be seen in Figure 5. From here the algorithm 'unzips' the curve, composing consecutive conformal maps which send $\mathbb{H}\backslash$S to $\mathbb{H}$, where S is the segment from z_i to z_{i+1} and z_i is the point on the curve currently at the origin. The result is sending z_i to a point along the real axis, and z_{i+1} to the origin. This process continues until the last point on the curve (z_n) is mapped to the origin. The result of the completed process is the set $\{z_i | 1 \leq i \leq n\}$ in order along the real axis, with the distances between them describing the conformal map.

Ideally, the individual conformal maps used are exactly the ones which we

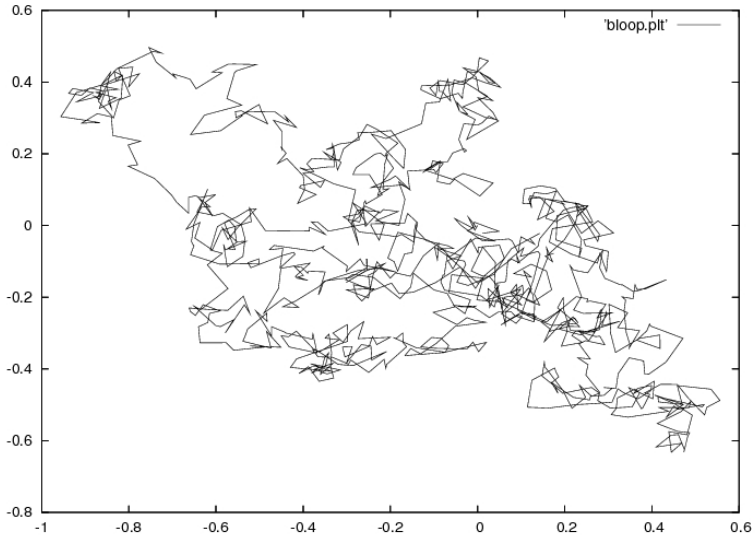

Figure 3: A Brownian loop of 1000 steps.

have described, sending $\mathbb{H}\backslash$S to $\mathbb{H}$, where S is a 'tilted slit'. In pratice this is a computational hassle because no explicit formula exists for such maps, requiring iterative approximation to determine them (via Newton's method). So instead we use similar maps which *do* have explicit formulas, of which two exist with the descriptive names 'circular arc' and 'vertical slit'. Let z_i be the point currently at the origin, and z_{i+1} the subsequent point on the curve. The circular arc map computes a conformal map from $\mathbb{H}\backslash$C to $\mathbb{H}$, where C is a circular arc orthogonal to the real axis from the origin to z_{i+1}. The vertical slit map computes the composition of a conformal map from $\mathbb{H}\backslash$V to $\mathbb{H}$ and f(z) = z - $\Re(z_{i+1})$ where V is the vertical segment from z_{i+1} to the real axis. In effect, it computes a conformal map, and then shifts the result. Both of these map versions are used in practice for computing the fingerprint with satisfactory results on most smooth shapes.

3.4 Initial Solution: Piecewise Computations

In the original implementation of the Fingerprint algorithm, both the vertical slit and circular arc maps were used in an attempt to produce valid fingerprints for all shapes. Both methods failed, yielding invalid fingerprints for a large number of common shapes and each technique failing on some shapes for which the other produced valid output. It was initially assumed that both methods failed because of increasing discrepancies between them and the tilted slit conformal map which they approximate.

Figure 5 below shows initial curves at the beginning of the unzipping process in which the curve is put through a sequence of small conformal maps. As

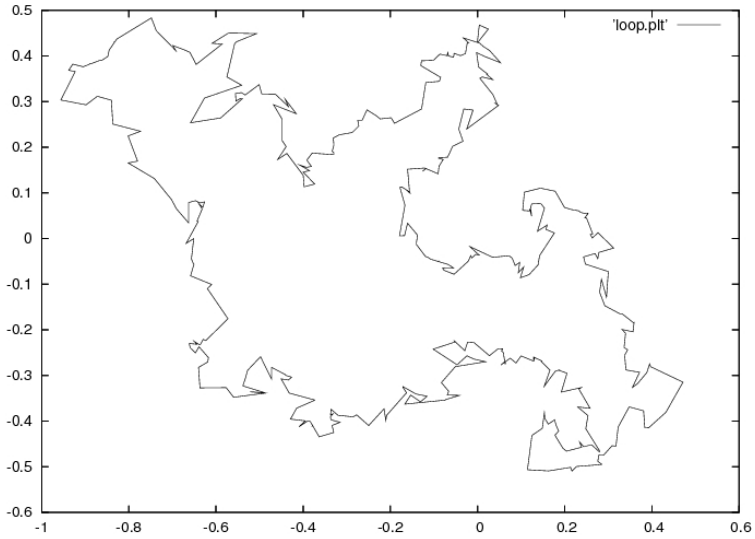

Figure 4: The boundary of the Brownian loop in Figure 3.

described previously, these conformal maps are simply maps which map $\mathbb{H}\backslash$S to $\mathbb{H}$, where S is the first segment of the remaining curve in $\mathbb{H}$. Recall that the circular arc map approximates these segments by an arc which is orthogonal to the real axis at its start, and whose end is the endpoint of the segment S. Thus a segment which is nearly horizontal requires this arc to be nearly half a circle. Moreover, the region of $\mathbb{H}$ underneath this arc must be distorted such that it constitutes a large portion of $\mathbb{H}$ in the image of the map, due to the angle-preserving nature of conformal maps. So nearly horizontal segments, such as those present in Figure 5 require massively-distorting conformal maps. The undesireable result of this is that these maps are extremely singular and thus are much more sensitive to rounding errors, producing unzippings which differ too dramatically from the correct unzipping to produce valid fingerprints.

The use of the vertical slit map suffers a similar fate, though for different reasons. Recall that the vertical slit map first performs a very nice conformal map removing a vertical segment V in $\mathbb{H}$ that extends from the real axis to the endpoint of the first segment on the curve. After this mapping, all of $\mathbb{H}$ is shifted towards origin. Both of these maps remain very consistent over the range of possible segments. However, we cannot forget all the previous points on the curve which have been mapped onto the real axis, and whose spacing and order on the axis define the overall conformal map sending the original region to $\mathbb{H}$. When we perform the mapping from $\mathbb{H}\backslash$V to $\mathbb{H}$, the endpoint of V (previously referred to as z_{i+1}) is mapped to $\Re(z_{i+1})$. So in order to use the method safely, we must be sure that there are no points between $\Re(z_{i+1})$ and the origin, otherwise we suddenly have a non-conformal map (not to mention a zipper in which you zip in both directions simultaneously). But

this requirement has no guarantee of being true. In particular, this is not true for nearly horizontal segments. Examining Figure 5, we see that these types of segments occur regularly in the curves generated by Brownian motion boundaries. So we cannot use the vertical slit mapping.

So we have two methods of approximating conformal maps, neither of which is viable. One fails because of its poor performance for points in $\mathbb{H}$. The other fails due to invalid mapping of points on the axis. The solution? Use both! The concept is straightforward: for each map, use the circular arc map for points on the real axis, and the vertical slit map for points above the axis in $\mathbb{H}$. The process is the same as before, except that for each point we push through the conformal map we decide whether it is 'on the axis' or not. Those that are get sent through the circular arc map, while the others are put through the vertical slit map. The conversion from one map to the other involves a resolution of a branch cut ambiguity. For a more detailed description of this issue, see Appendix A.

By combining both methods, the problems inherent to both methods have been eliminated. Testing this implementation, it is immmediately apparent that these issues are not the sole cause of invalid fingerprints. In the following sections, we discuss experimental results and possible causes for bad fingerprints.

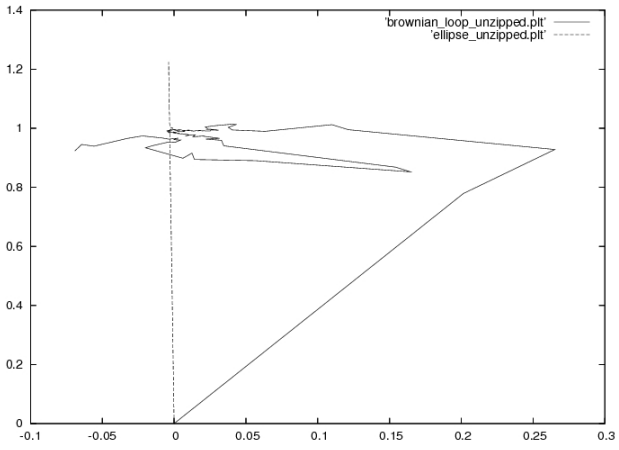

Figure 5: The resulting image of an ellipse and Brownian loop boundary after the conformal map $f(z) = i\sqrt{\frac{z-z_1}{z-z_0}}$. Though it may appear that the image of the ellipse is a nearly vertical line, there exist a small number of nearly horizontal segments on the curve. The horizontal scale of these segments is several magnitudes smaller than the scale of the horizontal segments of the Brownian loop boundary image, causing their lack of appearance.

3.5 Example: Triangle

Since our initial presumption of why the algorithm fails was incorrect, it is necessary to reevaluate in what cases the algorithm fails, and then attempt to formulate a list of possible reasons behind the behavior. Using a triangle as a template, several parameters were varied to obtain a well-defined list of which aspects of shape do and do not affect the validity of output conformal maps in the case of triangles. The list of parameters over which the validity of the fingerprint is invariant is short and unremarkable:

1. The number of points on each side of the triangle (provided n $\geq$ 2).
2. The size of the triangle.

The complementary list of parameters which do influence the validity of the fingerprint produced by the algorithm is longer and contains several points of interest.

1. The position of z_0, z_1 in relation to the nearest vertex.
2. The interior angles formed by adjacent sides of the triangle.
3. The position of the origin within the triangle (assuming, as we have, that the origin in the triangle is mapped to the origin in the unit disc).

One of the more serious issues present is the first item listed. Before careful examination of the triangle example, it was assumed that fingerprints are invariant over the z_0, z_1 chosen as starting points for the shape's boundary. This is not the case. One possible explanation of this is the variance of the curve of a shape at the beginning of the unzipping process, particularly the direction in which it heads to infinity. Recall that before performing the unzipping, we send the boundary of the shape to a curve in $\mathbb{H}$ which starts at the origin via the conformal map $f(z) = i\sqrt{\frac{z-z_1}{z-z_0}}$. Notice that z_1 is sent to the origin while z_0 is sent to infinity. Compare the end of the curves in Figure 5. The end of the ellipse curve is headed to infinity in the $\imath$ direction, while the end of the Brownian walk curve is heading to infinity in an undefined (though certainly not $\imath$) direction. In a confirmation of this idea, in general the unzipped curves for Brownian walks vary wildly depending upon the choice of z_0, z_1 while the unzipped curves for smoother shapes which produce valid fingerprints invariably go to infinity in the $\imath$ direction.

The second and third items can be lumped into a category of problems relating to the computation of the conformal maps. Particularly the third item indicates that conformal maps which are too singular simply cause the algorithm to produce incorrect output. This is very likely attributed to some aspect of the Zipper algorithm. It is not immediately apparent what governs thresholds (such as nearness of the origin to a side) at which the algorithm breaks. This may indicate that the change is not due to a sudden change in the input shape parameters, but instead a change in algorithm execution due to implementation specifics.

3.6 Example: Horned Square

This example shall serve as a study of how shapes which differ in fairly minor ways can fail or succeed to produce valid fingerprints when using our implementation of the fingerprint algorithm which utilizes the Zipper conformal mapping algorithm. Figures 6, 7 are two variations of a square, the first of which produces a valid fingerprint and the second does not. This behavior was seen in the previous triangle example, but within a triangle it is difficult to isolate which properties of the triangle cause problems since a change to the measure of an angle or side also changes other angles, sides as well as the area of the triangle and nearness of the origin to the sides. So in this example we vary only one parameter in a significant way: a distortion in a small portion of a square.

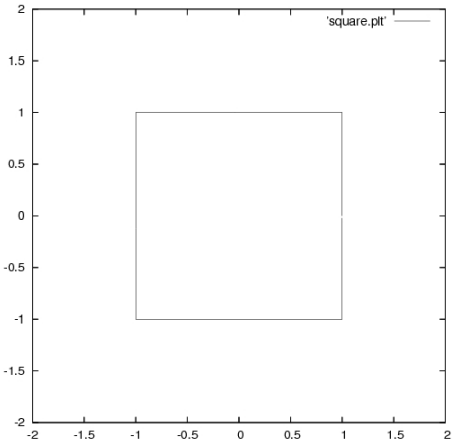

Figure 6: A square. For the purposes of the Zipper algorithm, $z_0 = (1.0, 0.0)$, $z_1 = (1.0, 0.02)$.

One of the more peculiar aspects of the algorithm failures is seen by varying the interior angle of the 'horn' present in the shape in Figure 7. If the angle of the horn is widened slightly to $\approx 4°$, a valid fingerprint (albeit a fairly singular one) is produced. That is, there is a 'breaking point' of the interior angle of the horn where larger angles produce valid fingerprints and smaller ones do not. This suggests that some aspect of the problem is a result not of singular cases such as z_0 moving from the interior of a segment to a corner, but to cases hidden within the algorithm, possibly from approximation values used during the unzipping process for determining whether a point is the origin, in $\mathbb{H}$ or on the real axis.

Additionally, this case serves as a terminal case for investigation of the fingerprint algorithm using the Zipper algorithm for computing conformal maps.

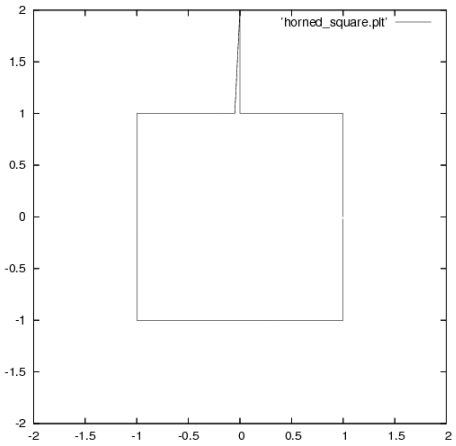

Figure 7: The previous square with a horn added. The interior angle of the horn is $\approx 3°$.

A square centered at the origin with the small horn attached is a 'nice' shape in that it has regularity and is relatively close to a circle in shape. Should the algorithm not be able to compute the fingerprint for this shape, the set of shapes over which the algorithm can be run reliably is limited to say the least. Due to this limitation, it is not reasonable to expect the algorithm be used in applications where the data is not heavily processed or checked by hand for smoothness.

3.7 Conclusion

The reason behind the algorithm's failings are still unknown. We have isolated general themes in the types of shapes which cause the algorithm to fail, as well as identified possible reasons that these types of shapes fail. Unfortunately, we have not been able to identify specific portions of code within the implementation or flaws within the algorithm that explicitly lead to incorrect fingerprint computations.

A Appendix

As referrenced in the body of the paper, a method of computing conformal maps which uses both established methods of vertical slit and circular arc was implemented in an attempt to eliminate problems due to inherent properties of the two methods. Within this implementation, an issue arose: a square root

ambiguity present in the computation of circular arc conformal maps meant that the image of a point that is mapped to the real is only known up to sign.

A.1 Branch Cut Ambiguity

This is an examination of the function g(z): $\mathbb{H}\backslash\gamma \to \mathbb{H}$ defined by

$$g(z) = \frac{ib^2\sqrt{\frac{-z^2+2az-R^2}{(R^2-az)^2}}}{ia\sqrt{\frac{-z^2+2az-R^2}{(R^2-az)^2}}+1}$$

The equation is the formula for the a circular arc conformal map where the end of the segment (previously called z_{i+1}) is a + bi. The ambiguity comes for points on the negative real axis, since the branch cut of the solution was previously assumed to be at $\theta = \pi$. Since the combined implementation uses the circular arc map exclusively for points on the real axis, it is quite important that this issue is resolved!

Let

$$l(z) = \frac{-z^2+2az-R^2}{(R^2-az)^2}$$

The ambiguity present in the square root of Equation 3 is what we intend to resolve. This ambiguity exists for $z \in \{z \in \mathbb{C} | z = r, r \in \mathbb{R}, r < 0\}$, the negative real axis, since the branch cut for this equation is located at $\theta = \pi$.

A.2 z = x + iϵ substitution

To resolve the sign of the square root, we will determine the actual result of l(z) on elements which are slightly above the real axis. That is, elements which are contained in $\mathbb{H}$. Once we know where l(z) lies, we can determine which root is taken (depending on the side of the branch cut l(z) lies). Let z = x + iϵ. Then

$$\begin{aligned}
l(z) &= l(x+i\epsilon) \\
&= \frac{-(x+i\epsilon)^2+2a(x+i\epsilon)-R^2}{(R^2-a(x+i\epsilon))^2} \\
&= \frac{-x^2-2ni\epsilon+-i^2\epsilon^2+2ax+2ai\epsilon-R^2}{R^4-2R^2ax-2R^2ai\epsilon+a^2x^2+2a^2xi\epsilon+a^2i^2\epsilon^2} \\
&\approx \frac{-x^2+2ax-R^2+i\epsilon(-2x+2a)}{R^4-2R^2ax+a^2x^2+i\epsilon(-2R^2a+2a^2x)}
\end{aligned}$$

Multiplying by the complex conjugate of the denominator yields

$$l(z) \approx l(x) + i\epsilon\frac{(-2x+2a)(R^4-2R^2ax+a^2x^2)+(-2R^2a+2a^2x)(x^2-2ax+R^2)}{(R^4-2R^2ax+a^2x^2)^2}$$

Let q(x), p(x) be real-valued polynomials defined by

$$\begin{aligned} q(x) &= (-2x+2a)(R^4 - 2R^2ax + a^2x^2) + (-2R^2a + 2a^2x)(x^2 - 2ax + R^2) \\ p(x) &= (R^4 - 2R^2ax + a^2x^2)^2 \end{aligned}$$

So we can write l(z) as

$$l(z) = l(x + i\epsilon) \approx l(x) + i\epsilon\frac{q(x)}{p(x)}$$

Since p(x) is always positive, the sign is dependent only upon the sign of q(x). So we simplify to find

$$\begin{aligned} q(x) &= -2R^4x + 4R^2ax^2 - 2a^2x^3 + 2R^4a - 4R^2a^2x + 2a^3x^2 - 2R^2ax^2 \\ &+ 4R^2a^2x - 2R^4 + 2a^2x^3 - 4a^3x^2 + 2R^2a^2x \\ &= -2R^4x + 2R^2ax^2 - 2a^3x^2 + 2R^2a^2x \\ &= -2R^4 + 2a^3x^2 + 2b^2ax^2 - 2a^3x^2 + 2R^4x - 2R^2b^2x \\ &= 2b^2ax^2 - 2R^2b^2x \\ &= 2b^2(ax^2 - R^2x) \end{aligned}$$

This is a quadratic with the following properties:

1. If a < 0, q(x) is negative for x $\in (-\infty, \frac{R^2}{a}) \cup (0, +\infty)$ and positive for x $\in (\frac{R^2}{a}, 0)$.
2. If a $= 0$, q(x) is positive for x $\in (-\infty, 0)$ and negative for x $\in (0, +\infty)$.
3. If a > 0, q(x) is positive for x $\in (-\infty, 0) \cup (\frac{R^2}{a}, +\infty)$ and positive for x $\in (0, \frac{R^2}{a})$.

So this criteria indicates which root should be taken as well (positive when q(x) positive, negative when q(x) is negative). The non-zero point at which the root sign suddenly changes if a $\neq 0$ can be explained by the fact that this is the point on the real axis at which l(z) is undefined, which means there is a massive distortion around this point, leading to no sudden 'switch' of maps.

References

[1] D. E. Marshall and S. Rohde. Convergence of the Zipper algorithm for conformal mapping. JAMS, 18: 763-778, 2005.

[2] E. Sharon and D. Mumford. 2D-Shape Analysis using Conformal Mapping. CVPR, 2: 350-357, 2004.

[3] C. A. Rothwell. Object Recognition through Invariant Indexing. Oxford University Press, New York, 1995.

[4] T. Driscoll and L. N. Trefethen. Schwarz-Chrsitoffel mapping. ACM Trans. Math.Soft., 22:168-186, 1996.