

Approximate Hyperbolic Splines and Their Transformations

A Thesis

submitted to the faculty of the graduate school
of the University of Minnesota

by

Amit Lath

In partial fulfillment of the requirements
for the degree of Master of Science

May 2002

Department of Computer Science

University of Minnesota

Duluth, Minnesota 55812

U.S.A.

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a master's thesis by

Amit Lath

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Douglas Dunham

Name of Faculty Adviser

Signature of Faculty Adviser

Date

GRADUATE SCHOOL

Acknowledgments

I would sincerely like to thank Dr. Douglas Dunham for persistent guidance, and support at times of doubt. I also thank Dr Gary Shute and Dr Joe Gallian for agreeing to serve on my committee. Appreciation is also due for the help I recieved from faculty members and fellow students in the computer science department in my graduate study.

Finally i would like to thank my parents and my wife for constant support and patience.

Abstract

The most widely used curves in graphical applications are parametric cubic curves in the Euclidean 3-space and various forms thereof. These curves enjoy various properties that make them very suitable for this kind of applications. For example they often have desirable continuity properties at the end points, and are relatively easy to compute while offering good flexibility in shape control. One of the nice properties these curves have is that they are invariant under the common transformations. This entails that the curve can be transformed by transforming the defining conditions only. The transformed version can then be computed from the transformed conditions, which offers a big computational advantage over having to transform the curve point-by-point.

The question is: Is there a way to define a curve using given control points on the hyperbolic plane, such that it is invariant under hyperbolic transformations? In the present work we make progress towards answering this question. This can lead to a vast improvement in computerized hyperbolic pattern generation, where using this kind of “smooth” curves has so far been computationally prohibitive. Applications can also be found in the fields of animation and hyperbolic curve approximation.

Contents

1	Introduction	1
2	Hyperbolic Geometry	3
2.1	Euclid's Postulates	3
2.2	The Hyperbolic Axiom	4
3	Models of Hyperbolic Geometry	6
3.1	The Weierstrass Model	7
3.2	The Poincaré Disk Model	10
3.3	Isomorphism between the Poincaré and Weierstrass Models	12
4	Parametric Cubic Curves	14
4.1	General Characteristics	15
4.2	Hermite Curves	17
4.3	Beziér Curves	18

5	Hyperbolic Isometric Transformations	20
5.1	Reflection	21
5.2	Rotation	22
5.3	Translation	22
5.4	Glide Reflection	22
6	Transformations of Approximate Hyperbolic Splines	24
6.1	Approximate Hyperbolic Splines	26
6.2	Investigation of Invariance Property	31
7	Results	33
8	Future Possibilities	48
9	Conclusion	49

List of Figures

3.1	A “line” in the Weierstrass Model [Fab83]	8
3.2	The vector l on the hyperboloid of one sheet $\langle X, X \rangle = k^2$ corresponds to a “line” in the Weierstrass Model [Fab83]	9
3.3	The Poincaré model, showing lines l , m and n . We say l and n are <i>divergently parallel</i> , m and n are <i>intersecting</i> and l and m are <i>asymptotically parallel</i>	11
3.4	Stereographic projection between the Weierstrass model (H^2) and the Poincaré model (D)	13
6.1	A spline drawn without the modification of control points. It clearly starts and ends in a direction tangential to the Euclidean straight line segment between the end point and the control point.	29
6.2	The same spline drawn after the modification now starts and ends tangential to the hyperbolic line between the end points and the control points.	30
7.1	An example of a rotation. Distance = 0.0435455476152937	35

- 7.2 Another example of a rotation. Notice that even though we are quite close to the bounding circle, the distance measure demonstrates a good approximation, and the visual quality of the result is very good. Distance = 0.0141032093927292 36
- 7.3 An example of a reflection. The visual quality of the result is excellent, and is reflected in the measure for distance as well. Distance = 0.00269178128091013 37
- 7.4 Another example of a reflection. An important thing to notice here is that although the visual quality of this result is almost as good as the previous one, the distance measure shows a significant difference. This is because here we are closer to the bounding circle and therefore the splines appear closer to each other than they actually are. Distance = 0.0109420035482458 38
- 7.5 An example of a translation. Distance = 0.0258546892815096 39
- 7.6 Another example of a translation. Distance = 0.0114738924405686 40

List of Tables

7.1	Results for tests on reflections using “smaller” splines.	41
7.2	Results for tests on rotations using “smaller” splines.	42
7.3	Results for tests on translations using “smaller” splines.	43
7.4	Results for tests on reflections using “bigger” splines.	44
7.5	Results for tests on rotations using “bigger” splines.	45
7.6	Results for tests on translations using “bigger” splines.	46

Chapter 1

Introduction

For a long time now, the most widely used curves in graphical applications have been parametric cubic curves in the Euclidean 3-space and various forms thereof. These curves enjoy various properties that make them very suitable for this kind of applications. For example they often have C^1 (and in some cases C^2) continuity at the end points and are relatively easy to compute while offering good flexibility in shape control. One of the nice properties these curves have is that they are invariant under the common transformations. This ensures that the curve can be transformed by transforming the defining conditions only. The transformed version can then be computed from the transformed conditions. This offers a big computational advantage over having to transform the curve point-by-point.

The question is: *Is there a way to define a curve using four given control points on the hyperbolic plane, such that it has such useful properties?* In the present work we make progress towards answering this question. This can lead to a vast improvement in computerized hyperbolic pattern generation, where using these kinds of “smooth” curves has so far not been implemented. Applications can also be found in the fields of animation and hyperbolic curve approximation.

More specifically, we investigate some ways to define a curve using the control points such that it is invariant under isometric transformations. The testing is done on each of reflection, rotation, translation and glide reflection. We use the Poincaré disk model of hyperbolic geometry for a visual display of the results, whereas all the computations are carried out using the Weierstrass Model of hyperbolic geometry.

Following, we give a brief introduction to hyperbolic geometry (Chapter 2), followed by a discussion on the relevant models of hyperbolic geometry (Chapter 3). In Chapter 4, we discuss parametric cubic curves, and in Chapter 5, we look at isometric hyperbolic transformations. This is followed by a discussion of an approximate hyperbolic spline that we have found and our investigation into its transformations (Chapter 6). Chapter 7 presents results and the following chapter provides some pointers to future work.

Chapter 2

Hyperbolic Geometry

2.1 Euclid's Postulates

In his book, *Elements*, Euclid laid out five postulates, and based on these, deduced theorems which constituted the rest of the volume. He postulated the following:

- To draw a straight line from any point to any other
- To produce a finite straight line continuously in a straight line
- To describe a circle with any center and distance
- That all right angles are equal to each other
- That, if a straight line falling on two straight lines makes the interior angles on the same side less than two right angles, the two straight lines, if produced indefinitely, meet on that side on which are the angles less than the two right angles

Of these postulates, perhaps the most intriguing is the fifth, known as the “parallel postulate”. The modern day version of this postulate is Playfair’s Axiom, given by John Playfair

in a commentary he wrote on Euclid's book. It states:

Given a line and a point not on the line, it is possible to draw exactly one line through the given point parallel to the line.

The Parallel Postulate was clearly not as natural and obvious as the first four and was a subject of controversy since almost the day it was written. Mathematicians felt that rather than being a postulate, it should be a theorem derived from the first four. Euclid himself was aware of its problems and refrained from using it in his deductions as long as possible, up until the twenty ninth proposition in his book.

For more than two thousand years, a string of mathematicians attempted to deduce the fifth postulate from the first four. However, none of these attempts met with success. Most of the time the proofs assumed something which was found to be equivalent to the postulate. In 1813, after having attempted the proof for more than twenty years, Gauss wrote:

In the theory of parallels we are even now not further than Euclid. This is a shameful part of mathematics ...

It was this failure to prove the fifth postulate that ultimately led to the birth of what is known as Non-Euclidean Geometry.

2.2 The Hyperbolic Axiom

By the beginning of 19th century, mathematicians began to explore a geometry in which the Parallel Postulate did not hold. Gauss had deduced a great deal of it but did not publish his work. Two mathematicians, namely Janos Bolyai and Nicolai Lobachevsky have been credited to have discovered the new geometry independently. After making the discovery, Bolyai wrote in a letter to his father:

... I have discovered things so wonderful that I was astounded ... out of nothing
I have created a strange new world.

This new geometry, popularly known as hyperbolic geometry, is defined as follows:

The geometry you obtain by assuming that all the axioms for neutral geometry (geometry without a parallel postulate) and replacing the Parallel Postulate by its negation, which we shall call the “Hyperbolic Axiom” [Gre94].

Lobachevsky formulated a new postulate to replace the parallel postulate. This postulate, known as Lobachevsky’s Parallel Postulate, states:

Through a point not on a given line there exists (in the plane determined by this point and line) at least two lines which do not meet the given line.

Subsequently a number of models were developed for the new geometry. All the models had to distort distance because it is not possible to smoothly isometrically embed the entire hyperbolic plane in Euclidean 3-space.

Chapter 3

Models of Hyperbolic Geometry

A model is an interpretation of the primitive terms that makes the postulates true statements. A “point”, a “line”, “lies between”, “is parallel to” are examples of primitive terms in case of a model for a geometry. So, any model for a geometry has to interpret or “give meaning to” these (and possibly other) primitive terms.

One way to prove the consistency of a geometry is to formulate a model for it. Non-Euclidean geometry was proved consistent by Eugenio Beltrami when he gave a model for it in 1868 [Fab83].

Several models have been formulated for hyperbolic geometry since its discovery. The Poincaré Disk model and the Klein-Beltrami disk model are examples of finite models, while models such as the Weierstrass model and the Poincaré Upper-Half Plane model are examples of infinite ones.

In the following sections, we discuss two of these which are relevant to the present work.

3.1 The Weierstrass Model

Weierstrass demonstrated the consistency of hyperbolic geometry by constructing a model for it on a certain surface in Euclidean 3-space [Fab83].

We now provide a brief development of this model following [Fab83]. We will look at the interpretation of the primitive terms like “point”, “line”, “lies on” and “between” given by it. Before that, we need some preliminaries.

A “point” in real 3-dimensional space is represented by a triplet of real numbers such as $X = (x, y, z)$. Such a triplet can also represent a vector in 3-space. This vector can be visualized as an arrow from the origin $O = (0, 0, 0)$ to X . This lets us use the terms “vector” and “point” interchangeably.

Consider two vectors $X_1 = (x_1, y_1, z_1)$ and $X_2 = (x_2, y_2, z_2)$. Their *Euclidean inner product* or *dot product* is defined as:

$$X_1 \cdot X_2 = x_1x_2 + y_1y_2 + z_1z_2$$

To construct the model we also need another inner product, called the *hyperbolic inner product*. For the above two vectors it can be defined as:

$$\langle X_1, X_2 \rangle = x_1x_2 + y_1y_2 - z_1z_2 \quad (3.1)$$

Two vectors X and Y are said to be *e-orthogonal* if $X \cdot Y = 0$, and *h-orthogonal* if $\langle X, Y \rangle = 0$.

Now the equation of a plane in R^3 is of the form $\langle A, X \rangle = b$, where A is an *h-normal* of the plane. This means that A is h-orthogonal to any point that lies on this plane.

We are now ready to develop the model. As noted earlier, we provide interpretations of “point”, “line”, “lies on” and “between”. We will use the symbol R^3 to mean the Euclidean 3-space.

A “point” is defined to be a point (or vector) $X = (x, y, z)$ of R^3 such that $\langle X, X \rangle = -k^2$ and $z > 0$. Such points comprise the “upper sheet” of a two sheeted hyperboloid in R^3 . We shall call this sheet H^2 . The constant k is a measure of the curvature of the hyperbolic plane.

A “line” is defined to be the intersection of H^2 with a plane through the origin of R^3 . We know from Euclidean solid geometry that such an intersection is one branch of a hyperbola (Fig. 3.1).

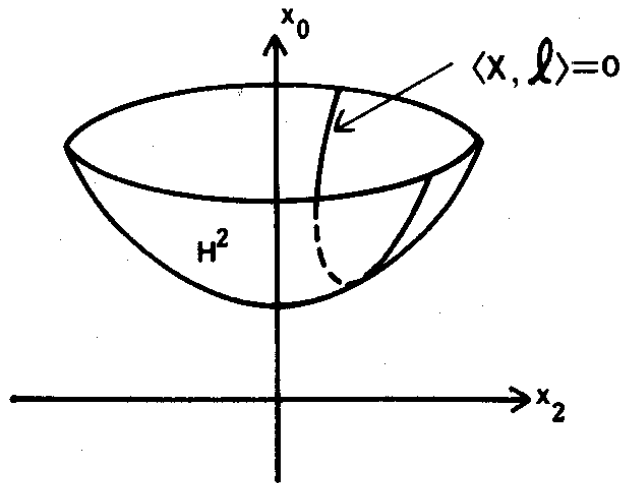


Figure 3.1: A “line” in the Weierstrass Model [Fab83]

Now, a plane through the origin is given by an equation of the form $\langle X, l \rangle = 0$, where l is an h-normal of the plane. Since l is an h-normal, any scalar multiple of l will also be an h-normal to the plane. We shall choose l in such a way that $\langle l, l \rangle = k^2$. This means that l is a point of the single-sheeted hyperboloid with the equation $\langle X, X \rangle = k^2$ (Fig 3.2).

Summarizing, a “line” is a section of H^2 by a plane through the origin of R^3 . There is a vector, l , such that $\langle l, l \rangle = k^2$ and the line consists of all X of H^2 such that $\langle X, l \rangle = 0$.

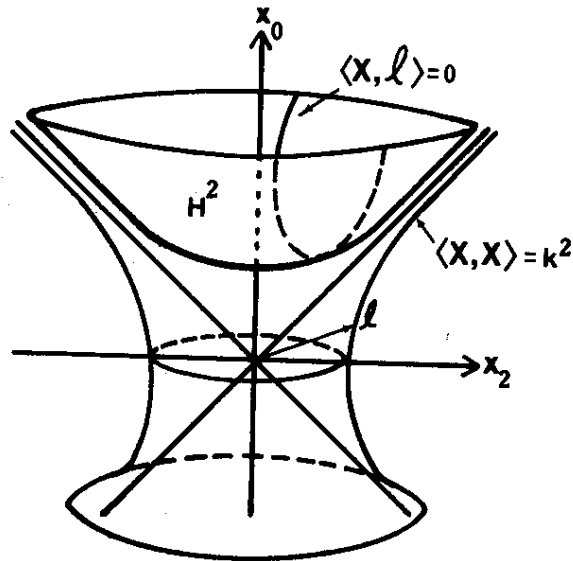


Figure 3.2: The vector l on the hyperboloid of one sheet $\langle X, X \rangle = k^2$ corresponds to a “line” in the Weierstrass Model [Fab83]

“Lines” are thus in one-to-one correspondence with the vectors l on $\langle X, X \rangle = k^2$. We shall now refer to a “line” by either giving its equation $\langle X, l \rangle = 0$ or by just giving its vector l .

A point X “lies on” the line l if and only if $\langle X, l \rangle = 0$.

Given three distinct points A , B and C on the line, we will say that C “lies between” A and B if, when the line is traversed in either direction, the three points are encountered in one of the orders ACB or BCA .

Now we define distance in H^2 .

The Lobachevskian distance between two points P and Q of H^2 is the unique

non-negative number $d = d_{PQ}$ satisfying

$$\cosh \frac{d}{k} = -\frac{1}{k^2} \langle P, Q \rangle \quad (3.2)$$

With distance thus defined, we can look at the concept of an *equidistant curve*. We saw that a “line” in the Weierstrass model is defined by the section of H^2 cut by a plane passing through the origin. *Equidistant curves* to one such line are defined as sections of H^2 cut by planes that are parallel to the defining plane of the line. Note that the term “parallel” is used in the Euclidean sense here.

This completes the discussion of concepts of the Weierstrass model that are relevant to the present work. The inquisitive reader may refer to Faber[Fab83] for a detailed treatment of the model.

3.2 The Poincaré Disk Model

This model was formulated by the French mathematician Henri Poincaré. It is an example of a finite model for hyperbolic geometry and is *conformal* in nature. This means that angles are represented faithfully in this model. However, distance is distorted. We will now look at how this model interprets the primitive terms.

A “point” in the Poincaré Model is defined to be a point $X = (x, y)$ in Euclidean 2-space (R^2), such that $x^2 + y^2 < 1$. Such points comprise the interior of a circle in R^2 with center at the origin and unit radius. Note that the points on the circumference of this circle are not included in the model. We shall refer to this circle as D or “The Poincaré Disk” interchangeably.

A “line” is defined to be a set of interior points of D that constitute an arc of a circle that is orthogonal to D (Fig 3.3). Open diameters of D are also defined as “lines” (open

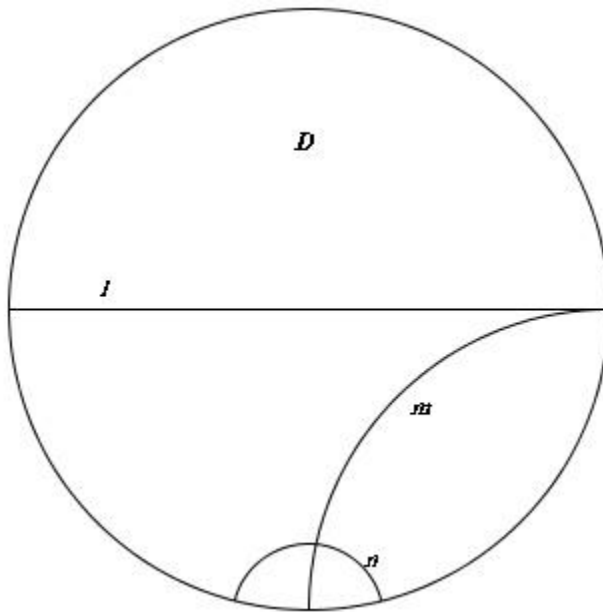


Figure 3.3: The Poincaré model, showing lines l , m and n . We say l and n are *divergently parallel*, m and n are *intersecting* and l and m are *asymptotically parallel*

diameters can be thought of as arcs of a circle of infinite radius and obviously intersect D orthogonally).

The terms “lies on” and “between” have the same meaning as in the Euclidean case in this model.

Since the Poincaré Model is conformal, the angle between two intersecting “lines” can be measured by measuring the angle between the tangents to the “lines” at the point of intersection.

3.3 Isomorphism between the Poincaré and Weierstrass Models

Quoting from Faber,

An isomorphism between two models of a given set of postulates is a one-to-one correspondence between the elements, relations and operations (respectively) of one model and those of the other, such that whenever a given relationship holds among certain elements in one model, the corresponding relationship holds among the corresponding elements in the other [Fab83].

In the present work we use the Poincaré Model for a visual display of the results and do all the computations for the transformations using the Weierstrass Model. The following is a discussion of various aspects of the isomorphism between the Weierstrass Model and the Poincaré Model that are relevant in this regard.

Consider the upper sheet of the hyperboloid $\langle X, X \rangle = -k^2$. Without loss of generality we can take the value of k^2 to be 1. Now we have the upper sheet of the hyperboloid $\langle X, X \rangle = -1$, which intersects the z -axis at the point $(0, 0, 1)$. If we project a point on this sheet down stereographically toward the point $(0, 0, -1)$, the projected point lies in a circle of unit radius with its center at the origin, in the xy -plane. This is the Poincaré Disk that corresponds to the upper hyperboloid sheet (Fig 3.4).

The projection from the Weierstrass to the Poincaré model is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \frac{1}{1+z} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \quad (3.3)$$

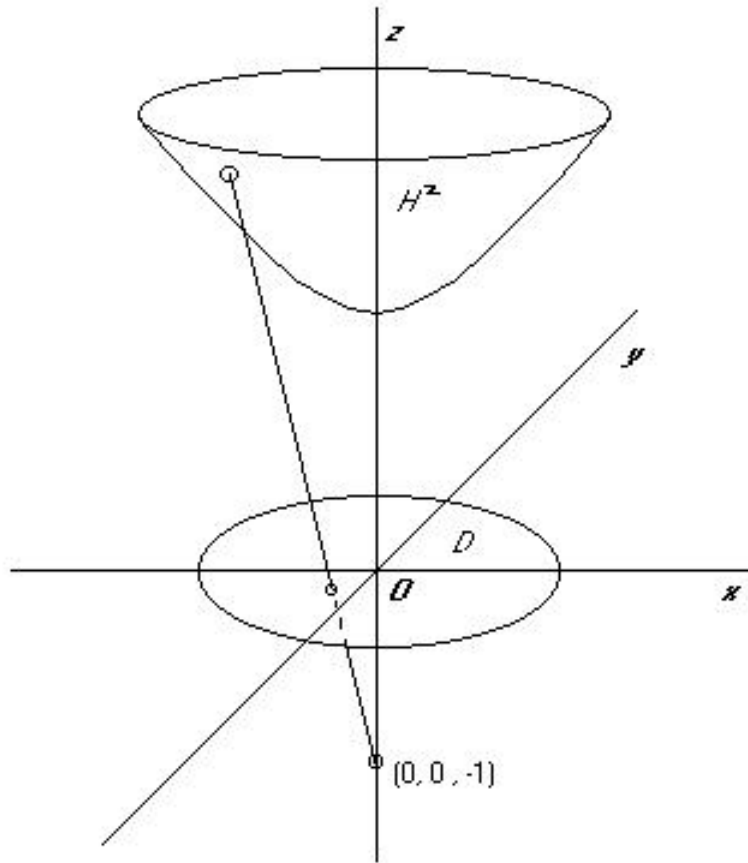


Figure 3.4: Stereographic projection between the Weierstrass model (H^2) and the Poincaré model (D)

And the inverse projection is given by:

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \rightarrow \frac{1}{1-x^2-y^2} \begin{bmatrix} 2x \\ 2y \\ 1+x^2+y^2 \end{bmatrix} \quad (3.4)$$

Chapter 4

Parametric Cubic Curves

Parametric cubics are piecewise polynomial curves in which any curve segment $P(t)$ is represented by $x(t), y(t), z(t)$, which are all cubic polynomials in the parameter t .

This representation is preferred in many graphical applications for various useful features. For example, to move from two to three dimensions, all that is needed is to add an equation for the z -coordinate. Also, multiple values of, say, y and z are possible for a given x . This makes the curve capable of “turning back” on itself or backtracking. It's also easier to handle difficulties like infinite slopes as slopes are replaced by tangent vectors in this representation [FvDF⁺94].

Cubic polynomials offer various advantages over non-cubic curves. They offer more flexibility in shape control than their lower degree counterparts while being computationally less expensive than the higher degree ones. They are also non-planar, unlike polynomials of a lesser degree.

In the sections to follow, we describe some general characteristics of parametric cubics and then provide a discussion of two types of such curves which are relevant to the present work, namely Hermite and Beziér curves. A reader who is familiar with these may skip

over the rest of the chapter.

4.1 General Characteristics

A curve segment $Q(t) = [x(t), y(t), z(t)]^T$ is defined by cubic polynomials of the form

$$\begin{aligned}x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z\end{aligned}\tag{4.1}$$

where $0 \leq t \leq 1$.

We can define the coefficient matrix of the above polynomials as

$$C = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix}\tag{4.2}$$

and the parameter matrix as $T = [t^3, t^2, t, 1]^T$, giving us a simplified form of eq. (4.1)

$$Q(t) = C \cdot T\tag{4.3}$$

The three cubics in eq. (4.1) can be determined if four conditions are specified, as there are four unknown coefficients in each of them. This is the basis for defining different types of cubic parametric curves. For example, Hermite curve segments are defined by specifying the two end points and the derivatives at the end points, giving four conditions.

It turns out that the curves are a weighted sum of the four geometric conditions or *restraints*. To see how the weights are determined, we provide a development following [FvDF⁺94].

The coefficient matrix of eq. (4.3) can be rewritten as:

$$C = G \cdot M\tag{4.4}$$

where, M is a 4×4 matrix called the *basis matrix* and G is a 4-element matrix called the *geometry matrix*. M basically specifies the effect of the defining conditions on the curve, while G represents the defining conditions themselves. For different types of curves, G or M or both of them are different. Thus eq. (4.3) can be rewritten as,

$$Q(t) = G \cdot M \cdot T \quad (4.5)$$

or,

$$Q(t) = [G_1 \quad G_2 \quad G_3 \quad G_4] \cdot \begin{bmatrix} m_{11} & m_{21} & m_{31} & m_{41} \\ m_{12} & m_{22} & m_{32} & m_{42} \\ m_{13} & m_{23} & m_{33} & m_{43} \\ m_{14} & m_{24} & m_{34} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

The right hand side of this equation is just three cubic polynomials in t as G and M are constants.

Let G_x refer to the row vector of just the x-components of G . Expanding out just $x(t) = G_x \cdot M \cdot T$ gives

$$\begin{aligned} x(t) &= (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) g_{1_x} \\ &\quad + (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}) g_{2_x} \\ &\quad + (t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}) g_{3_x} \\ &\quad + (t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}) g_{4_x} \end{aligned} \quad (4.6)$$

Equation (4.6) demonstrates the fact that the curve is a weighted sum of the geometric constraints.

Lastly, the weights, which are all cubic polynomials in t , are called *blending functions*.

4.2 Hermite Curves

Hermite curve segments are specified by two end points P_1 and P_2 and the tangent vectors at these end points, R_1 and R_2 . Together these give us four conditions required to solve for the coefficients in eq. (4.1).

The Hermite basis matrix, M_H , can be derived using the limiting conditions at $t = 0$ and $t = 1$ to be the following:

$$M_H = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

For the derivation the reader may refer to [FvDF⁺94].

The Hermite geometry matrix can be written as

$$G_H = \begin{bmatrix} P_1 & P_2 & R_1 & R_2 \end{bmatrix}$$

Let G_{H_x} be the x-component of the geometric constraints. We can write

$$G_{H_x} = \begin{bmatrix} P_{1_x} & P_{2_x} & R_{1_x} & R_{2_x} \end{bmatrix}$$

G_{H_y} and G_{H_z} have similar representations.

Thus we can solve $x(t) = G_{H_x} \cdot M_H \cdot T$, $y(t) = G_{H_y} \cdot M_H \cdot T$, and $z(t) = G_{H_z} \cdot M_H \cdot T$, to find

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}^T = G_H \cdot M_H \cdot T$$

Doing the math yields the following equation for the Hermite curve segment:

$$Q(t) = (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_2 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_2 \quad (4.7)$$

Finally, the above equation also shows the blending functions to be

$$B_H = \begin{bmatrix} 2t^3 - 3t^2 + 1 \\ -2t^3 + 3t^2 \\ t^3 - 2t^2 + t \\ t^3 - t^2 \end{bmatrix}$$

4.3 Beziér Curves

Beziér curve segments (named after the mathematician Pierre Beziér) are specified using four points. Two of these, P_1 and P_2 , are the endpoints of the curve, and the other two, C_1 and C_2 , called the control points indirectly specify the tangent vectors at P_1 and P_2 respectively.

The Beziér basis matrix can be derived to be the following:

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

For the derivation the reader may refer to [FvDF⁺94].

The Beziér geometry matrix can be written as

$$G_B = \begin{bmatrix} P_1 & C_1 & C_2 & P_2 \end{bmatrix}$$

The product $Q(t) = G_B \cdot M_B \cdot T$ is

$$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 C_1 + 3t^2(1-t) C_2 + t^3 P_2 \quad (4.8)$$

The above equation also shows the blending functions for the Beziér curve, known as Bern-

stein polynomials, to be the following:

$$B_B = \begin{bmatrix} (1-t)^3 \\ 3t(1-t)^2 \\ 3t^2(1-t) \\ t^3 \end{bmatrix}$$

In the present work we investigate if there is a way to define a parametric cubic curve using four points on the hyperbolic plane, such that it can be transformed by just transforming the four control points, instead of transforming it point by point. Cubics in the Euclidean 3-space already enjoy this nice property, being a linear combination of the elements of the geometry matrix. Those on the hyperbolic plane have additional constraints on them, in that, the tangent vectors at the endpoints should be tangent to the hyperbolic plane and all the points on the curve should satisfy the equation of a hyperboloid, i.e., $\langle X, X \rangle = -k^2$.

In the following, we present a brief discussion on the transformations that we have investigated.

Chapter 5

Hyperbolic Isometric Transformations

Isometric hyperbolic transformations are mappings of the n -dimensional hyperbolic space, \mathbf{H}^n , into itself, that preserve distance. Such transformations are also known as *isometries*. In the following discussion we will restrict ourselves to hyperbolic 2-space or \mathbf{H}^2 .

As an example, let T be such a transformation and P and Q be two arbitrary points of \mathbf{H}^2 . Then the distance between $T(P)$ and $T(Q)$ is equal to the distance between P and Q . Hence the term *isometric* in the name.

Four isometries are known for hyperbolic space, namely, *reflection*, *rotation*, *translation* and *glide reflection*. A discussion about each follows.

5.1 Reflection

A *reflection* is defined such that, if l is a line and P a point in \mathbf{H}^2 , then the reflection, P' , of P across l is a point for which l is the perpendicular bisector of the line segment from P to P' . All the points of the line l remain fixed in this isometry.

A reflection, *ReflectX*, across the x – *axis* is given by the matrix [Dun86]

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A reflection, *ReflectB*, across a line that is perpendicular to the positive x – *axis* and intersects it at a hyperbolic distance of b from the origin is given by the matrix [Dun86]

$$\begin{bmatrix} -\cosh(2b) & 0 & \sinh(2b) \\ 0 & 1 & 0 \\ -\sinh(2b) & 0 & \cosh(2b) \end{bmatrix}$$

where,

$$\begin{aligned} \cosh b &= (e^b + e^{-b})/2 \\ \cosh(2b) &= 2\cosh^2(b) - 1 \\ \sinh(2b) &= \sqrt{\cosh^2(2b) - 1} \end{aligned}$$

A reflection, *Reflect θ* , across a line passing through the origin at an angle of θ is given by the matrix [Dun86]

$$\begin{bmatrix} \cos(2\theta) & \sin(2\theta) & 0 \\ \sin(2\theta) & -\cos(2\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.2 Rotation

A *rotation* is defined by fixing a point in \mathbf{H}^2 , and rotating all the other points by a specified angle. We shall refer to the fixed point as the *pivot*.

Rotations, say by an angle 2θ , can be performed by reflection across two lines that intersect at the *pivot*, and make an angle equal to θ between themselves.

By the above principle, a counterclockwise rotation, *Rotate* O , around the origin by an angle of 2θ is the product $Reflect\theta \cdot ReflectX$, given by the matrix

$$\begin{bmatrix} \cos(2\theta) & -\sin(2\theta) & 0 \\ \sin(2\theta) & \cos(2\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.3 Translation

In general a *translation* is defined such that, it “slides” all points in the same direction by the same distance.

On the hyperbolic plane though, it is slightly more complicated than that. There it is realized through the concept of *equidistant curves*. In H^2 , a translation by a distance d is defined relative to a line l , as the transformation that “slides” all points lying on the *equidistant curves* of l , by a distance of d . Note that distance in this context is the hyperbolic distance.

5.4 Glide Reflection

Glide Reflection is a combination of a translation and reflection. It is defined relative to a line, as the transformation that translates all points in the direction of the line by a particular distance and then reflects them across the line.

On the hyperbolic plane though, translations are defined using *equidistant curves*. In this case, a glide reflection can be defined to be a transformation that translates all points along equidistant curves passing through them by a given distance, and then reflects them across the given line. Note that the points on the line itself only get translated.

Reiterating, all these transformations are “distance preserving”, in that, the distance between the transformed points is equal to the distance between the respective original points.

In the present work we have designed and implemented algorithms to perform the four isometries discussed in this chapter, for any arbitrary point of H^2 .

Chapter 6

Transformations of Approximate Hyperbolic Splines

The goal of the present work is to find computationally inexpensive splines primarily to be used in hyperbolic design applications.

The initial hope was to find splines that are analogs of the Euclidean Hermite and Beziér curves. The desired property was that these be invariant under hyperbolic transformations. This means that if S is a spline defined by control points $P_i (i = 1, 2, \dots, n)$, and T is a hyperbolic transformation, and S' is the spline defined by the control points $T(P_i)$, then $T(S(t)) = S'(t)$ (for all values of the parameter t). We have restricted ourselves to $n = 4$.

The advantage that this property offers is that to compute the transformed spline, it is only needed to transform the control points, and not each point on the original spline. This is especially useful when programming with Java or Postscript, which have built-in Beziér splines.

Two models of hyperbolic geometry are useful in this regard. The Weierstrass model has

been used for computations involving transformations. This model has the nice property that the transformations are represented by 3×3 matrices, which allows for the computation of a composite transformation as the product of the matrices representing the constituent transformations. The Poincaré model has been used for visualizing the results because of its conformal nature. These two models are related to each other by stereographic projection as given by Equations 3.3 and 3.4.

Thus, originally we hoped to be able to specify the control points in the Poincaré Model, project them up to the Weierstrass model, find a cubic spline (such as a Hermite or Beziér curve) on the Weierstrass model, and then project the spline back down to the Poincaré model – as a quotient of cubic polynomials.

To do this we need the curve to lie on the Weierstrass model as embedded in the Euclidean space. Unfortunately however, the control data (four control points or two end points and two tangent vectors) for the curve completely determines the curve as cubic polynomials of the parameter t . Thus, projecting the control points up from the Poincaré model gives us control points on (or vectors tangent to) the Weierstrass model, but not necessarily a curve which lies on it. In fact, as we discovered, generally such curves do not lie on the Weierstrass model.

In view of the above constraint, our work was divided into two distinct parts: 1) to find curves that could be used, and 2) to investigate these curves with respect to hyperbolic transformations.

In the following sections we deal with these two parts separately.

6.1 Approximate Hyperbolic Splines

To determine a spline that could be used for testing, we tried two methods: 1) We attempted to determine approximate splines that could be used in the Poincaré model, and 2) we attempted to compute a spline on the Weierstrass model that was a close approximation to the invariant splines that we originally sought. Doing this allowed us to compare the spline obtained from method (1) with that obtained from method (2) to see how close the one from method (1) was to being as invariant.

To get the best splines for method (1), we transformed arbitrary control points in the Poincaré disk, such that the end points lay equidistant on the u -axis on either side of the origin. We then used the Beziér curve determined by the transformed points as our approximate spline. The reason for taking this approach was that transforming the control points like this allows us to determine the curve in the “most Euclidean” part of the Poincaré model, where the distance is least distorted. Then we could transform the Beziér curve back using the inverse transformation, so that its control points coincided with the original curve. This uniquely determines the spline corresponding to the given control points. Finding the transformation and its inverse involved breaking the transformation up into simpler units that we could compute.

Generating the splines for method (2), however, was a more involved process. The goal was to find a spline that lies on the Weierstrass model. We began with transforming arbitrary control points on the Poincaré model up onto the Weierstrass, and then finding the 3D cubic spline given by these control points. As noted earlier, in general these splines do not lie on the Weierstrass model. The problem now was to get a spline that lies on the Weierstrass model, using the spline that we had. To do this we attempted to project the points on the spline to their nearest point on the Weierstrass hyperboloid.

Let $P' = (x', y', z')$ be such a point that we need to project to the hyperboloid. We are

looking for the point $P_0 = (x_0, y_0, z_0)$ that is the nearest point to P' on the hyperboloid. It is clear that the point would lie on the unique plane containing the point P' and the z -axis. This can be thought of as the rz -plane, where $r^2 = x^2 + y^2$, which allows us to do the calculations in rz -coordinates and finally convert them to corresponding x, y, z values. The calculations follow.

P' can now be represented as (r', z') , and P_0 as (r_0, z_0) . The distance, D , of P_0 from P' is given by

$$D^2 = (r_0 - r')^2 + (z_0 - z')^2 \quad (6.1)$$

Also we have the following equation for a hyperbola in the rz -plane:

$$r^2 - z^2 = -1 \quad (6.2)$$

Thus, to find the closest point we would have to minimize D at $r = r_0$ and $z = z_0$. Doing this however involved solving fourth degree equations. To overcome this constraint, we used Newton's iterative method.

The hyperbola on the rz -plane can be represented parametrically as $(\sinh t, \cosh t)$. This gives us,

$$D^2 = (\sinh t - r')^2 + (\cosh t - z')^2 \quad (6.3)$$

To minimize D , we take the derivative with respect to t , and equate it to 0, giving us the equation that we need to solve as

$$2 \cdot \sinh t \cdot \cosh t - \cosh t \cdot r' - \sinh t \cdot z' = 0 \quad (6.4)$$

In the usual case, Newton's method depends on a good choice for the "first guess" of the solution. Initially we tried to fix the first guess arbitrarily as $t = 1.0$. However, this did not give the necessary stability, and we had to come up with a better initial guess. Now, as we saw before, the Weierstrass hyperboloid is enclosed by the cone given by $\langle X, X \rangle = 0$. Therefore, in the rz -plane the hyperbola given by the equation (6.2) is enclosed by the lines

$r = \pm z$. Making use of this fact, we fixed the initial guess as the intersection of a line passing through P' , perpendicular to one of the two lines $r = \pm z$, and parallel to the other, with the line out of those two that it intersected.

However, we observed that Newton's method was not stable under some cases and therefore tried to add some iterations of the bisection method before iterating with the Newton's method. The bisection method needs two points as the initial guess, which we fixed to be the two points on the hyperboloid at the same z coordinate and at the same r coordinate as P' respectively. In other words the two points in a vertical and horizontal direction respectively from P' . In the case when P' had a z coordinate less than 1.0, we took the first point to be $(0, 1.0)$, and the second point at the same r coordinate as P' . Currently we are doing the computations with the number of iterations of the bisection method fixed at 5.

The procedure stated above gave us the spline we needed in most cases. Unfortunately though, the "nearest point" on the Weierstrass hyperboloid is ambiguous for some points in three space (points lying on the z -axis). To overcome this problem, we tried adding a fourth degree term to the z -coordinate of the splines, in order to "force" the center point (the point at $t = 0.5$) to lie on the hyperboloid. This would "pull" the spline away from the z -axis and closer to the hyperboloid. So if S is the spline, the polynomials representing the spline now looked like this:

$$\begin{aligned} S_x(t) &= S'_x(t) \\ S_y(t) &= S'_y(t) \\ S_z(t) &= S'_z(t) + C_z t^2(1-t)^2 \end{aligned} \tag{6.5}$$

where the S' terms represent the original spline, which can be either Hermite or Beziér.

The constant C_z could be computed using the condition that the point of the curve at $t = 0.5$ lies on the hyperboloid given by the equation $\langle X, X \rangle = -1$, where $X =$

$(x(t), y(t), z(t))$.

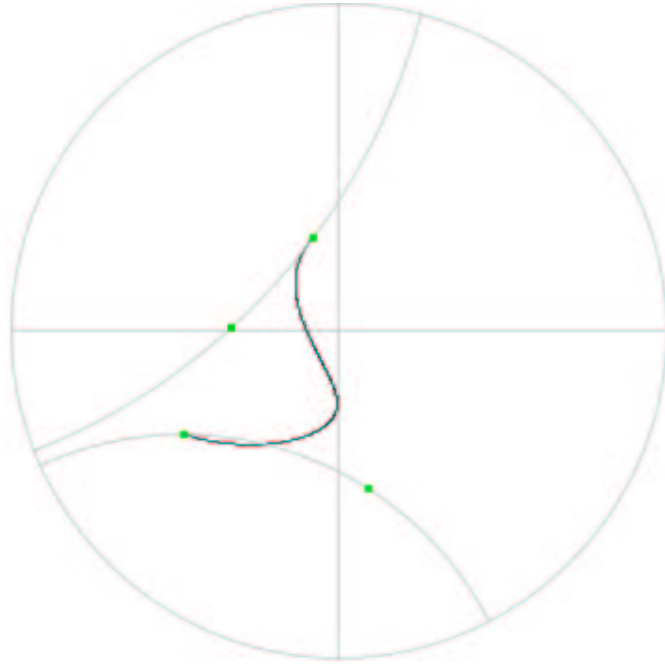


Figure 6.1: A spline drawn without the modification of control points. It clearly starts and ends in a direction tangential to the Euclidean straight line segment between the end point and the control point.

Thus we could generate a spline that lay on the Weierstrass hyperboloid. We could then project this spline back down to the Poincaré disk and compare it with the spline obtained from method (1). The two splines came out to be different for most cases and we decided to use the one obtained from method (2). Henceforth references to splines will imply a spline obtained from method (2).

The splines generated by both the methods above came out to be starting and ending tangential to the Euclidean line segment between the corresponding end point and control point (Fig. 6.1). However, a better way to generate the curve is to make it tangential to the hyper-

bolic line between the endpoints and the control points. This also has the effect of making the spline start and end closer to the Weierstrass hyperboloid.

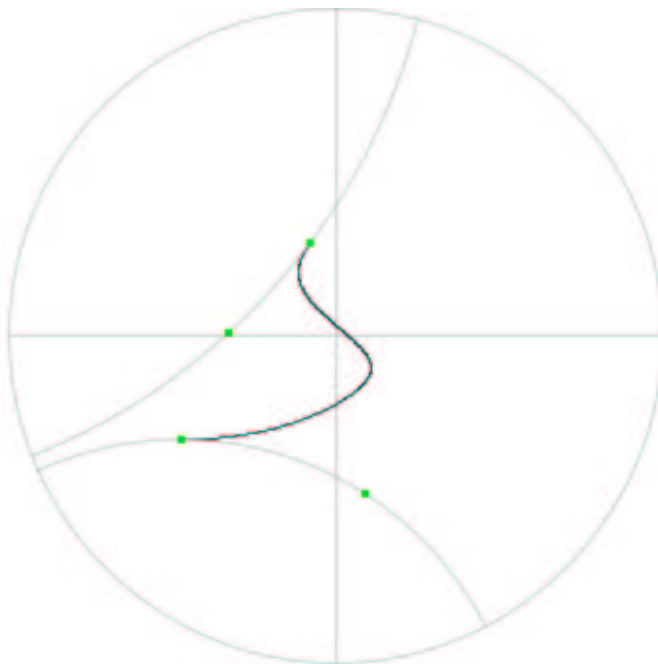


Figure 6.2: The same spline drawn after the modification now starts and ends tangential to the hyperbolic line between the end points and the control points.

Now, hyperbolic lines are represented as arcs of circles orthogonal to the Poincaré disk in the Poincaré model. Thus we calculated the modified control points to be in a direction tangential to the hyperbolic line between the end point and the control point, and at a distance equal to the hyperbolic distance between the end point and the original control point. We could use the Poincaré model for this purpose because of its conformal nature. The affect of the modification is illustrated by Fig. 6.2.

In the following section we discuss our investigations into hyperbolic transformations of the splines.

6.2 Investigation of Invariance Property

Given a spline and a hyperbolic transformation, the goal of this part was to compare the spline obtained by a point-by-point transformation and one obtained by just transforming the control points, and to find out how close these were to each other. If these turned out to be the same spline, we would conclude that the kind of spline we started with was indeed invariant under hyperbolic transformations. And if not so, we would measure how close it was to being invariant.

To perform a given transformation, we had to break it down into simpler transformations that we could easily compute. As noted earlier, some simple hyperbolic transformations are already known and we computed arbitrary transformations by breaking them down into these known ones.

For example, let T represent a counter-clockwise rotation about a given point P , by an angle of θ radians. This can be broken down into simpler transformations in the following steps:

- Step 1: Rotate P to the positive x -axis
- Step 2: Translate it along the axis to the origin
- Step 3: Perform the counter-clockwise rotation by an angle of θ around the origin
- Step 4: Invert the translation of step 2
- Step 5: Invert the rotation of step 1

It turned out, that in general, the spline generated by just transforming the control points was an approximation of the “proper” spline obtained by the point-by-point transform. So we had to come up with a way to measure how close an approximation it was. For this purpose, we averaged the hyperbolic distance between successive t points on the two trans-

formed splines. This average, which we now refer to as the *distance* between the splines, is obviously greater than or equal to 0, and the closer it is to 0 the better the approximation.

Also, due to the significance of the work in hyperbolic design applications, an important factor to consider was the visual quality of the result. In other words, how close to each other the splines appeared when drawn in the Poincaré disk. The Poincaré disk distorts distance as we get closer to the circumference, and so splines close to the circumference which have a large *distance* between them, may in fact appear very close to each other. So as we get closer to the edge, the closeness of approximation can decrease without affecting the visual quality of the result.

In the following chapter we present the results that we obtained.

Chapter 7

Results

We distinguish between two categories of results. One is visual results and the other is numerical results. Accordingly this chapter is divided into two parts; one displays some examples of the visual results that we obtained and the other presents numerical results.

The data sets of spline data that we used to generate splines for testing consisted of two categories. One category was generated under the following restraints:

$$\cosh D_{st-e} \leq 3.0, \cosh D_{p-cp} \leq 1.5$$

and the other under:

$$\cosh D_{st-e} \leq 6.0, \cosh D_{p-cp} \leq 3.0$$

where, D_{p-cp} is the hyperbolic distance between the end points of the splines and the corresponding control points, and D_{st-e} is that between the start and the end points of the curve. The first set of constraints, the tighter set, yields “smaller” splines and the second set yields “bigger” splines.

In general, we observed that as the control points got farther from the end points, the worse the results got. This can be expected because farther control points imply that the curve is

leaving the start point or coming into the end point with a greater velocity. In the case of the procedure that we use to generate the spline, this is a problem. We find the closest point on the Weierstrass hyperboloid to a given point on the spline. As the spline gets farther away from the hyperboloid, this distortion gets worse too, leading to the deterioration in results.

However, in hyperbolic design applications, and indeed in other possible applications like animation, this is unlikely to be a cause of great concern. It almost never happens that control points at great distances from the end points are used.

Following we present some figures illustrating the transformations. They are also accompanied by distance measures for the transformation shown.

The tables that follow show results averaged over datasets of randomly generated curves and transformations. There are two sets of tables. Tables number 7.1, 7.2 and 7.3 show results obtained using the former category of “smaller” splines, and the latter three tables (7.4, 7.5 and 7.6) show those obtained using the latter category of “larger” splines. As noted before, the results using the smaller splines are significantly better.

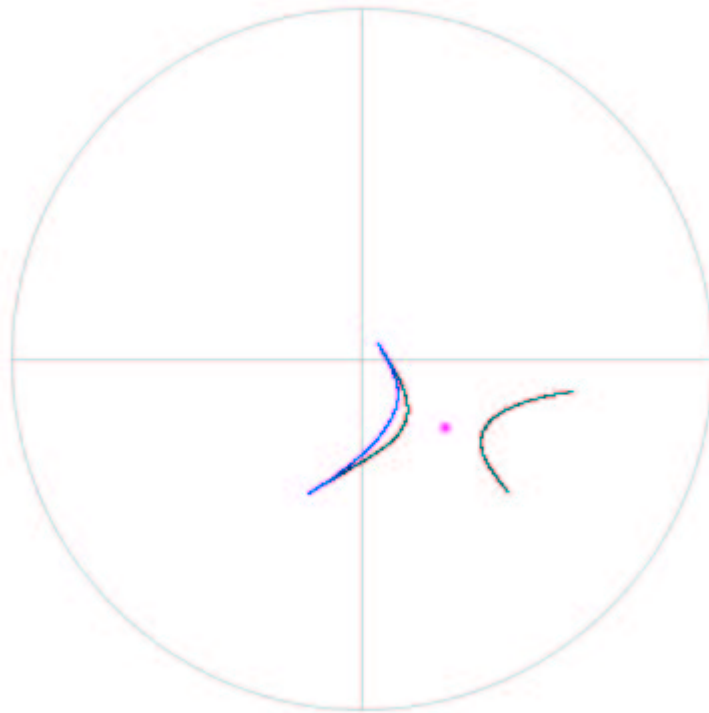


Figure 7.1: An example of a rotation. Distance = 0.0435455476152937

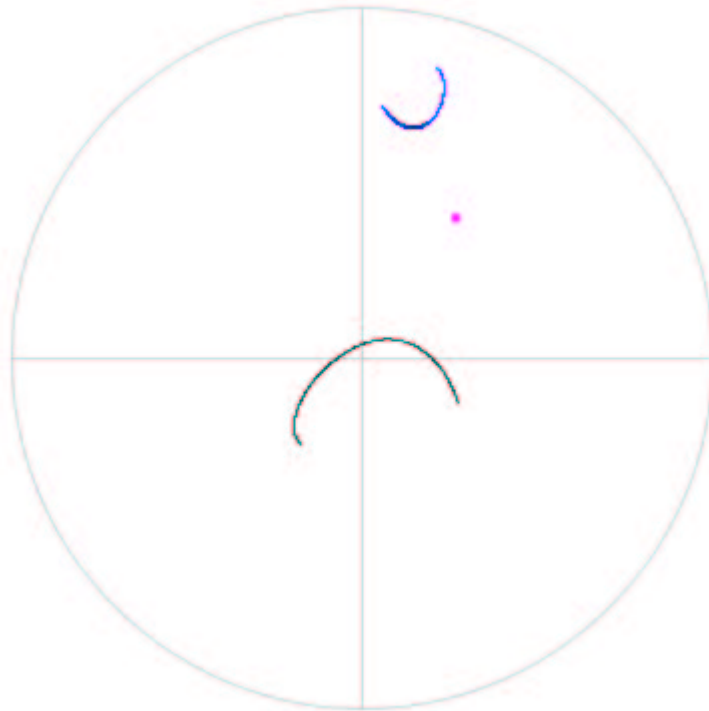


Figure 7.2: Another example of a rotation. Notice that even though we are quite close to the bounding circle, the distance measure demonstrates a good approximation, and the visual quality of the result is very good. Distance = 0.0141032093927292

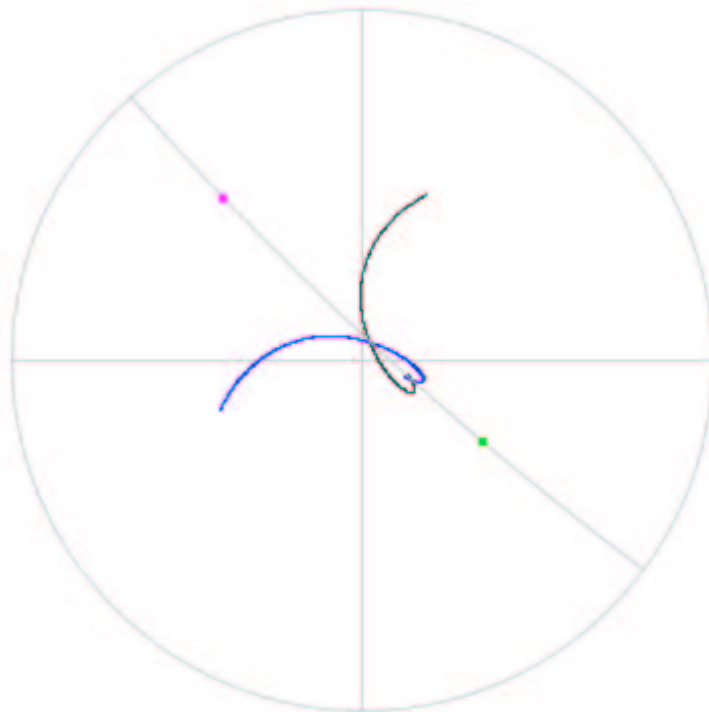


Figure 7.3: An example of a reflection. The visual quality of the result is excellent, and is reflected in the measure for distance as well. Distance = 0.00269178128091013

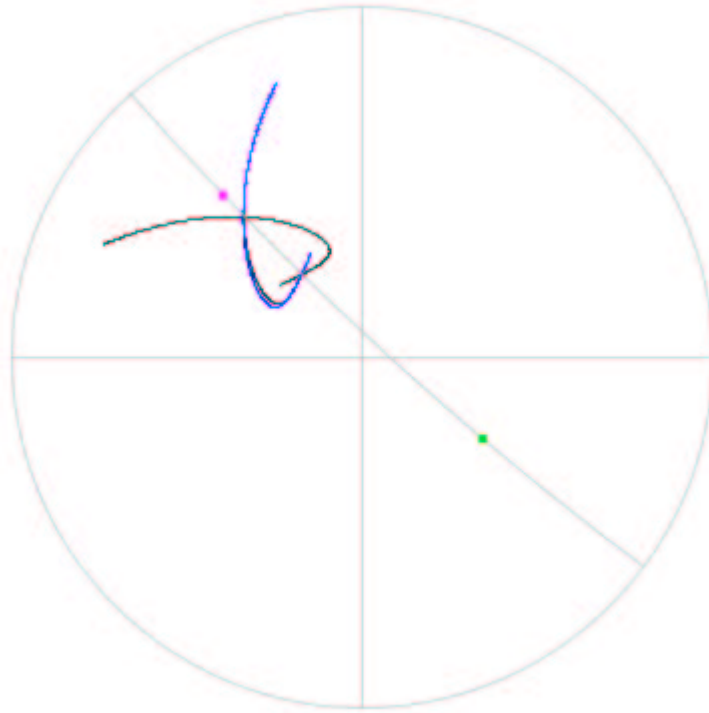


Figure 7.4: Another example of a reflection. An important thing to notice here is that although the visual quality of this result is almost as good as the previous one, the distance measure shows a significant difference. This is because here we are closer to the bounding circle and therefore the splines appear closer to each other than they actually are. Distance = 0.0109420035482458

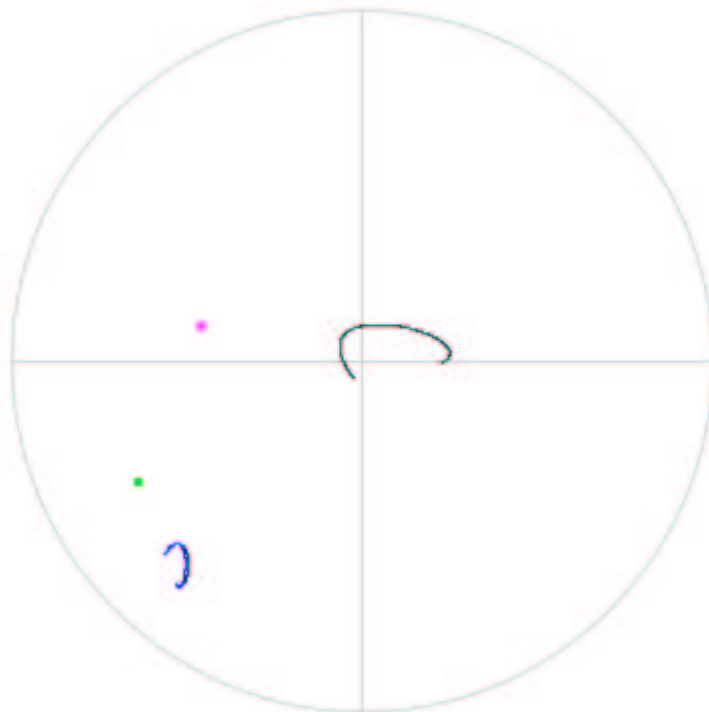


Figure 7.5: An example of a translation. Distance = 0.0258546892815096

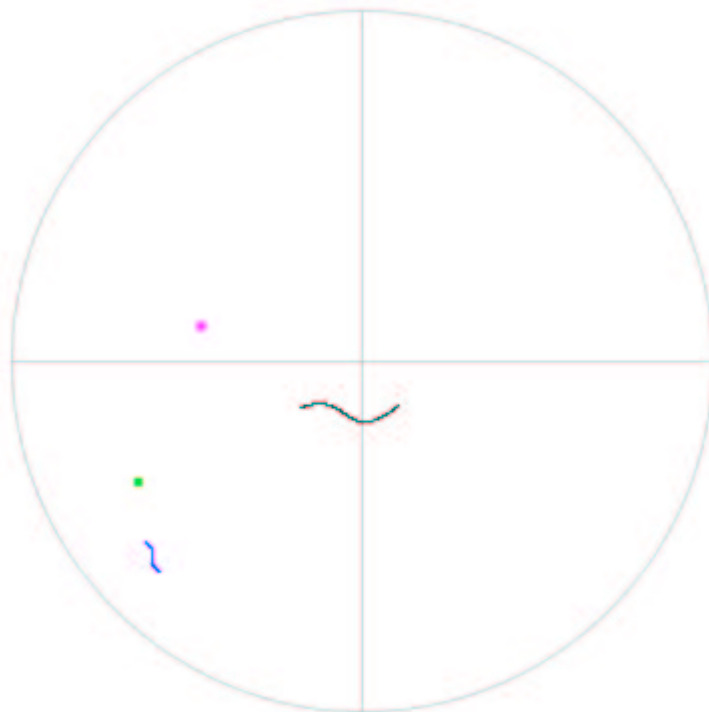


Figure 7.6: Another example of a translation. Distance = 0.0114738924405686

Table 7.1: Results for tests on reflections using “smaller” splines. Each row represents a different spline. Each column contains the mean and standard deviation of the distance calculated over twenty different reflections performed on the corresponding spline. The set of reflections is the same for each row.

spline no.	reflection	
	mean	std. deviation
1	0.09074907191186	0.03849831492933
2	0.04413098339948	0.01851047712963
3	0.01390871031456	0.00668362783522
4	0.06530247517171	0.02736078801349
5	0.07141472958803	0.06222468089613
6	0.03112659302034	0.01803494783386
7	0.05103098514916	0.02029454266853
8	0.02433702090279	0.01114449891385
9	0.02646942459994	0.01295185457799
10	0.06965220220430	0.03370714078628
11	0.01515872919653	0.00563913316124
12	0.03163699843335	0.01323787649124
13	0.03806028805545	0.02170797441315
14	0.01683052639629	0.01256985830737
15	0.01613930982411	0.00700544204881
16	0.02209205506493	0.01915274610527
17	0.05110178598993	0.02218087359154
18	0.03118512575595	0.03052715933614
19	0.01775247045888	0.01440520127323
20	0.13429989566771	0.07406166084172

Table 7.2: Results for tests on rotations using “smaller” splines. Each row represents a different spline. Each column contains the mean and standard deviation of the distance calculated over twenty different rotations performed on the corresponding spline. The set of rotations is the same for each row.

spline no.	rotation	
	mean	std. deviation
1	0.11281420714733	0.04960188074834
2	0.05036082476231	0.02058463363975
3	0.01186834790258	0.00616390000443
4	0.08418229256372	0.03886438098306
5	0.11104903054025	0.07051092404207
6	0.04289003465192	0.02053486280679
7	0.05445176299461	0.02376889722753
8	0.03255410968774	0.01486890727386
9	0.02609563764090	0.01309781948803
10	0.06643261078541	0.03876470027052
11	0.01780681331971	0.00909953987410
12	0.04026535439882	0.01907268691942
13	0.03465991565680	0.02046835486169
14	0.00984667054033	0.00843249392734
15	0.01976908561645	0.00837619462224
16	0.01534755915638	0.01527707643190
17	0.05209130734920	0.02512827875375
18	0.02154450220757	0.03872298852329
19	0.01416465805348	0.01396273471631
20	0.11900938437625	0.05992389974496

Table 7.3: Results for tests on translations using “smaller” splines. Each row represents a different spline. Each column contains the mean and standard deviation of the distance calculated over twenty different translations performed on the corresponding spline. The set of translations is the same for each row.

spline no.	translation	
	mean	std. deviation
1	0.11643298518364	0.02908047907656
2	0.05391767025518	0.00845916526627
3	0.01755364199592	0.00563544046481
4	0.09038082950078	0.02416769276515
5	0.08392325275167	0.03775227097658
6	0.03906188559797	0.01095048998801
7	0.06879594269537	0.01663736783249
8	0.03282090362188	0.00856377427447
9	0.03184708818017	0.01006241095041
10	0.09786867027047	0.04123146062956
11	0.02066999969893	0.00789866908547
12	0.04358837427556	0.01260013782790
13	0.04589385028256	0.02061747777653
14	0.01898716881058	0.00912192268300
15	0.02055214748707	0.00373283929352
16	0.02749590522061	0.02293441026364
17	0.06896623177206	0.02057868398707
18	0.04452237831307	0.03790803074454
19	0.02348946564058	0.02222106969100
20	0.17358172649861	0.05308047034855

Table 7.4: Results for tests on reflections using “bigger” splines. Each row represents a different spline. Each column contains the mean and standard deviation of the distance calculated over twenty different reflections performed on the corresponding spline. The set of reflections is the same for each row.

spline no.	reflection	
	mean	std. deviation
1	0.41423884120313	0.19897662720214
2	0.28082206884001	0.14342128091165
3	0.04741490759687	0.04271928268895
4	0.20738078997367	0.14151875349824
5	0.13447567665107	0.10161996912036
6	1.41022932803140	0.87689489006247
7	1.30575320863271	1.09230287984278
8	1.56427698729675	1.17644284675225
9	1.83777686823402	1.31802980922598
10	0.02964790231049	0.01726583854944
11	1.02064869963270	0.91737353873749
12	1.42964901079453	1.23600766074606
13	0.07145070174526	0.05131164123594
14	0.65536599355848	0.49377376088646
15	0.53379111713862	0.42062006595470
16	1.61398995306858	0.96360960635501
17	1.16714160270402	0.81611999399726
18	0.09560701878934	0.06089422760132
19	0.78190156437931	0.62915431656269
20	0.05607664999525	0.04615052209832

Table 7.5: Results for tests on rotations using “bigger” splines. Each row represents a different spline. Each column contains the mean and standard deviation of the distance calculated over twenty different rotations performed on the corresponding spline. The set of rotations is the same for each row.

spline no.	rotation	
	mean	std. deviation
1	0.60214689810358	0.43646794141635
2	0.27358223784640	0.13768018887196
3	0.03015862039904	0.03278081035341
4	0.18251728032639	0.19319362997169
5	0.10082601568039	0.07694810514837
6	3.57426550290739	2.97385083884546
7	1.67535075210902	1.58336478597213
8	2.34757745646432	1.75015865461362
9	2.64157060823435	2.08834207387484
10	0.05629550455474	0.04644003328406
11	1.23951304544350	1.18368091094263
12	1.80928921684443	1.45054775890032
13	0.11940051154576	0.07706658449521
14	1.09237149801804	0.90302675592395
15	0.61335934867249	0.50773303289393
16	3.52357536753898	2.78163137952210
17	3.15156556289846	2.77375883106845
18	0.06828876939324	0.04371752210033
19	2.34049874465652	2.30763117528019
20	0.03518294828688	0.03101675582114

Table 7.6: Results for tests on translations using “bigger” splines. Each row represents a different spline. Each column contains the mean and standard deviation of the distance calculated over twenty different translations performed on the corresponding spline. The set of translations is the same for each row.

spline no.	translation	
	mean	std. deviation
1	0.67866130571385	0.26538498763055
2	0.39220812249339	0.12832217954600
3	0.05937938759026	0.05056989101076
4	0.42653507052022	0.40440060947897
5	0.16180544744382	0.09817500808702
6	2.62116420994877	1.00449225544233
7	2.79571891321124	2.49377116585214
8	3.60487509209387	2.39619611680823
9	4.09080702936330	2.95849118317745
10	0.05146792102407	0.03655968560858
11	2.09474407023188	2.09443276024700
12	3.43700406531682	2.74916380036012
13	0.10705225695836	0.06986345518666
14	1.46341668437748	1.10387656389440
15	1.28625739485446	1.01532245242572
16	3.50221828230294	1.73818592188582
17	2.09158558197722	1.01089782572392
18	0.11414833911395	0.04554357315757
19	1.11355274276131	0.48412844638243
20	0.06440929182477	0.03789936028523

With these results, we show that the approximate hyperbolic splines that we have found are good approximations , certainly visually, and to a good degree numerically to the “ideal” splines obtained by the point-by-point transformation.

Chapter 8

Future Possibilities

In a recent paper, Buss et al. ([BP01]), describe a method to define spherical splines using spherical weighted means. This might be an interesting method to pursue in the case of hyperbolic splines as well. As we noted before, Euclidean splines are also defined as weighted sums.

We also believe that attempting to define hyperbolic splines in terms of fourth degree curves might give better results.

The tremendous amount of work done in the field of Euclidean splines and some amount in the field of spherical splines can be a good indicator of procedures to adopt to generate hyperbolic splines.

Chapter 9

Conclusion

Euclidean parametric cubic curves are widely used in graphical applications because of several nice properties they enjoy. It would be useful to find curves which have similar properties and lie on the hyperbolic plane.

In the present work, we have obtained an approximate hyperbolic spline that is reasonably close to being invariant under isometric transformations. It can certainly serve a good purpose when used in hyperbolic graphical applications. The approximate spline found by us offers a computational advantage over having to transform a spline point-by-point. This property can be of significant use in applications such as hyperbolic pattern generation, where using such “smooth” curves has been thus far computationally prohibitive.

We have also determined some directions that future work in this new field can take.

Bibliography

- [BP01] Samuel R. Buss and Fillmore Jay P. Spherical averages and applications to spherical splines and interpolation. In *ACM Transactions on Graphics*, volume 20, pages 95–126, April 2001.
- [Dun86] *Hyperbolic Symmetry*, volume 12B, 1986.
- [Fab83] Richard L. Faber. *Foundations of Euclidean and Non-Euclidean Geometry*. Marsel Dekker, Inc, New York and Basel, 1983.
- [FvDF⁺94] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Philips. *Introduction to Computer Graphics*. Addison-Wesley Publishing Company, 1994.
- [Gre94] M.J. Greenberg. *Euclidean and Non-Euclidean Geometries*. W.H. Freeman and Company, New York, 1994.