Generalized Helicoids for Hair Modeling

Emmanuel Piuze-Phaneuf

Master of Science

School of Computer Science

McGill University

Montreal,Quebec

2010-08-31

A thesis submitted to McGill University in partial fulfilment of the requirements for the degree of Master of Science in Computer Science

ACKNOWLEDGEMENTS

The past year has not been an easy one for me. I am truly grateful to the many friends and people that were there to listen, and offered their support. You guys made all the difference. I would like to thank Jonathan Guillemette for his omnipresence when I really needed it. What more could I ask for than a good friend, food, a frisbee, and a magnificent University campus during summer time to cheer me up? I would like to give a special thank to my awesome lab mates Parya MamayyezSiahkal and Svetlana Stolpner, for their friendship, guidance, and constant support. They made Room 404 a wonderful place to be and for that alone, I will always be there to debug their Java programs. I want to express sincere thanks to my supervisor Kaleem Siddiqi, for his wisdom and resourcefulness, and for his encouragement when things did not go so well. I thank him for believing in me and giving me the opportunity to work in a field I was not familiar with. I consider myself incredibly fortunate that our paths have crossed. I thank the School of Computer Science at McGill University, Kaleem Siddiqi, le Fonds Québécois de la recherche sur la nature et les technologies (FQRNT) and the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support. Finally, I would like to thank my girlfriend Judy Dumont, for so much. Not enough words or state-of-the-art renderings could fit in these pages to express how much I owe her for this thesis. She is the hands that have kept me above the waterline. Her cuisine is exquisite!

ABSTRACT

The modeling of hair and hair-like patterns, including those of grass and fur, is recurrent in the computer graphics literature. The reproduction of the extensive variety of these arrangements, and their generation from scarce information, remains a challenge for the computational models that describe them. This thesis introduces a novel mathematical model of hair-like patterns to the field of computer graphics, one that is based on a class of minimal surfaces called generalized helicoids. This representation is characterized by intuitive parameters that control the curvature of a hair along its tangent, normal and binormal directions, and its elevation angle. This representation equips a hair with information not only about its own geometry but also about the geometric behavior of other hairs in its vicinity. A hair modeling framework is proposed to investigate these properties. This framework includes an implicit surface sampling method based on interacting particles for determining hair root locations. It also introduces algorithms for interpolating generalized helicoids from sparse hair samples as well as a fitting algorithm for modeling unparameterized hair datasets. The usefulness of the generalized helicoid is motivated via several applications including the generation of different types of hair geometry, hair interpolation, hair fitting and wisp generation.

ABRÉGÉ

La modélisation de cheveux et de structures s'apparentant aux cheveux, incluant celles de l'herbe et de la fourrure, est courante dans la littérature en informatique graphique. La reproduction de divers arrangements de cheveux, et leur génération à partir d'information limitée, reste un défi pour les modèles mathématiques de cheveux. Cette thèse introduit une nouvelle représentation mathématique de structures de cheveux au domaine de l'informatique graphique. Cette représentation est basée sur une classe de surfaces minimales appelée hélicoïdes généralisés, et est caractérisée par des paramètres intuitifs contrôlant la courbure d'un cheveux le long de ses directions tangentes, normales et binormales, ainsi que son angle d'élévation. Non seulement cette représentation équipe-t-elle un cheveux avec de l'information sur sa propre géométrie, mais elle indique aussi le comportement géométrique des autres cheveux dans son voisinage. Afin d'étudier ces propriétés, une plateforme de modélisation de cheveux est proposée. Cette plateforme inclu une méthode d'échantillonnage pour surface implicite basée sur l'interaction de particules afin de déterminer l'emplacement de la racine des cheveux. Elle présente aussi des algorithmes pour interpoler des hélicoïdes généralisés à partir d'échantillons de cheveux clairsemés ainsi qu'un algorithme de *fitting* (ajustement) pour la modélisation d'ensembles de données comportant des cheveux non paramétrisés. L'utilité de l'hélicoïde généralisé est motivée par plusieurs applications, notamment la production de différents types de géométrie de cheveux, l'interpolation de cheveux, le *fitting* de cheveux, et la génération de brins de cheveux.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

Hair-like patterns, comprised of dense collections of curve-like elements, are ubiquitous in our world [60]. Consider examples arising in nature such as hair, fur, grass, feathers, and white matter fiber tracts in the brain, or those arising in man-made structures, such as textiles, optical fibers or threads. The realistic modeling of such patterns in computer graphics remains a significant challenge, in part because of the difficulty in obtaining accurate measurements from real data and in part because of the inability of current computational models to capture the extensive variety of hair-like arrangements. Whereas there is an intuitive sense these structures can be straight, wavy or curly, modeling them mathematically is not trivial. Hair patterns and their specific distribution and adaptation to internal and external forces form a complex physical system, making their study and effective reproduction computationally challenging. Furthermore, the typical large number of strands involved in pair-wise interactions currently makes the modeling and rendering of fiber patterns by manual placement and direct computations intractable for applications constrained by efficiency and time complexity.

The prospect of a simple and elegant mathematical model encompassing both the local flow and geometry of hair-like patterns is thus of particular interest for the field of computer graphics. To simplify the modeling of hairs, assumptions are often made about their length, their continuity and their local geometry. For example, when such patterns are generated through biological processes, there often is a coherency between the orientation and curvature of neighboring hairs. In the computer vision literature such locally parallel dense curves have been referred to as texture flows [6]. The coherency is a direct result of the local forces – static, interaction, and external – that apply on a larger scale than each individual hair. This suggests the use of local geometric models to incorporate neighboring information, without requiring the underlying forces to be explicitly modeled.

1.1 Overview

This thesis investigates an approach for generating, fitting, and interpolating hair patterns. An emphasis is placed on hair modeling as the driving application for the methods proposed. Many intricacies of hair modeling are shared with those of fur, feathers, and grass modeling, and so the proposed methods should hold for these and related cases as well. Important complications related to the modeling of hairs are addressed. First, the explicit manual construction of individual strands in a hair pattern is a laborious task. Considering that there are typically more than $100,000$ strands [73] on a human head, this also leads to a description of a hair volume that is computationally inefficient for further applications. Conversely, the interpolation of a subset of key hair strands, if not carried out carefully, can lead to a largely inhomogeneous and inconsistent hair pattern, where coherency between local neighbors is not enforced.

Motivated by these considerations, this thesis introduces a parametric model for generating, fitting and interpolating hair patterns to computer graphics, one that is based on the *generalized helicoid*, a well-known minimal surface [14]. The generalized helicoid has found use in the computer vision literature, in particular for texture analysis [6], and in the medical imaging literature [76]. Its application to hair modeling, however, is novel, and is the objective of this thesis.

The generalized helicoid can be expressed in such a way that it characterizes a hair strand using three parameters that control curvature in the tangent, normal, and binormal directions to the strand, and a fourth parameter to control elevation angle above a tangent plane [77]. This intuitive representation equips a strand with information not only about its own geometry but also about the "flow" of hairs in a local volumetric neighborhood. As such, the model is not merely a representation of a single hair, but rather, captures a volumetric bundle of hairs in the neighborhood of a hair strand. The model can be used for a variety of applications, including hairstyle synthesis and hairstyle reconstruction from sparse samples.

As a result, the model can be used for a variety of applications including hairstyle synthesis and hairstyle reconstruction from sparse samples.

1.2 Contributions of this Thesis

The major contributions of this thesis are the following:

1. The introduction of a generalized helicoid-based framework to model the geometry of a hair strand and its local (volumetric) neighborhood.
2. The synthesis of different hair types, including wisps, by sampling from a generalized helicoid-based representation.
3. The use of the model to "fill-in" a patch of hair between hair strand samples, by interpolation, and the use of a particle sampling method for implicit surfaces to determine hair root locations.
4. The use of the model to fit unparametrized hair strand data, which when combined with interpolation allows for efficient hairstyle reconstruction from sparse samples.

1.3 Outline

Chapter 2 presents a review of the computer graphics literature on hair modeling. Chapter 3 introduces the generalized helicoid model. Chapter 4 presents a particle method for sampling implicit surfaces. Chapter 5 investigates the interpolation of the generalized helicoid representation on possibly sparse and irregular datasets. Chapter 6 develops an approach for deriving a correspondence between an arbitrary hair strand (described as a series of points) and its generalized helicoid representation. Finally, Chapter 7 reviews the concepts discussed in this thesis and presents possibilities for future work.

CHAPTER 2
Review of Hair Modeling

The problem of modeling realistic hair patterns, from the construction of a single hair to the animation and rendering of a full hairstyle, is non-trivial. Nonetheless, hair modeling can be divided into three common themes: *hairstyling*, *hair simulation*, and *hair rendering* [84]. Hairstyling addresses the geometrical aspects of hair, and accounts for its general static shape. Hair simulation incorporates the notions of external or interaction forces (friction, collisions, wind, etc.) and time (dynamics), into the hair modeling framework. Hair rendering is in charge of polishing the visualization of the resulting hair.

Hairstyling, hair simulation and hair rendering each comprise numerous subproblems. What complicates matters is that many levels of the hierarchy of subproblems interact with one another i.e., they seldom work independently. Therefore, proponents of a novel hair modeling method typically need to incorporate it into an existing framework in order to investigate its usefulness. Take hair rendering for example – in order to experiment on a novel rendering algorithm, one must first decide what computational model of hair geometry will be used, and how it will be simulated.

The same computational model can also be used to generate hairs and to simulate them, in which case the parameters governing the dynamics of the hair are embedded into its geometric description. This is the case with the *super-helix* mechanical model, employed in Bertails et al. [10], where a hair strand is described by a parameterized piecewise super-helix with each piece corresponding to an individual helix with constant curvature and torsion. The dynamics of hair are directly encoded into the parameters of each helix.

There are other times when hairstyling, hair simulation and hair rendering are completely decoupled from each other, with different and possibly noninteracting methods being used for each. For instance, parametric curves and surfaces, generalized cylinders, cantilever beams, fluid flows, particle methods, vector fields and hand sketches are all different methods used for hairstyling [84], but many don't apply well to hair simulation. In these cases, other methods are used to ensure the continuity of the hair modeling process.

In the hair modeling literature, important issues have been somewhat ignored, with advances made in more "exciting" areas. Rendering is one such topic, and the literature on the subject has expanded considerably in the last few years, particularly in top conferences and journals in the field (I3D, SIGGRAPH, Eurographics, ACM Transactions on Graphics, to name a few). Moreover, the increasingly exquisite hair animations seen in blockbuster movies seem to indicate that the general problem of hair modeling is nearly solved. In reality there are other important and less discussed issues in the hair modeling literature, such as those of finding better geometrical models of hair, fitting hair to data, and interpolating hairs. These topics tend to be discredited as they can be addressed by – computationally and time expensive – brute-force methods. The cost of running these brute-force approaches to completion could be reduced by advances in such topics, and the amount of required human intervention could be lowered.

The generation of complete hairstyles from minimal or sparse data, which will be referred to as the *hair reconstruction* problem, is one such recurrent and unsolved problem in the computer graphics literature. Little work has been carried out on the development of approaches for efficiently and realistically interpolating hair from sparse samples. Moreover, hair reconstruction is highly dependent on the computational hair model at work. Hairs are often represented as series of points, vector fields, meshes, or textures, obtained through either an explicit (discrete) or implicit (continuous) generative process. In the case of explicit models, void spaces can be filled by directly interpolating the position of the points in key selected hair strands (commonly referred to as *guide hair strands*). On the other hand, for implicit methods, the interpolation problem is generally simpler and in some cases, it simply amounts to tracing additional streamlines in a continuous medium.

This thesis investigates an approach for hair reconstruction using a novel parametric model for hair, the *generalized helicoid*. This parametric representation of hair contrasts with the ones aforementioned in that each hair, described as a piecewise helicoid, carries information about its own geometry and also about the curvature of hair strands in its neighborhood. The generalized helicoid model makes it possible to produce a coherent hairstyle from sparse hair data. A small sample of hair strands is first required to shape the global hairstyle, and then additional hairs are interpolated from this sample. Hairs are generated from their implicit (parametric) representation using a streamline tracing algorithm, described in Chapter 3. As such, the generalized helicoid representation can be described as a *hybrid* between implicit and explicit approaches.

Going back to the more general problem of modeling the geometry of natural fibers, there is an extensive literature to cover. Fiber patterns – particularly hair, fur, and grass – have been studied and modeled extensively in computer graphics for the past three decades. This literature review focuses on work that is related to the modeling of hair geometry in computer graphics. The topics were selected to cover as much as possible of the relevant literature for this thesis. For a review of hair dynamics and hair rendering, two topics that are not directly addressed in this thesis, see Bertails et al. [9], Parke et al. [65], Ward et al. [84], Hadap et al. [32], and Magnenat et al. [55].

2.1 Optimizations in Hair Modeling

The human scalp is typically populated by more than 100,000 hair strands, that vary in width from 0.05 mm to 0.09mm [73]. Consequently, the computation of the geometry, dynamics and rendering of all hairs is intractable at the highest level of resolution, if hairs are considered individually. The generation of a small hair volume having a consistent and natural geometry is already a difficult problem on its own. Furthermore, the problem of hair dynamics does not scale well as the hair count increases, as the number of possible pairwise hair interactions increases quadratically. In hair rendering, the more hairs that are considered, the more complex it is to compute shadows and the scattering of light through the hair medium. These issues become considerably more important in real-time applications. Thus, simplifications and optimizations are an essential part of any hair modeling framework, regardless of the methods or regime (offline, real-time) at hand.

Most solutions avoid considering each hair individually, and instead work with a subsampling of the complete hair count, down to a few samples named *guide hairs*, representative of the full hairstyle. Additional hairs can be interpolated from this subset of guide hairs, giving it a globally consistent shape. Locally, hair *clusters* can be generated from individual guide hairs to improve both the texture of the hair and the efficiency of the computations. These two optimizations are respectively discussed in Sections 2.1.1 and 2.1.2. Other simplifications arise from the nature of the computational hair model that is used. For instance, the simplest way to model the appearance of hair is through the use of textures applied to polygon patches. Due the great variety of such methods, the possibilities are numerous, and they are described in

Section 2.2, grouped into the categories that best describe the mathematical processes they involve.

2.1.1 Guide Hairs

A useful optimization in hair modeling is the use of *guide hairs* that represent a coarse approximation to a full hairstyle. This optimization is employed in most hair modeling techniques, and is carefully investigated by Alter [2]. The process often goes as follows. A few selected guide hairs are first placed on the scalp. Ideally, these strands are selected such that they describe the large scale impression of the hairstyle. Then, a denser and consistent representation of the hairstyle is obtained by interpolating across guide hairs. The measure of how well interpolation scales with the number of hairs is highly dependent on the geometric hair model at work. The interpolation can be done before the hair dynamics are solved, or after, depending on the density of the initial sampling and the complexity of the hairstyle. Many issues arise with regard to the orientation and geometry of the hairs that are interpolated from guide hairs. This particular point is addressed in Chapter 5.

2.1.2 Wisps, Clusters, Strips and Generalized Cylinders

Depending on the degree of refinement in the guide hair interpolation step described in Section 2.1.1, the overall hairstyle may still look sparse and undersampled. The extrapolation of guide hairs to form what is referred to as *wisps*, *clusters*, *strips* or *generalized cylinders* can be used to both fill in missing hairs and to speed up the animation of the resulting hairstyle. This operation involves the grouping of many hairs together, in order to simplify the computations. Wisp extrapolation contrasts with guide hair interpolation in that consistency is not directly enforced between hairs belonging to different wisps. This is a cheap and efficient way to pack in additional hairs at a low computational cost. The wisps are often formed before solving for the hair dynamics, as they provide a way to constrain the interactions between larger volumes (e.g. piecewise generalized cylinders) and involve less pairwise interactions. For simple hairstyles, this optimization is highly convenient at the design, animation, and rendering stages. For more complex hairstyles, the level of detail might be so high that consistency plays a crucial role in the visual appearance of the hairstyle. In such cases, either individual hairs are generated in great number, leading to expensive computations, or interpolation is carried out at a finer level of detail.

Although the wisp model is often used in the hair modeling literature, some articles address it more specifically. Watanabe and Suenaga [86] is one of the first to introduce the hair wisp abstraction, and uses a trigonal prism model as a representation of hair. Wisps are generated by offsetting existing hairs in random directions. Chen et al. [16] also uses a trigonal prism wisp model, and the full hair is divided into several groups of wisps, following their position on the skull. Each wisp is constructed using continuous trigonal prisms, defined by three connected 3D b-spline curves. In contrast, Yang et al. [92] and Xu et al. [90] use a cluster-based hair model. A volume density model is embedded into a generalized cylinder that specifies the shape of the hair envelope. Then, more recently, Plante et al. [67] uses a wisp model for simulating interactions inside long hair, where hair strands are clustered into wisps consisting of a skeleton and its (deformable) envelope to capture the global and local motion of hair, respectively. In Choe et al. [17], wisps are represented as generalized cylinders and are generated by offsetting a (Catmull-Rom spline) master strand. In Section 3.5, the generalized helicoid model is presented as a way to efficiently generate coherent and realistic hair clusters around skeletal guide strands.

2.2 Hair Models

Computational models for hair can be classified into two main categories: those which are *explicit* and those which are *volumetric* or *implicit*. Explicit models use a representation for hair strands that is based on a collection of geometric primitives, for instance, generalized cylinders [39], mechanical rods [29], or super-helices [10]. Implicit models instead use the evaluation of a distribution function over space, for instance particles in a continuous medium [5], or a type of fluid flow [30]. Explicit approaches generally provide a greater control over the local shape of the hair than implicit approaches, and can fully benefit from the most advanced hair rendering techniques. However, the modeling and rendering of hair using explicit methods can be quite time consuming. On the other hand, whereas implicit approaches can be straightforward to use and incorporate useful mathematical properties such as continuity, smoothness and flow into the global shape of the hair, they typically do not offer sufficient precision and control. Furthermore, they can fail to reproduce hair styles where certain intrinsic assumptions related to the nature of the implicit method are violated.

A compelling argument against explicit methods is that since they model individual hairs in a discrete fashion, it is not immediately clear how they can capture geometric coherency between neighboring strands, or allow for hair curvature to vary consistently across the scalp. In contrast, implicit methods are prone to the oversimplification of hair geometry. The generalized model presented in this thesis addresses these limitations. It may be viewed as a hybrid between explicit and implicit approaches, benefiting from the advantages of both worlds. In particular, whereas individual hair strands are represented explicitly by a geometrical model, the model is truly implicit and enforces coherency between nearby strands while allowing hair curvature to vary arbitrarily across the scalp.

2.2.1 Explicit Methods

Explicit approaches to hair modeling describe hairs as a series of connected geometric primitives. The main advantage of explicit hair models is that they provide optimal precision and control (see Bergou et al. [7] and Singh et al. [79] for instance) and can fully benefit from the most advanced hair dynamics and rendering algorithms. Their major drawback is that they involve a great number of hair primitives, and thus require a lot of storage, and can be slow to render. Explicit approaches generally use interpolating splines for representing hairs in the final rendering stage.

Some methods use connected *trigonal prisms* as a representation of hair [16, 57, 86]. Other methods describe hairs as polygonal chains or *polylines* [2, 34, 44, 58, 80]. Some methods use straight *cylindrical segments* connected by points to describe a hair [56, 20, 47]. A cluster-based hair model extending the idea of simple cylindrical hairs can also be used [68, 17, 39, 90, 92]. In this case, a master strand or a volume density model is embedded into a *generalized cylinder* that specifies the shape of the hair envelope. Some make use of a pseudo-physical hair system where hairs are represented as serial *multi-body chains* or spring-connected masses rotating about hinges [29, 18, 4, 74]. Lee et al. [48] extends this idea by using a simplified cantilever beam model. Plante et al. [67] uses a multi-layer explicit wisp model to represent a hair. The hair is composed of three layers: a skeleton center polyline that defines large-scale motion and formations; a deformable envelope that defines the deformation of the wisp; an extrapolation of the skeletal polyline to fill the wisp with hair strands. Piecewise helices, called *super-helices*, can be used to describe hairs [13, 8, 12, 10]. Kmoch et al. [42] use *elastic rods* to describe hairs. In Reeves et al. [70, 71], the first model of hair-like objects making use of particles was introduced. Hair-like objects are represented as a cloud of primitive particles that define its volume. Similarly, Bando et al. [5] uses *loosely connected particles* to represent a hair strand. For particle methods, hair interaction is controlled via a particle system driven by physical forces.

2.2.2 Implicit Methods

Implicit approaches to hair modeling describe hair as a continuous medium, in which each location in space is associated information regarding the behavior of hair in its vicinity. Texture-based methods are included in this category. Although simple to use, in general implicit methods provide a less precise control over the local shape of the hairstyle. There are rendering algorithms designed specifically for some implicit methods that can be faster than those for explicit methods (see Lengyel et al. [51] for instance). However, these rendering algorithms are very specific and usually fail to generalize for other hair modeling methods.

Csuri et al. [19] introduce one of the first implicit methods to model hair-like objects in the graphics literature. Each hair is modeled as a texture applied on a triangle defined on a polygonal surface. A z-buffer algorithm is used for hidden surface removal. Similarly, Ivanov et al. [36, 49] use textured polygonal meshes. Others use a generalization of this texturing approach as an intermediate model between geometry and texture [61, 38]. The geometry of the hair is encoded in textures that are distributed everywhere in space, and are referred to as *texels*. Yuksel et al. [94] use *hair meshes* to define the large-scale topology of the hairstyle, and generates a refined version of it by growing hairs along the meshes. Perlin et al. [66] employ volume densities, controlled with pseudo-random functions, to generate soft fur-like objects. In Goldman et al. [27], a probabilistic model is used to approximate the lighting properties of a furry surface. The geometry of hair is completely disregarded, and thus, this method is only suitable for fur viewed at a great distance. Xu et al. [90] and Yang et al. [92] use a cluster-based hair model where a volume density model is embedded into a *generalized cylinder* that specifies the shape of the hair envelope. Others model the hair shape as streamlines of a *single* fluid flow, wrapped around the scalp and characterized by sources and vortices [93, 31, 30]. This work is perhaps the one that is most similar to the methods described in this thesis. The fluid flow described in these articles is defined once for the whole hair volume, while the generalized helicoid model assigns a different fluid flow to each hair.

2.2.3 Multiresolution Methods

Multiresolution approaches are very similar to MIP (*multum in parvo*, meaning "much in small space") maps in computer graphics in the context of texture filtering. In the typical rendering of a scene, a fully detailed texture is applied on an object, and then a scaling transformation is applied on it as the viewer moves closer or further from it. In contrast, mipmap methods instead precompute subsampled versions of the same texture, and depending on the level of resolution that is required (as determined by the distance from the object to the viewer), the appropriate subsampling is picked. In the context of hair modeling, multiresolution approaches work the same way. Depending on the distance from the viewer to the hair, a different level of detail or computational model of hair is used. For instance, a cluster of hair viewed from a great distance is almost fully opaque and could be replaced by a generalized cylinder. On the other hand, when located very close to a hair tuft, a fine resolution could be selected such as to make the distinction between individual hairs possible. This kind of optimization can dramatically speed up the dynamics and rendering of hair, although the transition from one level of resolution to the other must be dealt with carefully (e.g. see Ward et al. [85] for example).

Kim et al. [40] and Koh et al. [43] use a dual representation for hairstyles comprised of mostly long hairs: a large scales, hair is approximated by *connected surface patches* or *strips*; a smaller scales, hair is represented by a set of *thin cylinders* spread on these surfaces. This duality is combined to form what is referred to as a *thin shell volume*. Lengyel [51] and Lengyel et al. [50] combine explicit and simplified volumetric methods for rendering fur and short hair. When the viewer is close to the hair, a polyline hair representation is used, while from distant views, a fur coating is modeled using *concentric texture layers*. Ward et al. [82], Wang et al. [83] and Kim et al [39] use a hierarchy of *generalized cylinders* to represent hair. Ward et al. [85] use a hierarchical representation for hair – strips, clusters and then individual strands. Their framework switches seamlessly between different levels of detail using the methods described in Funkhouser and Séquin [25]. In Kong and Nakajima [44], a visible volume buffer is proposed, where hairs that are close to the camera are rendered as thin polylines, and the background hairs are rendered as thick polylines.

2.2.4 Capturing Hair Model to Images

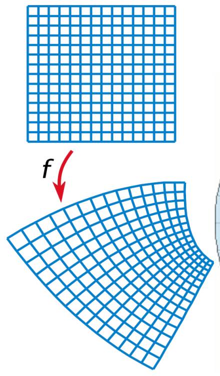
Kong et al. [45] is considered the first approach to use real hair pictures to automatically create hairstyles [84]. A 3D hair volume is approximated using the 2D outline of different viewpoints, and hair strands are reconstructed through this volume following a simple point matching heuristic. Grabli et al. [28] use a shape-from-inverse-lighting approach, where a stationary viewpoint is used along with a moving light source. A synthetic reflection profile is determined and used for extracting the orientation of hair strands. Paris et al. [63] extends this idea by placing a lower bound on the number of images and sequences required, and develops a more sophisticated methods for reconstructing all visible hairs. The 2D orientation of a hair is inferred from its anisotropic behavior, and its 3D orientation is reconstructed from the reflection of multiple light rays. Similarly, Wei et al. [87] use images captured from multiple views to recover the hair geometry. Hair strands are triangulated from local per pixel orientations at each viewpoint. Paris et al. [64] build on these ideas and simplifies the capture of complex features such as concavities and curls. Yamaguchi et al. [91] use an array of synchronized video cameras to capture dynamic hairstyles. Hair is reconstructed at each frame using an orientation-preserving algorithm. Jakob et al. [37] use macrophotographs with a shallow depth of field of a hair and extracts features from these. They use a sequence of least squares iterations to grow hair fibers in a cloud of hair features derived from the series of images.

2.3 Follicle Sampling

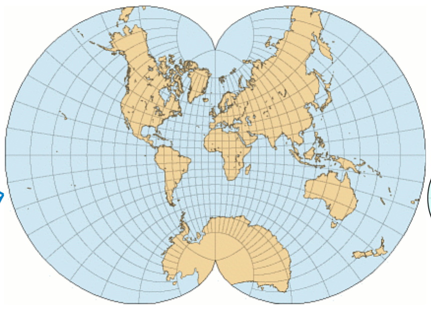
Regardless of the computational hair model used, the locations where hairs are grown need to be determined. The great variability in the topology of surfaces and in the clustering of follicles complicates this task. For instance, sampling might be required on sharp edges (e.g. near an ear) and there might be locations on the scalp where varying hair distributions are required (e.g. balding spots). Different sampling methods have been used in the graphics literature, and these can be divided in two broad categories, those making use of *surface mappings*, described in Section 2.3.1 and those using *surface interpolation* methods, described in Section 2.3.2.

2.3.1 Surface Mappings

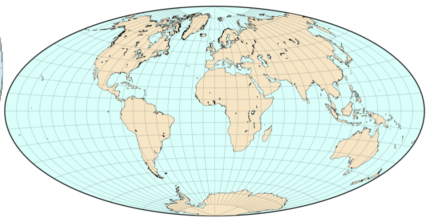
Surface mappings seek to find an invertible surface transformation that yields a representation of the original surface that is more convenient to work



(a) Conformal mapping



(b) August (conformal) map



(c) Aitoff (equidistance) map

Figure 2–1: Surface mappings. (Adapted from [26])

with. For instance, we find 2D world maps more often than 3D ones since they convey the same information, are cheaper to produce, and take less space. Typically, the transformation is applied on the vertices of a geometrical surface.

Certain types of constraints concerning how the geometry and relationship of neighboring points are preserved in the mapping can be enforced. For instance, *conformal mappings* locally preserve oriented angles between curves passing through the same point. Other properties include the preservation of distances (equidistance, isometry), geodesics (great circles), directions, area (equivalence), distortion patterns, and angle deformation patterns.

Figure 2–1 shows different types of mappings. In general, such analytical mappings are hard to find for arbitrary surfaces, except in some simple cases such as the plane mapping example shown in Figure 2–1a. Wang et al. [82] is a good example of how surface mappings can be used in the context of hair modeling. They use scalp space coordinates for modeling the curved hair volume bounded by the scalp from below. The coordinate system is defined as a mapping of the scalp points onto a sphere. The head scalp can then be sampled by using the parameterization obtained through a spherical projection. Figure 2–2 shows the scalp parameterization for this mapping.

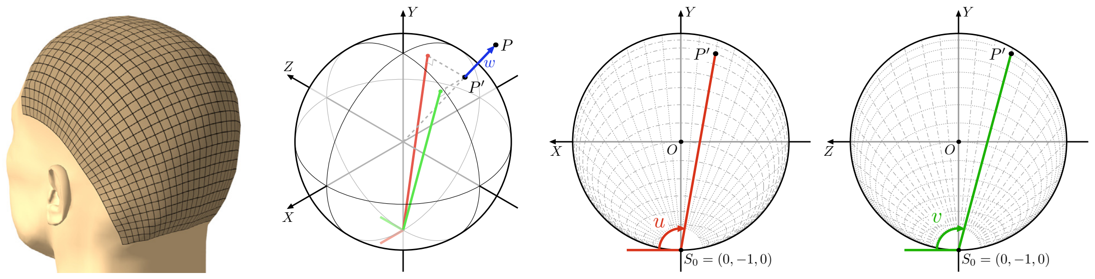


Figure 2–2: 3D Scalp Space Parameterization. A world space point $P(x, y, z)$ is represented with its spherical projection P' on the unit sphere. The scalp space coordinates (u, v) are given as two angles ranging from 0 to π. (Adapted from Wang et al. [82])

2.3.2 Surface Interpolation

There are cases where there might be no convenient mapping, or none flexible enough to include large variations of the surface being sampled. For instance, we might seek a hair sampling method that could be used for both a human scalp, and a topologically different surface such as the back of a dog, while remaining computationally efficient and allowing for user input. Wang et al. [82] showcases the inability of mapping methods to easily tackle these requirements altogether. The distortion induced by their parameterization increases as the value of one of their parameters (y) decreases, leading to inconsistent mappings in the case of highly irregular surfaces. Moreover, non-conformal mappings can yield a confusing representation for complex and folded surfaces, and are thus not well suited for interactive user input.

Surface interpolation methods are typically more useful when dealing with complex surfaces, and when users are required to interact. Moreover, they pair well with particle-based methods for sampling surfaces. They involve the making of an approximative surface, through an interpolation process driven by some constraints on the resulting topology, and on a similarity to the original surface. The approach used in this thesis, namely that of using a particle system to sample an implicit representation of an arbitrary three-dimensional mesh, is described in Chapter 4.

CHAPTER 3
Hairs as Generalized Helicoids

This thesis investigates an approach for generating, fitting, and interpolating hair patterns using a parametric model which is based on the *generalized helicoid*. Although the generalized helicoid has been used in the computer vision literature for texture analysis [6], and in the medical imaging literature [76], its application to hair modeling is new. The generalized helicoid provides an intuitive representation that equips a hair strand with information not only about its own geometry but also about the flow of hair in its neighborhood.

Using the generalized helicoid model, a hair strand $\mathbf{H}$ rooted at position $\mathbf{r}$ can be represented in explicit (discrete) form as a sequence of points $\mathbf{H} = \{\mathbf{r}, \mathbf{p}_1, \cdots, \mathbf{p}_n\}$, and parameterized in implicit (continuous) form using $\mathbf{H} \rightarrow (\mathbf{k}, \mathbf{r}, \mathbf{M})$. The significance of the parameter vector $\mathbf{k}$ is described in Section 3.2. $\mathbf{M}$ refers to the local transformation matrix aligning the hair at $\mathbf{r}$. Depending on the problem at hand, either the explicit or the implicit description of a hair will be used. There are other cases where both descriptions will be used simultaneously.

Section 3.1 first introduces topics on the differential geometry of space curves that are used throughout this thesis, more precisely, the Frenet frame is discussed. Then the generalized helicoid model is introduced in Section 3.2. Section 3.3 shows how the explicit form of a hair can be obtained from a spherical mapping of its implicit parameterization. In Section 3.4, intrinsic properties of the generalized helicoid are discussed. Finally, the concept of a single generalized helicoid is extended in sections 3.5 and 3.6 to respectively form wisps and composite (piecewise helicoidal) hairs.

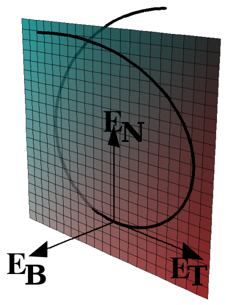


Figure 3–1: Frenet frame $\mathbf{E}_T, \mathbf{E}_N, \mathbf{E}_B$ traveling along a space curve $\mathbf{c}$. The osculating plane is spanned by $\mathbf{E}_T$ and $\mathbf{E}_N$.

3.1 Hair Framing: Computing the Frenet Tetrad

For various applications, the local frame describing the orientation of a hair at its root and at various locations along its arclength need to be computed. More generally, there is a need to determine a coordinate frame travelling along with the hair, a process refered to as *curve framing*. This coordinate frame should allow to study the differential geometry of the hair under investigation (specifically, its curvature and torsion). Different approaches exist for framing spatial curves, including quaternion-approaches, parallel frames, Frenet frames, and the simple "fixed-up" method [23, 11, 41]. In this thesis, the Frenet frame approach is used, since it naturally corresponds to the geometrical motivation behind the generalized helicoid model, it smoothly varies along the curve, is consistent, and is fairly easy to compute in the discrete case.

In $\mathbb{R}^3$, the Frenet frame of a curve $\mathbf{c}$ is a reference frame that moves along its length, and is uniquely determined by three orthonormal and positively oriented vectors $\mathbf{E}_T, \mathbf{E}_N, \mathbf{E}_B$, referred to respectively as the tangent, principal normal, and binormal vectors. The tangential and normal components span the osculating plane, as shown in Figure 3–1. In the case of a continuous and differentiable curve, the accompanying 3-frame (triad) is given by Kuhnel et al. [46]:

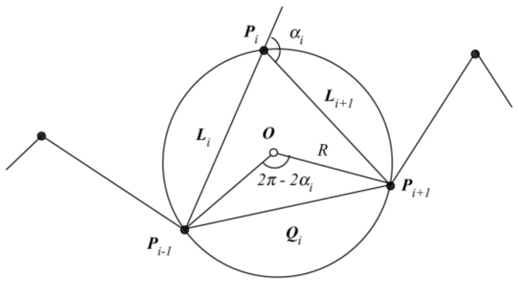


Figure 3–2: Discrete differential geometry on a polyline.

$$\mathbf{E}_T = \frac{\mathbf{c}'}{||\mathbf{c}'||} \tag{3.1}$$

$$\mathbf{E}_N = \frac{\mathbf{E}_T'}{||\mathbf{E}_T'||} = \frac{\mathbf{c}' \times (\mathbf{c}'' \times \mathbf{c}')}{||\mathbf{c}'||\,||\mathbf{c}'' \times \mathbf{c}'||} \tag{3.2}$$

$$\mathbf{E}_B = \mathbf{E}_T \times \mathbf{E}_N = \frac{\mathbf{c}' \times \mathbf{c}''}{||\mathbf{c}' \times \mathbf{c}''||}. \tag{3.3}$$

where $||.||$ is the Euclidean norm, and $\mathbf{c}'$, $\mathbf{c}''$ respectively denote the first and second derivatives of the curve with respect to its arclength. Furthermore, $\mathbf{c}$ is uniquely determined, up to Euclidean motion, by its curvature κ and torsion τ, which are directly related to the Frenet frame of the curve:

$$\kappa = ||\mathbf{E}_T'|| \tag{3.4}$$

$$\tau = \mathbf{E}_N' \cdot \mathbf{E}_B \tag{3.5}$$

where $\cdot$ is the inner product in $\mathbb{R}^3$. A complete relationship between the Frenet triad and the curvature and torsion of the curve is defined by the Frenet-Serret theorem [62]. The discrete curvature κ_i and torsion τ_i at a point $\mathbf{p}_i$ with tangent $\mathbf{L}_i$ can be estimated by ([3, 52, 53]):

$$\mathbf{L}_i = \mathbf{p}_i - \mathbf{p}_{i-1} \tag{3.6}$$

$$\cos(\alpha_i) = \frac{\mathbf{L}_i \cdot \mathbf{L}_{i+1}}{L_i L_{i+1}} \tag{3.7}$$

$$\kappa_i = \frac{2\sin(\alpha_i)}{\left|\left|\mathbf{p}_{i+1} - \mathbf{p}_{i-1}\right|\right|} \tag{3.8}$$

$$\mathbf{V}_i = \mathbf{L}_i \times \mathbf{L}_{i+1} \tag{3.9}$$

$$\mathbf{N}_i = \frac{\mathbf{V}_i}{V_i} \tag{3.10}$$

$$\cos(\beta_i) = \mathbf{N}_{i-1} \cdot \mathbf{N}_i \tag{3.11}$$

$$\Delta_i = \det(\mathbf{L}_{i-1}, \mathbf{L}_i, \mathbf{L}_{i+1}) \tag{3.12}$$

$$\tau_i = \operatorname{sgn}(\Delta_i)\frac{\sin(\beta_i)}{L_i} \tag{3.13}$$

where $L_i = ||\mathbf{L}_i||$ and Δ_i is the (signed) area formed by the triangle p_{i-1}, p_i, p_{i+1}.

Since backward tangent estimations are used, the first tangent is not well-defined. The first tangent is set identical to the second one. This assumption makes sense since the polylines forming hairs are opened, and no prior information exists for how the curve should behave in the vicinity of the first tangent. The first tangent is therefore obtained by:

$$\mathbf{L}_1 = \mathbf{L}_2 = \mathbf{p}_2 - \mathbf{p}_1. \tag{3.14}$$

For a hair represented as a polygonal chain (defined by a connected series of points), there is no direct analytical expression $\mathbf{c} = \mathbf{c}(t)$ that describes the curve along its arclength t. Therefore, a discrete version of the Frenet frame is used that would approximate that of the continuous curve if its description was available. A polygonal chain or *polyline* is a discrete curve formed by a sequence of vertices $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n$. A rough but sufficient approximation of the differential geometry in the case of a discrete curve for three successive poins

$\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}$ can be found in Sapidis et al. [75]. The Frenet frame is given by:

$$\mathbf{E}_T = \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{||\mathbf{p}_i - \mathbf{p}_{i-1}||} \tag{3.15}$$

$$\mathbf{E}_B = \frac{\mathbf{E}_{T,i} - \mathbf{E}_{T,i-1}}{||\mathbf{E}_{T,i} - \mathbf{E}_{T,i-1}||} \tag{3.16}$$

$$\mathbf{E}_N = \mathbf{E}_B \times \mathbf{E}_T. \tag{3.17}$$

which respectively correspond to backward tangents, tangent differences, and cross products of tangent differences. A transformation matrix $\mathbf{R} \in \mathbb{R}^{4\times 4}$ from the canonical basis of $\mathbb{R}^3$ (world coordinates) to the basis $\mathbf{E}_T, \mathbf{E}_B, \mathbf{E}_N$ located at a location $\mathbf{p}$ can then be represented as the orthogonal matrix

$$\mathbb{W} = \begin{bmatrix} | & | & | & 0 \\ \mathbf{E}_T & \mathbf{E}_N & \mathbf{E}_B & 0 \\ | & | & | & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & | \\ 0 & 0 & 0 & \mathbf{p}_i \\ | & | & | & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

$$= \mathbb{E} + \mathbb{T} \tag{3.19}$$

Since $\mathbb{E}$ is orthogonal, $\mathbb{E}^T = \mathbb{E}^{-1}$ and $\mathbb{E}^T - \mathbf{T}$ is therefore the inverse transformation matrix to the canonical basis. The Frenet frame is not defined at locations where the second derivative vanishes and the curve is locally flat (i.e. three consecutive points are collinear):

$$\mathbf{E}_{T,i} = \mathbf{E}_{T,i-1} \tag{3.20}$$

$$\text{or } \mathbf{E}_{T,i} = 0 \tag{3.21}$$

$$\text{or } \mathbf{E}_{T,i-1} = 0. \tag{3.22}$$

A heuristic based on the magnitude of the normal component is used to determine when this is the case:

$$||\mathbf{E}_N|| < \epsilon. \tag{3.23}$$

where $\epsilon \to 0$. If equation 3.23 is satisfied at a point $\mathbf{p}_i$, the framing algorithm backtracks along the curve until it finds an appropriate frame $\mathbb{E}$ or the identity matrix, if the backtracking reaches the root.

3.2 The Generalized Helicoid Model

The generalized helicoid is a well-known mathematical object and is the only nonplanar ruled minimal surface. More generally, Catalan [15] showed that any minimal ruled submanifold of a Euclidean space is part of a generalized helicoid. To get a more intuitive feel of the generalized helicoid, consider the following. If a twisted curve rotates about a fixed axis and is displaced parallel to this axis such that the displacement rate is always proportional to the angular velocity of rotation, then this curve generates a generalized helicoid [89, 69]. In the context of Diffusion MRI analysis, the generalized helicoid model has been shown to describe the two orientation angles $\theta(x, y, z)$ and $\phi(x, y, z)$ of a 3D flow via four parameters: $\mathbf{k} = (k_T, k_N, k_B, \alpha)$, in a Cartesian reference frame [76]. In spherical coordinates, these two orientation angles are given by

$$\theta(x, y, z) = \arctan\left(\frac{k_T x + k_N y}{1 + k_N x - k_T y}\right) + k_B z, \tag{3.24}$$

$$\phi(x, y, z) = \alpha\theta(x, y, z). \tag{3.25}$$

Here θ represents the orientation in the xy plane, ϕ represents the elevation angle out of this plane, and α is a constant. As shown by Savadjiev [77], this model has two key mathematical properties. First, the hypersurfaces (x, y, z, θ) and (x, y, z, ϕ) are both minimal surfaces, i.e., surfaces with mean curvature zero. As such, the orientation of the flow is guaranteed to vary smoothly in a local neighborhood. Second, the parameters k_T, k_N, k_B are each related to a notion of a tangential, normal and bi-normal curvature, respectively, of the underlying flow field.[1]

In the context of texture flow, the scalar orientation function $\theta(x, y, z)$ represents the local behavior of the flow at a point in space, and is the solution to the projection of the frame field of a curve satisfying Cartan's connection equations [62]. The texture flow manifold $s(x, y, z)$ is then represented by the 4-tuple

$$(x, y, z, \theta(x, y, z)). \tag{3.26}$$

By imposing the notion of a slowly varying dominant orientation, certain constraints are enforced on the behavior of $\theta(x, y, z)$. More precisely, the following harmonic energy is minimized:

$$\int ||\nabla\theta||^2 \, dV \tag{3.27}$$

and also the surface tension by the hyperarea functional

$$\int \sqrt{1 + \theta_x^2 + \theta_y^2 + \theta_z^2} dV. \tag{3.28}$$

The right or "generalized" helicoid is a solution to these constraints. See Savadjiev et al. [76] and Ben-Shahar and Zucker [6] for more information regarding the theory supporting the generalized helicoid model and for a derivation of the curvature parameters. Although this thesis does not extend the

[1] In fact, if one drops the terms related to the coordinate z in the model, one obtains the right-helicoid which is known to be a minimizer of a harmonic energy. This model has been used by Ben-Shahar and Zucker [6] to model 2D (i.e. planar) texture flows in computer vision.

theory concerning the generalized helicoid, it proposes a novel way to represent it in explicit form and proposes its use for hair modeling as a direct application.

A more intuitive way to describe this model is to define a frame field at the origin, with its tangential, normal and bi-normal components given in spherical coordinates by

$$\mathbf{E}_T = (\cos\phi\cos\theta, \cos\phi\sin\theta, \sin\phi) \tag{3.29}$$

$$\mathbf{E}_N = (-\sin\theta, \cos\theta, 0) \tag{3.30}$$

$$\mathbf{E}_B = (-\sin\phi\cos\theta, -\sin\phi\sin\theta, \cos\phi)\,. \tag{3.31}$$

Let the tangential component be aligned with the direction of a local flow pattern. The generalized helicoid then describes the fashion in which the frame field must be rotated, in a local neighborhood, to fit the flow pattern at a point (x, y, z). The three curvature parameters k_T, k_N, k_B describe the curvature along flow lines in the tangential plane, the curvature across flow lines in the tangential plane and the curvature of flow lines out of this plane, respectively[2] . These planes define a Frenet frame along the helicoid.

[2] Another, more elaborate way of describing the effect of the generalized helicoid parameters k_T, k_N, k_B, α in the context of hair tracing is the following:

- k_T defines the main orientation quadrant of the hair, either in the positive-y or negative-y portion of the x-y plane, with (always) positive x.
- k_N defines the amount of normal curvature in the hair, pointing towards its osculating circles. This is the same as pulling the tip of the hair and wrapping it in a coil-like fashion.
- k_B defines the amount of local torsion in the hair. This torsion twists the hair away from the osculating frame.
- α defines the main growth quadrant of the hair, either in the positive-z or negative-z portion of the x-z plane, with (always) positive x.

It is important to note that the generalized helicoid differs from the super-helix model presented by Bertails et al. [10]. Single streamlines in the generalized helicoid behave qualitatively like super-helices and in fact, a direct relationship might exist. However, the correspondence does not go beyond this point without extending or simplifying one or the other of the models. The generalized helicoid is itself a higher dimensional object and carries information (e.g. smoothness of the surrounding flow) about a volumetric neighborhood of streamlines, not just of a single hair strand.

3.3 Tracing Vector Field Lines

The generalized helicoid representation can be used to generate a hair, by tracing a path along an initial tangential direction of the flow field, which is assumed to be aligned with the x axis at the origin. More precisely, starting from the origin, the orientation $\theta(x, y, z)$ is evaluated in space and a step is taken in the tangential direction $(\cos\phi\cos\theta, \cos\phi\sin\theta, \sin\phi)$ of the generalized helicoid frame field. Additional steps are taken until a desired fiber length step count is reached. This essentially follows a forward Euler approach, described in Algorithm 1. The scale of the actual polyline generated with this algorithm can be adjusted appropriately, following the length scale predetermined by the problem at hand. Figure 3–4 illustrates the effect of varying the curvature parameters k_T, k_N, and k_B on both the generalized helicoid (a planar slice is shown in blue) and the resulting hair that is traced (shown in black).

3.3.1 Helical Waviness Offset

Instead of directly encoding the waviness of a hair in its helicoid parameterization, a helical waviness offset can be used. This waviness offset is intuitive (controlled by one principal parameter, the frequency), efficient to compute (it reuses the precomputed Frenet frame moving along the curve), and couples well with the generalized helicoid tracing algorithm. It preserves

Algorithm 1 Generalized helicoid-based fiber tracing

```
1: function trace (float $stepSize$, float $stepCount$, Point $\mathbf{p}_0$)
2: List<Point> $points \leftarrow \{\}$
3: Point $\mathbf{p} \leftarrow \mathbf{p}_0$
4: for $step = 1$ to $stepCount$ do
5:     $\theta(\mathbf{p}) \leftarrow \arctan\left(\frac{k_T p_x + k_N p_y}{k_N p_x - k_T p_y}\right) + k_B p_z$
6:     $\phi(\mathbf{p}) \leftarrow \alpha\theta(\mathbf{p})$
7:     $\mathbf{E}_T \leftarrow (\cos\phi\cos\theta, \cos\phi\sin\theta, \sin\phi)$
8:     $\mathbf{p} \leftarrow \mathbf{p} + stepSize \cdot \mathbf{E}_T$
9:     $points \leftarrow points \cup \mathbf{p}$
10: end for
11: return $points$
12: end function
```

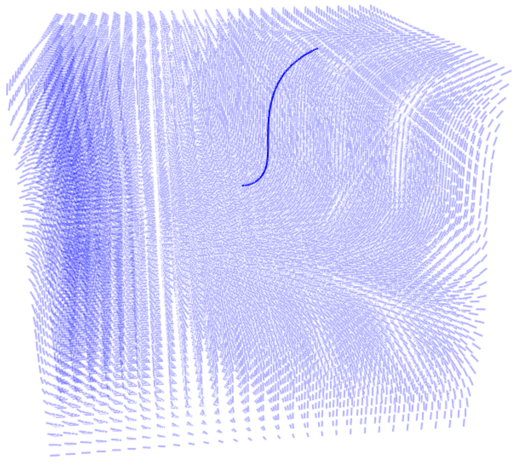
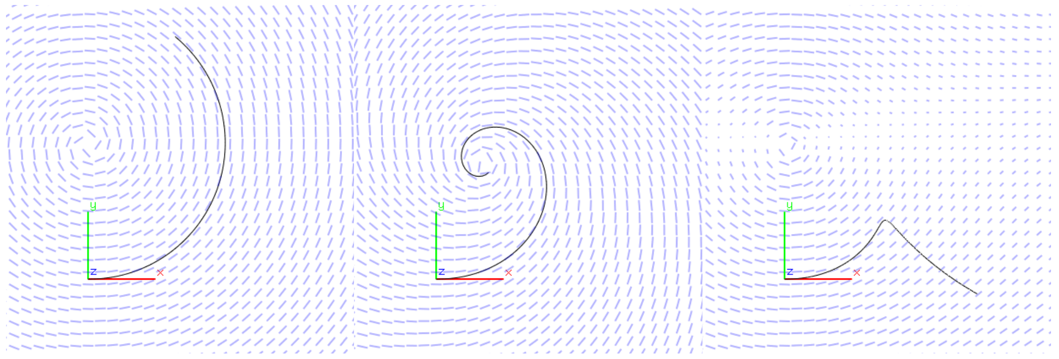


Figure 3–3: Tracing a path in the 3D field defined by curvature parameters $k_T = 1, k_N = 0.05, k_B = 0.5, \alpha = 1$.

the large scale geometry of the target hair, while adding local helical perturbations. This allows one to completely decouple the notion of waviness from the generalized helicoid parameterization, at the benefit of concentrating its effects on the overall shape and neighborhood of a generated hair, not on its local behavior.

A helix of radius a and pitch (frequency) $2\pi b$ is described as a curve in 3-dimensional space parameterized by its arclength t in Cartesian coordinates



(a) $\mathbf{p} = (0.5, 0, 0, 0)$ (b) $\mathbf{p} = (0.5, -0.2, 0, 0)$ (c) $\mathbf{p} = (0.5, 0, 0.5, 1)$

Figure 3–4: The effect of varying the curvature parameters of a generalized helicoid, of which a slice in the xy plane is shown in blue. The parameter vector is defined as $\mathbf{k} = (k_T, k_N, k_B, \alpha)$. The α parameter allows the helicoid to grow out of plane, and must be non-zero to see the effects of the binormal curvature k_B. Figures 3–4a, 3–4b and 3–4c show the effects of varying the tangential, normal, and binormal components of the parameter vector, respectively. The hairs, shown in blue, are traced from an initial direction that is tangential to the x axis, shown in red.

by

$$x(t) = a\cos(t) \tag{3.32}$$

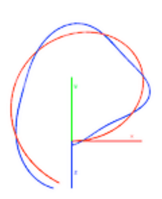
$$y(t) = a\sin(t) \tag{3.33}$$

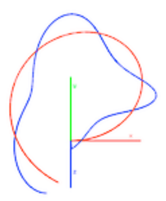
$$z(t) = bt. \tag{3.34}$$

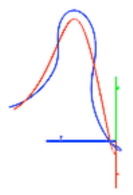
In the case of a generalized helicoid, the Cartesian coordinates are replaced by the Frenet coordinates, moving along with the curve. The waviness offset is mapped in the normal/binormal plane spanned by $\mathbf{E}_B, \mathbf{E}_N$. The offset $\mathbf{W} = a\cos(t_0 + t/b)\mathbf{E}_B + a\sin(t_0 + t/b)\mathbf{E}_N$ is applied to the tangential component of the representation in spherical coordinates of the generalized helicoid:

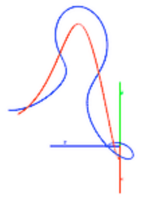
$$\tilde{\mathbf{E}_T} = \mathbf{E}_T + \mathbf{W} \tag{3.35}$$

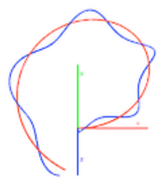
where a controls the radius of the waviness offset, b its frequency, and t_0 determines the phase of this helix. a is typically set to a small constant value, since it is the frequency of the waviness offset that mostly drives its qualitative impression. Noise can be added to the initial phase t_0 in order to break the similarity between hair neighbors. Note that the waviness offset is applied only to the perceived geometry of the generalized helicoid, but its trace remains the same. This way, the original generalized helicoid actually lives at the center line of the helix responsible for its offset, and can be used for later purposes (e.g. interpolation). Figure 3–5 shows an original hair inside its waviness offset for radii and frequencies of different magnitudes.

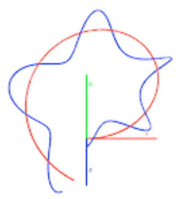

(a) $a = 0.1, b = 3$

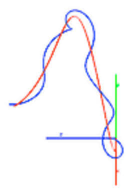

(b) $a = 0.2, b = 3$

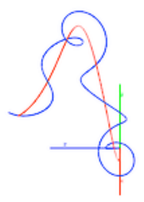

(c) $a = 0.1, b = 3$

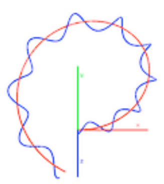

(d) $a = 0.2, b = 3$

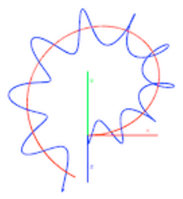

(e) $a = 0.1, b = 5$

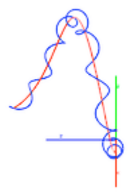

(f) $a = 0.2, b = 5$


(g) $a = 0.1, b = 5$


(h) $a = 0.2, b = 5$


(i) $a = 0.1, b = 10$


(j) $a = 0.2, b = 10$


(k) $a = 0.1, b = 10$

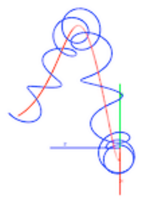

(l) $a = 0.2, b = 10$

Figure 3–5: Waviness offset (in blue) for helical radii a and frequencies b. The helicoidal parameter of the hair is $\mathbf{k} = (1.1, -0.01, 0.2, 0.1)$. The original hair is shown in red. The first two columns show a front view, and the last two a side view.

3.3.2 Smoothed Stochastic Perturbations

Stochastic perturbations can be directly added to the generalized helicoid tracing algorithm. If Gaussian random noise of mean 0, standard deviation 1.0 and magnitude r is used, the following orientation update equation is obtained, replacing equation 3.25 for θ:

$$\theta \to \theta + rN(0,1). \tag{3.36}$$

This perturbation is not cumulative in the sense that it is applied independently at each step of the tracing algorithm. Smoothing can be applied on the resulting geometry of the hair in order to improve its visual appearance and filter the higher frequencies in the resulting noise. Smoothing by weighted averaging is used herein, because of its simplicity and sufficient efficiency. The smoothing is done by traversing the polyline describing the helicoid and applying the following update rule at each point $\mathbf{p}_i$ of the polyline:

$$\mathbf{p}_{i+1} \to \mathbf{p}_{i+1} + \alpha \left((\mathbf{p}_{i+1} - \mathbf{p}_i) - \frac{1}{2} (\mathbf{p}_{i+2} - \mathbf{p}_i) \right) \tag{3.37}$$

$$= (1+\alpha)\,\mathbf{p}_{i+1} - \frac{\alpha}{2} (\mathbf{p}_i + \mathbf{p}_{i+2}) \tag{3.38}$$

where $0 \le \alpha \le 1$ determines how much smoothing is applied at each iteration. Figure 3–6 shows the direction $\mathbf{n}$ towards which the point $\mathbf{p}_{i+1}$ is pulled with a magnitude α. The effect of the smoothing is illustrated in Figure 3–7 for different numbers of smoothing cycles, and different values of α.

3.4 Properties of the Explicit Form

This section derives some properties relating to the symmetries and singularities in the trace of a generalized helicoi. Recall that a generalized helicoid is defined by the orientation angle

$$\theta(x,y,z) = \arctan\left(\frac{k_T x + k_N y}{1 + k_N x - k_T y} \right) + k_B z. \tag{3.39}$$

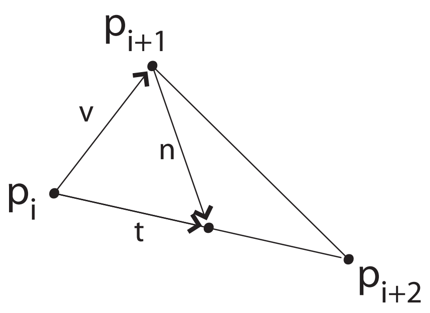


Figure 3–6: Smoothing by weighting averaging of a point $\mathbf{p}_{i+1}$ on its midline t

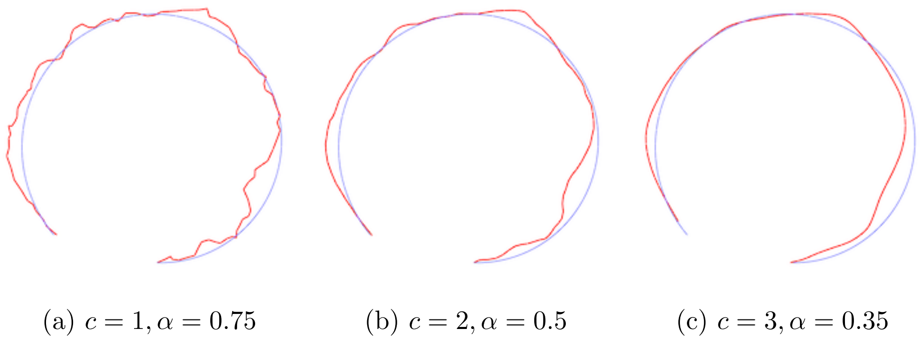


Figure 3–7: Effect of varying the number of smoothing cycles c and the magnitude of α. The original generalized helicoid is shown in blue and the noisy and smoothed version in red. The magnitude of the noise is $r = 0.03$.

3.4.1 Restrictions on ϕ and k_B

In the helicoid tracing Algorithm 1, the assignment of $\phi = \alpha\theta$ needs to be dealt with carefully. Since the orientation angle θ varies monotonically, it eventually completes full 2π cycles in the $x - y$ plane. The assignment of $\phi = \alpha\theta$ should thus be constrained in order to prevent cusps from forming in the helicoidal trace where θ suddenly goes from 2π to 0, and therefore ϕ goes from $2\alpha\pi$ to 0. It is therefore necessary to count the number of complete turns that θ cycles through, and the direction (positive or negative) in which it is turning. This directly leads to a restriction on k_B, as it should not be greater than 2π, otherwise θ starts skipping turns and the trace will become inconsistent.

3.4.2 Reflection by k_T

Flipping the sign of k_T merely results in a reflection of the trace about a rotated x-z plane. The reflection occurs near the origin, and immediately determines the quadrant into which the trace will be located. Let the current location $\mathbf{p} = (x, y, z)$ in the tracing algorithm be near the origin, that is $\mathbf{p} = (\epsilon, \epsilon, \epsilon)$ where $\epsilon \to 0$, from any direction. Then the orientation angle can be rewritten as follows:

$$\lim_{x,y,z\to\epsilon} \theta(x, y, z) = \lim_{x,y,z\to\epsilon} \arctan\left(\frac{k_T x + k_N y}{1 + k_N x - k_T y}\right) + k_B z \tag{3.40}$$

$$= \lim_{x,y,z\to\epsilon} \arctan\left(k_T x + k_N y\right) + k_B z \tag{3.41}$$

$$= k_T x + k_N y + k_B z \tag{3.42}$$

where in the last step the taylor series $\arctan\phi = \phi - \frac{1}{3}\phi^3 + \frac{1}{5}\phi^5 + \cdots$ was used and the $O(\phi^3)$ terms were omitted for small ϕ. The tangential component of the trace is given by:

$$\mathbf{E}_T = (x, y, z) \tag{3.43}$$

$$= \left(\cos\alpha\theta\cos\theta, \cos\alpha\theta\sin\theta, \sin\alpha\theta\right). \tag{3.44}$$

The planar case $\alpha \to 0$ is first considered. If the orientation angle flips sign, that is $\theta \to -\theta$, and is substituted in equation 3.44, the following is obtained:

$$\mathbf{E}_T = \left(\cos\theta, -\sin\theta, 0\right). \tag{3.45}$$

Thus flipping the orientation angle results in a reflection of the trace about the x-axis. This behavior can be obtained by flipping the sign of k_T in equation 3.39. Since arctan is an odd function,

$$\arctan\left(\phi\right) = -\arctan\left(-\phi\right) \tag{3.46}$$

using equation 3.42 yields

$$\lim_{x,y,z\to\epsilon} -\theta(x,y,z) = (-k_T)x + (-y)k_N \tag{3.47}$$

which shows that flipping the sign of k_T changes that of both θ and y, resulting in the aforementioned reflection. For nonzero α, $\cos\alpha\theta \neq 1$, $z = \sin\alpha\theta$ is still small since θ is small, and the result is a reflection that is now done in a rotated version of the x-z plane.

3.4.3 Singularities

Equation 3.39 is not defined if both the numerator and the denominator in the argument of the arctan are 0. That is, if the following two equations are satisfied:

$$k_T x + k_N y = 0 \tag{3.48}$$

$$1 + k_N x - k_T y = 0. \tag{3.49}$$

This happens when

$$x_c = -\frac{k_N}{1+\left(\frac{k_T}{k_N}\right)^2} \tag{3.50}$$

$$y_c = \frac{k_T}{1+\left(\frac{k_T}{k_N}\right)^2}. \tag{3.51}$$

The critical point (x_c, y_c) is responsible for the helicoidal behavior of the trace. As shown in Figure 3–8, the closer the singularity is to the origin, the more compact and coiled the trace becomes.

3.4.4 Rotational Ambiguity

This section showcases three simple cases to reveal ambiguities in the way hairs are described by generalized helicoids when aligned in different local frames. Let two generalized helicoids face each other. Their helicoidal parameters are assumed to all be zero, except for tangential curvature, which takes either the value k_T or $-k_T$. The helicoid on the right is either aligned with the left one ($\theta = 0$), or rotated about the normal by $\theta = \pi$ radians. The three cases that are considered are the following:

1. $k_T(a) = -k_T(b), \theta = 0$. should produce opposing generalized helicoids, one rotated by π radians from the other.
2. $k_T(a) = k_T(b), \theta = \pi$. should produce opposing generalized helicoids, one rotated by π radians from the other.
3. $k_T(a) = -k_T(b), \theta = \pi$. should produce dentical and aligned helicoids.

In cases 1 and 2, the hairs look exactly the same. In case 3, they look different. In all three cases, the generalized helicoids should have the exact same geometry (as per the argument mentioned in Section 3.4.2). These ambiguities are dealt with automatically by imposing k_T to be always positive and using normal rotations instead when necessary.

3.5 Wisps

By offsetting the origin $\mathbf{p}_0$ in Algorithm 1, a dense distribution of locally parallel hair strands can be generated. Looking at Figure 3–4, this amounts to tracing additional streamlines surrounding a master strand. The offset has the same qualitative effect as interpolating between the master hair and other strands of higher curvature. However, the offsetting method is much simpler since it requires no additional information about the location and geometry of neighboring hairs. This feature is of particular interest for proponents of the hair wisp model. The computation of a wisp using the offsetting method is fast since it only involves tracing streamlines in the helicoidal field. Noise and random orientation perturbations can be added to further emphasize the fanning of the wisp.

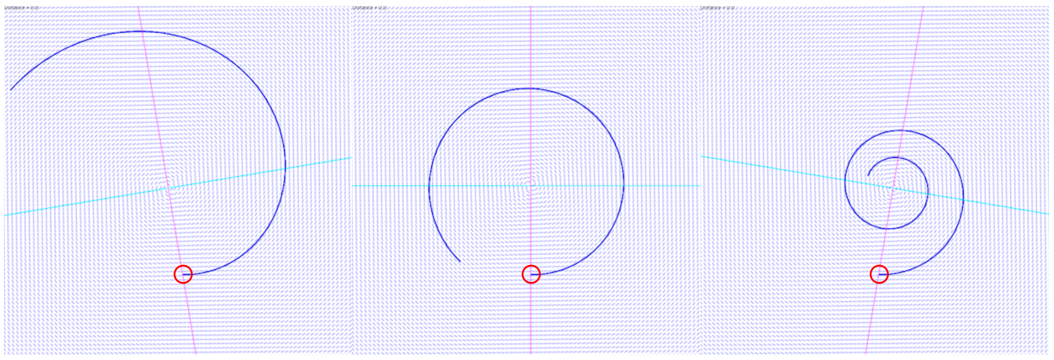
Figure 3–9 shows a wisp generated from extrapolated strands around a guide hair. The diameter of the sampling region around the guide hair is 0.01

and the offset scaling with respect to the displacement vector from a sample location to the guide hair is 0.1. 256 hairs are extrapolated from the guide hair.

3.6 Hairs as Piecewise Helicoids

A more powerful and flexible way of representing hairs with the generalized helicoid model is to make use of a composite parameterization, referred to as piecewise-helicoids or *p-helicoids*. P-helicoids are simply the union of multiple generalized helicoids, aligned together by their Frenet frame using equation 3.31. This characterization of hair strands enforces C^0 continuity (pieces are always connected), and C^2 continuity (connected pieces share the same tangent and second derivative at their connection point). C^2 continuity is obtained from the generalized helicoid parameterization. The transformation matrix that maps a point at the root of a p-helicoid to its tip is found by traversing its helicoidal pieces, and carrying over the premultiplication of the Frenet frame at the tip of each piece.

Piecewise helicoids can be used to generate the geometry of more complex hairstyles. As described in sections 3.3.2 and 3.3.1, waviness and noise offsets can be used to produce more realistic strands. The examples shown in this section are represented in their wisp form, to give a more appreciable result. For instance, Figure 3–10 shows a long hair composed of 3 helicoidal pieces with varying curvature parameters. Figure 3–11 shows a “messy” hair using 5 helicoidal pieces and a curly hair composed of 3 helicoidal pieces of similar curvature. All wisps are formed of 256 extrapolated hairs.



(a) $k_T = 1.2, k_N = 0$ (b) $k_T = 1.2, k_N = 0.2$ (c) $k_T = 1.2, k_N = -0.2$

Figure 3–8: Moving singularity for different k_T, k_N and $k_B = \alpha = 0$. The trace is shown in blue. The singularity is located at the intersection of the two lines defined in equation 3.51, shown in cyan and pink. The origin is shown as a red circle.

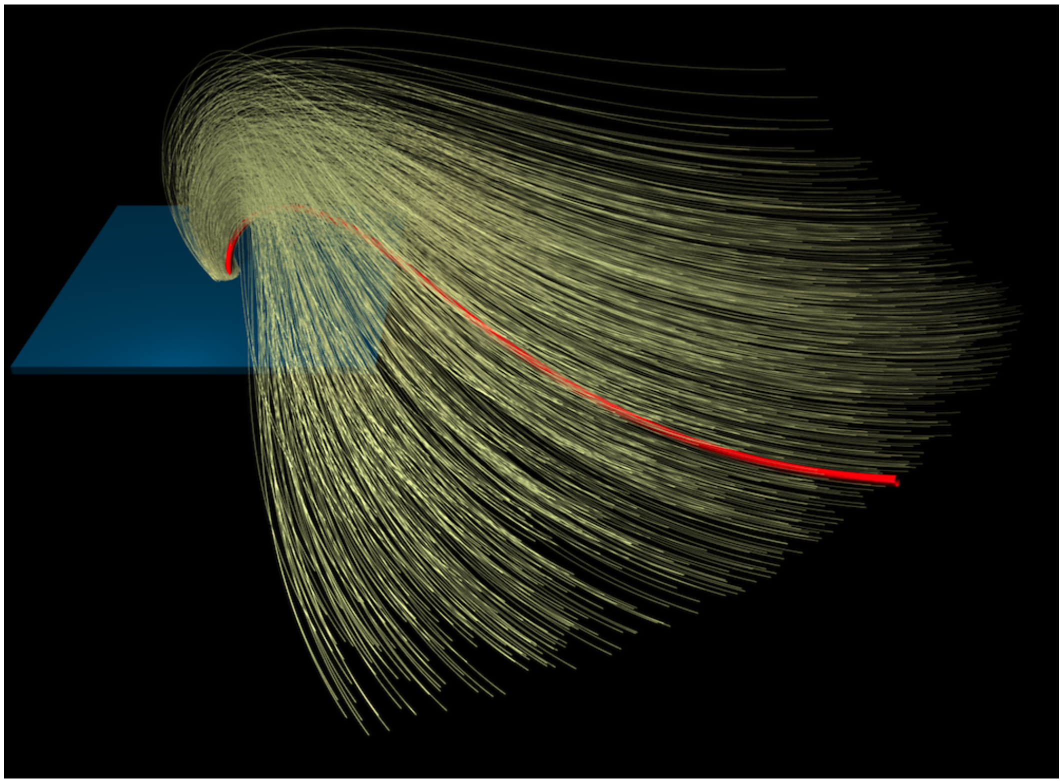
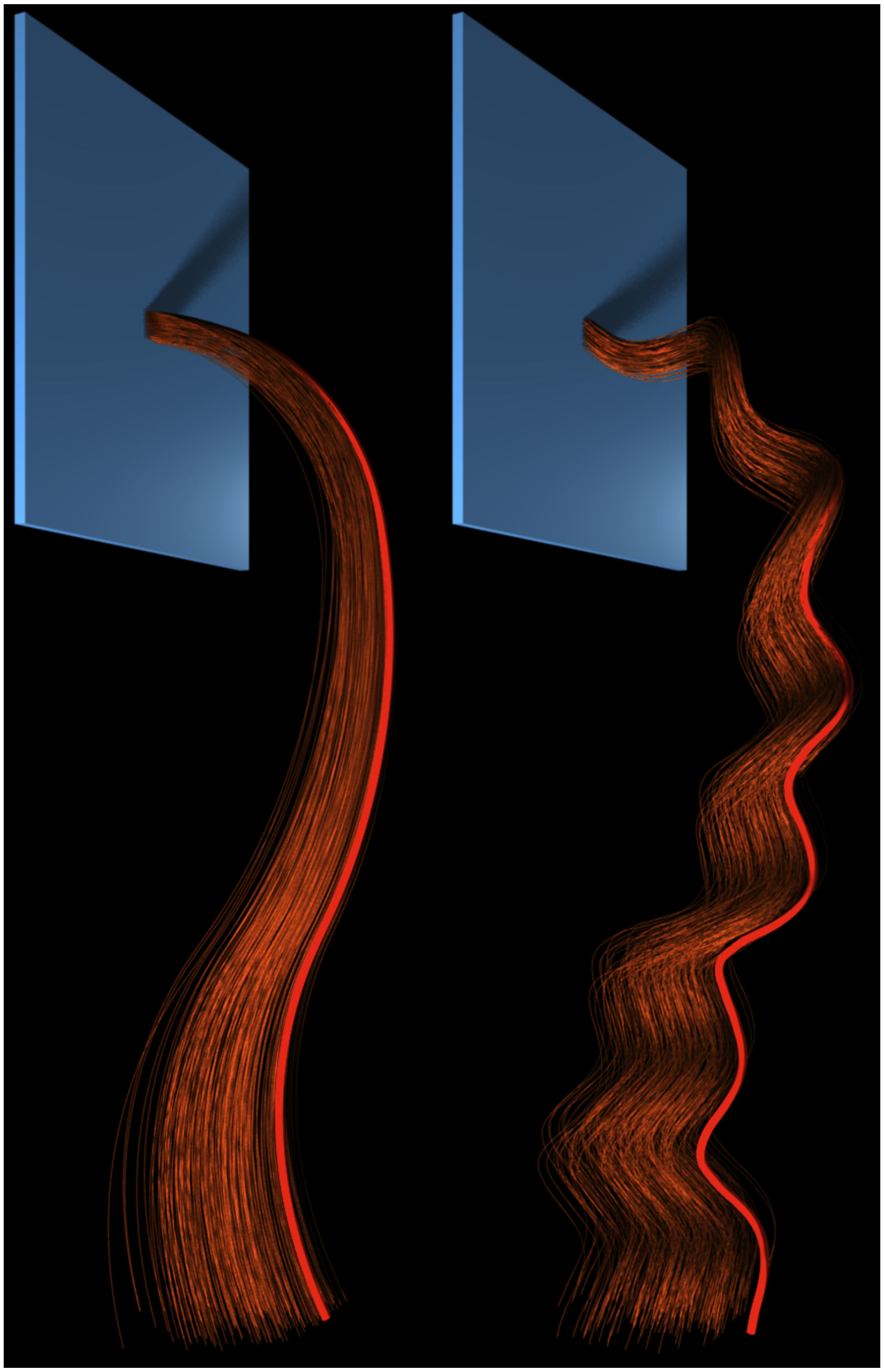
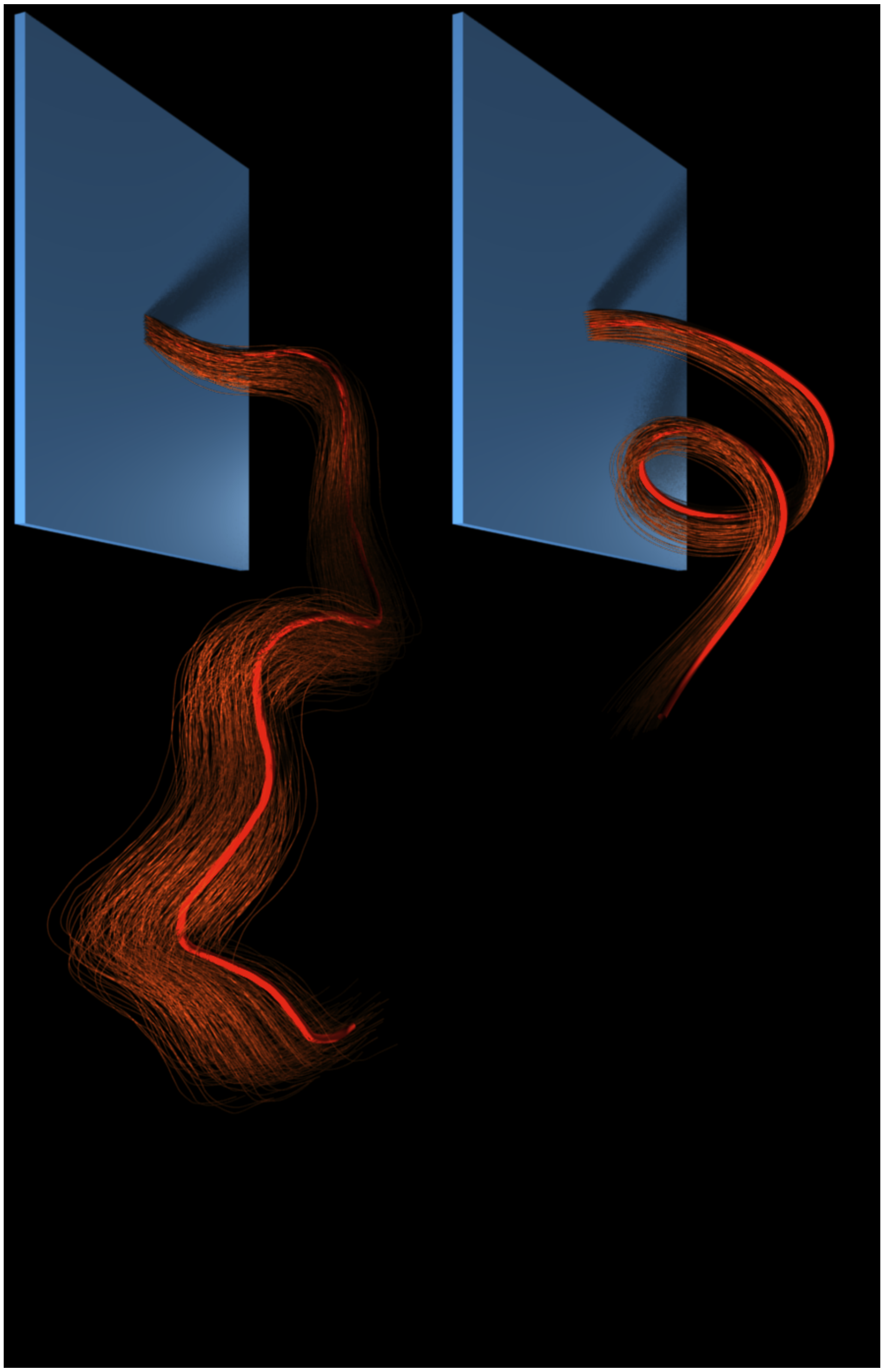


Figure 3–9: A wisp generated by offsetting the origin in the streamline trace. The guide strand of the helicoid is shown in red and the extrapolated hairs are shown in gold.



(a) A long hair. (b) A long hair with a waviness offset.

Figure 3–10: A long hair (left) combined with a waviness offset (right), shown in red. The associated extrapolated wisps are shown in brown.



(a) A “messy” hair.

(b) A curly hair.

Figure 3–11: A “messy” hair (in red, on the left) and a curly hair (in red, on the right) and their extrapolated wisps (in brown).

CHAPTER 4
Follicle Sampling

To interpolate across guide hairs, root locations (or follicles, if the biological terminology is used) have to be determined. The method used is an adapted implementation of Turk and O'brien [81] and Witkin and Heckbert [88] for using particles to sample implicit surfaces. Although the particle methods that are proposed closely follow the ideas introduced in Witkin and Heckbert [88], the use of contour particles for outlining the shape of the hair surface is novel in the hair modeling literature. Section 4.1 first presents the implicit surface interpolation method. Section 4.2 then describes how particles can be spread and constrained on the implicit surface. Finally, Section 4.2.9 shows the results of the sampling method applied on different surfaces.

4.1 Implicit Surface Interpolation

The method by Turk and O'brien [81] is used to generate an interpolated representation of the growth surface for sampling hairs. More specifically, the interpolated surface is an *implicit surface*. An implicit surface is defined by a real-valued function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$. The locus of the zero-crossings $\{\mathbf{x} | f(\mathbf{x}) - c = 0\}$ represents the level sets of the implicit surface, for varying c. Figure 4–1 shows different level sets of a gaussian-like function. In 3D, these level sets would actually correspond to different surfaces, of which only one (for a fixed c, determined through some constraints with respect to the original surface) is considered.

In the context of surface interpolation, an implicit function is sought such that at least one of its level sets passes through (or within an epsilon-radius

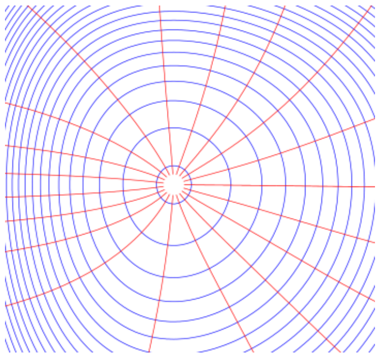


Figure 4–1: Visualization of an implicit function. The blue curves are the the level sets $f(\mathbf{x}) - c = 0$ for different values of c. The red curves follow the direction of the gradient of f.

of) a set of constraints (control points). The nature of the implicit function determines how the surface varies in the vicinity of the control points.

In Turk and O'brien [81], the energy function

$$E(f) = \int_{\Omega} f_{xx}^2(\mathbf{x}) + 2f_{xy}^2(\mathbf{x}) + f_{yy}^2(\mathbf{x}) d\mathbf{x} \tag{4.1}$$

is used to evaluate the smoothness of the implicit function f over the region of interest Ω. This energy measures how much the surface changes in the region, and is often used in the context of *regularization*, in the Computer Vision literature. Points of high curvature contribute to a large value of E, while smooth regions tend to minimize it. Different applications yield different smoothness criterion regarding E. If the implicit function satisfies the set of constraints implied by the interpolation problem, and minimizes the energy E defined in equation 4.1, then it is denoted a *thin-plate* solution.

The thin-plate solution gets its name from an analogy with a thin sheet of metal, laid flat and bent such that it grazes the ends of a collection of vertical poles distributed following the constraints of the interpolation problem. A metal plate naturally resists bending such that it smoothly changes in shape in between the poles, which is exactly the kind of neighborhing behavior that minimizes the energy function E defined above. [81]

In the 3D interpolation case, the thin-plate radial basis function (RBF) $\phi(\mathbf{x}) = |\mathbf{x}|^3$, for which the positional constraints are automatically satisfied and the smoothness is enforced, is typically used. It is the thin-plate function defined by Turk and O'brien [81], and the one adopted in this thesis. Constraints $\mathbf{c}_j$ are selected by the user by picking vertices on a 3D model. Interior and exterior constraints are added automatically by adding an offset in the direction of the normal at the vertex corresponding to a constraint. The implicit function that generates the interpolation surface is written as

$$f(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi(\mathbf{x} - \mathbf{c}_j) + P(\mathbf{x}) \tag{4.2}$$

where $\mathbf{w}_j$ are the weighting factors to be found for each RBF, $\mathbf{c}_j$ are the location of the constraints, and $P(x) = p_0 + p_1 x + p_2 y + p_3 z$ is a polynomial of degree one that describes the constant behavior of f. The gradient of this function, pointing in the normal direction to the implicit surface, is given by:

$$\nabla f(\mathbf{x}) = n(p_1, p_2, p_3) + 3 \sum_{j=1}^{n} w_j \left|\left|\mathbf{x} - \mathbf{c}_j\right|\right|^2 (\mathbf{x} - \mathbf{c}_j) \, . \tag{4.3}$$

and the constraint equation that solves for the weights w_j's is obtained by

$$h_i = \sum_{j=1}^{k} w_j \phi(\mathbf{c}_i - \mathbf{c}_j) + P(\mathbf{c}_i). \tag{4.4}$$

Now let $\mathbf{c}_i = (c_i^x, c_i^y, c_i^z)$ and $\phi_{ij} \equiv \phi(\mathbf{c}_i - \mathbf{c}_j)$. Equation 4.4 can then be expanded as the linear system

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1k} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2k} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \cdots & \phi_{kk} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \cdots & c_k^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \cdots & c_k^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \cdots & c_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.5}$$

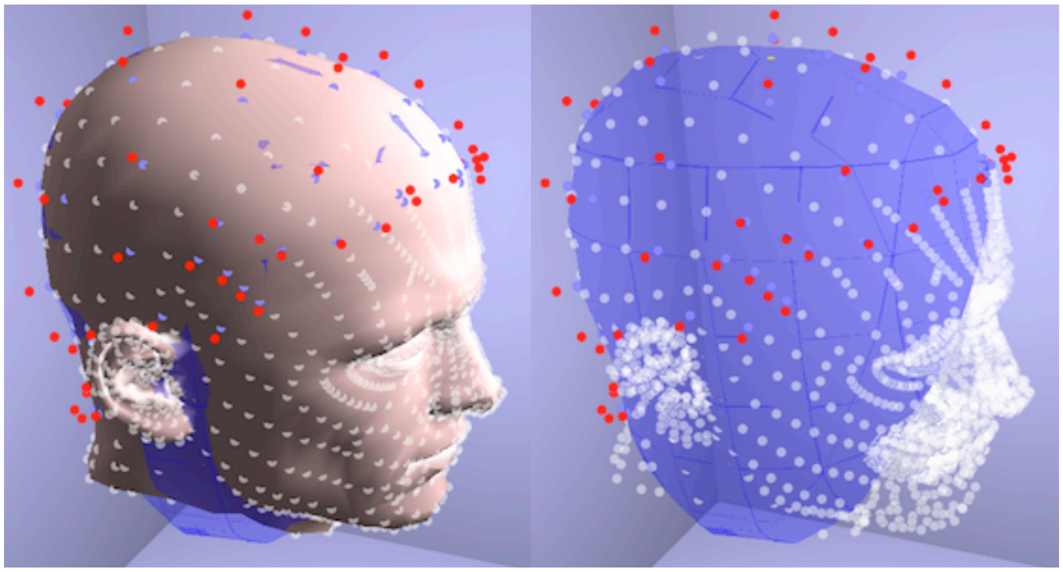
which can be rewritten in compact form:

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{G} \\ \mathbf{G}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{h} \\ 0 \end{bmatrix}. \tag{4.6}$$

Equation 4.6 can be solved by computing the pseudoinverse of the constraint matrix using its singular value decomposition, and zeroing singular values that fall below a certain threshold (thus smoothing the implicit representation). Other least-squares methods could be used by exploiting the symmetrical structure of the constraint matrix, such as QR decomposition. Figure 4–2 shows the implicit constraints and the resulting interpolated implicit surface. The visualization of the interpolation surface can be done using the methods described in Appendix B.

4.2 Particle sampling

By having an implicit representation of the growth surface, methods that have been developed in particle-based implicit surface sampling can be directly incorporated. By their adaptive and dynamic nature, particle-based sampling



(a) *Model* (b) *Implicitfit*

Figure 4–2: Implicit fit of a 3D model. The vertices are shown in white. Interior constraints lie inside the surface, surface constraints are shown in blue, and exterior constraints in red.

approaches typically provide (and require) user control, and offer great flexibility. The particle-based implicit surface sampling method by Witkin and Heckbert [88] was adapted, in which the sampling is done with a particle system driven by repulsion forces and implicit constraints. This method efficiently distributes particles across the scalp in a near-regular manner, by allowing particles to fission (divide in two), and die.

The system is updated through a number of computational steps.

1. Initialization. Force accumulators are zeroed and the system is initialized.
2. Connectivity. Particle neighbors are assigned through a particle-mesh (PM) scheme.
3. Repulsion. A pairwise repulsion energy is computed amongst particle neighbors.

4. Constraints. Hard constraints are applied on the velocity of the particles in order to keep them in contact with the implicit surface.
5. External forces. Drag, user interaction forces, and other external forces are added to the force accumulators.
6. Force integration. Particle positions and velocities are updated using the force accumulators.
7. Fission. Particles satisfying certain requirements are allowed to fission.
8. Death. Particles satisfying certain requirements are allowed to die.

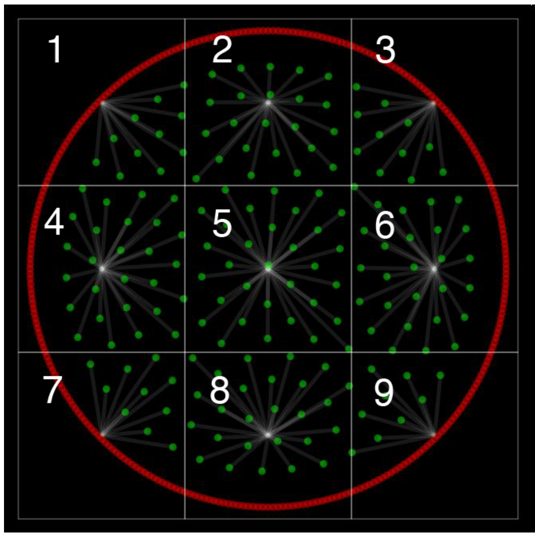
4.2.1 PM Connectivity

Particle neighbors are set using a particle-mesh scheme. Particle-mesh methods dramatically reduce the computational time for updating the particle system, especially when it is very large. In a simple particle-particle approach, for n particles, $O(n^2)$ distance comparisons are required, while using particle-mesh approach with a grid composed of m cells, $O(mn)$ comparisons are required.

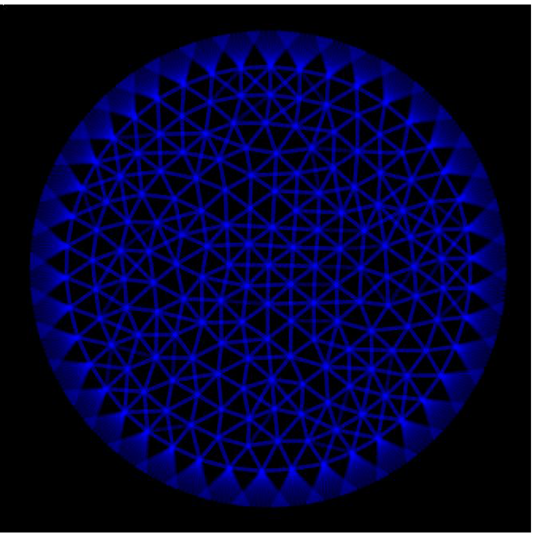
First, the volume of the particle system is divided into grid cells. For a fixed volume, the cell size determines their number, and has to be adjusted carefully. Too many cells will result in many cells being empty, and more comparisons will be made than when using a simple particle-particle based approach. Not enough cells will result in a poor improvement with respect to a particle-particle implementation.

Each particle is then assigned to the cell of which the center is closest to its position. The connectivity is assigned using the following specifications:

1. Particles that do not connect through their neighboring radius cannot be neighbors.
2. Cells within one-cell radius of each other are set as neighbors.



(a) Particle-mesh grid. The white transparent lines connect the particles to their respective cell centers.



(b) Connectivity graph.

Figure 4–3: Example of the particle-mesh connectivity for 500 particles in a volume composed of 9 cells.

3. Particles that share the same cell or are within neighboring cells are set as neighbors if rule 1 is also satisfied.

Figure 4–3 shows the particle mesh for a system composed of 500 particles, and a volume divided into 9 cells. Following our connectivity rules, particles in the cell no 5 are neighbors to all other particles, since it is within a 1-cell radius to all other cells. In contrast, particles in cell no 1 are only neighbors to other particles in cell no 2, 4, 5.

4.2.2 Repulsion Energy

Neighboring particles acquire a non-symmetrical interaction energy based on their pairwise distances. The energy of a particle i interacting with a particle j at a distance $\mathbf{r}_{ij}$ is defined as

$$E_{ij} = \alpha \exp\left(-\frac{|\mathbf{r}_{ij}|^2}{2\sigma_i^2}\right) \tag{4.7}$$

where α is a global *repulsion amplitude* parameter, and σ the local *repulsion radius.*

The total interaction energy of a particle i with n neighbors is then defined as

$$E_i = \sum_j^n (E_{ij} + E_{ji}) \tag{4.8}$$

$$= \alpha \sum_j^n \left[\exp\left(-\frac{|\mathbf{r}_{ij}|^2}{2\sigma_i^2}\right) + \exp\left(-\frac{|\mathbf{r}_{ij}|^2}{2\sigma_j^2}\right)\right] \tag{4.9}$$

The particle system is required to eventually converge to a more or less static configuration. This is done by enforcing the energy of each particle to reach a local minimum, using gradient descend in space at each system update:

$$\mathbf{P}_i = -\sigma_i^2 E_i \tag{4.10}$$

$$= \sigma_i^2 \sum_j^n \left(\frac{\mathbf{r}_{ij}}{\sigma_i^2} E_{ij} - \frac{\mathbf{r}_{ij}}{\sigma_j^2} E_{ji} \right) \tag{4.11}$$

where $\mathbf{P}_i$ is the change in velocity for this system update.

The repulsion radius follows an adaptive scheme based on local energy measurements. For a particle i, the amount of energy that is determined by its own repulsion radius is

$$D^i = \sum_j^n E_{ij} \tag{4.12}$$

A linear feedback equation is used to keep this energy near the desired value $\hat{E}$:

$$\dot{D}_i = -\rho \left(D_i - \hat{E} \right) \tag{4.13}$$

where ρ is the feedback constant.

4.2.3 Constraint Forces

Although the velocity update defined in Section 4.2.2 will ensure that the global repulsion energy will be minimized, it does not impose any constraints on the actual location of the particles in space. Particles are constrained by projecting equation 4.11 on the implicit surface F:

$$\dot{\mathbf{p}}_i = \mathbf{P}_i - \frac{\nabla F_i \cdot \mathbf{P}_i}{\nabla F_i \cdot \nabla F_i} \nabla F_i \tag{4.14}$$

where $\nabla F_i = \nabla F(p_i)$ is the gradient of the implicit function describing the implicit surface evaluated at the position of particle i, and $\dot{\mathbf{p}}_i$ is the update for the velocity of particle i.

4.2.4 External Forces

In the context of a particle system, drag refers to forces that oppose the relative motion of the particles as they move in space. Drag forces suck energy out of the system, making sure that it stays numerically stable through its integration. Linear drag $\mathbf{F}_d$ was used for a particle i moving a velocity $\dot{\mathbf{p}}_i$:

$$\mathbf{F}_d = -b\dot{\mathbf{p}}_i \tag{4.15}$$

where b is the *drag coefficient*. For small time integration steps Δt, adding drag forces to the system might not be required. However, for larger time steps, the drag coefficient needs to be adjusted carefully. A large value of b will result in a very stable integration, but the particle system will be nearly static and the time to reach equilibrium will be increased. On the other hand, a small drag coefficient might make the system converge faster to equilibrium, but numerical errors might accumulate and result in a catastrophic scenario where particles become so unstable that they take enormous jumps in space (the particle system is said to "blow up").

4.2.5 Force Integration

Many particle system integration methods exist, and their usefulness depends on the requirements of the problem at hand. In this context, all the constraints are explicitly handled in the construction of the velocity update $\mathbf{P}_i$.

Moreover, numerical errors are tolerated, as long as the system behaves relatively smoothly (i.e. does not blow up) for small time steps. Euler's (explicit, first order) method is a fast and sufficient method of updating the system:

$$\mathbf{p}_i^{t+\Delta t} = \mathbf{p}_i^t + \Delta t \dot{\mathbf{p}}_i \tag{4.16}$$

where Δt is the *time step* that discretizes the temporal evolution of the particle system and $\mathbf{p}_i^t$ is the position of particle i at time t.

4.2.6 Particle Birth and Death

Particle fissioning and death ensure a uniform sampling of the implicit surface. Particles with a repulsion radius that is too large become isolated within a predetermined region of space, and should fission (split in two). On the other hand, particles that lie in an overcrowded region should die. The conditions for deciding which option applies for a particle i are based on particle velocity $\dot{\mathbf{p}}_i$ and repulsion radius σ_i.

Fission. The requirements for fissioning a particle i are the following:

1. The particle is near equilibrium: $\dot{\mathbf{p}}_i < \gamma\sigma_i$.
2. The particle's repulsion radius is large: $\sigma_i > \sigma_{max}$.
3. The particle is adequately energized and its radius is above the desired radius: $D_i > \nu\hat{E}$ and $\sigma_i > \hat{\sigma}$.

The particle is fissioned if the condition 1) is met, with either 2) or 3) also holding. The two particles emerging from the fission of a particle i are given repulsion radii of $\sigma_i/\sqrt{2}$, and a velocity in a random direction with a magnitude proportional to σ_i. Their initial location is set as a short step in the direction of their velocity.

Death. The requirements for killing a particle i are the following:

1. The particle is near equilibrium: $\dot{\mathbf{p}}_i < \gamma\sigma_i$.
2. The particle's repulsion radius is too small: $\sigma_i < \delta\hat{\sigma}$.

3. A biased randomized test based on a uniform random number R between 0 and 1 succeeds: $R > \sigma_i / (\delta\hat{\sigma})$.

The particle is killed if all three conditions are met. The third stochastic criterion prevents the mass killing of all particles in an overcrowded region.

4.2.7 Parameters

The particle sampling method requires many parameters to be finely hand-tuned and Witkin and Heckbert [88] give recommended values for most of them. These parameter recommendations were subsequently hand-tuned. The value and meaning of each parameter are listed in Table 4–1.

Table 4–1: Parameter values for sampling implicit surfaces using particles.

Parameter	**Value**	**Purpose**
Δt	0.03	time step
ϕ, ρ	15	feedback coefficients to keep particles from drifting off the surface, and keep particles energized, respectively
α	6	repulsion amplitude
$\hat{E}$	0.8α	desired particle energy
β	10	prevents division-by-zero in the adaptive repulsion scheme
$\hat{\sigma}$	?	desired repulsion radius
$\sigma^{\max}$	?	maximum repulsion radius
γ	$4\sigma_i$	equilibrium speed
ν	$0.2\hat{E}$	controls particle fission
δ	0.2σ	controls particle death

4.2.8 Boundary

Although the particle sampling method in Witkin and Heckbert [88] performs well for closed implicit surfaces, additional work must be done in order to extend it to open surfaces, such as that of a scalp. The novel concept of *contour particles* was designed as a way of restraining the evolution of the particle system within the boundary of a closed surface. In practice, contour particles have a large and fixed repulsion radius, and a high repulsion magnitude. They define a boundary that prevents other floating particles from drifting

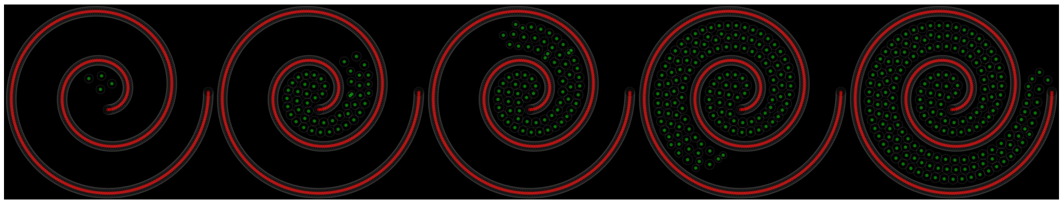
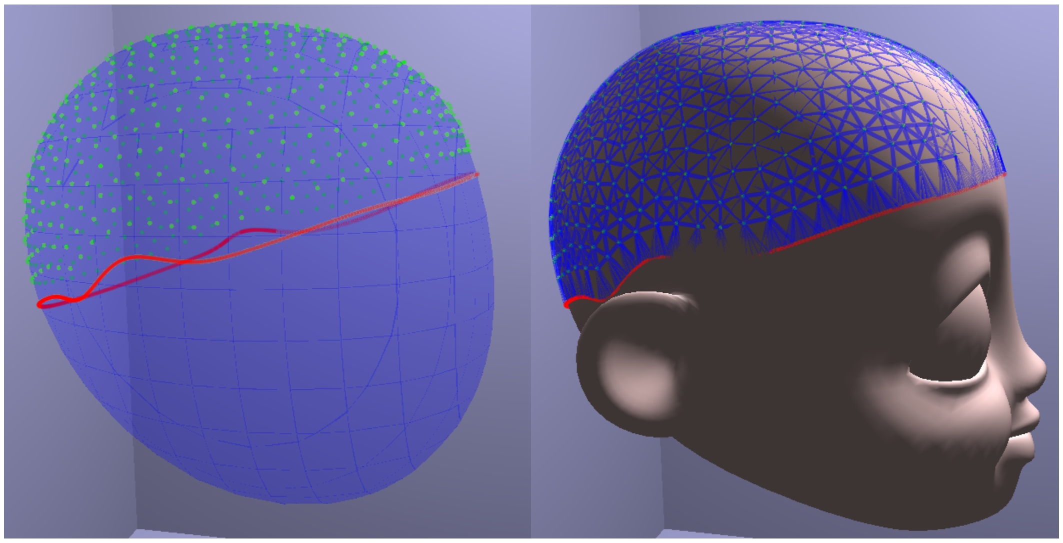

(a) t_0 (b) t_1 (c) t_2 (d) t_3 (e) t_4

Figure 4–4: Evolution of the sampling for a 2D spiralling contour.

below the implied contour line. This contouring method, to the extent of our knowledge, has not been employed in the graphics literature before.

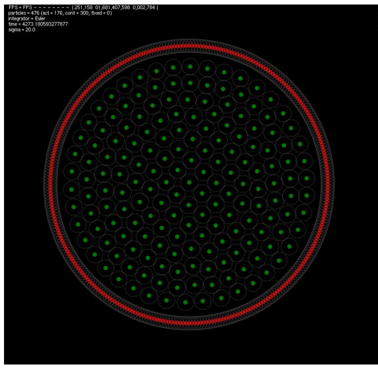
4.2.9 Sampling Results

Figure 4–4 shows the particle sampling method applied to a 2D spiraling contour. Starting from an initial set of parameters and a 4-particle seed, the particles are automatically fissioned/killed until they spread the interior of the contour uniformly. The spiral is opened at one end and particles are allowed to drift out. Figure 4–5 shows the sampling of 1000 particles on a 3D model delimited by contour particles. Figure 4–6 shows the sampling of a circle and the associated particle connectivity and system grid. Figure 4–7 shows the implicit constraints and the sampling of a scalp for 5000 interacting particles.

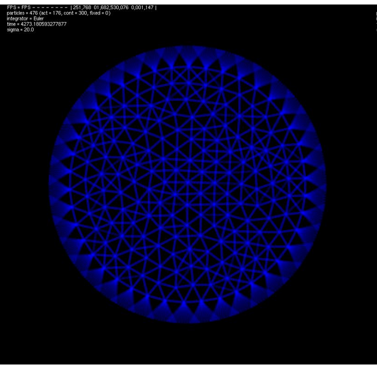


(a) Implicit surface view (b) Model view

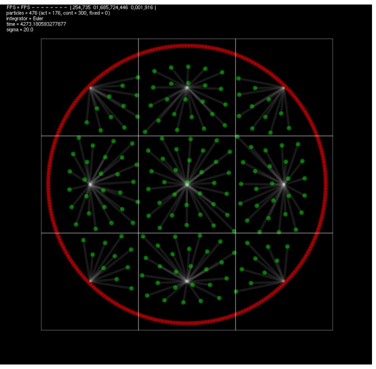
Figure 4–5: 1000 contour particles (in red) keep 1000 floating particles (in green) from drifting below the implied contour line.



```
FPS = FPS - - - - - - - - - | 251,158  01,681,407,598  0,002,784 |
particles = 476 (act = 176, cont = 300, fixed = 0)
integrator = Euler
time = 4273.180593277877
sigma = 20.0
```

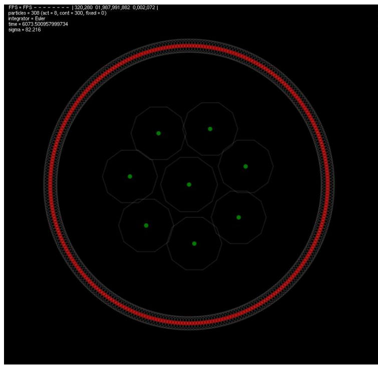


```
FPS = FPS - - - - - - - - - | 251,768  01,682,530,076  0,001,147 |
particles = 476 (act = 176, cont = 300, fixed = 0)
integrator = Euler
time = 4273.180593277877
sigma = 20.0
```

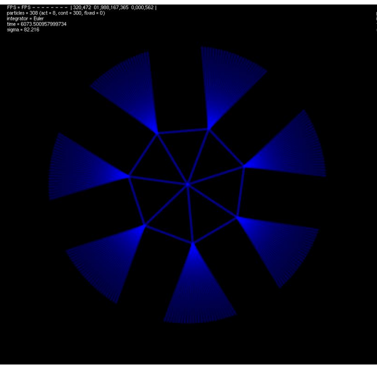


```
FPS = FPS - - - - - - - - - | 254,735  01,685,724,446  0,001,916 |
particles = 476 (act = 176, cont = 300, fixed = 0)
integrator = Euler
time = 4273.180593277877
sigma = 20.0
```

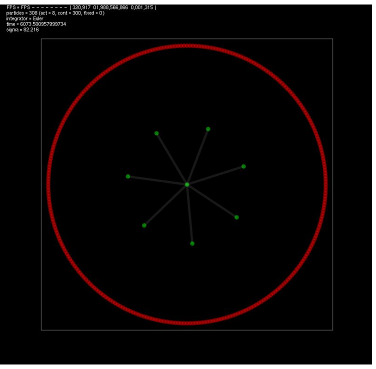
(a) Particles ($\sigma = 20, n = 176$) (b) Connectivity (c) System grid



```
FPS = FPS - - - - - - - - - | 320,280  01,987,991,882  0,002,072 |
particles = 308 (act = 8, cont = 300, fixed = 0)
integrator = Euler
time = 6073.500957999734
sigma = 82.216
```



```
FPS = FPS - - - - - - - - - | 320,472  01,988,167,365  0,000,562 |
particles = 308 (act = 8, cont = 300, fixed = 0)
integrator = Euler
time = 6073.500957999734
sigma = 82.216
```



```
FPS = FPS - - - - - - - - - | 320,917  01,988,566,866  0,001,315 |
particles = 308 (act = 8, cont = 300, fixed = 0)
integrator = Euler
time = 6073.500957999734
sigma = 82.216
```

(d) Particles ($\sigma = 80, n = 8$) (e) Connectivity (f) System grid

Figure 4–6: Circle sampling for varying repulsion radii σ.

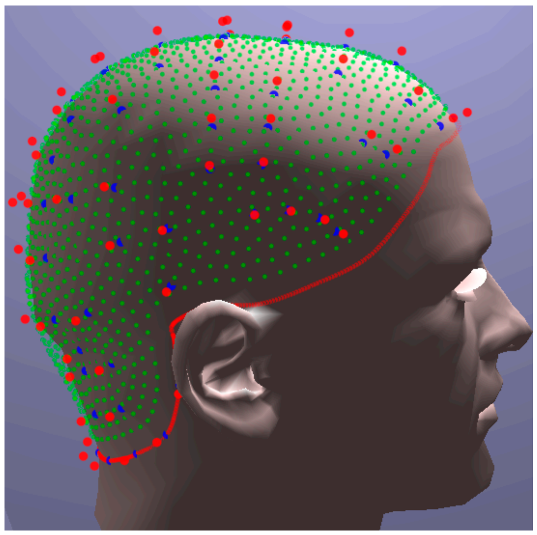


Figure 4–7: Sampling of 5000 follicles on the scalp using particle methods. Contour particles are shown in red following the scalp contour. Implicit constraints are shown as pairs of red/blue points.

CHAPTER 5
Interpolating the Generalized Helicoid Model

The generalized helicoid parameterization, because of its inherent simplicity, smoothness, and ability to ensure neighborhood coherency, can be efficiently used for interpolating hairs in sparse datasets. This chapter aims to demonstrate this claim. Section 5.1 introduces the inverse distance weighting scheme that is used for interpolating generalized helicoids and their local frames. Section 5.2 discusses the interpolation of orientation frames. The notions developed in sections 5.3 and 5.4 regard the interpolation of generalized helicoids, and p-helicoids, on a planar surface. These sections are further expanded into a more general framework in Section 5.5, where the interpolation is carried out on surfaces of arbitrary topology.

5.1 Inverse Distance Weighting

This section describes an inverse distance weighting (IDW) scheme for interpolating scattered data points. An IDW method describes a weighting scheme in which the interpolating surface is more influenced locally by nearby points, and less by distant points. The method used in this thesis is also referred to as Shepard's method [24]. Given a set of scattered data points $\mathbb{Q} = \{\mathbf{q}_i | i = 1, \cdots, n\}$ defined by

$$\mathbf{q}_i = (\mathbf{p}_i, \mathbf{k}_i) \tag{5.1}$$

where $\mathbf{p}_i$ is the position of the data point and $\mathbf{k}_i$ its (possibly vector) value, weights $w_i = w_i(\mathbf{p}_i, \tilde{\mathbf{p}})$ are determined in order to interpolate a value $\tilde{\mathbf{k}}$ at an arbitrary point $\tilde{\mathbf{p}}$. The interpolated value from the collection of data points is obtained by

$$\tilde{\mathbf{k}} = \sum_{i=1}^{n} w_i \mathbf{k}_i. \tag{5.2}$$

The weights are computed from a metric $\phi = \phi(\mathbf{p}, \tilde{\mathbf{p}})$ that is a function of the distance from the interpolant to the interpolation location. The weights are then normalized

$$w_i = \frac{\phi_i}{\sum_{i=1}^{n} \phi_i} \tag{5.3}$$

such that

$$\sum_{i=1}^{n} w_i = 1. \tag{5.4}$$

The interpolation is thus given by

$$\tilde{\mathbf{k}} = \frac{\sum_{i=1}^{n} \phi_i \mathbf{k}_i}{\sum_{i=1}^{n} \phi_i}. \tag{5.5}$$

The metric ϕ determines the behavior of the interpolating surface in the vicinity of control points. Different metrics were experimented with, including α-norms, power functions, exponential functions, and other harmonic functions. Section 5.3 discusses these metrics in more details. Cut-offs are typically used when the interpolation location hits an interpolant within an ϵ-radius, in order to ensure a perfect match. In this case the weight for this interpolant is set to 1 and the others are set to 0. This also prevents the computation of an ∞/∞ in the normalization, as is the case for the metric $\phi = ||\mathbf{p} - \tilde{\mathbf{p}}||^{-1}$ when $\mathbf{p} = \tilde{\mathbf{p}}$.

5.2 Interpolation of Local Frames

The problem of interpolating coordinate frames is often found in computer animations, where the path of an object is interpolated from a selection of key orientations located in space. There are many ways by which coordinate frames can be interpolated, and we refer to Dam et al. [21] for more information. In this Chapter we describe three ways of interpolating coordinate frames, using rotation matrices, Euler angles and quaternions.

5.2.1 Interpolation of Rotation Matrices and Euler Angles

As described in Section 3.1, a transformation matrix $\mathbf{E} \in \mathbb{R}^{4\times 4}$ from the canonical basis of $\mathbb{R}^3$ to another basis $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3$ located at position $\mathbf{p}$ can be represented as

$$\mathbb{E} = \begin{bmatrix} | & | & | & | \\ \mathbf{E}_T & \mathbf{E}_N & \mathbf{E}_B & \mathbf{p} \\ | & | & | & | \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.6}$$

This transformation matrix is actually a *rigid transformation*, and can be factorized out into its rotational $\mathbf{R}$ and translational $\mathbf{T}$ components:

$$\mathbb{E} = \mathbf{TR}. \tag{5.7}$$

The translation $\mathbf{T}$ is simply given by

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

and the rotation matrix $\mathbf{R}$ can be further factorized as a series of rotations of magnitude $\theta_x, \theta_y, \theta_z$ about the x, y and z axes respectively:

$$\mathbf{R} = \mathbf{R}_x(\theta_x)\mathbf{R}_y(\theta_y)\mathbf{R}_z(\theta_z). \tag{5.9}$$

The three rotation matrices can be expressed explicitely:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}, \mathbf{R}_y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \mathbf{R}_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.10}$$

The order in which the rotations are applied is important (rotation matrices are non-commutative) and there is no unique combination of $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$ such that $\mathbf{R} = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z$ since their ordering can be permuted and different angles can be used to obtain the same result. Moreover, one rotation about a given x, y or z axis can be completely ignored (only two are needed in order to get access to all degrees of freedom). For instance, the rotation ordering ZXZ and its corresponding rotation angles (ϕ, θ, ψ) are the Euler angles. The rotation ordering XYZ is known as TaitBryan angles or roll (x), pitch (y) and yaw (z).

Both the rotation matrix and the Euler angles representations can be directly used to interpolate coordinate frames. As an example, consider the linear interpolation of coordinate frames between two control points $\mathbb{E}_1$ and $\mathbb{E}_2$ with a free parameter $0 \leq t \leq 1$ that describes the line joining them. In matrix form, the interpolation can be viewed as

$$\tilde{\mathbb{E}} = (1-t)\mathbb{E}_1 + (t)\mathbb{E}_2. \tag{5.11}$$

Similarly, interpolating Euler angles as 3-tuples $\boldsymbol{\Theta} = (\phi, \theta, \psi)$ yields

$$\tilde{\mathbb{E}} = (1-t)\boldsymbol{\Theta}_1 + (t)\boldsymbol{\Theta}_2. \tag{5.12}$$

Both the matrix and Euler angle interpolation methods suffer from what is called a *Gimbal lock* (when a degree of freedom is lost) and the inter-dependencies between axes are ignored, amongst other disadvantages [21]. The quaternion representation is more robust and has many advantages over these two methods in the context of coordinate frame interpolation.

5.2.2 Interpolation of Quaternions

A unit quaternion is defined by a 4-vector

$$\mathbf{q} = (q_0, q_1, q_2, q_3) \tag{5.13}$$

with $||q||^2 = \mathbf{q}_0^2 + \mathbf{q}_1^2 + \mathbf{q}_2^2 + \mathbf{q}_3^2 = 1$. A quaternion $\mathbf{q}$ is represented in $\mathbb{R}^4$ as

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{5.14}$$

where i, j, k are the elements of this basis and are fully described by the following properties:

$$i^2 = j^2 = k^2 = ijk = -1. \tag{5.15}$$

Quaternions can be used to describe coordinate frames. The interpolation of coordinate frames using their quaternion representation involves the components of the 4-vector $\mathbf{q} = (q_0, q_1, q_2, q_3)$. The coordinate frame can be reconstructed from a quaternion using ([33]):

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}\,(2(q_0 q_1 + q_2 q_3, 1 - 2(q_1^2 + q_2^2)) \\ \arcsin\,(2(q_0 q_2 - q_3 q_1) \\ \text{atan2}\,(2(q_0 q_3 + q_1 q_2, 1 - 2(q_2^2 + q_3^2)) \end{bmatrix} \tag{5.16}$$

and subsequently applying rotation matrices $\mathbf{R}_x(\phi), \mathbf{R}_y(\theta), \mathbf{R}_z(\psi)$ and a translation matrix $\mathbf{T}$.

5.3 Single Helicoid Interpolation

The interpolation of generalized helicoids defined by a single set of curvature parameters, referred to as *single* generalized helicoids, is now described.

In this context, the parameter nodes are defined directly on the growth surface, and nowhere else in space. It is assumed that an initial sampling of a region is available with *parameter nodes* $\mathbf{w}_i$ defined by $\mathbf{w}_i = (\mathbf{k}_i, \mathbf{p}_i)$ and $\mathbf{k}_i = (k_T, k_N, k_B, \alpha)$ where $\mathbf{k}_i$ is the curvature vector introduced in Chapter 3.2 and $\mathbf{p}_i$ the spatial location of the node. These parameter nodes are actually control points through which the interpolation process is carried out. These curvature vectors can be interpolated using an IDW scheme at new locations $\mathbf{p}'_i$ in space and a dense collection of additional parameter nodes can be generated. Hair strands are then traced using the methods described in Chapter 3.3, in this volume of curvature vectors. A key property is that since the interpolation is carried out on curvature parameters, the resulting interpolants are themselves guaranteed to be generalized helicoids. The application of an arbitrary rotation to the parameter nodes about the surface normal allows to generate hairs of a variety of possible geometries in in any quadrant.

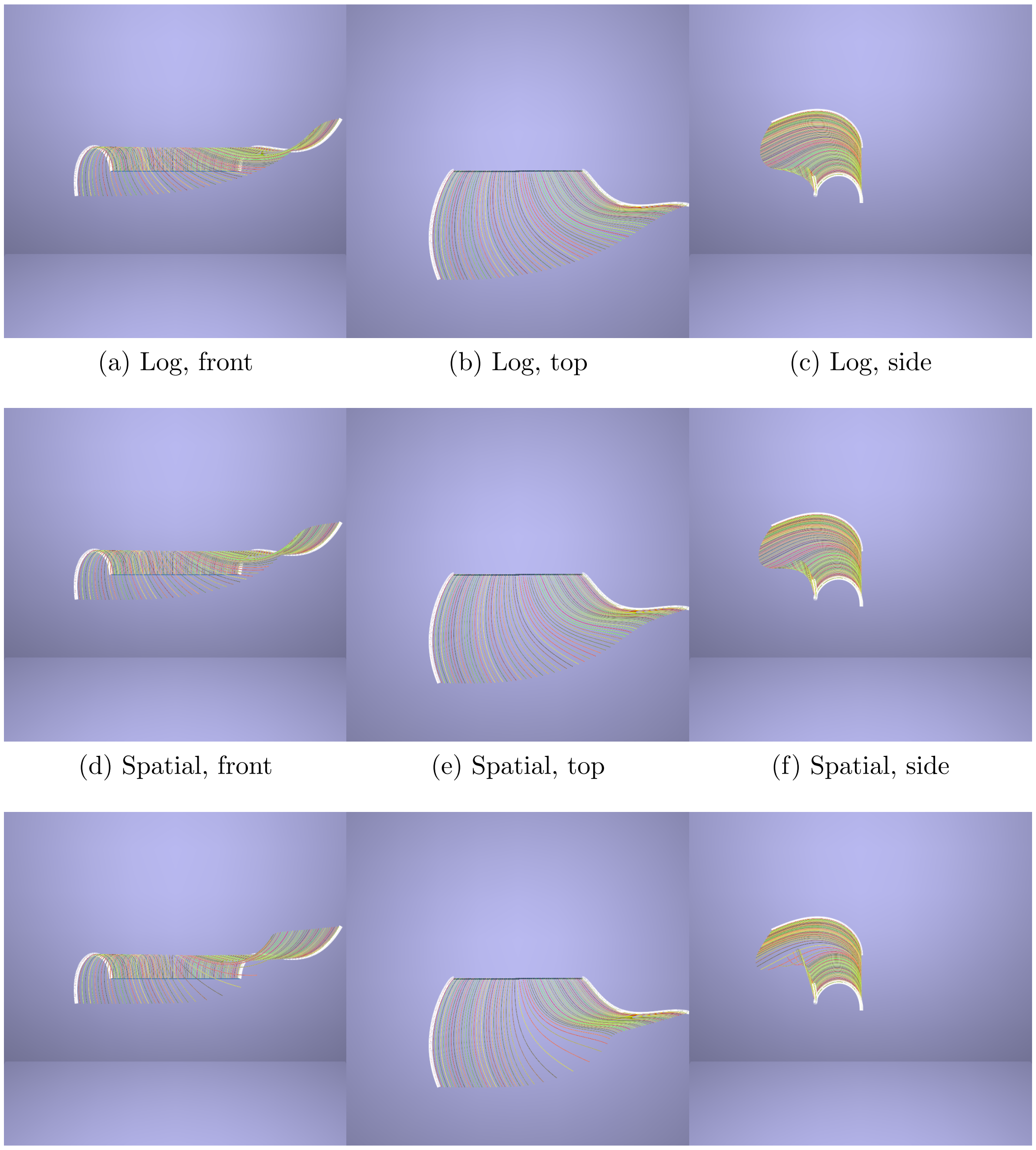
Different interpolation strategies were tested and the spatial metric was adopted. It was determined that the spatial metric provides consistent and smooth results, while allowing the user to control the degree of consistency with the set of guide hair strands, and so this is the metric that is used for the rest of this thesis. Table 5–1 shows the list of metrics investigated. Figures 5–1 and 5–2 show the difference between the metrics in the context of generalized helicoid interpolation in both the 1D, and 2D case. For comparison, a simple IDW Euclidean interpolation scheme was implemented on the positions of the polyline vertices. A comparison of this point-based interpolation with the parametric interpolation using generalized helicoids and the logarithmic metric is shown in Figure 5–3. Qualitatively, interpolating generalized helicoids yields slightly smoother results. More importantly, since the geometry and local frames of guide hairs are interpolated in the reduced coordinate system of the generalized helicoid parameters, the interpolated hairs flow naturally from one guide hair to the other while preserving their shape. This is a key property in the reconstruction of hair styles from sparse initial hair strand samples. In contrast, the interpolation of vertices breaks the geometrical similarity between the two guide hairs, and leads to unwanted geometries, for instance a straight hair at their midpoint, as seen in Figure 5–3b.

Table 5–1: Various metrics used for determining interpolation weights.

ϕ	**Description**
$\log\left(1 + \|\|\mathbf{p} - \tilde{\mathbf{p}}\|\|_2\right)^{-1}$	Logarithmic norm. Produces smoother turning points around the interpolants.
$\|\|\mathbf{p} - \tilde{\mathbf{p}}\|\|_\alpha^{-1}$	Spatial norm. Produces sharper turning points around the interpolants. Different values of α yield similar results, with a lower α giving smoother results, at the cost of reduced precision.
$\exp\left(-\frac{\|\|\mathbf{p}-\tilde{\mathbf{p}}\|\|}{\sigma^2}\right)$	Exponential norm. Produces very sharp turning points around interpolants. σ determines the length scale at which the interpolants cease to influence their neighborhood and can be adjusted to regularize the interpolation.

5.4 Procedural Piecewise Helicoid Interpolation

The method used for p-helicoid interpolation builds on top of the ideas developed in Section 5.3 for single generalized helicoids. The interpolation scheme and metric remain the same. The major difference lies in the space of helicoidal interpolants, which is now allowed to be populated everywhere (following the parameter nodes of the p-helicoids that serve as guide hairs), not only on the growth surface. This implies major changes in the way the interpolation of hairs is carried out. Interpolated p-helicoids are computed procedurally, piece by piece. The number of helicoidal pieces composing each guide hair also has to be considered. When a hair is interpolated, the following two questions must be answered: how many pieces should compose the interpolated p-helicoid, and what should their individual length be? It

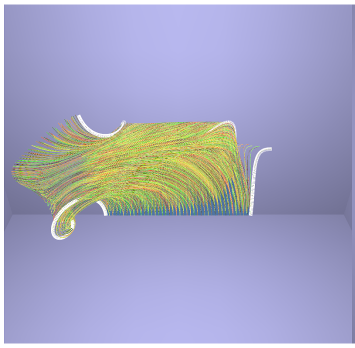


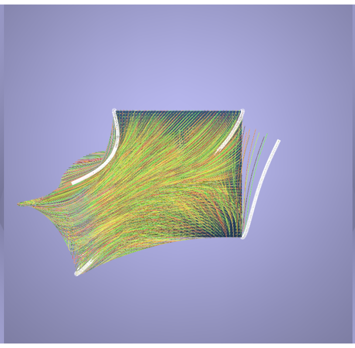
(a) Log, front (b) Log, top (c) Log, side

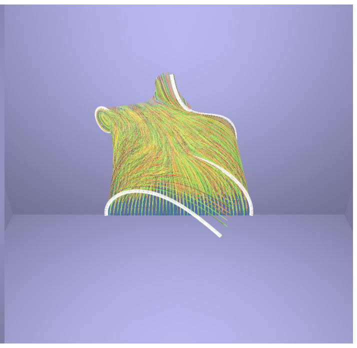
(d) Spatial, front (e) Spatial, top (f) Spatial, side

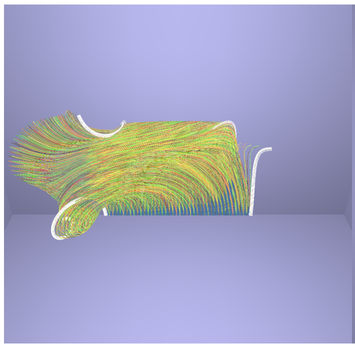
(g) Exponential, front (h) Exponential, top (i) Exponential, side

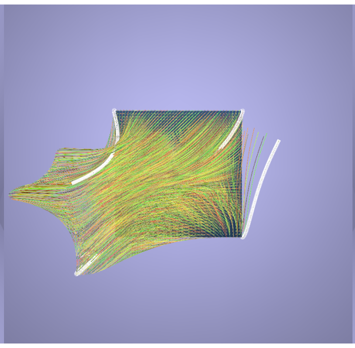
Figure 5–1: Comparison of different weighting schemes for interpolating generalized helicoids on a one-dimensional manifold for oriented curvature parameters. Guide hairs are shown in white, and interpolated hairs in random colors.

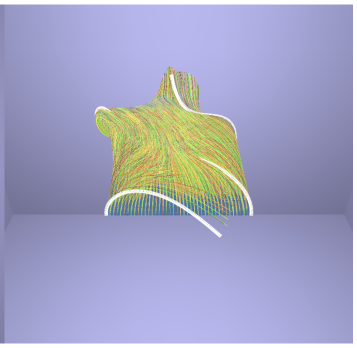

(a) Log, front

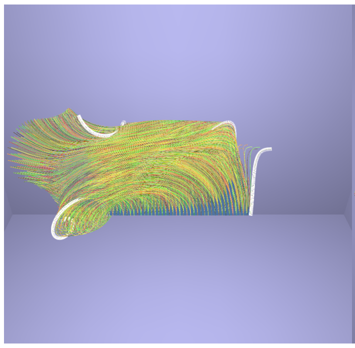

(b) Log, top

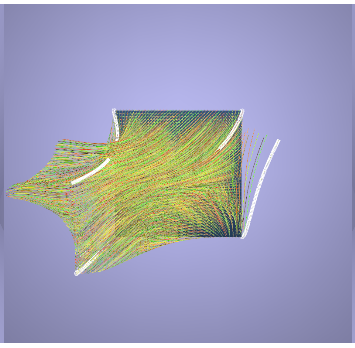

(c) Log, side


(d) Spatial, front


(e) Spatial, top


(f) Spatial, side


(g) Exponential, front


(h) Exponential, top

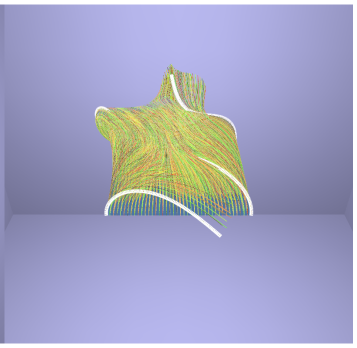

(i) Exponential, side

Figure 5–2: Comparison of different weighting schemes for interpolating generalized helicoids on a two-dimensional manifold for oriented curvature parameters. Guide hairs are shown in white, and interpolated hairs in random colors.

turns out that the answer to these questions can also be found by means of interpolation.

The procedural p-helicoid interpolation method is carried out on a set of p-helicoid guide hairs, and it produces an interpolated p-helicoid rooted at location p. A p-helicoid is interpolated procedurally by interpolating multiple single helicoids, starting from the location of the root $\mathbf{p} = \mathbf{r}$, and aligning them together at their end. In point form, the algorithm works as follows:

1. Interpolate the number $\tilde{n} \in \mathbb{R}$ of helicoidal pieces at $\mathbf{p}$. $\tilde{n}$ is rarely an integer, except when all guide hairs have the same number of pieces. The number of pieces in that p-helicoid is thus defined as the ceiling of that number, $n = \lceil \tilde{n} \rceil$.
2. Fix the step size, and step count in the tracing algorithm. The length of the current interpolated helicoid is determined by scaling it uniformly by some scaling factor s. For a single helicoid, $s = 1$ is used. Therefore, to be consistent with the varying length across p-helicoids, $s = 1/\tilde{n}$ is used for the first $n - 1$ helicoids, and $s = r/\tilde{n}$ for the last one, where the residual r is defined as the fractional difference between $\tilde{n}$ and n: $0 \leq r = 1 - (n - \tilde{n}) \leq 1$. This ensures that the last helicoid will only be long enough such that there is a smooth transition between p-helicoids having different number of pieces.
3. Interpolate a generalized helicoid parameter vector from the guide hairs.
4. Trace a generalized helicoid piece from this parameter vector. Orient the piece properly by using the Frenet frame at the tip of the previous one.
5. Repeat with the next helicoid piece using $\mathbf{p}$ was the tip of the current piece.

Figure 5–4 and 5–5 show the interpolation of p-helicoid guide hairs on a line, and on a plane. This interpolation is compared with the vertex-based approach described in Section 5.3. Again, in the helicoidal approach, the interpolation of local frames makes hairs flow and turn naturally from one guide hair to the other while preserving their shape. In the vertex-based approach, although the interpolation is smooth over guide hairs, the geometry of interpolated hairs is inconsistent when guide hairs are facing each other. If desired, this *parting* behavior can be emulated in the generalized helicoid interpolation by placing a nearly straight guide hair where the hair is required to split on either sides.

5.5 Interpolation on an Implicit Surface

In the interpolation of guide hairs on arbitrary surfaces, the concept of local frame interpolation, addressed in Section 5.2, becomes more subtle. Two frames now have to be considered: one representing the coordinate system $\mathbf{W}$ of the surface, and the other representing the orientation frame $\mathbf{R}_\theta$ in which the helicoid is grown. The two frames are defined by

$$\mathbf{W} = \begin{bmatrix} \mathbf{E}_T & \mathbf{E}_N & \mathbf{E}_B & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17}$$

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.18}$$

where $\mathbf{E}_N$ is set according to equation 4.2. A helicoid $\mathbf{H}$ traced at the origin can be transformed to the coordinate system on the surface at position $\mathbf{p}$ using

$$\mathbf{H} \rightarrow \mathbf{W}\mathbf{R}_\theta\mathbf{H}. \tag{5.19}$$

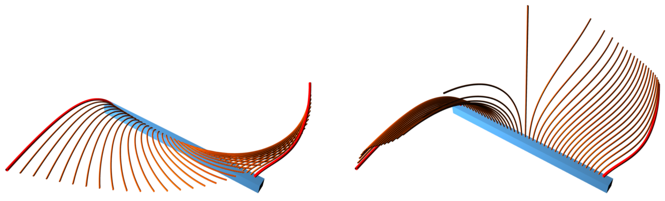
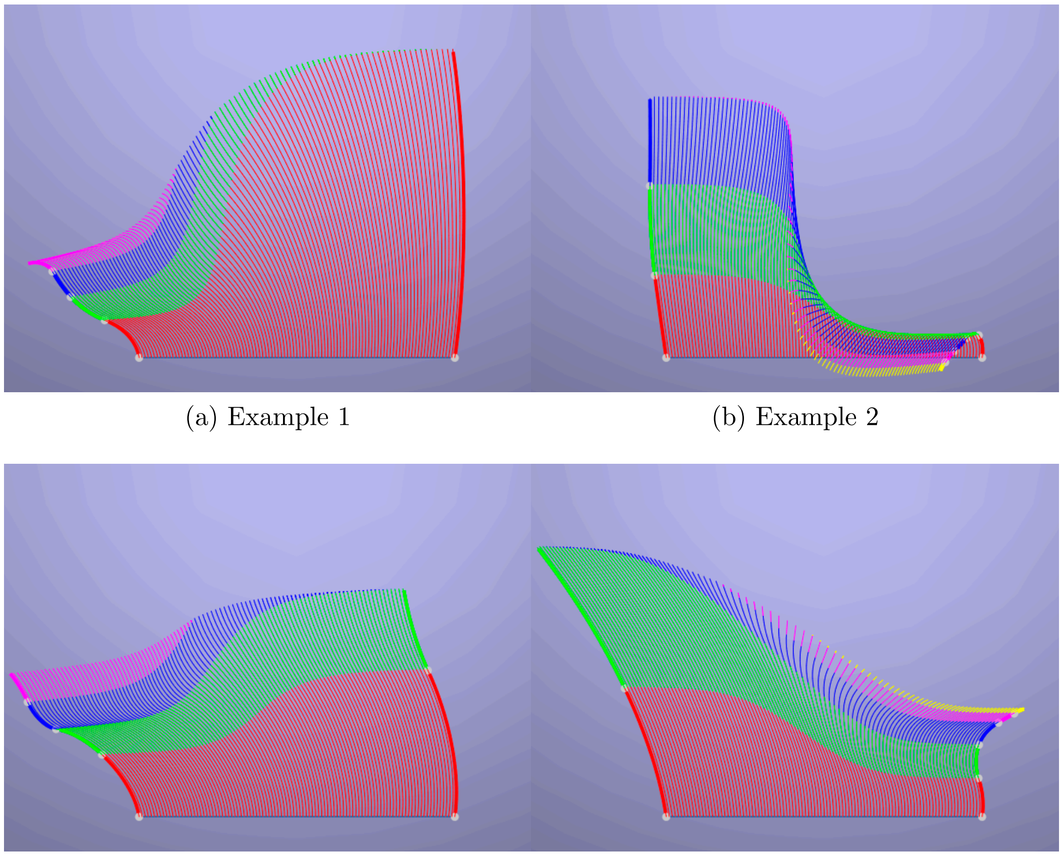
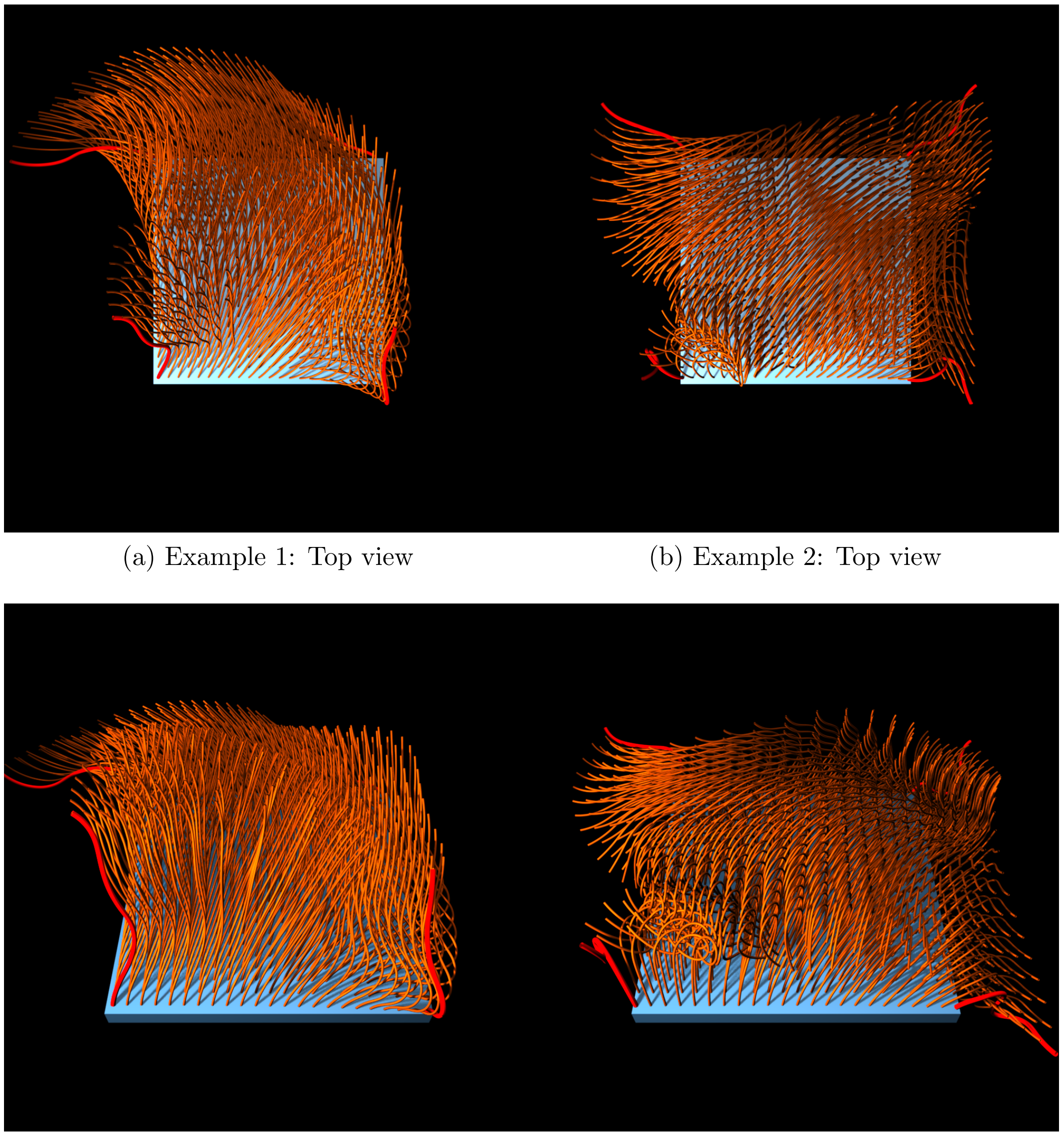

(a) Helicoid-based (b) Sampled Point-Based

Figure 5–3: A comparison of interpolation between guide hairs using generalized helicoids (left) and using only points sampled along the guide strands (right). See text for a discussion. The two guide hairs (in red) are π-rotated copies of each other, with a parameter vector $k = (1.39, -0.008, 0.904, -0.225)$.


(a) Example 1 (b) Example 2

(c) Example 3 (d) Example 4

Figure 5–4: Comparison of the interpolation of curvature parameters and vertices, for piecewise helicoid guide hairs. A color code is used to distinguish between helicoid pieces. Two guide hairs are used, one having three helicoid pieces on the left and the other having two helicoid pieces on the right.



(a) Example 1: Top view

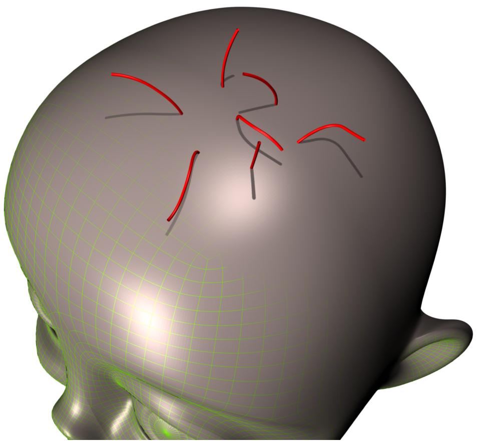
(b) Example 2: Top view

(c) Example 1: Front view

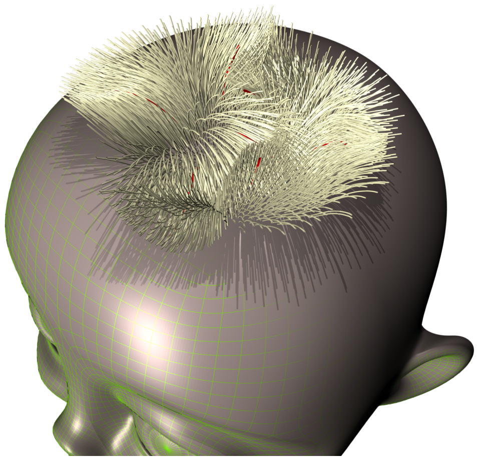
(d) Example 2: Front view

Figure 5–5: Two examples of interpolation between p-helicoids on a plane. The four guide hairs are shown in bright red and the interpolated hair strands are shown in brown.

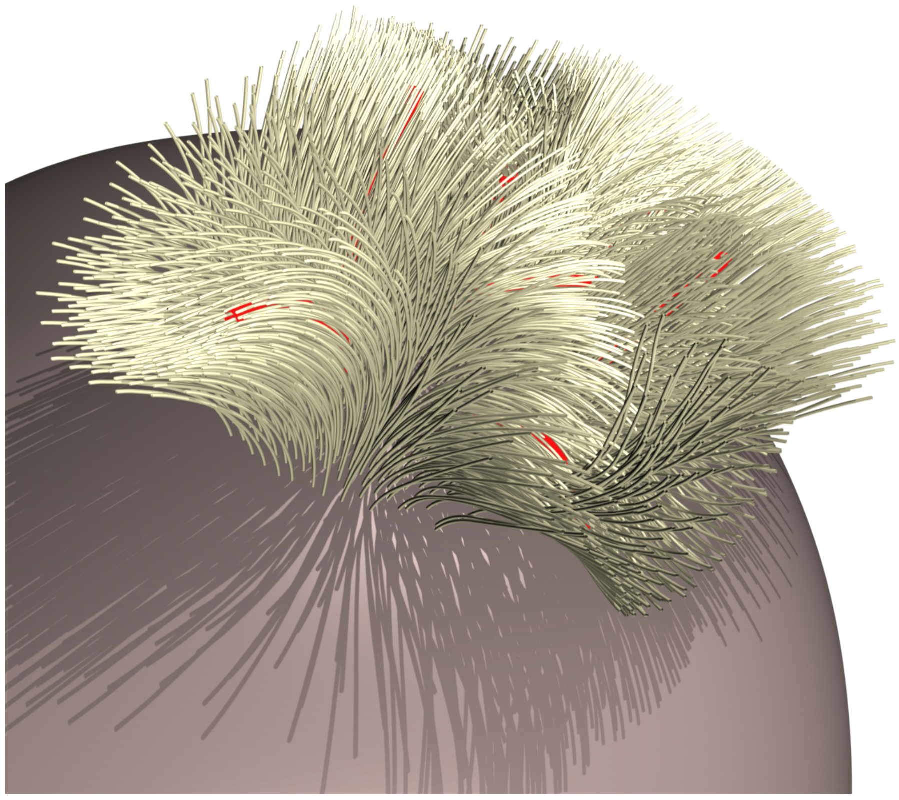
using the fact that the helicoid orientation frame is just a rotation of magnitude θ about the world normal $\mathbf{e}_2$. Since $\mathbf{W}$ is defined everywhere on the implicit surface, the only frame that needs to be interpolated is $\mathbf{R}_\theta$, so it should be stored separately. Figure 5–6 shows the interpolation of guide hairs on an implicit surface. A parting is obtained by using opposite normal rotations and widely spaced guide hairs, as shown in Figure 5–7.



(a) Guide hairs

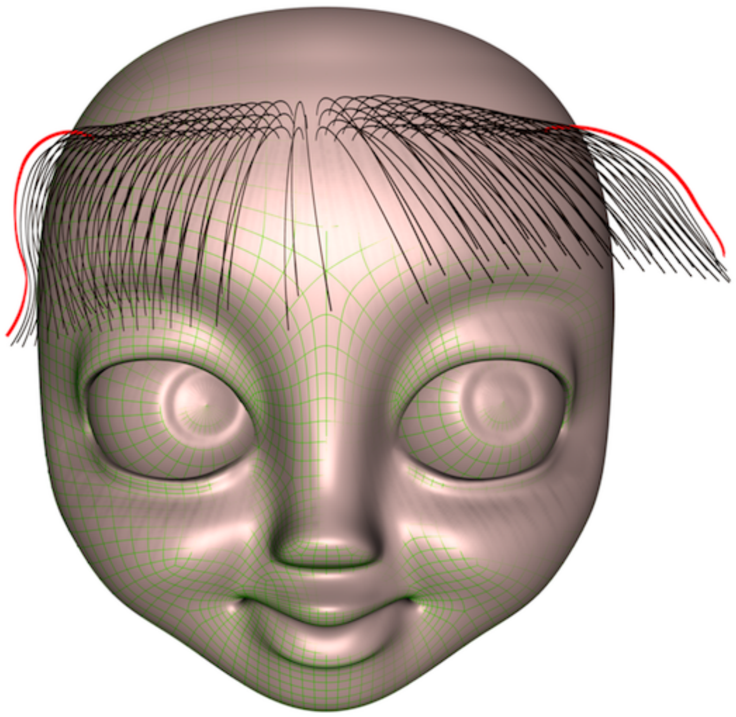


(b) Interpolated, side view

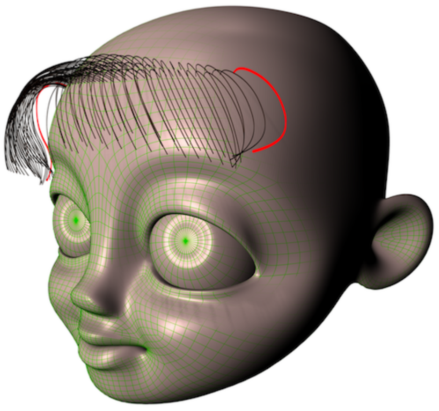


(c) Interpolated, front view

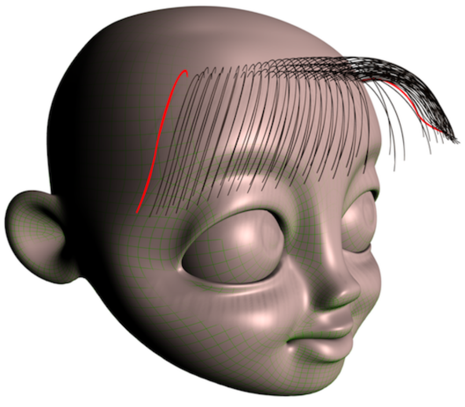
Figure 5–6: Interpolation of a tuft on an implicit surface. The guide hairs are shown in red, and the interpolated ones in brown.



(a) Front view



(b) Side view



(c) Side view

Figure 5–7: Hair parting using normal rotations. The two guide p-helicoids (in red) are composed of three helicoid pieces and have different curvature parameters. The interpolated hairs are shown in brown.

CHAPTER 6
Fitting the Generalized Helicoid Model

Chapters 3 and 5 demonstrated that the generalized helicoid framework can generate hairstyles from scratch. A small number of helicoidal hair samples are initially generated by hand and following their interpolation a complete hairstyle is constructed. In this case, the samples are of synthetic nature and their helicoidal parameterization is available. There are other times when the hair samples consist of raw data, represented as polylines having no helicoidal parameterization. This chapter demonstrates that generalized helicoids can be exploited to capture, manipulate, and reconstruct the flow of an existing hair pattern. To this end, a hair fitting algorithm is proposed that draws a correspondence between a set of polylines and their piecewise helicoid parameterization. Combined with the interpolation method described in Section 5.4, this parameterization can be used to generate dense hair patterns tailored to external sparse hair distributions.

Section 6.1 describes the local frame subtraction that transforms target hairs to their natural reference frame. Section 6.2 introduces the Fréchet distance, the distance metric that is minimized in the fitting of a target hair to a generalized helicoid. Section 6.3 describes the fitting algorithm for single generalized helicoids. Section 6.4 describes a neural network strategy for accelerating the fitting process. Section 6.5 extends the case of single generalized helicoids to the case of p-helicoids for fitting target hairs.

6.1 Local Frame Subtraction

Before a polyline is fit, the local coordinate system $\mathbf{Q}$ in which it is located has to be determined. The fitting process uses this frame to translate and orient generated polylines such that they are aligned with the target hair (or equivalently, the fitting could be done in world coordinates, only then $\mathbf{Q}^T$ would be used instead). $\mathbf{Q}$ is the product of two transformation matrices. The first, $\mathbf{T}$, corresponds to the local frame on the surface where the hair is located. The normal $\mathbf{E}_T$ is known, and two additional orthogonal vectors $\mathbf{E}_N$, $\mathbf{E}_B$ on the surface are found by taking the cross product of the normal with an arbitrary reference vector. The second transformation matrix, $\mathbf{R}$, corresponds to the alignment of the hair with respect to the first frame. $\mathbf{R}$ is built such that the first segment (tangent) of the polyline corresponds to the first orthogonal vector $\mathbf{v}_T$ of the basis. The second and third vectors of the basis, $\mathbf{v}_N$ and $\mathbf{v}_B$, are found from the normal and binormal components of the discrete Frenet frame at the origin of the polyline. The transformation from a polyline $\mathbf{h}$ to a polyline $\tilde{\mathbf{h}}$ in the local coordinate of the target hair located at a location $\mathbf{p}$ on the surface is summarized as follows:

$$\begin{aligned}
\tilde{\mathbf{h}} &= \mathbf{Qh} && (6.1)\\
&= \mathbf{TRh} && (6.2)\\
&= \begin{bmatrix} | & | & | & | \\ \mathbf{E}_T & \mathbf{E}_N & \mathbf{E}_B & \mathbf{p} \\ | & | & | & | \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} | & | & | & 0 \\ \mathbf{v}_T & \mathbf{v}_N & \mathbf{v}_B & 0 \\ | & | & | & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{h} && (6.3)
\end{aligned}$$

6.2 Computing the Similarity Measure: The Fréchet Distance

The Fréchet distance is proposed as a natural measure of the similarity between two polylines. Alt and Godau [1] provides an intuitive description of the Fréchet distance, using the analogy of a dog and its handler walking on their respective paths, as illustrated in Figure 6–1. Both the man and the dog can control their speed independently but can never backtrack. The Fréchet distance between these two curves is the minimal length that a leash must have for the dog and the handler to move from the starting points of the two curves to their respective endpoints. A discrete variant of the Fréchet distance, which is referred to as the Fréchet coupling, was introduced in Eiter and Mannila [22], and better represents a global measure of the distance between two polygonal curves, represented as a list of points. This is the measure that is used in the fitting of generalized helicoids to data. Returning to the walking man and his dog analogy, this measure represents the minimum total length of leashes needed for men walking their dogs – one on each point of each curve – such that leashes do not cross over. The simplest dynamic algorithm for computing the Fréchet coupling has a computational complexity of $O(pq)$, where p and q are respectively the number of points on the two discrete curves. Algorithm 2 shows the pseudocode for computing the Fréchet coupling between two curves. ca is the array in which the dynamic programming algorithm stores the pairwise Fréchet distances from one polyline to the other. Figure 6–2 shows the mean running time in ms for computing the discrete Fréchet coupling between random polygonal curves of varying length. The quadratic fit on the plot reflects the expected complexity of $O(pq)$.

6.3 Discrete Optimization of Single Generalized Helicoid Fitting

The simplest case of hair fitting uses a single generalized helicoid to approximate a target hair. A parameter vector $\mathbf{k} = (k_T, k_N, k_B, \alpha)$ that generates the trace of a hair is optimized iteratively. The objective function is the minimization of the Fréchet distance between a target hair h located at r with local frame $\mathbf{Q}$ and a helicoidal trace H:

$$\mathbf{k}_0 = \underset{\mathbf{k}}{\operatorname{argmin}}\ \mathbf{fc}\left(c_1, H(\mathbf{k}, \mathbf{r}, \mathbf{Q})\right). \tag{6.4}$$

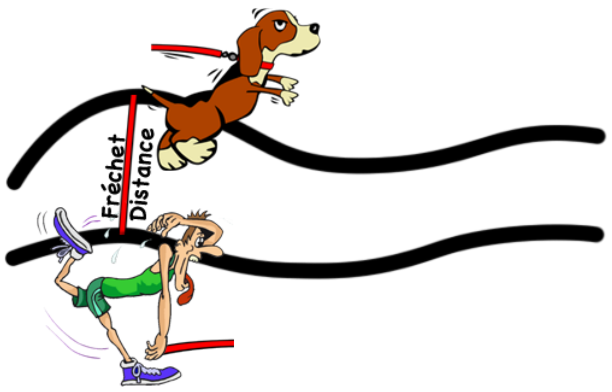


Figure 6–1: Fréchet distance analogy to a man walking his dog on a leash, on two separate paths. The Fréchet distance between these two curves is the minimal length that a leash must have for the dog and the handler to move from the starting point of each curve to their respective endpoint.

Algorithm 2 Frechet coupling

```
function fc(Polyline c1, Polyline c2, float array ca, int $i$, int $j$)
if $ca(i,j) \geq -1$ then
    return $ca(i,j)$
else if $i = 1$ and $j = 1$ then
    $ca(i,j) := d(c_1(1), c_2(1))$
end if
elseif $i > 1$ and $j = 1$ then $ca(i,j) := \max\{fc(i-1,1), \mathbf{d}(c_1(i), c_2(1)\}$
elseif $i = 1$ and $j > 1$ then $ca(i,j) := \max\{fc(1,j-1), \mathbf{d}(c_1(1), c_2(j))\}$
elseif $i > 1$ and $j > 1$ then $ca(i,j) :=$
        $\min\{fc(i-1,j), fc(i-1,j-1), fc(i,j-1)\} + \mathbf{d}(c_1(i), c_2(j))\}$
else $ca(i,j) = \infty$
return $ca(i,j)$
end function

function d(Point $a$, Point $b$)
return $||a-b||_2$
end function
```

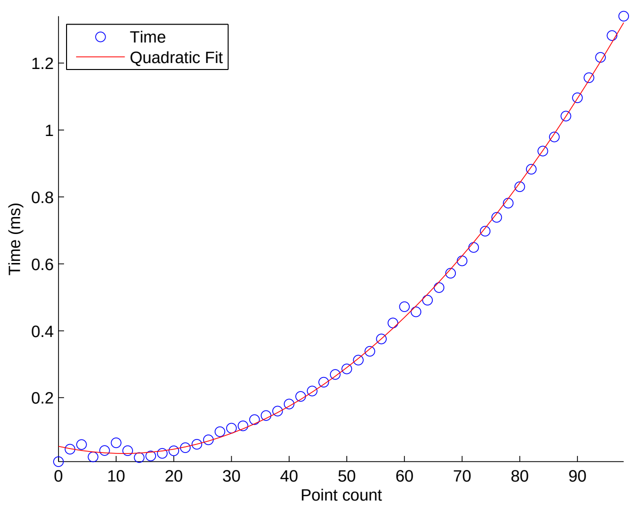


Figure 6–2: Mean computational time (ms) involved in evaluating the Fréchet distance between random curves of increasing point count.

In order to avoid numerical approximation errors – the objective function equation 6.4 is not analytical – it is preferable to use a gradient-free method. The Nelder-Mead (NM) algorithm, first proposed by Nelder and Mead [59], is a technique for optimizing a function in a multidimensional space that generally performs well with arbitrary seed points. The algorithm only uses the evaluation of the function and does not require the computation of an approximate gradient in its optimization process.

The optimized parameter is the curvature vector $\mathbf{k} = (k_T, k_N, k_B, \alpha)$ that is used to trace a hair using Algorithm 1. The distance between this trace and the target hair is evaluated using the Fréchet coupling measure defined in Algorithm 2. The optimization is done iteratively over a simplex constructed of $N + 1$ vertices, where $N = 4$ is the dimensionality of the function to optimize. Figure 6–3 shows an example of NM optimization for the case $N = 2$, where the simplex of dimension 3 forms a triangle in the plane. Each vertex corresponds to a different test point, and is assigned a value following the Fréchet coupling measure. From an initial simplex, the Nelder-Mead optimization scheme uses a combination of heuristics in order to climb towards

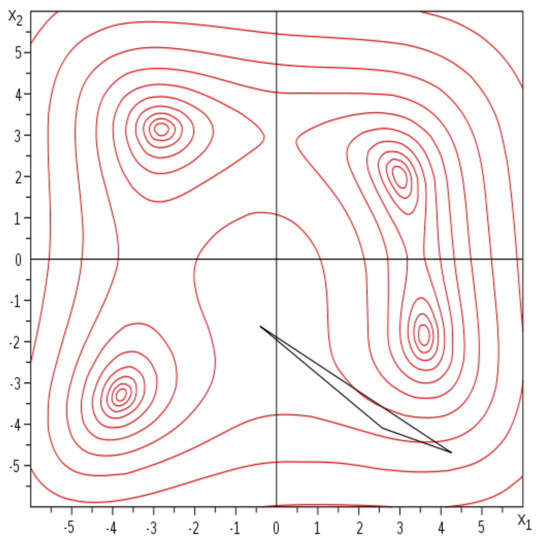


Figure 6–3: Tridimensional simplex for finding the optima of a bivariate Himmelblau function.

the local optima of the function. Put simply, the NM method extrapolates the behavior of the objective function at each vertex, and follows the direction of the worse updated vertex at each iteration. The algorithm iteratively loops over a series of intermediate steps:

1. Order values
2. Calculate centroid
3. Compute a simplex transformation (reflection/expansion/contraction) along the worse updated vertex
4. Compute simplex reduction if the transformation step failed

6.4 Neural Networks Seeds

There are instances for which the optimization will fail to converge. This happens if the parameter vector **k** reaches a local minima that has a large magnitude and the increase in one or more curvature parameter has no significant effect on the resulting trace. The output from a neural network trained with generalized helicoid data can be used as a seeding point, and this is often crucial in making the optimizer converge faster and closer to the global optima. A multilayer feedforward backpropagation neural network is used for fitting generalized helicoid parameters to hair strands. A resilient propagation (RPROP) learning [72] strategy is used that updates neural weights based on local gradient information (see Appendix C, Hornik et al. [35] and Scarselli and Chung Tsoi [78] for more details). The inputs to the neural network are the characteristics of the hair strands used in the training process. More precisely, the curvature κ, torsion τ, and tangential vector $\mathbf{E}_T$ evaluated at multiple locations along each strand using the methods described in Section 3.1. In general, a good rule of thumb is to use a sampling of 10% of the vertices describing the hairs. For the examples shown in this paper, 10 tangent, curvature, and torsion evaluations for generalized helicoids containing 100 vertices were sufficient to describe the hair strand under investigation. The output of the neural network is a curvature parameter tuple $\mathbf{k} = (k_T, k_N, k_B, \alpha)$ that describes a particular hair strand given as input.

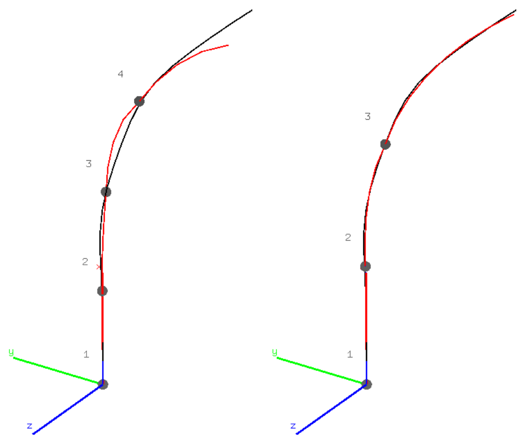
For training the neural network, a data set was generated containing synthetic hairs by sampling and tracing the parameter space $\mathbf{k} = (k_T, k_N, k_B, \alpha)$. The random parameter vectors $\mathbf{k}$ were generated in the following range: $k_T \in [-1, 1], k_N \in [-0.1, 0.1], k_B \in [-0.5, 0.5], \alpha \in [-1, 1]$. Noise offsets were added to the trace of the training data, as described in Section 3.3.2, to make the neural network more robust to noise.

6.5 Piecewise Helicoid Fitting

The examples in Section 3.2 demonstrate that whereas a *single* generalized helicoid can represent a simple hair strand, it cannot accurately capture the behavior of a more complex geometry. In practice, a single generalized helicoid is best used to fit a hair that contains up to one inflection point. In order to extend the applicability of this representation to the modeling of natural hair patterns, piecewise generalized helicoid fits are used. The essential idea is to connect together a few generalized helicoids such that they approximate a target hair strand. Each individual helicoid is fit to different pieces of the target strand, following the methods described in Section 6.3. In order to do this, the locations where the target hair is fragmented first need to be determined. For this purpose, a recursive algorithm was designed for the piecewise helicoid fitting of hairs.

The fitting process is initially carried out on the complete target hair strand. The neural network is given this target as input, and the resulting parameter vector is given as input to the direct search algorithm. The optimized parameter vector is used to generate a helicoid fit using Algorithm 1. If the Fréchet coupling measure from the target hair piece to the fitted helicoid piece is small enough, the fit is accepted. Otherwise, the target hair piece is fragmented into two pieces, at a location determined by differential and Fréchet heuristics.

The efficiency of the piecewise fit depends largely on the distribution of the target fragments that the individual helicoid pieces are fit to. The following two heuristics are used for determining this fragmentation. First, a thresholding is used on the Fréchet coupling between the target and candidate hairs. Using a threshold of 100% of the Fréchet coupling results in fitting the entire target with a single helicoid. It was experimentally determined that 60% is a reasonable threshold percentage to use, with a suitable trade off between overfititng and avoiding large biases. The objective of this thresholding is to provide an upper bound on the fragmentation location. However, more information is needed regarding the geometry of the hair to prevent a candidate hair from being accepted prematurely, in which case the fit can suffer from inconsistencies in the orientation of the hair near the fragmentation location, as seen in Figure 6–4.



(a) Thresholding only (b) Thresholding + curvature

Figure 6–4: Accepting a candidate fit prematurely will end up propagating an inherent differential error between the target and candidate hairs. The target is shown in black, and the fit in red. Fragmentation points are shown as dots along the fit.

In order to ensure that the orientation of the target hair is preserved, it is required that the candidate hair share similar differential characteristics to the target hair at the fragmentation location. This coherency is enforced between the corresponding Frenet frames on the target and candidate hairs. A curvature profile is computed on both hairs, starting from the fragmentation location, and backtracking to the origin of the fiber. The new fragmentation location is picked such that it minimizes the difference in curvature between the two fibers.

An additional hair alignment step must also be taken in order to ensure an acceptable fit. The tracing algorithm generates curves along the tangential axis of the Frenet frame of the generalized helicoid, as explained in Section 3.2. The inverse Frenet frame can be computed on the target hair, and applied on itself, thereby producing a target hair that is aligned with the growth axis of the generalized helicoid tracing algorithm. In order to preserve C^2 continuity between the helicoid pieces, once fit, the separate pieces are connected back

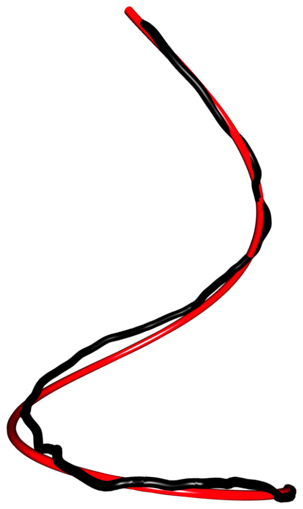


Figure 6–5: Single helicoid fit (in red) of a synthetic hair (in black) obtained with a parameter vector $(1.768, 0.024, -0.227, -0.115)$ and with noise of magnitude 0.013. The fit has a parameter vector $(1.740, -0.034, -0.136, -0.105)$.

together using the Frenet frame at the tip of their parent. The natural frame $\mathbf{Q}$ of the target hair is premultiplied to properly align the resulting p-helicoid to the target hair. The result of fitting a hair with a single helicoid is shown in Figure 6–5. A piecewise helicoid fit is shown in Figure 6–6.

6.6 Application to the Reconstruction of Real Hair

This section illustrates the fitting of real hair data to the generalized helicoid representation and its interpolation. Figure 6–7 first shows piecewise helicoid fits of unparameterized wavy hairs. This sample was extracted from a real wavy hairstyle reconstructed from different viewpoints using the methods described in Paris et al. [64]. Figure 6–8 (top left) shows a sparse sampling of the “straight” hairstyle, along with the individual p-helicoid fits to each hair strand sample (top right). The fit closely resembles the sampled hair strands. The remaining rows focus on the process of reconstructing a selected tuft at the resolution of the original data (Figure 6–8 c)) by means of subsampling and interpolation. The salient geometrical features of the original hairstyle can be recovered from as little as 10% of the fit to the guide hairs.

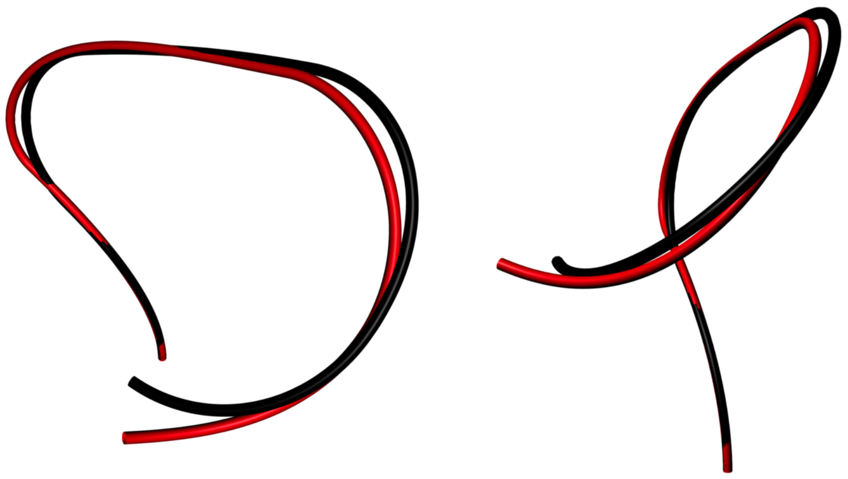


Figure 6–6: Piecewise helicoid fitting. The target hair (in black) was obtained by aligning 5 single generalized helicoids together. The fit (in red) is composed of 3 helicoidal pieces.

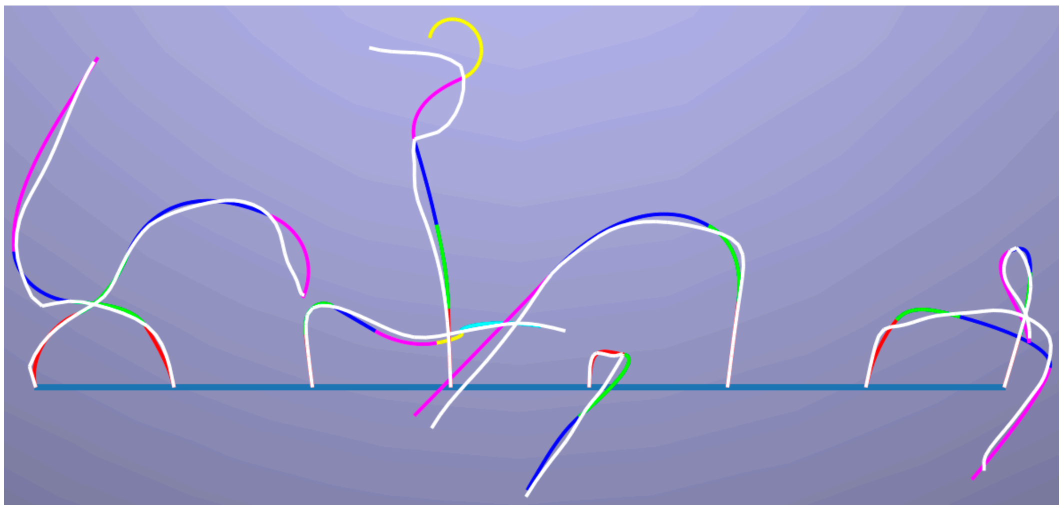
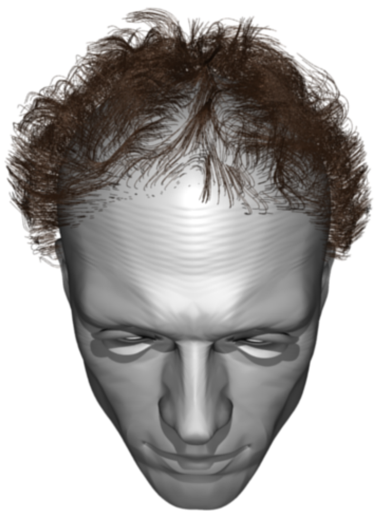
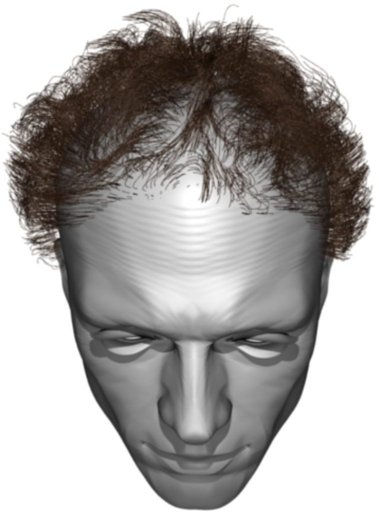
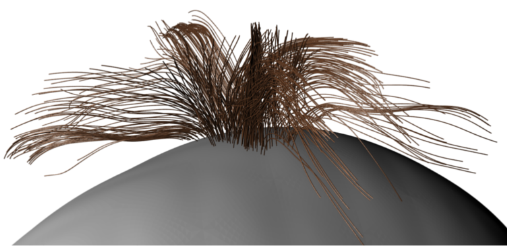
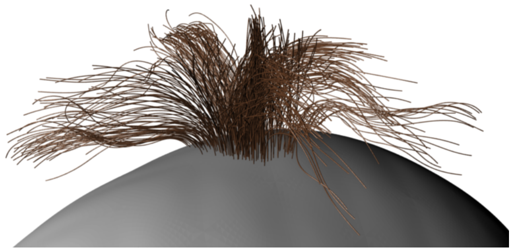


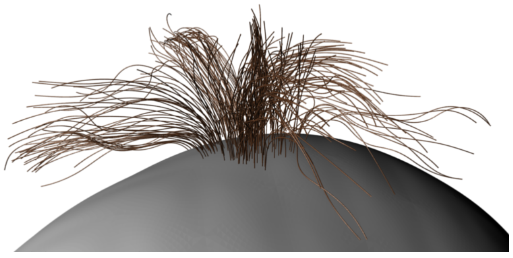
Figure 6–7: P-helicoid fitting (in color) of unparameterized data (in white) selected randomly from a hairstyle in Paris et al. [64].

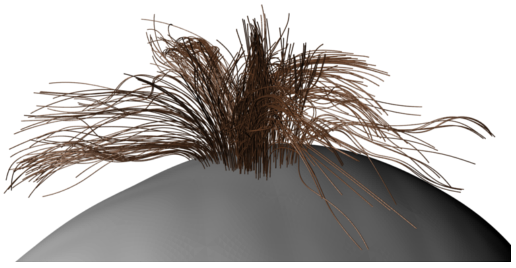

(a) Original hairstyle

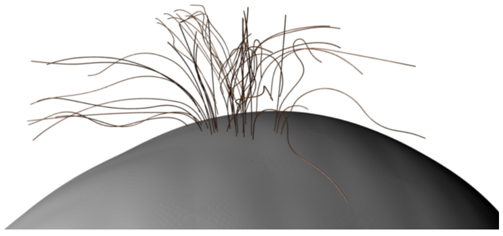

(b) Fitting each strand in a)


(c) A tuft at full resolution


(d) Fitting each strand in c)


(e) Sampling 50% of d)


(f) Interpolating from e) to 100%


(g) Sampling 10% of d)

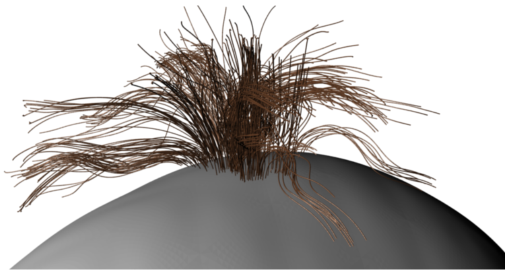

(h) Interpolating from g) to 100%

Figure 6–8: The reconstruction of the "straight" hairstyle from Paris et al. [64] by fitting and interpolation of p-helicoids.

CHAPTER 7
Conclusion

A novel mathematical model based on generalized helicoids was presented in Chapter 3 for modeling the geometry of hair-like patterns. Hair modeling was used as the driving application, resulting in a parameterized representation of hairs strands. The properties of the generalized helicoid were exploited for hair modeling. It was determined that the composition of a hair using multiple generalized helicoids, referred to as piecewise helicoids or p-helicoids, provides a richer description that allows the synthesis of complex hairstyles. A physically-driven particle-based method was designed in Chapter 4 for sampling and contouring an implicit representation of the growth surface. Chapter 5 presented methods for interpolating composite generalized helicoids on surfaces of arbitrary topology. Chapter 6 introduced algorithms for fitting the generalized helicoid to unparameterized hair data. In the following sections we discuss the many possibilities afforded by the incorporation of generalized helicoids in hair modeling. We then list several possible directions for future work.

7.1 Achievements

The hair modeling framework presented in this thesis opens up many possibilities to the computer graphics community. Piecewise generalized helicoids can be used to generate hair strands having a rich and diverse geometry. The intuitive nature of the helicoid parameters make this task simple. Waviness and noise offsets act as modifiers to the local shape of hair strands, while preserving the global geometry predetermined by their curvature parameters. The ability to generate many additional hairs from a few of these piecewise helicoids has a particular advantage. Dense hair patterns can be produced from sparse guide hairs by the interpolation of their curvature parameters, as described in Chapter 5. Similarly, a dense wisp can be generated from a single guide hair, by offsetting the origin of its streamline trace. The wisp follows the principal direction of the guide hair, while fanning out, corresponding to the adjustment of a single parameter.

Motivated by progress in reconstruction methods for acquiring real hair data [63, 64, 37] where individual hairs are described by the vertices of a polyline, Chapter 6 developed a procedure for fitting a p-helicoid to a hair strand using a measure of similarity based on Fréchet distance. The fitting process can be carried out for a set of sampled hair strands, following which their p-helicoid parametrization can be interpolated densely to fill in the hairstyle. Results indicate that both the fits and the interpolated strands preserve the qualitative geometry of the hairstyle. In fact, a complex tuft of hair can be reconstructed from as little as 10% of the original unparametrized hair data. Thus, the p-helicoid representation may be used to parametrize and then compress a hairstyle, an application that will likely grow in interest as databases containing detailed hair capture data become available.

7.2 Future Work

This thesis paves a way for many possible further directions. In particular, whereas the examples in this article focused on human hair, the methods developed are readily applicable to the modeling of other hair-like patterns including, for example, fur, grass and feathers. Individual strands in fur typically have a simpler geometry than human hair and a higher degree of geometric coherency. Similarly, the geometry of the barbs of a feather emerging from its main shaft closely resembles that of a generalized helicoid. Several other potential directions are listed below.

7.2.1 Reconstructing with Adaptive Sampling

A proper hair sampling – more precisely the task of selecting which hairs are to be fit and where they are to be interpolated – is critical when reconstructing sparse hairstyles. For instance, a coarse hairstyle made of widely spaced hair tufts will not be reconstructed properly if too many sample points are placed in-between those tufts. An adaptive sampling based on the homogeneity of hair neighborhoods could improve the visual appearance of the hair as a whole, and naturally recover structures such as clumps and wisps.

7.2.2 Fitting Hair Volumes

The fitting approach presented in this thesis goes through all hairs, one by one. An extension of this work is to fit a volumetric neighborhood surrounding a hair at the same time. The volumetric nature of the generalized helicoid suggests such a task should be possible and perhaps even more desirable. Similar work carried out in Savadjiev et al. [76] points towards that direction.

7.2.3 Embedded Dynamics

Given the success of the piecewise helical super-helix representation for modeling hair dynamics [10, 8], a second area of research that could prove fruitful is the coupling of these ideas with the p-helicoid representation. Since the p-helicoid captures the geometry of the flow of hair strands in a volumetric neighborhood of a guide hair, it allows for sparsity and hence efficiency. It would be interesting to determine whether the physics of hair, such as the modeling of external forces including wind and contact, or internal properties such as torsion, bending and stretching, can be embedded directly into the p-helicoid parameterization.

7.2.4 Hairstyle Compression

Although the reconstruction of sparse hairstyles was investigated qualitatively, more work has to be done in order to extend this application in the context of hair compression. A parameterization using generalized helicoids could be used as a way to store a hair strand using less information than a listing of its vertices. This application could be of interest for large databases containing highly detailed hairstyles.

7.2.5 Detail Transfer

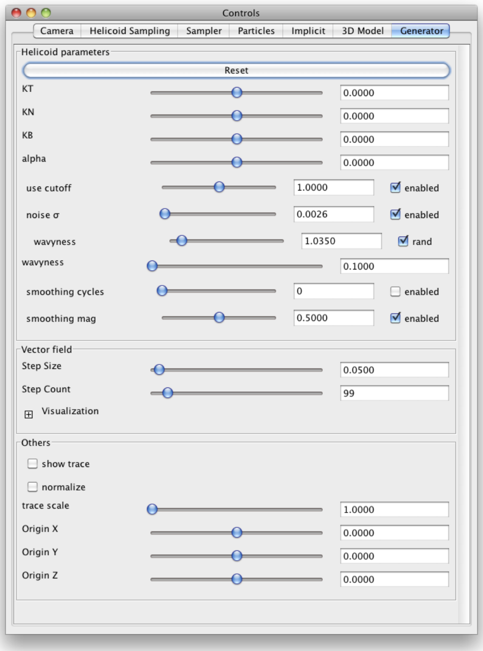
As in Wang et al. [82], it should be possible to use the generalized helicoid in the context of hair detail transfer. P-helicoids can serve as the skeleton for a variety of hairstyles, obtained by using waviness, noise, and possibly other kinds of offsets. Imagine for instance building a template for a general hairstyle, and then refining it depending on the type of hairstyle wanted (e.g. straight, wavy, or curly hair).

7.2.6 Curve Framing

The effect of the curve framing approach used in this thesis was not investigated thoroughly. Additional methods should be considered to find more efficient approaches for determining Frenet frames on discrete polylines [11], [41] and [7].

APPENDIX A
Implementation Details

The hair modeling framework was implemented in the Java programming language, under Mac OS X. OpenGL was used for real-time rendering. Blender was used for rendering more detailed depictions of hairstyles, using NURBS for hair strands. Figures A–1 and A–2 show a selection of snapshots from the hair modeling framework.

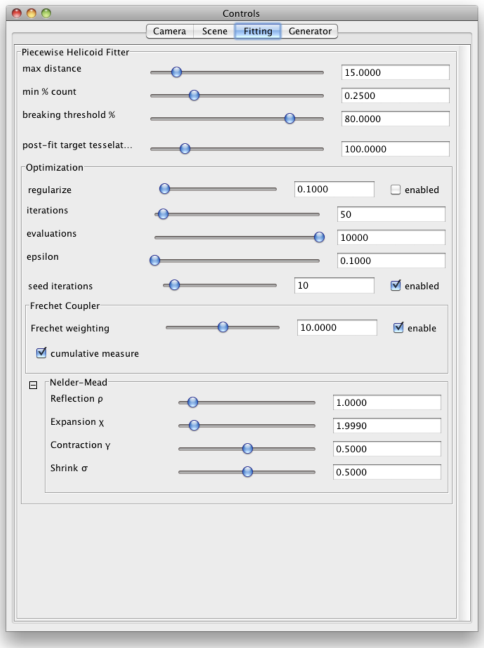


Helicoid parameters
KT 0.0000
KN 0.0000
KB 0.0000
alpha 0.0000
use cutoff 1.0000 [✓ enabled]
noise σ 0.0026 [✓ enabled]
wavyness 1.0350 [✓ rand]
wavyness 0.1000
smoothing cycles 0 ☐ enabled
smoothing mag 0.5000 [✓ enabled]

Vector field
Step Size 0.0500
Step Count 99

Others
☐ show trace
☐ normalize
trace scale 1.0000
Origin X 0.0000
Origin Y 0.0000
Origin Z 0.0000

(a) Generation



Piecewise Helicoid Fitter
max distance 15.0000
min % count 0.2500
breaking threshold % 80.0000
post-fit target tesselat... 100.0000

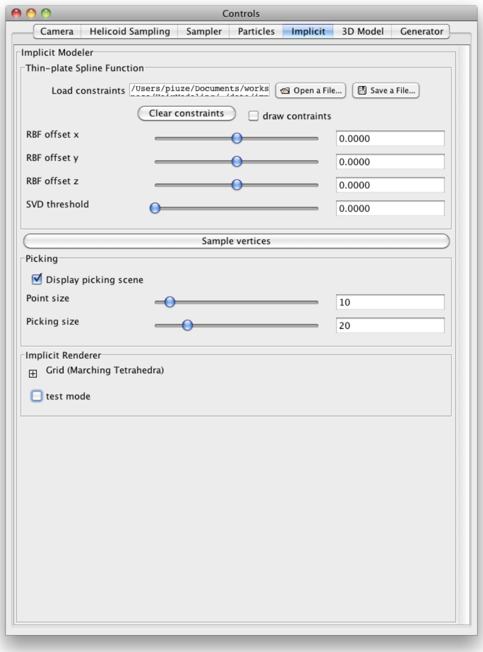
Optimization
regularize 0.1000 ☐ enabled
iterations 50
evaluations 10000
epsilon 0.1000
seed iterations 10 [✓ enabled]

Frechet Coupler
Frechet weighting 10.0000 [✓ enable]
[✓ cumulative measure]

Nelder-Mead
Reflection ρ 1.0000
Expansion χ 1.9990
Contraction γ 0.5000
Shrink σ 0.5000

(b) Fitting

Figure A–1: GUI and user-controllable parameters for the generation and fitting modules of the hair modeling framework.



Implicit Modeler

Thin-plate Spline Function

Load constraints /Users/piuze/Documents/works

[ ] draw contraints

RBF offset x 0.0000

RBF offset y 0.0000

RBF offset z 0.0000

SVD threshold 0.0000

Picking

[✓] Display picking scene

Point size 10

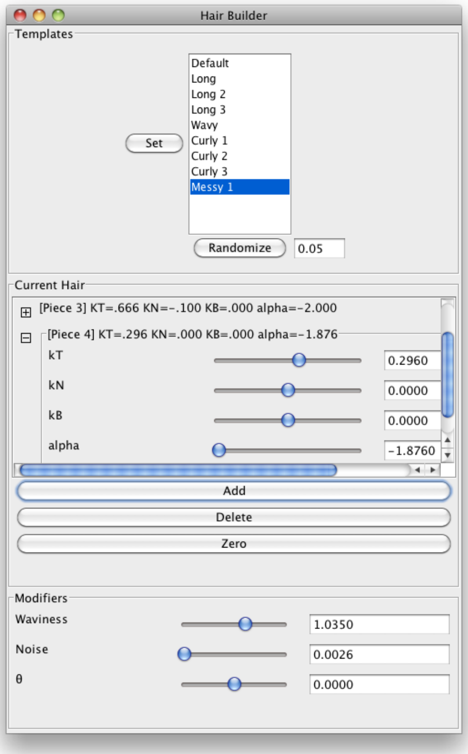
Picking size 20

Implicit Renderer

⊞ Grid (Marching Tetrahedra)

[ ] test mode

(a) Implicit surface



Templates

Default
Long
Long 2
Long 3
Wavy
Curly 1
Curly 2
Curly 3
Messy 1

Randomize 0.05

Current Hair

⊞ [Piece 3] KT=.666 KN=-.100 KB=.000 alpha=-2.000

⊟ [Piece 4] KT=.296 KN=.000 KB=.000 alpha=-1.876

kT 0.2960

kN 0.0000

kB 0.0000

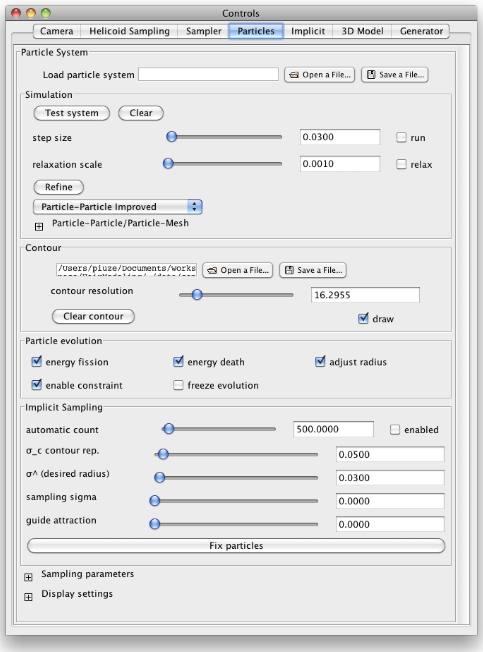
alpha -1.8760

Modifiers

Waviness 1.0350

Noise 0.0026

θ 0.0000



Particle System

Load particle system

Simulation

step size 0.0300 [ ] run

relaxation scale 0.0010 [ ] relax

Particle-Particle Improved

⊞ Particle-Particle/Particle-Mesh

Contour

/Users/piuze/Documents/works

contour resolution 16.2955

[✓] draw

Particle evolution

[✓] energy fission [✓] energy death [✓] adjust radius

[✓] enable constraint [ ] freeze evolution

Implicit Sampling

automatic count 500.0000 [ ] enabled

σ_c contour rep. 0.0500

σ^ (desired radius) 0.0300

sampling sigma 0.0000

guide attraction 0.0000

⊞ Sampling parameters

⊞ Display settings

(b) Hair builder (c) Sampling

Figure A–2: GUI and user-controllable parameters for the implicit surface fitting, hair builder, and particle sampling modules of the hair modeling framework.

APPENDIX B
Visualizing Implicit Surfaces

Implicit surfaces are harder to visualize than explicit ones. The difficulty resides in the continuous description of the surface, defined for all points in space. In this case, the level set of interest is the zero-crossing. In a brute-force manner, all points in space could be evaluated using the implicit function describing the implicit surface, and the ones that yield a value of 0 would be marked for rendering. Obviously, this is not an efficient way to go, and different methods have been designed for this purpose. One of the most well-known method for rendering implicit surfaces is the *marching cube* algorithm. This method is not described here. See [54] for more information. A similar method, the *marching tetrahedron*, was used. The method works by dividing the rendering space into a grid of tetrahedra, and evaluating the implicit function at the four corners of the cells composing that grid. The division of a cube into six tetrahedra is shown in Figure B–1. The value (positive, zero, negative) of each cell constitutes a signature, for which only one possibility exists, locally, as the planar approximation of the surface. The cell signature determines which of the cases listed in Figure B–2 applies.

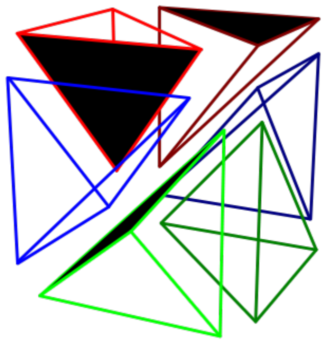


Figure B–1: Division of a cubic cell into six tetrahedra.

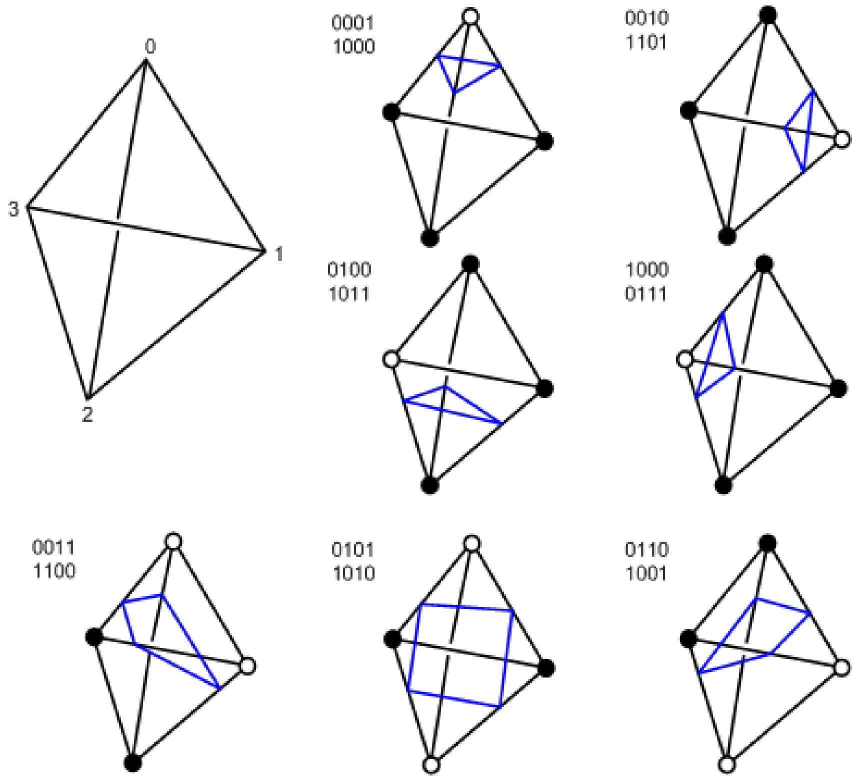


Figure B–2: Marching tetrahedra test cases. A binary 0/1 value is assigned at the corners of each tetrahedron depending on whether the evaluation of the implicit function yields a negative or positive value, respectively. By default, evaluations to zero (within one machine-epsilon) are assigned the smallest non-zero positive value.

APPENDIX C
Learning with Resilient Propagation

Given an error function E, each neural weight w_{ij} is updated according to an adaptive individual update-value Δ_{ij} that depends on the gradient of the error function over the previous time steps. The update rule for the weights w_{ij} is thus given by:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ * \Delta_{ij}^{(t-1)} & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} > 0; \\ \eta^- * \Delta_{ij}^{(t-1)} & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} < 0; \\ \Delta_{ij}^{(t-1)} & \text{else.} \end{cases} \tag{C.1}$$

$$w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} > 0; \\ +\Delta_{ij}^{(t)} & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} < 0; \\ 0 & \text{else.} \end{cases} \tag{C.2}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \tag{C.3}$$

where $0 < \eta^+ < 1 < \eta^+$. If the partial derivative changes sign, in the case where the minimum was missed, a backtracking weight step is carried out and the update rule is reverted:

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)}, \text{ if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} * \frac{\partial E^{(t)}}{\partial w_{ij}} > 0. \tag{C.4}$$

To prevent the same backtracking in the following time step, the update value of the weights should stay the same, so the following is enforced:

$$\frac{\partial E^{(t-1)}}{\partial w_{ij}} = 0. \tag{C.5}$$

The original article ([72]) suggests the use of $\eta^+ = 1.2$ and $\eta^- = 0.5$.

References

[1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5(1):75–91, 1995.

[2] J.S. Alter. Hair generation and other natural phenomena with surface derived control volumes in computer graphics and animation, April 13 2004. US Patent 6,720,962.

[3] S.B. Andersson. Discrete approximations to continuous curves. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2546–2551. Citeseer, 2006.

[4] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, page 120. ACM, 1992.

[5] Y. Bando, B.Y. Chen, and T. Nishita. Animating hair with loosely connected particles. In *Computer Graphics Forum*, volume 22, pages 411–418. Citeseer, 2003.

[6] O. Ben-Shahar and S.W. Zucker. The perceptual organization of texture flow: A contextual inference approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):401–417, 2003.

[7] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. Discrete elastic rods. In *ACM SIGGRAPH ASIA 2008 courses*, page 14. ACM, 2008.

[8] F. Bertails. Linear time super-helices. In *Computer Graphics Forum*, volume 28, pages 417–426. John Wiley & Sons, 2009.

[9] F. Bertails, S. Hadap, M.P. Cani, M. Lin, T.Y. Kim, S. Marschner, K. Ward, and Z. Kačić-Alesić. Realistic hair simulation: animation and rendering. In *ACM SIGGRAPH 2008 classes*, page 89. ACM, 2008.

[10] Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean-Luc Lévêque. Super-helices for predicting the dynamics of natural hair. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1180–1187, New York, NY, USA, 2006. ACM.

[11] J. Bloomenthal. Calculation of reference frames along a space curve. *Graphics gems*, 1:44–54, 1990.

[12] U. Bonanni and P. Kmoch. Virtual hair handle: A model for haptic hairstyling. In *Proc. Eurographics*, pages 135–138, 2008.

[13] U. Bonanni, P. Kmoch, and N. Magnenat-Thalmann. Interaction Metaphors for Modeling Hair using Haptic Interfaces. *International Journal of CAD/CAM*, 9(1), 2010.

[14] E. Catalan. Sur les surfaces réglées dont l'aire est un minimum. *Journal de Mathématiques (1)*, 7:203–211, 1842.

[15] B.Y. Chen. A report on submanifolds of finite type. *Soochow Journal of Mathematics*, 22(2):117–337, 1996.

[16] L.H. Chen, S. Saeyor, H. Dohi, and M. Ishizuka. A system of 3d hair style synthesis based on the wisp model. *The Visual Computer*, 15(4):159–170, 1999.

[17] B. Choe and H.S. Ko. A statistical wisp model and pseudophysical approaches for interactive hairstyle generation. *IEEE Transactions on Visualization and Computer Graphics*, 11(2), 2005.

[18] Byoungwon Choe, Min Gyu Choi, and Hyeong-Seok Ko. Simulating complex hair with robust collision handling. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 153–160, New York, NY, USA, 2005. ACM.

[19] C. Csuri, R. Hackathorn, R. Parent, W. Carlson, and M. Howard. Towards an interactive high visual complexity animation system. In *Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, pages 289–299. ACM, 1979.

[20] A. Daldegan, N.M. Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating and rendering hair. In *Computer Graphics Forum*, volume 12, pages 211–221. John Wiley & Sons, 1993.

[21] E.B. Dam, M. Koch, and M. Lillholm. Quaternions, interpolation and animation. Technical report, Institute of Computer Science, University of Copenhagen, 1998.

[22] T. Eiter and H. Mannila. Computing discrete Fréchet distance. *Technische Universitat Wien Technical Report CD-TR*, 94:64, 1994.

[23] J.D. Foley, A. Van Dam, S.K. Feiner, J.F. Hughes, and R.L. Phillips. *Introduction to computer graphics.* Addison-Wesley Reading, MA, 1994.

[24] R. Franke. Scattered Data Interpolation: Tests of Some Methods. *Mathematics of Computation*, 38(157):181–200, 1982.

[25] T.A. Funkhouser and C.H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 247–254. ACM, 1993.

[26] Carlos A. Furuti. Map projections. http://www.progonos.com/furuti/mapproj/. last checked august 2010.

[27] D.B. Goldman. Fake fur rendering. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 127–134. ACM Press/Addison-Wesley Publishing Co., 1997.

[28] S. Grabli, F.X. Sillion, S.R. Marschner, and J.E. Lengyel. Image-based hair capture by inverse lighting. *Graphics Interface 2002: proceedings: Calgary, Alberta, 27-29 May, 2002*, page 51, 2002.

[29] S. Hadap. Oriented strands: dynamics of stiff multi-body system. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, page 100. Eurographics Association, 2006.

[30] S. Hadap and N. Magnenat-Thalmann. Interactive hair styler based on fluid flow. *Computer Animation and Simulation'00*, pages 87–100, 2000.

[31] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3):329–338, 2001.

[32] Sunil Hadap, Marie-Paule Cani, Ming Lin, Tae-Yong Kim, Florence Bertails, Steve Marschner, Kelly Ward, and Zoran Kačić-Alesić. Strands and hair: modeling, animation, and rendering. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 1–150, New York, NY, USA, 2007. ACM.

[33] A.J. Hanson and H. Ma. Quaternion frame approach to streamline visualization. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):164–174, 1995.

[34] B. Hernandez and I. Rudomin. Hair paint. In *Computer Graphics International, 2004. Proceedings*, pages 578–581, 2004.

[35] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[36] D. Ivanov, V. Lempitsky, A. Shokurov, A. Khropov, and Y. Kuzmin. Creating Personalized Head Models from Image Series. In *Proc. of the International Conference on Computer Graphics & Vision GraphiCon*, 2003.

[37] W. Jakob, J.T. Moon, and S. Marschner. Capturing hair assemblies fiber by fiber. In *ACM SIGGRAPH Asia 2009 papers*, pages 1–9. ACM, 2009.

[38] JT Kajiya and TL Kay. Rendering fur with three dimensional textures. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 271–280. ACM New York, NY, USA, 1989.

[39] Tae-Yong Kim and Ulrich Neumann. Interactive multiresolution hair modeling and editing. *ACM Trans. Graph.*, 21(3):620–629, 2002.

[40] T.Y. Kim and U. Neumann. A thin shell volume for modeling human hair. In *Computer Animation 2000*, pages 104–111, 2000.

[41] F. Klok. Two moving coordinate frames for sweeping along a 3D trajectory. *Computer Aided Geometric Design*, 3(3):217–229, 1986.

[42] P. Kmoch, U. Bonanni, and N. Magnenat-Thalmann. Hair simulation model for real-time environments. In *Computer Graphics International*, pages 5–12. ACM, 2009.

[43] CK Koh and Z. Huang. Real-time animation of human hair modeled in strips. In *Computer animation and simulation 2000: proceedings of the Eurographics Workshop in Interlaken, Switzerland, August 21-22, 2000*, page 101. Springer Verlag Wien, 2000.

[44] W. Kong and M. Nakajima. Visible volume buffer for efficient hair expression and shadow generation. *ca*, page 58, 1999.

[45] W. Kong, H. Takahashi, and M. Nakajima. Generation of 3d hair model from multiple pictures. In *Proceedings of Multimedia Modeling*, pages 183–196, 1997.

[46] W. Kühnel and B. Hunt. *Differential geometry: curves-surfaces-manifolds*. AMERICAN MATHEMATICA, 2006.

[47] A.M. LeBlanc, R. Turner, and D. Thalmann. Rendering hair using pixel blending and shadow buffers. *The Journal of Visualization and Computer Animation*, 2(3):92–97, 1991.

[48] D.W. Lee and H.S. Ko. Natural hairstyle modeling and animation. *Graphical Models*, 63(2):67–85, 2001.

[49] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 55–62. ACM, 1995.

[50] J. Lengyel. Real-time fur. In *Eurographics Rendering Workshop*, pages 243–256, 2000.

[51] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 227–232. ACM New York, NY, USA, 2001.

[52] T. Lewiner, J.D. Gomes, H. Lopes, and M. Craizer. Curvature and torsion estimators based on parametric curve fitting. *Computers & Graphics*, 29(5):641–655, 2005.

[53] GH Liu, YS Wong, YF Zhang, and HT Loh. Adaptive fairing of digitized point data with discrete curvature. *Computer-Aided Design*, 34(4):309–320, 2002.

[54] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, page 169. ACM, 1987.

[55] N. Magnenat-Thalmann, S. Hadap, and P. Kalra. State of the art in hair simulation. In *International Workshop on Human Modeling and Animation*, pages 3–9, 2000.

[56] N. Magnenat-Thalmann, M. Montagnol, R. Gupta, and P. Volino. Interactive virtual hair-dressing room. *Computer-Aided Design & Applications*, 3(5):535–546, 2006.

[57] G.S.P. Miller. From wire-frames to furry animals. In *Proceedings on Graphics interface'88*, pages 138–145. Canadian Information Processing Society, 1989.

[58] M. Nakajima, S. Saruta, and H. Takahashi. Hair image generating algorithm using fractional hair model. *Signal Processing: Image Communication*, 9(3):267–273, 1997.

[59] JA Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308, 1965.

[60] A.C. Neville. *Biology of fibrous composites: development beyond the cell membrane.* Cambridge Univ Pr, 1993.

[61] F. Neyret. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):55, 1998.

[62] B. O'Neill. *Elementary differential geometry.* Academic Pr, 1997.

[63] S. Paris, H.M. Briceño, and F.X. Sillion. Capture of hair geometry from multiple images. *ACM Transactions on Graphics (TOG)*, 23(3):712–719, 2004.

[64] S. Paris, W. Chang, O.I. Kozhushnyan, W. Jarosz, W. Matusik, M. Zwicker, and F. Durand. Hair photobooth: geometric and photometric acquisition of real hairstyles. In *ACM SIGGRAPH 2008 papers*, page 30. ACM, 2008.

[65] F.I. Parke and K. Waters. *Computer facial animation.* AK Peters Ltd, 2008.

[66] K. Perlin and EM Hoffert. Hypertexture. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, page 262. ACM, 1989.

[67] E. Plante, M.P. Cani, and P. Poulin. A layered wisp model for simulating interactions inside long hair. In *Computer Animation and Simulation 2001: Proceedings of the Eurographics Workshop in Manchester, UK, September 2-3, 2001*, page 139. Springer Verlag Wien, 2001.

[68] Proceedings of EGSBM. *A Sketching Interface for Modeling and Editing Hairstyles*, 2005.

[69] H. Reckziegel. Mathematical Models from the Collections of Universities and Museums. *Munich, Germany: Braunschweig*, 1986.

[70] W.T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics (TOG)*, 2(2):108, 1983.

[71] W.T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *ACM Siggraph Computer Graphics*, 19(3):313–322, 1985.

[72] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE international conference on neural networks*, volume 1993, pages 586–591. San Francisco: IEEE, 1993.

[73] C.R. Robbins. *Chemical and physical behavior of human hair.* Springer Verlag, 2002.

[74] R.E. Rosenblum, W.E. Carlson, and E. Tripp. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *The Journal of Visualization and Computer Animation*, 2(4):141–148, 1991.

[75] N.S. Sapidis. *Designing fair curves and surfaces: Shape quality in geometric modeling and computer-aided design.* Society for Industrial Mathematics, 1994.

[76] P. Savadjiev, S.W. Zucker, and K. Siddiqi. On the differential geometry of 3d flow patterns: Generalized helicoids and diffusion mri analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct. 2007.

[77] Peter Savadjiev. *Perceptual Organization in Diffusion MRI: Curves and Streamline Flows.* PhD thesis, McGill University, 2008.

[78] F. Scarselli and A. Chung Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural Networks*, 11(1):15–37, 1998.

[79] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, page 414. ACM, 1998.

[80] N.M. Thalmann, S. Carion, M. Courchesne, P. Volino, and Y. Wu. Virtual clothes, hair and skin for beautiful top models. In *Computer Graphics International*, pages 132–141. Citeseer, 1996.

[81] G. Turk and J.F. O'brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG)*, 21(4):855–873, 2002.

[82] L. Wang, Y. Yu, K. Zhou, and B. Guo. Example-based hair geometry synthesis. In *ACM SIGGRAPH 2009 papers*, page 56. ACM, 2009.

[83] T. Wang and X.D. Yang. Hair design based on the hierarchical cluster hair model. In *Geometric modeling*, page 359. Kluwer Academic Publishers, 2004.

[84] K. Ward, F. Bertails, T.Y. Kim, S.R. Marschner, M.P. Cani, and M.C. Lin. A survey on hair modeling: Styling, simulation, and rendering. *IEEE transactions on visualization and computer graphics*, pages 213–234, 2007.

[85] K. Ward, M.C. Lin, J. Lee, S. Fisher, and D. Macri. Modeling hair using level-of-detail representations. In *Computer Animation and Social Agents, 2003. 16th International Conference on*, pages 41–47. IEEE, Published by the IEEE Computer Society, 2003.

[86] Y. Watanabe and Y. Suenaga. Drawing human hair using the wisp model. *The Visual Computer*, 7(2):97–103, 1991.

[87] Y. Wei, E. Ofek, L. Quan, and H.Y. Shum. Modeling hair from multiple views. In *ACM SIGGRAPH 2005 Papers*, page 820. ACM, 2005.

[88] A.P. Witkin and P.S. Heckbert. Using particles to sample and control implicit surfaces. In *ACM SIGGRAPH 2005 Courses*, page 260. ACM, 2005.

[89] G. Xu and G. Wang. Control mesh representation of a class of minimal surfaces. *Journal of Zhejiang University-Science A*, 7(9):1544–1549, 2006.

[90] Z. Xu and X.D. Yang. V-hairstudio: an interactive tool for hair design. *IEEE Computer Graphics and Applications*, pages 36–43, 2001.

[91] T. Yamaguchi, B. Wilburn, and E. Ofek. Video-Based Modeling of Dynamic Hair. *Advances in Image and Video Technology*, pages 585–596, 2009.

[92] X.D. Yang, Z. Xu, J. Yang, and T. Wang. The cluster hair model. *Graphical Models*, 62(2):85–103, 2000.

[93] Yizhou Yu. Modeling realistic virtual hairstyles. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, page 295, Washington, DC, USA, 2001. IEEE Computer Society.

[94] C. Yuksel, S. Schaefer, and J. Keyser. Hair meshes. *ACM Transactions on Graphics (TOG)*, 28(5):1–7, 2009.