Cómputo y Simulación del Cabello Humano

Roger Steven Ramírez Espejo, Álvaro José Vera Soto

22 de julio de 2010

Índice general

Capítulo 1

Resumen

En la actualidad, la computación gráfica ocupa un papel preponderante en las ciencias de la computación. A saber, películas animadas, juegos de video y efectos artificiales son parte del vocabulario común para alguien que trabaja en cine, televisión ó entretenimiento en el siglo XXI. ¿Pero cómo se logra todo esto? Las matemáticas y la gran velocidad combinada con una inmensa capacidad de computo que poseen los computadores actuales, han combinado esfuerzos para encontrar soluciones que permiten virtualizar el comportamiento natural que vemos hoy en día, en cualquier parte.

En el transcurso de este documento se describen características de las estructuras delgadas y flexibles, como lo es el cabello del ser humano, los que en su estructura original fueron caraterizadas por una configuración curva sin ningún tipo de deformación. El modelamiento de estas superficies tres dimensionales con las que nos encontramos a diario, correspondían anteriormente a modelos bastante complejos que eran difíciles de entender y simular. Por otro lado el proyecto aunque no es sencillo, logra dar forma y mostrar de una manarera mas clara y concisa dicho modelo que comprende la geometría convencional y el cálculo de variables e integrales, simplemente usando lo ya conocido en el ámbito de la Ingeniería, de esta forma rápidamente se obtendrá un simulador del cabello humano, siendo este capáz de mostrar el movimiento causado por diversos entes externos (viento , un simple toque, etc).

Por otra parte, la combinación de tecnologías existentes como `Java`[1], `Processing`[2], son utilizadas para realizar la correspondiente implementación del simulador. Usando el core de java y la facilidad de implementar métodos especiales que posee processing ayudan a generar la ejecución de la virtualización correspondiente al problema planteado.

Durante cuatro capítulos se recorre y explica detalladamente el problema y la solución que plantea la simulación mediante una computadora, de las membranas flexibles del cabello humano. En el capítulo I se hace una recopilación de conceptos necesarios que son aplicados para solucionar el problema planteado; entre ellos tendremos la física de las barras propuesta por Kirchhof, energías elásticas, marco paralelo de transporte de Bishop, modelos discretizables y sus respectivas ecuaciones para la implementación exitosa del problema planteado. El capítulo II presenta toda la teoría matemática sobre la generación de el cabello humano basado en ecuaciones representadas en derivadas, integrales y ecuaciones relacionadas con el álgebra. El capítulo III presenta la arquitectura especial del simulador que abarca los dos capítulos anteriores y realiza la implementación de los métodos anteriores. Los resultados y las conclusiones obtenidas se presentan en el capítulo IV que tomará los resultados obtenidos con la respectiva implementación del programa.

[1] Lenguaje de programación orientado a objetos desarrollado por la empresa Sun Microsystems a principios de los años 90, usa una sintaxis similar a los lenguajes de programación C y C++, con la característica que elimina herramientas de bajo nivel que suelen conducir a errores en apuntadores a posiciones de memoria.

[2] Lenguaje y entorno de programación de código abierto basado en java, sirve para producción de proyectos de multimedia y diseño digital.

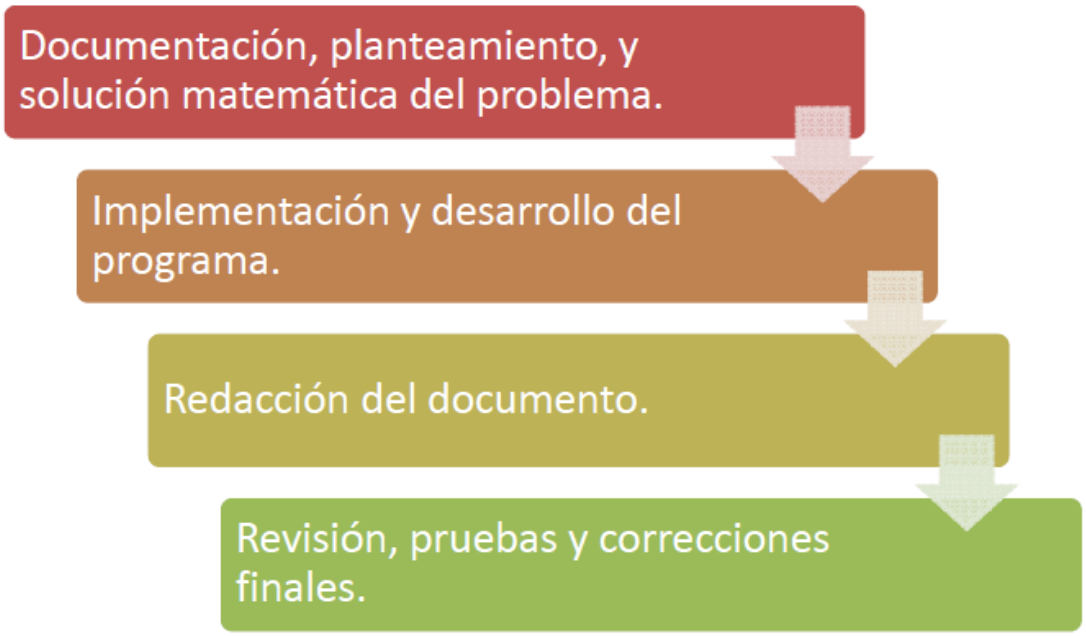


Figura 1.1: Etapas de ejecución del proyecto

1.1. Objetivo General

Implementar algoritmos para la simulación de cabello humano bajo el esquema de programación orientada a objetos.

1.2. Objetivos específicos

- Utilizar geometría diferencial para modelar un cabello como barra de Kirchhoff.
- Incluir fuerzas externas para simular la física del movimiento de cabello siguiendo el modelo de Terzopoulos.
- Implementar el modelo de Kmoch para la dinámica de cabelleras humanas.
- Producir una librería de simulación de cabello humano utilizando Processing y Java.

1.3. Flujo del proyecto

El proyecto fue divido en 4 etapas secuenciales cada una imediatamente dependiente de la anterior (ver figura 1.1)

1.3.1. Documentación, planteamiento y solución matemática del problema

En esta etapa se comprende el modelo matemático que definen las soluciones al sistema, es decir a través del cálculo variacional se darán las respuestas al problema planteado. Esta misma es muy importante porque a través del modelo matemático se dará solución al proyecto.

1.3.2. Implementación y desarrollo del programa

Un buen diseño define una buena aplicación, por esta razón en esta etapa se realizará un análisis de las soluciones y se definirá un esquema (realizado a través de diagramas UML) que dará un oriente claro a la implementación computacional del aplicativo.

Adicionalmente, se tomará el modelo anteriormente desarrollado para crear una correcta estructura funcional del programa.

1.3.3. Redacción del documento

Tomando como base esencial el modelo matemático propuesto además del desarrollo funcional del programa se pretende realizar la correspondiente etapa de documentación y formalización de los teoremas resueltos y demostrados en las etapas anteriores.

Además se dará un relato completo del desarrollo del proyecto, así como, experiencias vividas, pendientes y una sección de apéndices que mostrarán extras interesantes relacionados con el proyecto.

1.3.4. Revisión, pruebas y correcciones finales

Lo ideal de esta etapa es la de realizar profundas revisiones del proyecto para refinar y pulir detalles que no se han tenido en cuenta anteriormente para que de esta manera se establezca un buen estándar de presentación del proyecto.

Capítulo 2

Preliminares

A lo largo de este capítulo se establecerán todos los conceptos básicos que se necesitan conocer y usar para plantear la solución al problema de simular y/o virtualizar el cómputo del cabello humano que son el objetivo común del proyecto.

2.1. Introducción

Los métodos que formulan y representan formas instantáneas de los objetos son el centro del modelamiento por computación gráfica. Estos métodos han sido particularmente útiles, para modelar objetos rígidos los cuales su superfices no cambian durante el tiempo[18]. Este proyecto desarrolla un enfoque al modelamiento el cual incorpora dinámicas de los materiales delgados y flexibles que están sujetos a cambios permanentes, dentro de modelos geométricos los cuales han sido usados tradicionalmente. Modelos basados en teorías de elasticidad, torsión y agentes externos son convenientes para representar la forma y movimiento del cabello humano cuando este sufre algún tipo de deformación respecto a su estado original.

La animación realista y presentación del cabello es parte crucial en la virtualización del ser humano. En investigaciones realizadas anteriormente se determinó que la simulación de barras elasticas, introducen una mejora técnica que utiliza propiedades específicas de cada unos de los filamentos del pelo humano mejorando así su rendimiento y simulación[13]. Aunque se tiene en cuenta todo esto , no es del todo fácil la representación del cabello humano de manera computable, pues la simulación de este se ve sujeta a múltiples agentes como lo son, la torsión, el patrón crecimiento del cabello, teniendo en cuenta que éste puede ser un crecimiento uniforme o como lo es en la mayoría de los casos de crecimiento no uniforme; adicionalmente a esto interactúa también la gran cantidad de cabello que puede tener una personas estimándose esta sobre los 100.000 cabellos invividuales[4] en donde en cada fibra de cabello interactúan todo tipo de simulaciones físicas.Por otra parte, estucturas delgadas y rígidas (ej., uñas de los dedos, botellas de vidrio), son objetos que son naturalmente curvos y no requieren de un gran estudio, ya que estos no sufren ningún tipo de deformación y/o adaptación a un entorno determinado; más aún pensando en deformaciones plásticas (ej., sombreros, latas, cajas de cartón, chasis de automóviles, botellas de detergentes), son membranas delgadas que no pueden ser modeladas usando formulaciones como las utilizadas en el modelamiento y cómputo del cabello humano.

Actualmente se tienen, o mejor, se trabajan dos categorías generales para la representación del cabello; la primera consiste en trabajar hebra por hebra de cabello independiente como un todo logrando de esta manera expresarla como una cadena de hebras rígidas, la segunda forma es de volumen y esta es representada y simulada utilizando implementaciones de mecánica continua sobre partículas suavizadas.

Fundamental en el estudio de la simulción de pelo es la teoría de barras de Kirchoff[25, 4]. Estos modelos se basan en la construcción de operaciones sobre curvas paramétricas tanto vectoriales como escalares.

En el enfoque de este documento una curva paramétrica es una inyección de un intervalo cerrado de $\mathbb{R}$ en un espacio euclideo $\mathbb{R}^n$[23]. A saber,

Definición 2.1.1 (CurvaParametrica) *Sea $[a,b] \subset \mathbb{R}$ un intervalo cerrado. Una* curva paramétrica *con dominio $[a,b]$ es una función vectorial $\mathbf{c} : [a,b] \to \mathbb{R}^n$ de la forma:*

$$\mathbf{c}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} \tag{2.1}$$

para $t \in [a,b]$. Las funciones $x_i : [a,b] \to \mathbb{R}$ se conocen como las componentes *de $\mathbf{c}$.*

Las curvas paramétricas constituyen el objeto geométrico más simple después del punto. Dado que una curva paramétrica es una función, su diferenciabilidad y continuidad dependen de sus funciones componente.

Teorema 2.1.2 *Una curva paramétrica $\mathbf{c}$ dada en la forma 2.1 se dice* continua *si cada componente x_i es continua.*

Teorema 2.1.3 *Una curva paramétrica $\mathbf{c}$ dada en la forma 2.1 se dice* diferenciable *n-veces si cada una de sus componentes es al menos n-veces diferenciable.*

2.2. Campos vectoriales

Los campos vectoriales son vistos desde la perspectiva de la física como una ayuda para el modelamiento de la velocidad, las direcciones que toman los líquidos através del espacio, o la intensidad y dirección de algún tipo de fuerza; en nuestro caso se ve reflejada en el moviemeinto del cabello humano y todas las fuerzas que actúan sobre éste, como lo son las velocidades en la dirección en que éste se esta moviendo, la gravedad que se ejerce sobre cada hebra de cabello, entre otras.

Es importante que se entienda con claridad que es un campo vectorial, antes de adentrarnos en las nociones de rapidez, torsión y curvatura, ya que esta nos da las bases para poder lograr entender matemáticamente y físicamnente, el comportamiento que toma el cabello humano obedeciendo a actuadores externos y así poder comprender como funcióna y por qué se dan cambios repentinos en su comportamiento.

Definición 2.2.1 (Campo vectorial en $\mathbb{R}^n$) *Un campo vectorial es una función $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ que a cada punto $\mathbf{x} \in \mathbb{R}^n$ le asigna un vector $\mathbf{F}(\mathbf{x})$. A saber un campo vectorial está descrito por una función de la forma:*

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} F_1(\mathbf{x}) \\ \vdots \\ F_n(\mathbf{x}) \end{pmatrix} \tag{2.2}$$

donde las funciones F_i son escalares y se conocen como las componentes *del campo $\mathbf{F}$.*

Si bien, entre la definición de un campo vectorial y una curva paramétrica se habla de componentes, una curva paramétrica asigna a una posición en el espacio un vector que indica direccionalidad. Los campos vectoriales son útiles en física y matemáticas para describir fuerzas, tensiones y son de utilidad en la descripción del trabajo y la energía.

La noción de campo vectorial se puede restringir a una curva. A saber, si un campo vectorial asigna vectores de dirección a cada punto del plano, se puede hablar de la restricción de éstos a una curva paramétrica ó bien de un *campo vectorial sobre una curva paramétrica*.

Definición 2.2.2 (Campo vectorial sobre una curva paramétrica) *Sea* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *una curva paramétrica continua y sea* $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$. *La restricción de* $\mathbf{F}$, $\mathbf{F_c}$ *a la curva* $\mathbf{c}$ *constitye un campo vectorial sobre la curva* $\mathbf{c}$. *A saber:*

$$\mathbf{F_c}(t) = \mathbf{F} \circ \mathbf{c}(t) \tag{2.3}$$

2.3. Propiedades y mediciones en curvas paramétricas

Con las propiedade s y mediciones de la curva paramétrica estamos introduciéndonos más a fondo en el tema que nos atañe respecto a la virtualización del cabello humano, ya que logramos identificar componentes importantes que se estudiaran más adelante y con más profundidad como lo son las medidas vectoriales que serán utilizadas en el siguiente capítulo, para la implementación y utilización de *Torsión* y *Flexión* y las medidas escalares, representadas pór vectores, dándonos la información necesaria sobre las fuerzas y el direccionamiento que actúan sobre el objeto que estamos trabajando.

2.3.1. Velocidad

Una curva paramétrica se utiliza para representar la trayectoria de una partícula bajo fuerzas ó causas variadas. Dentro de este enfoque, la *velocidad* de una partícula se puede cuantificar a partir de las derivadas de la curva paramétrica describiendo su trayectoria ó bien a partir de su *vector tangente*.

Definición 2.3.1 (Vector tangente a una curva) *Sea* $\mathbf{c} : [a, b] \to \mathbb{R}$ *una curva paramétrica con componentes al menos una vez diferenciables. El* vector tangente *a* $\mathbf{c}$, $\mathbf{c}'$ *es la función definida por:*

$$\mathbf{c}'(t) = \begin{pmatrix} x_1'(t) \\ x_2'(t) \\ \vdots \\ x_n'(t) \end{pmatrix} \tag{2.4}$$

el vector tangente a la curva $\mathbf{c}$ *se conoce también como el* vector de velocidad *de la curva.*

Si bien una curva paramétrica sirve en un contexto físico para describir el movimiento de una partícula en el espacio o en el tiempo, una curva paramétrica es en sí misma un objeto geométrico, susceptible de análisis mediante métodos de cálculo[23]. En particular, la *longitud* de una curva paramétrica se puede estimar mediante la longitud de su vector tangente, por medio de la siguiente definición:

Definición 2.3.2 (Longitud de Arco) *Sea* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *una curva paramétrica continua y al menos una vez diferenciable. La* longitud *ó* longitud de arco *de la curva* $\mathbf{c}$, $\mathcal{L}(c)$ *está dada por:*

$$\mathcal{L}(\mathbf{c}) = \int_a^b ||\mathbf{c}'(t)||dt \tag{2.5}$$

Nótese que la longitud de arco de una curva depende intrínsecamente del dominio de la curva. De este modo, una curva con dominio $[a, b]$ ó con dominio $[c, d]$ pueden tener longitudes distintas a pesar de que tengan la misma gráfica en el plano. Con objeto de superar esta dualidadd, se propone la *parametrización por longitud de arco* de una curva. A saber, si la longitud de una curva $\mathbf{c}$ con dominio $[a, b]$ está dada por $L = \mathcal{L}(\mathbf{c})$, se puede construir una reparametrización de $\hat{\mathbf{c}}$ de modo tal que $\hat{\mathbf{c}}$ tiene la misma gráfica que $\mathbf{c}$ pero el dominio de $\hat{\mathbf{c}}$ es el intervalo $[0, L]$[17].

Definición 2.3.3 (Curva parametrizada por longitud de arco) *Si* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *es una curva paramétrica continua y al menos una vez diferenciable, con longitud* $L = \mathcal{L}(\mathbf{c})$, *existe una función* $D : [0, L] \to [a, b]$ *y una curva paramétrica* $\hat{\mathbf{c}}$ *tal que*

$$\mathbf{c} = \hat{\mathbf{c}} \circ D \tag{2.6}$$

a la función $\hat{\mathbf{c}}$ *se conoce como la* parametrización de $\mathbf{c}$ por longitud de arco.

En el sentido geométrico de las curvas paramétricas, la reparametrización por longitud de arco en una curva permite imaginarse a la curva como si fuera una línea recta que es deformada en el espacio para ocupar una forma específica: la gráfica de la curva[17].

2.3.2. Curvatura

Teniendo en cuenta que el concepto de curvatura en una superficie es la que mide la velocidad en la que la curva abandona el plano tangente a esta, en un d eterminado punto es importante mencionar que:

Definición 2.3.4 (Vector de Curvatura) *Sea* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *una curva paramétrica con componentes al menos dos veces diferenciable . El* vector de curvatura *de la curva* $\mathbf{c}$ *es el vector* $\mathbf{c}''$ *definido por:*

$$\mathbf{c}''(t) = \begin{pmatrix} x_1''(t) \\ \vdots \\ x_n''(t) \end{pmatrix} \tag{2.7}$$

La curvatura de una curva paramétrica es una generalización del concepto de *concavidad* para funciones de una variable. En general, la dirección del vector de curvatura define un círculo conocido como el *círculo osculante*[17]; localmente, la curva se puede ver como una sección o trozo del círculo osculante, pues el vector de curvatura apunta siempre hacia el centro de éste.

Al mismo tiempo, se puede definir la *curvatura* $\kappa(t)$ de una curva paramétrica por medio de la siguiente definición:

Definición 2.3.5 (Curvatura de una curva paramétrica) *Si* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *la* curvatura *de una curva paramétrica está definida por la función* $\kappa(t)$ *dada por:*

$$\kappa(t) = ||\mathbf{c}''(t)|| \tag{2.8}$$

El *radio de curvatura* de una curva paramétrica está definido como el recíproco de la función κ y corresponde con el radio R del círculo osculante:

$$R = \frac{1}{\kappa} \tag{2.9}$$

como corolario a estas definiciones es importante decir que una curva que es *plana* (recta) tiene curvatura cero y un radio de curvatura infinito.

2.3.3. Torsión

Una curva paramétrica describe el movimiento de una partícula en el espacio y constituye un objeto geométrico de dimensión 1 en el sentido de la geometría diferencial[17]. Si el vector de velocidad de la curva describe la rapidez de una partícula siguiendo la trayectoria de ésta y el vector de curvatura describe qué tanto difiere ésta de un plano, la *torsión* describe la manera como una partícula rotaría, de seguir la trayectoria de la curva.

Definición 2.3.6 (Vector de Torsión) *Sea* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *una curva paramétrica al menos dos veces diferenciable. Si* $\mathbf{c}'(t)$ *y* $\mathbf{c}''(t)$ *identifican los vectores de velocidad y curvatura, se define la* torsión $\mathbf{B}$ *como el vector*

$$\mathbf{B}(t) = \mathbf{c}'(t) \times \mathbf{c}''(t) \tag{2.10}$$

Es importante notar que existe una relación geométrica entre los vectores de velocidad, curvatura y torsión. Del mismo modo que la primera y segunda derivadas de una función identifican su velocidad de cambio y su concavidad, la velocidad y la curvatura miden el cambio en el espacio de la curva como objeto geométrico. De la misma manera que a partir del vector de curvatura se define la curvatura de una curva, la *torsión* de una curva está definida como la norma de su vector de torsión:

Definición 2.3.7 (Torsión de una curva paramétrica) *Si* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *es una curva paramétrica continua, al menos dos veces diferenciable, su* torsión *está definida por la función* $\tau(t)$ *dada por:*

$$\tau(t) = ||\mathbf{B}(t)|| \tag{2.11}$$

Existen relaciones fuertes entre la velocidad, la curvatura y la torsión de una curva paramétrica. Estas relaciones se condensan en las conocidas ecuaciones de *Frenet-Serret* que relacionan las tasas de cambio espaciales de los vectores de velocidad, curvatura y torsión y permiten el estudio y clasificación de las curvas paramétricas.

2.3.4. Formulas de Frenet

Uno de los puntos claves dentro de la virtualización y renderización del cabello humano tiene que ver con la torsión que éste sufre por diversos entes externos; en esta sección entraremos a analizar su comportamiento y las medidas matemáticas que tiene éste en una curva $\mathbb{R}^n$; es de tener en cuenta que solo trataremos curvas de *rapidez unitaria.*

Teorema 2.3.8 (Formulas de Frenet) *Si* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *es una curva paramétrica continua al menos tres veces diferenciable, denótese por* $\mathbf{T}$ *al vector de velocidad de la curva normalizado,* $\mathbf{N}$ *al vector de curvatura normalizado y* $\mathbf{B}$ *al vector de torsión normalizado. Las derivadas de* $\mathbf{T}, \mathbf{N}$ *y* $\mathbf{B}$ *satisfacen las siguientes relaciones:*

$$\begin{aligned} \mathbf{T}' &= \kappa c' \\ \mathbf{N}' &= -\kappa c + \tau c'\prime \\ \mathbf{B}' &= \tau c' \end{aligned} \tag{2.12}$$

Cabe anotar que estas relaciones sin independientes de la parametrización o reparametrización de la curva y constituyen un *invariante* para éstas[17].

2.4. Campos de sistemas de referencias

Si bien un campo vectorial es una función que a cada punto del plano le asigna una dirección de movimiento (un vector), las componentes de un campo vectorial dependen del *sistema de referencia* de $\mathbb{R}^n$ utilizado: la dirección de éste depende de la escogencia de un sistema.

Definición 2.4.1 (Sistema de referencia en $\mathbb{R}^n$) *Una colección* $\beta = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \mathbb{R}^n$ *se dice un* sistema de referencia *en* $\mathbb{R}^n$ *si satisface las siguientes dos condiciones:*

- *La colección es ortonormal:* $||\mathbf{v}_i|| = 1$ *para* $i = 1, \ldots, n$ *y* $\mathbf{v}_i \cdot \mathbf{v}_j = \delta_{ij}$.[1]
- *Todo punto en* $\mathbb{R}^n$ *se puede escribir como una combinación lineal de los vectores en* β.

Si bien en álgebra lineal, los sistemas de referencia ó *bases* permiten escribir vectores en $\mathbb{R}^n$ de manera resumida, no hay razón para pensar que la base de vectores esté siempre definida desde un punto fijo -generalmente el origen-. En efecto, la traslación de los vectores desde el origen hasta un punto arbitrario $\mathbf{x} \in \mathbb{R}^n$ no cambia la geometría del espacio sino simplemente la manera como se describen los puntos en éste como *coordenadas.* A saber, para punto $\mathbf{x} \in \mathbb{R}^n$ se puede definir un sistema de referencia desde el cual describir vectores, por medio del *movimiento rígido.*

Definición 2.4.2 (Movimiento rígido) *Si* $\mathbf{x} \in \mathbb{R}^n$ *es un vector, dado* $\mathbf{y} \in \mathbb{R}^n$*, el vector* $T_{\mathbf{x}}(\mathbf{y})$ *definido por:*

$$T_{\mathbf{x}}(\mathbf{y}) = \mathbf{x} + \mathbf{y} \tag{2.13}$$

se denomina el movimiento rígido *de* $\mathbf{y}$ *hacia la posición* $\mathbf{x}$.

[1] δij se conoce como el *delta de Kronecker*: $\delta_{ij} = 1$ si $i = j$ ó cero de lo contrario

En particular, un sistema de referencia se puede trasladar rígidamente a una posición arbitraria $\mathbf{x}$ del espacio. Todo vector en $\mathbb{R}^n$ se puede escribir en este nuevo sistema de referencia trasladado aprovechando las operaciones de vectores. Dado que una curva paramétrica define una trayectoria en el espacio, un sistema de referencias puede se trasladado de manera rígida a través del recorrido de la curva, constituyendo lo que se conoce como un *sistema de referencia móvil*

Definición 2.4.3 (Traslación de un sistema de referencia por una curva) *Sea* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *una curva paramétrica continua y sea* $\beta = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ *un sistema de referencia en* $\mathbb{R}^n$*. Un sistema de referencia trasladado sobre la curva* $\mathbf{c}$*,* $\beta_{\mathbf{c}}(t)$ *es un sistema de referencia de* $\mathbb{R}^n$ *que sigue la trayectoria de la curva. A saber,*

$$\beta_{\mathbf{c}} = \left\{T_{\mathbf{c}(t)}(\mathbf{v}_1), \ldots, T_{\mathbf{c}(t)}(\mathbf{v}_n)\right\} \tag{2.14}$$

Un sistema de referencia móvil sobre una curva puede estar determinado por la curva y no solamente por la traslación rígida de un sistema arbitrario. De esta manera, podemos generalizar la noción de un sistema de referencia móvil por medio de la siguiente definición:

Definición 2.4.4 (Sistema de Referencia móvil) *Sea* $\mathbf{c} : [a, b] \to \mathbb{R}^n$ *una curva paramétrica continua. Un sistema de referencia móvil a un curva es una aplicación* $\mathbf{F}$ *que a cada punto sobre la curva, le asigna un sistema de referencia:*

$$\mathbf{F}(t) = \{\mathbf{v}_1(t), \ldots, \mathbf{v}_n(t)\} \tag{2.15}$$

Como consecuencia de ésto, los campos vectoriales se pueden escribir como combinaciones lineales de vectores en un sistema de referencia móvil sobre una curva[17]. En particular, los vectores de velocidad, curvatura y torsión constituyen un ejemplo de un sistema de referencia móvil sobre curvas paramétricas.

2.5. Mecánica Lagrangiana

La mecánica lagrangiana es una reformulación de la mecánica clásica que combina la conservación del momento con la conservación de la energía para tratar problemas de física de una manera novedosa. En ésta, la trayectoria que un sistema de partículas sigue está intrínsecamente relacionada con principios como la conservación de la energía y el *principio de mínima acción*[1].

En la mecánica lagrangiana se procede primero a cuantificar la energía del sistema a partir de su posición espacial, su velocidad y aceleración en lo que se conocen como *coordenadas generalizadas*. Desde aquí, es posible definir un sistema de ecuaciones diferenciales, que al resolverse encuentran la configuración del sistema que minimiza el gasto de energía y predicen el movimiento de éste[1]. Estas ecuaciones son conocidas como las *ecuaciones de Euler-Lagrange*.

2.5.1. Modelos físicos vs modelos cinemáticos

La mayoría de los métodos tradicionales de modelamiento gráfico mediante una computadora son cinemáticos[23]. En un modelo cinemático, se modela la interacción de las partículas de manera causal: el movimiento es ocasionado por fuerzas externas ó internas a éstas y ocasionan movimiento[24, 6]. Al contrario, un modelo físico integra las fuerzas tanto internas como externas a un sistema y permite predecir su comportamiento tanto desde los estímulos que ocasionan el movimiento, como desde los principios que subyacen a éste.

Al mismo tiempo, el tratamiento analítico más simple de un problema físico particular se encuentra a veces dentro de lo puramente cinemático (ó newtoniano) a veces se encuentra dentro de lo lagrangiano. En particular, los problemas físicos donde se involucra elasticidad, deformación ó ruptura son intratables desde una postura newtoniana, mientras que la notación, ecuaciones y soluciones se pueden estudiar desde una perspectiva lagrangiana de modo natural[24].

2.5.2. Coordenadas generalizadas

Considérese un sistema de n partículas moviéndose en $\mathbb{R}^n$ de acuerdo con fuerzas y estímulos varios. Una descripción newtoniana de éste presupone la utilización de cuando menos n funciones vectoriales

$$\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_n(t) \tag{2.16}$$

En una postura lagrangiana del mismo sistema, la configuración total del sistema depende de las posiciones individuales de cada partícula en éste: a partir de las componentes de las funciones $\mathbf{x}_i$. En particular, estas funciones dependen entre otras de una selección de una base de $\mathbb{R}^n$ de modo tal que si la base del espacio cambia ó si el sistema de referencia en el cual están inscritas estas funciones cambia, el problema debe replantearse totalmente.

Si se considera más bien que la posición de cada partícula depende de un número de parámetros $s_1, s_2, \ldots, s_p$ podría generalizarse estas funciones $\mathbf{x}_i$, de modo tal que el estado ó *configuración* del sistema completo dependa de una colección de funciones

$$\mathbf{q}_i(s_1, s_2, \ldots, s_p; t) \tag{2.17}$$

de modo tal que la configuración del sistema depende de los paráme ros $s_1, \ldots, s_p$, independiemente del espacio en el que las partículas se encuentren. Estos parámetros se conocen como los *grados de libertad* del sistema. La construcción de cada función $\mathbf{q}_i$ depende de la naturaleza del problema y de la dinámica de éste.

Las leyes básicas de la física como la ley de la inercia, la conservación de masa y energía se pueden escribir mediante las derivadas de estas funciones $\mathbf{q}$, de modo tal que se puede aplicar *cálculo variacional* a los modelos lagrangianos, resultando en las ecuaciones de *Euler-Lagrange*.

2.5.3. Funcionales de Energía

Dado un sistema de partículas $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$ con k-grados de libertad $s_1, s_2, \ldots, s_k$ se puede cuantificar la energía en una posición arbitraria del sistema mediante la velocidad y posición del sistema de modo generalizado. Por ejemplo, la ecuación para calcular la energía cinética de una partícula:

$$K = \frac{1}{2} m ||\dot{\mathbf{x}}||^2 \tag{2.18}$$

En términos lagrangianos:

$$K = \frac{1}{2} m ||\dot{\mathbf{q}}||^2 \tag{2.19}$$

Así bien, en un sistema de n-partículas se puede construir una función

$$F(s_1, s_2, \ldots, s_k, \mathbf{q}_1, \ldots, \mathbf{q}_n, \dot{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_n) \tag{2.20}$$

esa función cuantifica la energía del sistema en una configuración particular de éste. A saber, entrega la *energía instantánea*[1]. De este modo, para la energía total $E(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n)$ del sistema en todas sus configuraciones posibles se tiene en virtud de

$$E(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n) = \int\int\int \ldots \int F(s_1, s_2, \ldots, s_k, \mathbf{q}_1, \ldots, \mathbf{q}_n, \dot{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_n) ds_0 ds_1 \ldots ds_k \tag{2.21}$$

Esta integral debe ser tomada sobre un dominio para los grados de libertad acorde con el sistema estudiado. Por ejemplo, en el caso de las curvas paramétricas, su energía interna se debe a factores como la curvatura y la flexión. En este sentido, si $\mathbf{c} : [a, b] \to \mathbb{R}^n$ es una curva paramétrica, su energía $E(\mathbf{c})$ está dada por:

$$E(\mathbf{c}) = \int_a^b \alpha ||\mathbf{c}'(t)|| + \beta ||\mathbf{c}''(t)||^2 dt \tag{2.22}$$

Esta noción es aprovechada en modelos de curvas flexibles y elásticas satisfaciendo propiedades especiales y es explorada por una variedad de autores: Terzopoulos et al.[4],Baraff et al.[25] y Hadap[20] entre otros.

Nótese que la función E de energía toma una función (la configuración del sistema) y la transforma en un número (su energía), a saber, una función de funciones ó un *funcional*.

2.5.4. Ecuaciones de Euler-Lagrange y el principio de mínima acción

Dentro del planteamiento Lagrangiano de la física, a cada sistema físico se le asigna un funcional de energía. Los sistemas tienden a minimizar su energía y adoptan una configuración que hace esto. Matemáticamente hablando, en la mecánica lagrangiana todos los sistemas adoptan configuraciones que son mínimos de su funcional de energía, de manera que para predecir un sistema es necesario *minimizar* el funcional E.

Este proceso es análogo a la conocida técnica de cálculo para hallar puntos críticos de funciones de funciones univariadas: derivar, igualar a cero y despejar. En el caso particular de los sistemas Lagrangianos, el *derivar* el funcional de energía del sistema resulta en un sistema de ecuaciones diferenciales conocidas como las ecuaciones de *Euler-Lagrange* que el sistema debe satisfacer en un óptimo de la energía. De este modo, solucionando estas ecuaciones se obtiene la configuración más óptima del sistema[24].

Si un sistema de n partículas posee un funcional de energía de la forma

$$E(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n) = \int\int\int \ldots \int F(s_1, s_2, \ldots, s_k, \mathbf{q}_1, \ldots, \mathbf{q}_n, \dot{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_n) ds_0 ds_1 \ldots ds_k \tag{2.23}$$

Sus *ecuaciones de Euler-Lagrange* en cuya solución está el óptimo de la energía se encuentran mediante la solución de las ecuaciones diferenciales determinadas por

$$\frac{d}{dt}\left(\frac{\partial F}{\partial \dot{\mathbf{q}}_i}\right) - \frac{\partial F}{\partial \hat{\mathbf{q}}_i} = 0 \tag{2.24}$$

Estas ecuaciones son deducidas a partir del cálculo de la *primera variación* del funcional de energía: una suerte de primera derivada de éste[24].

Capítulo 3

Modelo de simulación de cabello en tiempo real

En este capítulo se mostrará un modelo dinámico de animación de cabello diseñado para usar en ambientes virtuales que se ejecutan en tiempo real. Con base en recientes investigaciones y resultados de la simulación de barras elásticas, se introdce una técnica la cual utiliza propiedades específicas de los filamentos de cabello para lograr una simulación estable y de gran rendimiento. Las animaciones realistas y la renderización del cabello son parte crucial para modelar *humanos virtuales* [18]. Sin en el comportamiento natural del cabello, el realismo de las escenas presentadas no sería el indicado ni el más preciso, la cabeza de los humanos es un punto trascendental para los observadores [18]. Desafortunadamente, la animación del cabello no es una tarea fácil, debido a que esta forma geométrica exhibe muchas características físicas específicas que se deben tener en cuenta a la hora de simular. Los cabellos tienen un caracter natural anisotrópico [1]; Prácticamente el cabello no se puede ni estirar ni romper. Al mismo tiempo este se dobla y se tuerce fácilmente, pero vuelve a su estado original cuando la tensión externa es removida, adicionalmente la longitud de un cabello en orden de magnitudes es mucho más grande que su diámetro. Estas propiedades, combinadas con el hecho que una persona común posee alrededor de 100.000 cabellos hacen que la precisión y la velocidad de la simulación sea una tarea bastante compleja [13].

El modelo que presenta Kmoch et al. [13] se concentra en la animación del cabello en escenarios en tiempo real, abandonando el realismo estricto al cambio de velocidad [3], pero manteniendo plausibilidad y una base física. Para lograr este efecto, con base en el modelo introducido en [18], originalmente diseñado para objetos grandes y flexibles como las cuerdas. Esto, combinado con un paso en el tiempo, permite el uso de un rápido esquema de integración explícito el cuál es calculable en tiempo real.

Los métodos para simular el cabello generalmente se dividen en dos categorías: basados en los cabellos o basados en el volumen del mismo. la idea de representar el cabello cono un volumen fue introducido en [16]. El cabello es tratado como un volumen del *pelo en cuestión* y simulado usando mecánica continua implementada en partículas suaves. Las hebras individuales de cabello son modeladas como una cadena de nodos rígidos, unidos mediante partículas continuas.

Los métodos basados en el volúmen abandonan la noción de los cabellos individuales aún más. En [14], el volumen es modelado como una malla con nodos acutando como partículas simuladas. Las hebras son unidas a la malla por resortes eslásticos. El enfoque más radical fue adoptado en [21], donde el cabello es simulado como partículas suavizadas conectadas por resortes, sin nociones de hebras de cabello donde la apariencia de éste es obtenida usando *texture splatting*[2].

[1] Cuando un cabello es de caracter anisotrópico se dice que posee propiedades distintas a lo largo de su longitud, por ejemplo, un cabello en su raíz tiene más resistencia a la flexión que en su punta

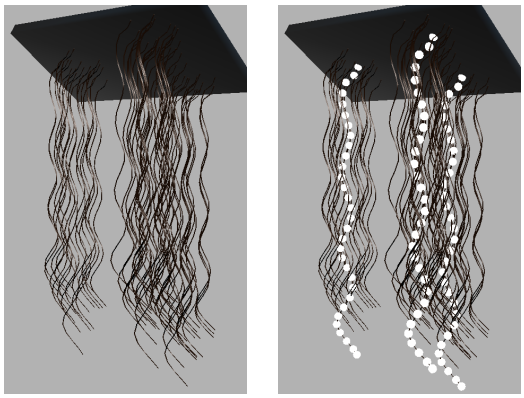
[2] *texture splatting* es denominado en computación gráfica al método de combinar varias texturas

Los métodos volumétricos generalmente son más rápidos, pero ellos tienden a producir deformaciones y no capturan la complejidad y comportamiento del cabello [21]. El enfoque es modelar el cabello explícitamente [3], así como analizar los cabellos individualmente o como mechones, un enfoque para el modelamiento de cabellos dinámicos son los sistemas de resortes (ver figura 3.1).

Las las barras de Kirchhoff han sido recientemente estudiadas en áreas que no están relacionadas con el cabello [3][10]. El método de[10] representa la torsión usando ángulos, mientras que en [3], la rotación escalar desde un marco de referencia es usada y la torsión. Para materiales de baja rigidez, estos métodos son útiles para lograr una buena simulación.

Un modelo dinámico diferente es la teoría de las barras elásticas de Kirchhoff [4] presentado por Kmoch et al[13], al principio introducidas dentro de [2]. Una aplicación de esta teoría de simulación de cabello fue presentada en [8], usando las ecuaciones de Kirchhoff discretizadas como *súper-hélices*, una estructura helicoidal. Las hélices como simulación del cabello reducen considerablemente el número de variables, pero resulta un sistema no lineal y muy costoso de computar. Así mismo modelar las interacciones del cabello es un trabajo bastante complejo.

Diferentes modelos de cabellos son usados para computar el comportamiento complejo del cabello así como la torsión[19] o efectos de productos de belleza[11]. El paradigma empleado es el de cadenas rígidas[7][12][20]. A lo largo de este capítulo se expone el modelo de barras de Kirchhoff tal como se presenta en [13] utilizando los conceptos expuestos en el capítulo 1 (Preliminares).



(a) (b)

Figura 3.1: Barras elásticas de Kirchoff utilizadas en la simulación de cabello ondulado en estado natural. Parte (a) barras como se mostrarían en la simulación combinando hebras físicas y hebras interpoladas, (b) barras físicas en blanco, barras interpoladas en negro. Figura tomada del paper de Kmoch et al.[13]

3.1. Algoritmo para sumulación de pelo

Según el modelo planteado por Kmoch et al. [13] se realiza la simulación y cómputo de las popiedades de cada uno de los cabellos. A saber, el volumen entero del cabello es visto como una colección de hebras individuales, sujetas a una simulación física. Así bien, un gran número de cabellos se simulan mediante

[3] *Modelar el cabello explícitamente* se refiere al método de evaluar cada uno de los cabellos (o un conjunto pequeño de ellos, como los mechones) y computar sus propiedades individualmente, es decir, durante la ejecución no se infiere ningún procedimiento.

[4] En la sección 3.2 se tratará el tema con mayor profundidad

la interpolación, de esta manera el estado de cada una de las hebras de cabello es interpolada desde la raíz (ver figura 3.1(a)). Cabe citar que el modelo para optimizar el rendimiento y evitar el cómputo de muchas variables usa una teoría de hebras físicas y réplicas; es decir, el modelo no computa todos los cabellos toma algunos y replica su estado en las hebras[5] más cercanas (es decir simular mechones de cabellos diminutos, ver figura 3.1(b) .Éste mantiene el número de de hebras simuladas a un nivel manejable, mientras que aún permite un comportamiento no uniforme en el volumen del cabello.

El modelo propuesto por Kmoch et al [13] se expone en el Algorítmo 1. Nótese la división de todas las posibles fuentes de la ecuación relacionadas con la rigidez que son calculadas en pasos de posteriores a los cálculos de las restricciones.

Algoritmo 1. Esquema general simulador de cabello.

```
1.  calcular valores iniciales
2.  while(simulación corriendo){
3.   computar fuerzas
4.   integrar ecuaciones de movimiento
5.   detectar colisiones entre cabellos
6.   while(restricciones sobre las colisiones no están resueltas){
7.    mejorar el paso de cálculo de las restricciones
8.   }
9.   actualizar marco de Bishop
10.  computar torsión de la curva
11. }
```

El algoritmo presentado por Kmoch et al [13] básicamente se divide en 9 procedimientos. Primero se calcula el estado inicial de las curvas, entre ellos la posición incial, su velocidad, su aceleración, su marco de Bishop, su marco material, etc [6]. Posteriormente se incializará la simulación. Durante la simulación se calcularán y actualizarán las fuerzas que intervienen en el sistema en ese instante del tiempo. A continuación se integrarán las ecuaciónes de movimiento que describen el movimiento que tendrá la curva al aplicar las fuerzas calculadas en el paso anterior. Después se realizará un proceso de iteración el cual se encarga de definir y aplicar las restricciones necesarias y de calcular el efecto de las colisiones entre los cabellos. Por último se realizará la correspondiente acutalización del marco de bishop de cada una de las curvas así como computa r la torsión existente en ellas.

3.2. Barras de Kirchhoff

En escencia una barra elástica de Kirchhoff es una forma geométrica que posee ciertas propiedades físicas que la hacen flexible e irrompible, por ejemplo una cuerda. Habiendo descrito el algoritmo de Kmoch et al. se muestra en esta sección una construcción de las barras de Kirchhoff básicas para el modelo de simulación de cabello. Veremos los marcos de referencia presentes en las barras de Kirchhoff (Frenet[7], Material y Bishop). Por último y no menos importante hablaremos de la energía elástica presente en las barras de Kirchhoff. En secciones posteriores realizaremos una discretización del modelo así como una ampliación matemática del modelo planteado por Kmoch [13] et al.

3.2.1. Curvas paramétricas extendidas

Dentro del modelo de simulación de cabello utilizado por Kmoch[18] y Baraff et al.[25], un cabello es tratado como un cilíndro deformando su eje central y sobre el cual se define un *marco de referencia móvil*

[5]Entiéndase por hebra a un cabello individual
[6]Estos conceptos son vistos en profundidad en las secciones posteriores
[7]Refiérase al capítulo anterior

(revísese la definición 3.2.1).Nótese que una barra elástica de Kirchhoff se mueve tanto en el espacio como en el tiempo. Consecuentemente defínase una *curva paramétrica móvil*:

Definición 3.2.1 (Curva paramétrica móvil (extendida)) *Una función* $\mathbf{x} : [a, b] \times \mathbb{R}^+ \to \mathbb{R}^3$ *se dice que es una* curva paramétrica móvil *ó* extendida *si para* $t > 0$ *la función*

$$\mathbf{x}(s,t) = \begin{pmatrix} x_1(s,t) \\ \vdots \\ x_n(s,t) \end{pmatrix} \tag{3.1}$$

es una curva paramétrica en el sentido de la definición 2.1.1.

Entiéndase una curva paramétrica extendida como una curva paramétrica en la variable s que identifica la posición en el espacio de la curva, mientras que el parámetro t denota el instante de tiempo en el que la curva está.

Si se asume que las componentes de la curva son diferenciables tanto en la variable s como en t se puede hablar de su *variación* tanto espacial como temporal. En este sentido, se denota por $\dot{\mathbf{x}}(s)$ a la derivada de una curva paramétrica extendida $\mathbf{x}$ con respecto al tiempo:

$$\dot{\mathbf{x}}(s,t) = \frac{\partial \mathbf{x}}{\partial t}(s,t) \tag{3.2}$$

de modo similar, la derivada *posicional* de la curva paramétrica $\mathbf{x}(s,t)$ se denota como $\mathbf{x}'(s,t)$. A saber:

$$\mathbf{x}'(s,t) = \frac{\partial \mathbf{x}}{\partial s} \tag{3.3}$$

A lo largo de este capítulo se tratará en general con curvas paramétricas extendidas. Consecuentemente, se obviará el término extra t en la parametrización de la curva. La notación de $\dot{\mathbf{x}}$ se puede extender para cubrir derivadas con respecto al tiempo de nivel superior: $\ddot{\mathbf{x}}$ para la segunda derivada ó aceleración, etc.

Una barra de Kirchhoff se puede pensar como un cilindro deformable que se mueve en el tiempo y el espacio, en el cual una curva paramétrica extendida según la definición 3.1 actúa como un eje principal ó *eje central*[25]. Los demas ejes de movimiento se pueden expresar a través de un sistema de referencia móvil añadido a la curva que permiten generar la estructura cilíndrica de ésta, y la describen unívocamente. Dado que la curva paramétrica extendida se mueve en el tiempo y en el espacio es preciso definir un *marco de referencia dependiente del tiempo* que permita generar la barra en cualquier posición y en cualquier instante de tiempo.

Definición 3.2.2 (Sistema de referencia dependiente del tiempo) *Sea* $\mathbf{x} : [a, b] \times \mathbb{R}^+ \to \mathbb{R}^3$ *una curva paramétrica extendida conforme a la definición 3.2.1. Un* sistema de referencia dependiente del tiempo *asociado a la curva* $\mathbf{x}$ *es una tripleta de vectores:*

$$\{\mathbf{m}_1(s,t), \mathbf{m}_2(s,t), \mathbf{m}_3(s,t)\} \tag{3.4}$$

que asigna un sistema de referencia de $\mathbb{R}^3$ *a cada punto sobre la curva en cada instante de tiempo* t*.*

En el modelo de Kmoch, Baraff et al.[13, 25] los cabellos se modelan mediante *barras Kirchhoff* ó *barras elásticas*. En una barra elástica, la longitud es significativamente más grande que en sus otras dos dimensiones (ancho y profundidad ó la *sección transversal*). La configuración de una barra es enteramente descrita por la posición de su centro ó eje central (una curva paramétrica extendida), mientras que su forma transversal, su curvatura y su torsión a través de marcos materiales.

Definición 3.2.3 (Barra de Kirchhoff) *Una barra de Kirchoff* $\Gamma(s,t)$ *es una tupla:*

$$\Gamma(s,t) = \{\mathbf{x}(s,t), \mathbf{t}(s,t), \mathbf{m}_1(s,t), \mathbf{m}_2(s,t)\} \tag{3.5}$$

donde $x(s,t)$ *es una curva paramétrica extendida describiendo el eje central de la curva y* $t(s,t), m_{1,2}(s,t)$ *es un sistema de referencia móvil asociado a* $\mathbf{x}$*. El vector* $\mathbf{t}(s,t)$ *es paralelo al vector tangente de la curva* $\mathbf{x}'(s,t)$*.*

Gracias a la definición anterior podemos discretizar un cabello humano como una barra de Kirchhoff, permitiéndonos calcular los marcos de referencia que determinan el comportamiento y forma de la barra, ajustando sus propiedades para cumplir con el modelo.

3.2.2. Marcos de referencia usuales

Se pueden definir varios marcos en una barra de Kirchhoff: Frenet (refiérase al capítulo 1), Bishop y material. Estos marcos de referencia determinan el comportamiento

Al saber que la torsión es un escalar, ésta se podría expresar usando una variable; por ejemplo expresar el marco material como una rotación del marco de libre de torsión (Ver figura 3.2 para una representación del marco de bishop):

Definición 3.2.4 (Vector de Darboux) *Sea* $\mathbf{x}(s,t)$ *una curva paramétrica extendida y denote por* $\{\mathbf{T}(s), \mathbf{N}(s), \mathbf{B}(s)\}$ *su aparato de Frenet. El* Vector de Darboux *del marco* $\{\mathbf{T}(s), \mathbf{N}(s), \mathbf{B}(s)\}$ *es el vector* $\Omega(s,t)$ *tal que:*

$$\begin{aligned} \mathbf{T}' &= \boldsymbol{\Omega} \times \mathbf{T} \\ \mathbf{N}' &= \boldsymbol{\Omega} \times \mathbf{N} \\ \mathbf{B}' &= \boldsymbol{\Omega} \times \mathbf{B} \end{aligned} \tag{3.6}$$

El vector de Darboux es una combinación lineal del vector de torsión $\mathbf{B}$ y el de curvatura $\mathbf{N}$. A saber,

Teorema 3.2.5 *Para una curva paramétrica* $\mathbf{x}$ *con vector tangente* $\mathbf{T}$*, vector normal* $\mathbf{N}$ *y vector binormal* $\mathbf{B}$ *su vector de Darboux* $\boldsymbol{\Omega}$ *satisface la igualdad:*

$$\boldsymbol{\Omega} = \tau\mathbf{N} + \kappa\mathbf{B} \tag{3.7}$$

donde κ *y* τ *representan la curvatura y la torsión.*

Entre los posibles marcos de referencia móviles sujetos a una curva $\mathbf{x}$, tenemos un interés particular en un marco que sea *libre de torsión*[25]. Un marco libre de torsión es rígido y se mueve de manera natural con la curva y permite la definición de la curvatura, la energía y la elasticidad de manera más simple. En un marco libre de torsión, $\tau = 0$, con lo cual el vector de Darboux satisface la ecuación:

$$\boldsymbol{\Omega} = \kappa\mathbf{B} \tag{3.8}$$

El marco de Bishop se construye a partir del *transporte paralelo* de un vector a lo largo de la curva: una traslación sin rotación que sigue la dirección de la curva.

Definición 3.2.6 (Transporte paralelo de un vector) *Sea* $\mathbf{x} : [a,b] \to \mathbb{R}^n$ *una curva paramétrica y* $\mathbf{v} \in \mathbb{R}^n$ *un vector tangente a la curva* $\mathbf{x}$*. El transporte paralelo del vector* $\mathbf{v}$ *desde la posición* $\mathbf{x}(s_0)$ *hasta la posición* $\mathbf{x}(s_1)$ *está definido por una función* $S : [s_0, s_1] \to \mathbb{R}^n$ *de tal manera que*

$$\mathbf{S}(u) || \mathbf{v} \tag{3.9}$$

para todo $u \in [s_0, s_1]$*.*

En general, el transporte paralelo se puede efectuar mediante una matriz de rotación que rota el vector apropiadamente para mantener el paralelismo a lo largo de la trayectoria de éste. La rotación debe efectuarse definiendo un eje fijo. A saber, la evolución del marco de Bishop a lo largo de la línea central puede ser expresada usando el vector de Darboux $\Omega(s)$ [25].

Definición 3.2.7 (Marco de Bishop) *si* $\mathbf{x}(s,t)$ *es una curva paramétrica extendida con dominio* $[a,b] \subset \mathbb{R}$*, Sea un marco adaptado:*

$$\{\mathbf{t}(s), \mathbf{u}(s), \mathbf{v}(s)\} \tag{3.10}$$

que es libre de torsión se denomina el Marco de Bishop *de la curva* $\mathbf{x}$*. Éste marco se obtiene definiendo* $\mathbf{t}(s,t)$*, fijando* $\mathbf{u}(s_0)$ *y transportando paralelamente* $\mathbf{u}(s_0)$ *por medio de una matriz de rotación* $\mathbf{S}(s)$ *identificando una rotación alrededor del vector de Darboux de la curva* $\mathbf{\Omega}(s,t)$*. El vector* $\mathbf{v}(s,t)$ *se deduce haciendo*

$$\mathbf{v}(s,t) = \mathbf{t}(s,t) \times \mathbf{u}(s,t) \tag{3.11}$$

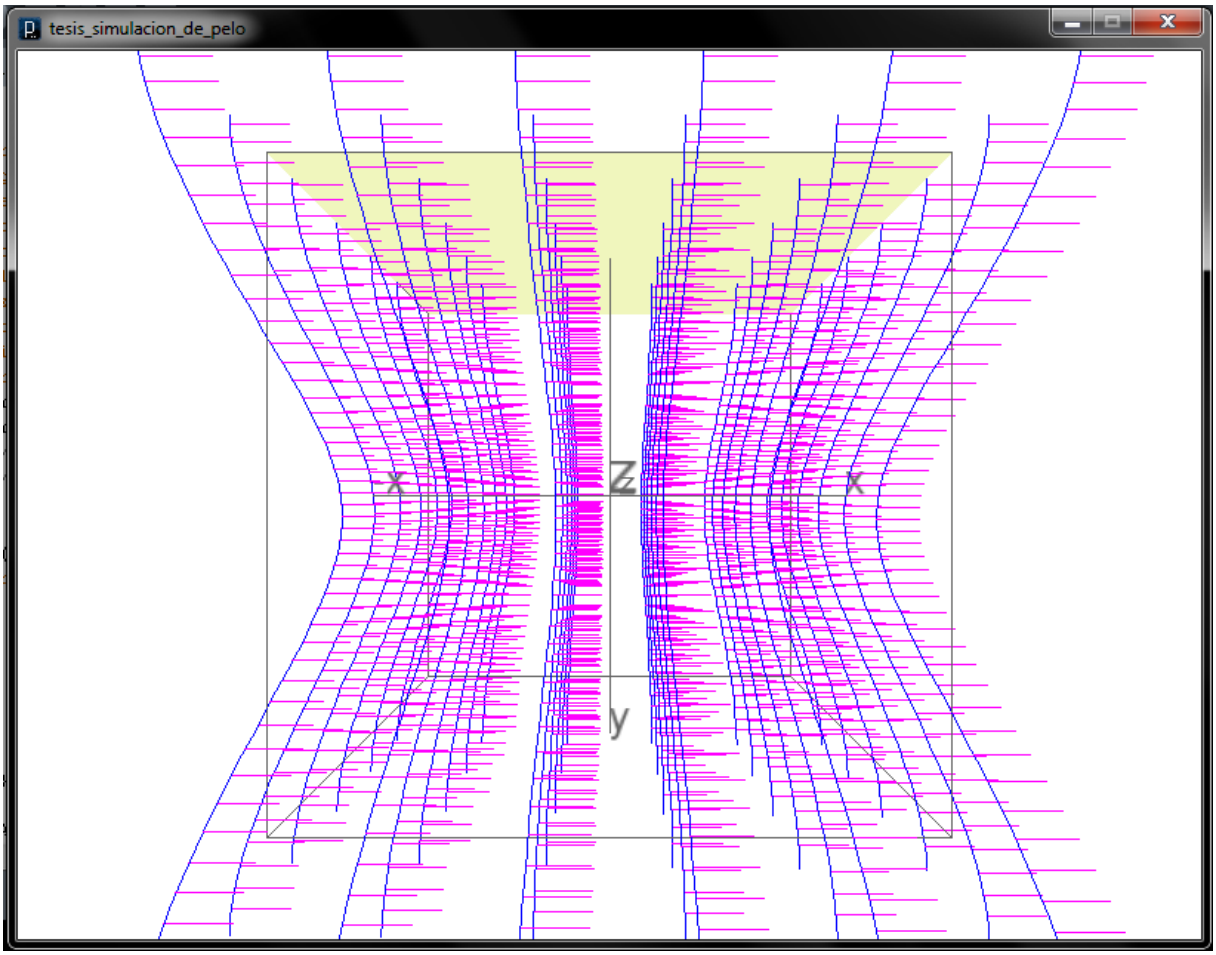


Figura 3.2: Curva paramétrica (azul) con su representación del marco de Bishop (violeta)

3.2.3. Relaciones entre los marcos

Representación del marco material dado que tanto el marco material (vea figura 3.3) como el marco de Bishop son definidos en los bordes:

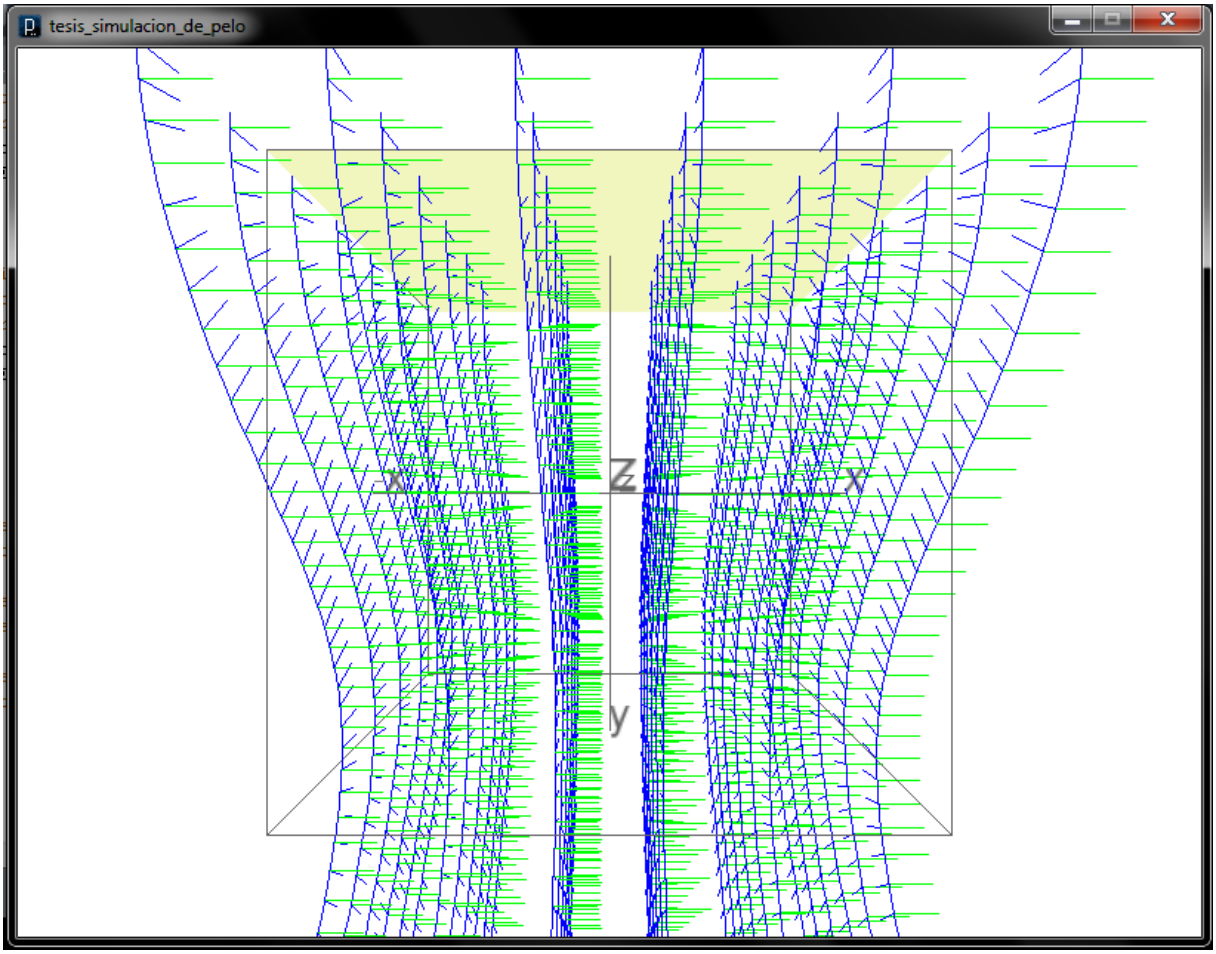


Figura 3.3: Marco material (vectores verdes y azules perpendiculares a la curva)

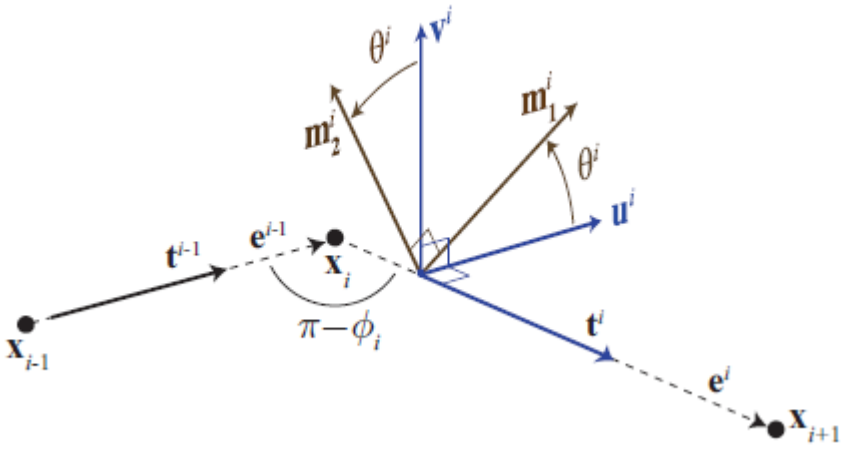


Figura 3.4: Diagrama de vectores influyentes en la curva. Imagen tomada de [3]

Teorema 3.2.8 (Marco Material) *Dada θ^i como el ángulo necesario para rotar el marco de Bishop dentro del marco material vea figura 3.4. Por lo tanto los vectores del marco material son:*

$$\begin{aligned} \mathbf{m}_1^i &= \cos\theta(s,t)^i \cdot \mathbf{u}^i + \sin\theta(s,t)^i \cdot \mathbf{v}^i \\ \mathbf{m}_2^i &= \sin\theta(s,t)^i \cdot \mathbf{u}^i + \cos\theta(s,t)^i \cdot \mathbf{v}^i \end{aligned} \tag{3.12}$$

El marco material y el marco de Bishop se relacionan a través de un ángulo que midel el ángulo entre ellos y que ayuda a determinar la torsión que hay en ese punto de la curva es decir la torsión se puede definir por:2.3.6

3.2.4. Propiedades físicas

Las barras de Kirchhoff poseen adicionalmente a la torsión propiedades físicas que el modelo considera, calcula y evalúa, durante la simulación. Además de estructura, estas propiedades definen el comportamiento que tendrá la barra en cierto instante de tiempo y se adaptan de acuerdo al escenario que estén expuestas:

Inextensibilidad Esta propiedad se refiere a la capacidad que tiene el cabello a no extenderse más de lo común, es decir, las hebras mantienen su longitud original desde que inicia hasta que termina la simulación, manteniendo de esta manera los cálculos de la fuerza estables sin necesidad de computar cuerpos con nuevos valores de longitud.

Nodos El modelo presentado por Kmoch et al. define una barra de Kirchhoff como un conjunto de puntos unidos. Un nodo es considerado como un punto que se debe evaluar entre la curva. Específicamente se considera que entre más puntos se definan en la curva mayor será el realismo de la simulación pero así mismo el rendimiento bajará considerablemente.

Segmentos Los segmentos son líneas que unen a dos nodos dentro de la curva; mientras existan $n+1$ nodos se necesitan n segmentos para unirlos y construir la curva.

Isotropía La isotropía se refiere a la caracterísa de los cuerpos físicos que no dependen de la dirección que se encuentren. A saber, los nodos y los segmentos de las barras de Kirchhoff son isotrópicos porque sin importar su dirección poseen las mismas características físicas.

Anísotropía En contraste, la anisotropía es la característica de los cuerpo físicos que dependen de la dirección que se evalúen estando esta presente en las barras de Kirchhoff, igualmente como se dijo en la página 15 es relevante tener en cuenta que este posee propiedades distintas a lo largo de la longitud del cabello tratado.

3.2.5. Energía elástica

Entiéndase como la energía elástica de una barra $E(\Gamma)$ donde:

$$\Gamma = \Gamma(s,t) = \{\mathbf{x}(s,t), \mathbf{t}(s,t), \mathbf{u}(s,t), \mathbf{v}(s,t)\} \tag{3.13}$$

es dada por la teoría de Kirchhoff de la barras elásticas, compuesta por dos componentes: torsión y flexión (debido a nuestra hipótesis de inextensibilidad, no hay componentes de estiramiento), la energía depende de la *tensión*: la tasa de cambio del marco material, es representada por:

$$\omega_1 = \mathbf{t}' \cdot \mathbf{m_1}, \quad \omega_2 = \mathbf{t}' \cdot \mathbf{m_2}, \quad m = \mathbf{m_1} \cdot \mathbf{m_2} \tag{3.14}$$

Si se denota el centro de la curvatura $\mathbf{k} = \mathbf{t}'$, se puede ver que $\omega_{1,2}$ corresponde a la flexión de la curva a través de los ejes de la sección transversal $\mathbf{m_{1,2}}$ mientras m mide la *torsión* del marco material.

El modelo marca que la flexión de la barra esta descrita por $\omega = (\omega_1, \omega_2)^T$. La energía debida a la flexión depende de sus propiedades de flexión de la sección transversal y su desviación de la flexión natural: ‘

Proposición 3.2.9 (Energía debido a la flexión) *Sea Γ una barra de Kirchhoff de la forma 3.2.3 y considere $\omega = (\omega_1, \omega_2)^T$ la tasa de cambio del marco material en el espacio (véase 3.14), la energía debido a la flexión $E_{bend}(\Gamma(s,t))$ esta definida por la expresión:*

$$E_{bend}(\Gamma(s,t)) = \frac{1}{2}\int_0^L (\omega(s,t) - \hat{\omega}(s,t))^T \hat{B}(\omega(s,t) - \hat{\omega})ds \tag{3.15}$$

donde $\hat{B}$ es una matriz simétrica de 2×2 con valores positivos. La matriz $\hat{B}$ describe la energía debida a la flexión de la barra. $\hat{\omega}$ es la flexión de la cuerda en su estado natural. (Se usa la notación $\hat{a}$ para denotar los valores iniciales de la cuerda antes de que la simulación de inicio)

Así mismo la energía causada por la torsión es definida por:

Proposición 3.2.10 (Energía causada por la torsión) *Sea Γ una barra de Kirchhoff (3.2.3 y sean $\omega = (\omega_1, \omega_2)^T$ la tasa de cambio del marco material en el tiempo 3.14, se representa la energía debido a la torsión como la expresión:*

$$E_{twist}(\Gamma(s,t)) = \frac{1}{2}\int_0^L \beta(s,t)(m(s,t) - \hat{m}(s,t))^2 ds \tag{3.16}$$

donde β es la rigidez de la torsión; a diferencia del modelo original, nosotros consideramos el estado inicial de la torsión de la barra, $\hat{m}$, para permitir la naturalidad de la torsión de las barras.

Esto admite modelar el cabello estilizado, el cual usualmente envuelve los valores iniciales de la torsión (rulos). Ver figura 3.1 para un ejemplo de la estimación del estado inicial. La fórmula anterior relaciona la energía de torsión de los vectores del marco material. Siguiendo las ideas presentadas en [3], fuerón utilizados algunos conceptos de geometría diferencial expresados usando pocas variables.

3.3. Discretización del modelo

Gracias a la teoría de barras de Kirchhoff (presentada en la sección anterior) tenemos los conceptos necesarios para realizar una discretización del modelo que se plantea a la hora simular el cabello humano como una barra de Kirchhoff (representada por una curva paramétrica diferencial). A lo largo esta esta sección se presentará el planteamiento matemático discretizado, se verán las características del pelo que se simularán, se realizará una representacíón discreta del modelo y se mostrará la forma de calcular la energía elástica discreta.

3.3.1. Características del pelo

Para obtener una representación discreta de una barra elástica, seguimos las ideas presentadas en [3]. Sin embargo, como el modelo está destinado para el cabello, también consideramos varios aspectos específicos del mismo, los cuales nos permiten simplificar el modelo.

Una de las consideraciones más importantes es que las hebras del cabello generalmente tienen una sección transversal elíptica, por lo tanto, ellos tienden a inclinarse a través del eje de su sección transversal [9]. Esta idea ha sido usada para reducir directamente el grado de libertad del modelo del cabello presentado en [18]. Sin embargo, nosotros podemos usar esta idea para guiar nuestra computación de la torsión, determinando la flexión de la línea central.

3.3.2. Representación discreta

Dada una barra $\mathbf{\Gamma}(s,t) = \mathbf{x}(s,t), \mathbf{m_1}(s,t), \mathbf{m_2}(s,t)$, nosotros discretizamos la línea central en $n+2$ nodos: $\mathbf{x}_0, \ldots, \mathbf{x}_{n+1}$ conectados por $n+1$ segmentos $\mathbf{e}^0, \ldots, \mathbf{e}^n$ donde cada segmento es definido por:

Definición 3.3.1 (Segmento) *Dado el conjunto de $n+2$ puntos $\mathbf{x}_0, \ldots, \mathbf{x}_{n+1}$ se define un segmento como:*

$$\mathbf{e}^i = \mathbf{x}^i - \mathbf{x}^{i-1} \tag{3.17}$$

Donde la tupla $\{\mathbf{x}^i, \mathbf{x}^{i-1}\}$ son nodos continuos en la curva. Vea figura 3.6

A lo largo del documento se expresarán como subíndices las cantidades asignadas a los nodos y como superíndices las cantidades asignadas a los segmentos.

El ángulo entre el marco material y el marco de Bishop define discretamente la torsión que existe en un punto de la curva. En la siguiente subsección se definen otras propiedades físicas que se tienen en cuenta al momento de realizar el cómputo de las fuerzas influyentes en el sistema.

Teorema 3.3.2 (Representación de la torsión) *Definimos una función escalar $\theta(s)$ la cual mide el ángulo (alrededor del tangente) entre el marco material y el marco de Bishop:*

$$\begin{aligned} \mathbf{m_1} &= \cos(\theta)\mathbf{u} + \sin(\theta)\mathbf{v} \\ \mathbf{m_2} &= \sin(\theta)\mathbf{u} + \cos(\theta)\mathbf{v} \end{aligned} \tag{3.18}$$

Esto nos permite expresar la torsión como $m(s) = \theta'(s)$. Así, nosotros hemos expresado la energía elástica de la cuerda usando 4 dimensiones: una posición central tres-dimensional $\mathbf{x}(s)$ y un ángulo escalar entre el marco material y el marco de Bishop $\theta(s)$.

El modelo le asigna un marco material (vea definición 3.2.8) $\mathbf{t}^j, \mathbf{m_1^j}, \mathbf{m_2^j}$ a cada segmento j (ver figura 3.5). Se debe tener en cuenta que la asinación es única, considerando que un nodo tangente a una curva poligonal es generalmente ambiguo. Nosotros mantenemos el requerimiento de que el marco debe ser adaptado al centro de la curva, significa que:

$$\mathbf{t}^j = \frac{\mathbf{e}^j}{||\mathbf{e}^j||} \tag{3.19}$$

Cantidades integradas como se muestra en [3], una distinción puede ser hecha entre cantidades definidas punto a punto y estas representan un valor integrado sobre el dominio. Cuando una cantidad integrada es asociada con un nodo, este dominio es más cercano a las mitades de los segmentos asignados a los nodos. Por cada nodo $\mathbf{x}_i$, el dominio tiene la longitud $l_i/2$, donde $l_i = |\hat{e}^{i+1}| + |e^i|$.

3.3.3. Energía elástica discreta

A lo largo de esta sección se discretizará la energía elástica usando las definiciones estudiadas anteriormente, energía debido a la torsión 3.16 y la energíá causada por la flexión 3.15. Recuérdese la definición del vector de Darboux 3.2.4 para una curva paramétrica extendida 2.1.1. La versión discreta del vector de Darboux $\mathbf{\Omega}(s,t)$, a saber, $\mathbf{\Omega}_i$ está dado por:

Definición 3.3.3 (Discretización vector de Darboux) *Sea $\mathbf{\Omega}(s,t)$, a saber $\mathbf{\Omega}_i$ discretamente se puede representar por la ecuación:*

$$\mathbf{\Omega}_i = 2\frac{\mathbf{e}^{i-1} \times \mathbf{e}^i}{|\hat{\mathbf{e}}^{i-1}||\hat{\mathbf{e}}^i| + \hat{\mathbf{e}}^{i-1} \cdot \mathbf{e}^i} \tag{3.20}$$

Reresentado una combinación lineal del vector de torsión y el vector de curvatura.

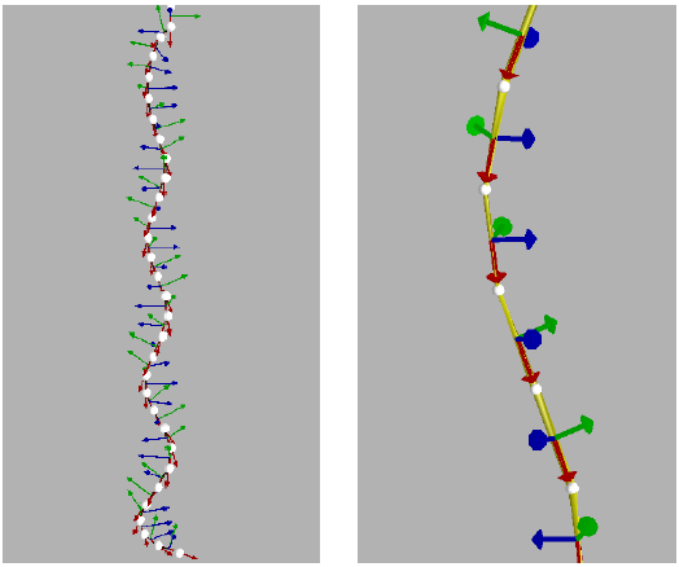


Figura 3.5: Cabello discretizado en 30 segmentos de 1cm. Imagen tomada de [13].

Gracias a la anterior definición podemos discretizar también la tasa de cambio del marco material teniendo en cuenta la ecuación 3.14:

Definición 3.3.4 (Razón de cambio del marco material en el espacio en razón del tiempo) *Sea* $\mathbf{\Omega}_i$ *una combinación lineal del vector de torsión y el vector de curvatura, discretizando la ecuación 3.14 definimos que:*

$$\omega_i^j = ((\Omega)_i \cdot \mathbf{m}_2^j, -(\mathbf{\Omega})_i \cdot \mathbf{m}_1^j)^T \text{ para } j \in \{i-1, i\} \tag{3.21}$$

La cual describe la razón de cambio que tendrá el marco material en razón del tiempo.

Para $i = 1, \ldots, n$ (por segmentos). A partir de éste se puede discretizar la energía debida a la flexión (véase proposición 3.2.10) por medio de:

$$E_{bend}(\Gamma) = \sum_{i=1}^{n} \frac{1}{2\hat{l}_i} \sum_{j=i-1}^{i} (\omega_i^j - \hat{\omega}_i^j)^T \hat{B}^j (\omega_i^j - \hat{\omega}_i^j) \tag{3.22}$$

En un modelo de simulación de pelo basado en barras de Kirchhoff cada una de estas se flexiona sobre su eje de la sección transversal (sobre el plano $\mathbf{m}_{1,2}$)[25]. En este sentido, la energía debida a la flexura depende de los vectores $\mathbf{m}_1$ y $\mathbf{m}_2$. En general, la energía debida a la flexión de una barra elástica se mide mediante la curvatura de su eje principal. A saber,

$$E_{bend}(\Gamma) = \int \kappa(s)^2 ds = \int ||\mathbf{N}(s)||^2 ds \tag{3.23}$$

En efecto, revísese [25, 4, 3] entre otros. Sin embargo, en aras de permitir comportamiento anisotrópico, la norma del vector de curvatura debe ser reemplazada por una *forma bilineal* tal y como se muestra en la proposición 3.2.10. Esta matriz B se conoce como la matriz de Rigidez de la barra elástica. Discretamente, no hay razón por la cual la barra sea isotrópica ó anisotropica en su totalidad. Teniendo esto en cuenta, se define para cada segmento una matriz B_j que permite cuantificar individualmente la energía debida a la flexión en un segmento dado.

Proposición 3.3.5 (Matriz de Rigidez) *Si* Γ *es una barra de Kirchhoff elástica discreta de n segmentos, defina* $\alpha^j,\ j = 1, \ldots, n$ *la resistencia de la barra a la flexión en el segmento* j*. Si* μ *es una constante arbitraria se define la* matriz de rigidez *de* Γ *en el segmento* j *como la matriz,*

$$B^j = \begin{pmatrix} \mu\alpha^j & 0 \\ 0 & \alpha^j \end{pmatrix} \tag{3.24}$$

esta matriz determina una forma bilineal con la cual se calcula la energía de flexión de la barra Γ *en el segmento* j *de manera discreta, en conformidad con la proposición 3.2.10.*

El marco de Bishop se puede discretizar utilizando transporte paralelo. discretizando directamente la definición 3.2.7:

Proposición 3.3.6 (Marco de bishop discretizado) *Sea* $\mathbf{u}^0$ *definido como el nodo raíz de la barra de Kirchhoff y el transporte paralelo a lo largo de la curva, A saber obtentemos* $\mathbf{u}^i = P_i(u^{i-1})$*. Es decir el marco de bishop es representado por:*

$$\mathbf{v}^i = \mathbf{t}^i \times \mathbf{u}^i \tag{3.25}$$

Según la definición de energía elástica dada (3.2.10), es preciso discretizar el vector ω (3.14). Una discretización por diferencias finitas es fácil de obtener y se omite en este punto (véase [25] y [3] para más información). De manera similar, es posible definir la *energía debida a la torsión* utilizando el marco de Bishop definido, discretizando la ecuación 3.16. En este caso, definimos θ^j como el ángulo por el cual hay que rotar el marco de Bishop del segmento j para obtener el marco material. A saber:

$$\begin{array}{rcl} \mathbf{m}_1 & = & \cos\theta^j \mathbf{u}^j + \sin\theta^j \mathbf{v}^j \\ \mathbf{m}_2 & = & -\sin\theta^j \mathbf{u}^j + \cos\theta^j \mathbf{v}^j \end{array} \tag{3.26}$$

En analogía directa del caso continuo, se define la energía debida a la torsión como:

$$E_{twist}(\mathbf{\Gamma}) = \sum_{i=1}^{n} \beta \frac{(m_i - \hat{m}_i)^2}{\hat{l}_i},\ \textit{donde}\ m_i = \theta^i - \theta^{i-1} \tag{3.27}$$

De esta manera se discretiza las ecuaciones influyentes en el modelo presentado por Kmoch et al. [13]

3.4. Actualización del marco material

Anteriormente discretizamos el modelo que se usará para simular el comportamiento del cabello humano. A lo largo de esta sección presentaremos la forma de iterar el sistema para que se adapte a los cambios que tendrá con el paso del tiempo. Es decir, mostraremos como realizar la actualización del marco material en cada iteración del algoritmo.

3.4.1. Torsión optimizada del Cabello

Anteriormente discutimos que los cabellos no se flexionan sobre el eje menor (es decir la raíz). El camino teórico para representar esta situación podría ser una rigidez infinita sobre el eje menor. En la práctica, esto podría ser representado como un valor muy alto de rigidez μ. Desafortunadamente, esto podría resultar en ecuaciones rígides de movimiento y la minimización de Newton podría también ser numéricamente inestable.

Nosotros tomamos un enfoque diferente, manteniendo μ muy pequeño para obtener una pequeña penalización. Entonces en luhar de una minimización de Newton, empleamos un método diferente de computación de la torsión sujeta al hecho de que la flexión sobre el eje menos es despreciable.

Para eliminar la flexión sobre el eje menor completamente, el cabello tendría que torcer a medida de que el último eje sea paralelo al binormal de la curvatura en cada nodo. En nuestro modelo, la flexión ocurre en los nodos, mientras que los marcos materiales son asignados a los segmentos. Esto significa que no es posible prevenir completamente la flexión sobre los primeros ejes de la curva, como los nodos $\mathbf{x}_j, \mathbf{x}_{j+1}$ probablemente requerirá una dirección diferente para el eje final $\mathbf{m}_1^j$. Entonces en lugar de realizar este procedimiento, nosotros computamos la torsión la cuál minimiza minimiza la flexión. En efecto, nuestra computción es dividida en dos pasos:

1. Encontrar (desorientadas) direcciones de los ejes mayores de cada segmento j así este minimiza el la flexión sobre el eje menor en los nodos $\mathbf{x}_j$ y $\mathbf{x}_{j+1}$. Esto fija el valor de θ^j hasta un múltiplo entero de π.

2. Encontrar el eje de orientación dentro de la dirección obtenida en el paso anterior, entonces esto minimiza la energía elástica. Este determina θ^j completamente encontrando el múltiplo de π a usar.

El primer paso es ilustrado en la figura 3.6 . Este procesa los segmentos independientemente y entonces puede ser calculado en paralelo. Para cada segmento j el cuál no está sujeto, nosotros encontramos los ángulos $\eta_{j,j+1}$ entre los ejes del marco de Bishop $\mathbf{u}^j$ y los respectivos nodos binormales, $(\mathbf{kb})_{j,j+1}$. Nótese que como los binormales de la curvatura son perpendiculares a los segmentos, todos los vectores envueltos son complanares. θ^j es obtenida desde $\eta_{j,j+1}$ de la siguiente manera:

1. Si ninguno de los dos $(\mathbf{kb}_{j,j+1})$ es $\mathbf{0}$, $\theta^j = \frac{1}{2}(\eta_j + \eta_{j+1})$.

2. Si solo $(\mathbf{kb}_k)$ no es cero, $\theta^j = \eta_k$.

3. Si $((\mathbf{kb}_{j,j+1})$ ambos son $\mathbf{0}$, $\theta^j = \hat{\theta}^j$

El segundo paso no determina cuál de los $\theta^j, (\theta^j + \pi)$ y $\theta^j - \pi$ minimiza la energía elástica. Denotamos la energía computada con este marco material E^0, E^+ y E^-, respectivamente. Desde la definición de $E(\Gamma)$ los siguientes criterios son definidos:

$$\begin{aligned} E^+ < E^- &\Leftrightarrow \theta^j < \hat{\theta}^j \\ E^0 < E^+ &\Leftrightarrow -2\hat{B}^j\bar{\omega}_i^j < 2\beta\pi(\theta^j - \hat{\theta}^j) + \beta\pi^2 \\ E^0 < E^- &\Leftrightarrow -2\hat{B}^j\bar{\omega}_i^j < -2\beta\pi(\theta^j - \hat{\theta}^j) + \beta\pi^2 \end{aligned} \tag{3.28}$$

donde $\bar{\omega}_i^j$ es el producto por componentes de ω_i^j y $\hat{\omega}_i^j$. Usando estos criterios, el valor correcto de θ^j a usar puede ser encontrado fácil y rápidamente.

3.5. Simulación

Ahora nosotros ensamblamos las ecuaciones de movimiento. Recuerde que la torsión es tratada cuasi-estáticamente y por lo tanto no es parte de la simulación dinámica. Entonces, para la ecuaciones de movimiento, θ^j no son variables independientes, pero pueden ser expresadas usando $\mathbf{x}_1$.

3.5.1. Ecuaciones de movimiento

Las ecuaciones de movimiento manejan el comportamiento de la barra elástica son para $i = 0, \ldots, n+1$:

$$M_1\ddot{\mathbf{x}}_i = F_i^{elastic}(\mathbf{x}) + F_i^{external}(\mathbf{x}, \dot{\mathbf{x}}) \tag{3.29}$$

M_i es la masa del nodo i. $F_i^{elastic}$ es la fuerza interna elástica que afecta al nodo i. $F_i^{external}$ es la fuerza externa total que afecta al nodo i. En el escenario de la simulación, usamos gravedad y fricción combinado

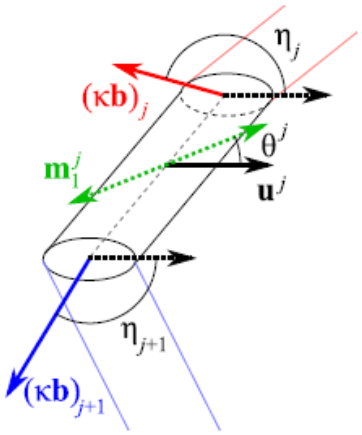


Figura 3.6: Computando la orientación del marco material θ^j del segmento j para minimizar la flexión sobre los ejes menores de los nodos $j, j+1$ Estableciendo θ^j a la medida de los ángulos orientados η_j y η_{j+1}. Imágen tomada de **??**

con el ambiente estático del aire: $F_i^{external} = M_i\mathbf{g} - \mathbf{v}\dot{\mathbf{x}}_i$, donde $\mathbf{g}$ es la aceleración gravitacional y $\mathbf{v}$ es el coeficiente de fricción del aire. Las ecuaciones de movimiento están integradas usando el método de Euler [5].

3.5.2. Fuerzas elásticas

La fuerza elástica es esfuerza en minimizar la energía elástica, entonces nosotrso podríamos simplemente escribir $F_i^{elastic} = -\dfrac{dE(\Gamma)}{d\mathbf{x}_1}$ la derivdad total de la energía elástica se tiene en cuenta tnato en la dependencia explícita sobre la posición central y una dependencia implícita en esta via el marco material. Sin embargo, para obtener una fórmula integrable, nosotros debemos substitiur dentro del total derivativo:

$$-\frac{dE(\Gamma)}{d\mathbf{x}_i} = -\frac{\partial E(\Gamma)}{\partial \mathbf{x}_i} - \sum_{j=0}^{n} \frac{\partial E(\Gamma)}{\partial \theta^j}\frac{\partial \theta^j}{\partial \mathbf{x}_i} \tag{3.30}$$

Ahora analizaremos los componentes individuales de esta expresión; usaremos la siguiente notación: $\nabla_i = \dfrac{\partial}{\partial \mathbf{x}_i}, \nabla^j = \dfrac{\partial}{\partial \theta^j}$.

Holonomía: Para expresar las derivadas de la energía, usaremos el concepto de holonomía. En nuestro caso, holonomía ψ es una diferencia (escalar) de la rotación del marco material causada por una temporal evolución del centro de la barra. Diremos que ψ_i es el ángulo entre ${}_tP(P_i(\mathbf{t}^{i-1}))$ y $P_i({}_tP(\mathbf{t}^{i-1}))$. Para una descripción completa de este concepto, refiérase a [3], de donde nosotros tomamos el gradiente holonómico:

$$\begin{aligned} \nabla_{i-1}\psi_i &= \frac{(\mathbf{kb})_i}{2|\hat{\mathbf{e}}^{i-1}|}, \ \nabla_{i+1}\psi i = -\frac{(\mathbf{kb})_i}{2\hat{e}^i} \\ \nabla_i\psi_i &= -(\nabla_{i-1}\psi_i + \nabla_{i+1}\psi_i) \end{aligned} \tag{3.31}$$

Nosotros podemos extender este concepto para atravesar más de un segmento, como la holonomía es aditiva. Nosotros definimos $\Psi^j = \sum_{i=1}^{j}\psi_i$ para ser la rotación necesaria para alinear ${}_tP(P_1(\ldots(P_j(\mathbf{u}^0))\ldots))$

hasta $P_1(\dots(P_j({}_tP(\mathbf{u}^0)))\dots)$. El gradiente de este ángulo es simple $\nabla_i\Psi^j = \sum_{k=1}^{j}\nabla_i\psi_k$. Tenga en cuenta que $\nabla_i\psi_k$ no pueden ser en al menos 3 valores de k.

Ψ_j describe que tanto rota el marco de Bishop cuando el centro se envuelve sobre el tiempo. Desde que nosotrso definimos el marco material relativo al marco de Bishop, debemos substraer este ángulo para matener el marco material alientado correctamiente. En orden de ideas, esto nos da una rotación del marco material con respecto a las posiciones centrales: $\dfrac{\partial\theta^j}{\partial\mathbf{x}_i} = -\nabla_i\Psi^j$.

Derivadas Posicionales: Nosotros ahora analizamos $\dfrac{\partial E(\Gamma)}{\partial\mathbf{x}_i} = \dfrac{\partial E_{bend}(\Gamma)+E_{twist}(\Gamma)}{\partial\mathbf{x}_i}$. $E_{twist}(\Gamma)$ no depende en $\mathbf{x}$ explícitamente, entonces $\nabla_i E(\Gamma) = \nabla_i E_{bend}(\Gamma)$. El gradiente toma la siguiente forma:

$$\frac{\partial E(\Gamma)}{\partial\mathbf{x}_i} = \sum_{k=1}^{n}\frac{1}{\hat{l}_k}\sum_{j=k-1}^{k}(\nabla_i\gamma_k^j)^T\hat{B}(\gamma_k^j-\hat{\gamma}_k^j) \tag{3.32}$$

Derivadas de torsión: Pasamos ahora a $\dfrac{\partial E(\Gamma)}{\partial\theta^j}$ A raíz de las expresiones de energía relevantes, obtenemos:

$$\begin{aligned}\frac{\partial E_{bend}(\Gamma)}{\partial\theta^j} &= \nabla^j W_j + \nabla^j W_{j+1}\\ \frac{\partial E_{twist}(\Gamma)}{\partial\theta^j} &= 2\beta(\frac{m_j}{\hat{l}_j}-\frac{m_{j+1}}{\hat{l}_{j+1}})\\ \texttt{donde}\nabla^j W_i &= \tfrac{1}{\hat{l}_i}(\omega_i^j)^T J\hat{B}^j(\omega_i^j-\hat{\omega_i^j})\end{aligned} \tag{3.33}$$

Fuerza elástica total: Recordando que la torsión de segmentos no sujetados son computados para minimizar la energía elástica. Por lo tanto, $\nabla^j E(\Gamma) = 0$ para $j \notin C$. Esto significa que el total de la fuerza elástica que actúa sobre el nodo i es:

$$F_i^{elastic} = -\frac{\partial E(\Gamma)}{\partial\mathbf{x}_i} + \sum_{j\in C}\frac{\partial E(\Gamma)}{\partial\theta^j}\nabla_i\Psi^j \tag{3.34}$$

3.6. Límites y colisiones

Nuestro esquema de integración no posee ningún mecanismo para mantener la inextensibilidad, evitando la complejidad de la ecuación. En [3], un paso de postintegración refuerza las restricciones para la inextensibilidad y el acomplamiento de barra a los cuerpos rígidos. Nosotros tomamos esta idea además para usar resctricciones para manejar eficientemente las colisiones entre los cabellos.

3.6.1. Tipos de restricciones

El modelo puede usar alguna de las siguientes restricciones. Cada restricción es representada por un valor el cual es 0 cuando la restricción es satisfecha; estas son reunidas dentro de un vector de restricciones $\mathbf{C}$.

Inextensibilidad (ver secciónes anteriores): Para cada segmento, definimos una restricción de inextensibilidad $\mathbf{CI}^j = \mathbf{e}^j\cdot\mathbf{e}^j - \hat{\mathbf{e}}^j\cdot\hat{\mathbf{e}}^j$.

Acoplamiento a un cuerpo rígido: Cualquier nodo o segmento puede ser acoplado (unido) a un cuerpo rígido. En el escenario del cabello, esto es solo usado para ajustar la raíz del segmento a la cabeza. Esto

nos permite expresar las restricción de acoplamiento de una forma simple: $\mathbf{CR}_0 = \hat{\mathbf{x}}_0 - \mathbf{x}_0$, $\mathbf{CR}_1 = \hat{\mathbf{x}}_1 - \mathbf{x}_1$

Colisiones entre los cabellos Después del paso de integración, todos los nodos son la prueba de impenetración con la cabeza. Los nodos son unidos dentro de un conjunto P y están sujetos a la restricción $\mathbf{CH}_i = (\mathbf{x}_i - \mathbf{h}) \cdot (\mathbf{x}_i - \mathbf{h}) - \hat{r}^2$ para $i \in P$, donde $\mathbf{h}$ es el centro de la cabeza y r es el radio de la misma. En teoría, los nodos orginialmente no están en P podrían no chocar con la cabeza gracias a las ubicaciones durante la ejecución de la restricción. Por motivos de rendimiento olvidamos esta situación; los nodos que chocan serán detectados y tratados en el siguiente paso del tiempo. Gracias a la duración del paso del tiempo es solo 1ms, tales interpretaciones ligeras no se notan.

3.6.2. Ejecución de las restricciones

Las restricciones son ejecutadas por una rápida proyección múltiple [23]. Este método toma una configuración sin restricciones e iterativamente computa una configuración cercana la cual satisface las restricciones. Definimos en términos de energía cinética $\frac{1}{2}\mathbf{y}\tilde{M}\mathbf{y}^T$, donde $\tilde{M}$ es una $3(n+3) \times 3(n+3)$ matriz de masa diagonal y $\mathbf{y} \in \mathbb{R}^{3)(n+3)}$ es un sistema de velocidad de:

$$diag(\tilde{M}) = (\mathbf{M}_H, \mathbf{M}_0, \ldots, \mathbf{M}_{n+1}) \quad \mathbf{y} = (\dot{\mathbf{h}}, \dot{\mathbf{x}})^T \tag{3.35}$$

donde $\mathbf{M}_H$ es la masa de la cabeza con respecto al nodo i.

Después la iteración convergem actualizamos las velocidades de los nodos, como en [23]: $\dot{\mathbf{x}}_i \leftarrow \dot{\mathbf{x}}_i - \frac{1}{h}\Delta\mathbf{x}_i$. Tenga en cuenta que por la masa insignificamnte del pelo comparada con la masa de la cabeza, no consideramos el movimiento de la cabeza en respuesta a las reubicaciones de los nodos del cabello. La proyección del método reposiciona los nodos ligeramente. Especialmente es útil para nuestro tratamiento de las colisiones de cabello, como la salida no exhibe un comportamiento inestable, a menudo causado por el uso de fuerzas de penalización..

Capítulo 4

Simulador de cabello

En los dos capítulos anteriores estudiamos tanto los conceptos preliminares como el planteamiento hecho por *Kmoch* et al. [13]. En el capítulo 2 estudiamos la teoría matemática que se debe manejar para poder comprender la teoría de Barras de Kirchhoff y el planteamiento hecho por [13]. Así mismo, en el capítulo 3 estudiamos el modelo planteado para simular el cabello como una barra elástica de Kirchhoff, discretizamos el modelo y realizamos una descripción general de la estructura del algoritmo a implementar. A lo largo de este capítulo mostraremos la implementación que se realizó para intentar construir un simulador que implemente toda la teoría que muestra el modelo planetado por Kmoch et al. [13].

4.1. Herramientas utilizadas

Durante esta sección describiremos las herramientas utilizadas para el desarrollo de la aplicación, el ambiente operacional, y la estructura de archivos utilizados en la implementación. Así mismo se realizará explicación de cada una de las herramienta utilizadas para el desarrollo de la aplicación.

4.1.1. Java y Processing

Processing es un lenguaje de programación y un ambiente de código abierto[1] creado para computación gráfica y poder desarrollar, animaciones, e interacciones. Éste es usado por estudiantes, artistas, diseñadores e investigadores para el aprendizaje, simulación y producción. Este es creado para enseñar técnicas fundamentales de programación dentro de un contexto visual y sirve como herramienta fundamental de producción. Processing usa el núcleo del lenguaje de programación Java permitiéndonos de esta manera usar todas las ventajas que posee este lenguaje de programación orientado a objetos.

Adicionalmente, processing añade métodos propios que facilitan la simulación de varios objetos y permite una programación más sencilla de ambientes gráficos permitiendo programar simuladores de manera cómoda y estable, entre ellos, usa dos métodos importantes para la incialización y ejecución del programa `void setup()` y `void draw()` que son definidos dentro del archivo principal usados por processing para incializar la simulación. El método `void setup()` es llamado tan pronto como inicia la simulación, siendo éste utilizado para definir el ambiente principal así como el tamaño que ocupará la ventana de simulación, el color de fondo, cargar imágenes, cargar fuentes, etc. Adicionalmente a esto es utilizado para la inicialización de las variables usadas durante la ejecución del programa. El método `void draw()` es llamado directamente después del método `setup` y continuamente ejecuta las lineas de código que estan contenidas dentro del método, éste nunca para su ejecución a menos que el programa sea finalizado o el

[1] Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

método `noLoop()` sea llamado. La función `draw()` es llamada automáticamente y no debe ser llamada explícitamente.

En las secciones posteriores veremos con mayor profundidad el uso de estas herramientas para crear la aplicación. A medida que se avance se explicará con detalle el uso de java y processing para crear el simulador que implementa el modelo planteado por Kmoch et al. [13].

4.2. La arquitectura de la aplicación

A lo largo de esta sección describiremos la arquitectura de la aplicación, plantearemos el diagrama de clases y discretizaremos cada uno de los componentes, métodos y atributos que posee cada una de las clases. Así mismo El diagrama de clases describe la estructura funcional de la aplicación. En este caso se presentan cuatro clases que son las que componen la aplicación. A continuación describiremos cada una de las clases su estructura, sus atributos y sus métodos más importantes. (Véase figura 4.1)

4.2.1. Clase tesis simulación de pelo

Esta clase es la principal (equivalente al main en java) dentro de programa, este es el archivo raíz desde el cual processing comenzará la ejecución de cada una de las instrucciones escritas en él. En esta clase se encuentran los dos métodos descritos en la sección anterior `setup` y `draw`.

Atributos: Esta clase posee 15 atributos los cuales son mencionados a continuación:

1. `nDVector[] aceleraciones`: Es un arreglo de objetos de clase nDVector (véase sección de clase nDVector) esta estructura de datos almacena las aceleraciones que poseen cada una de las curvas paramétricas en la interacción del programa.

2. `float factorEscala`: El factor de escala es un escalar de tipo float que nos define la escala a la cual voy a mostrar la curva, esta misma me ayuda a definir el centro de la curva porque me ayuda a calcular las coordenadas donde se encuentra.

3. `PFont fuente`: La variable fuente es un tipo de archivo que me define la fuente que voy a utilizar dentro del simulador en las inserciones de texto.

4. `float fuerza`: el atributo fuerza captura la magnitud de un evento asociado al mouse que captura el movimiento mientras se oprime el click izquierdo del mismo.

5. `CurvaParametricaDiferencial[][] miCurva`: miCurva es una matriz que almacena todas y cada una de las curvas que se van a definir, es decir, si se define que se modelarán 25 cabellos se define miCurva como una matriz de 5×5. Esta variable es llamada constantemente en el programa debido a que será la que se debe iterar constantemente.

6. `int numPuntos`: Este entero será el que define que tantos nodos estarán en cada una de las curvas (recuérdese el capítulo anterior la definición de nodo), entre más nodos se definan mayor será la precisión del programa pero menor será su rendmiento.

7. `float rotX, rotY` como el programa es capaz de rotar el plano a la direccion escogida por el usuario con el movimiento de su mouse `rotX` y `rotY` indican que tanto se debe rotar el plano en cada una de las direcciones x y y.

8. `nDVector[] velocidades`: Es un arreglo de tipo nDVector que almacenla las velocidades que poseen cada una de las curvas paramétricas en la interacción del programa.

Métodos: Los métodos que posee esta clase son mencionados a continuación:

1. `void dibujarPlano()`: Dibuja el plano donde todos as curvas estan ancladas se considera la raíz de todos los cabellos.
2. `void draw()`: Método iterativo propio de processing que nunca para de ejecutarse a menos que exista un `noLoop()` o se pare el programa.
3. `void setup()`: Primer método que se llama en processing es usado para incialización de variables y del ambiente de programación.
4. `void mousePressed()`: método que captura el evento de cuando un mouse es presionado. Es utlizado para capturar las coordenadas inciales del mouse cuando se oprime, para calcular la fuerza.
5. `void mouseDragged()`: método que captura el evento cuando el click del mouse se encuentra oprimido, es utilizado para mover el plano cartesiado y apreciar de otro ángulo la simulación.
6. `void mouseReleased()`: método que captura el evento cuando el click del mouse se suelta, es utilizado para capturar su coordenadas finales y calcular la fuerza que se hizo con el mouse para mover la curva.
7. `void plotAxis()`: Función que dibuja un cubo en el simulador para realzar el ambiente de un plano cartesiano 3D.

Esta clase es la encargada de la ejecución total del simulador y es la que se encarga de llamar a los objetos de las otras clases y realizar los cálculos necesarios.

4.2.2. Clase nDVector

Un vector es una entidad física con magnitud y dirección que representa un elemento en un espacio vectorial, durante los capítulos anteriores definimos como vectores varias cantidades físicas. A continuación definiremos la clase vector nDVector que nos define una estructura dos o tres dimensional según se necesite:

Atributos: Esta clase posee 2 atributos los cuales son:

1. `float components[]` este arreglo de tipo float almacena los componentes x, y, z propios de un vector.
2. `int dimension` esta variable de tipo int define de qué dimensión será el vector que se estará creando si de dos o tres dimensiones.

Métodos: las operaciones básicas entre vectores estan definidas en la clase vector lo que nos facilita muchísimo el trabajo para operaciones posteriores entre vectores.

1. `nDVector addVector(nDVector v)` Este método lo que hace es recibir un vector como parámetro y suma sus componentes con el vector que hace el llamado al método.
2. `float angle(nDVector v)` angle es la función que recibe un vector y retorna una variable de tipo float que determina el ángulo que se forma entre ese parámetro y el vector que hace el llamado al método.
3. `float getComponent(int i)` De acuerdo al entero que el método reciba, entre $0, 1, 2$ el método retorna el valor del componente x, y, z del vector.
4. `float getComponent(char c)` De acuerdo al caracter que el método reciba, entre x, y, z el método retorna el valor del componente x, y, z del vector.
5. `int getDimension()` Retorna la dimensión a la que pertenece el vector.
6. `nDVector()` Constructor por defecto de la clase.

de clases simualdor.png

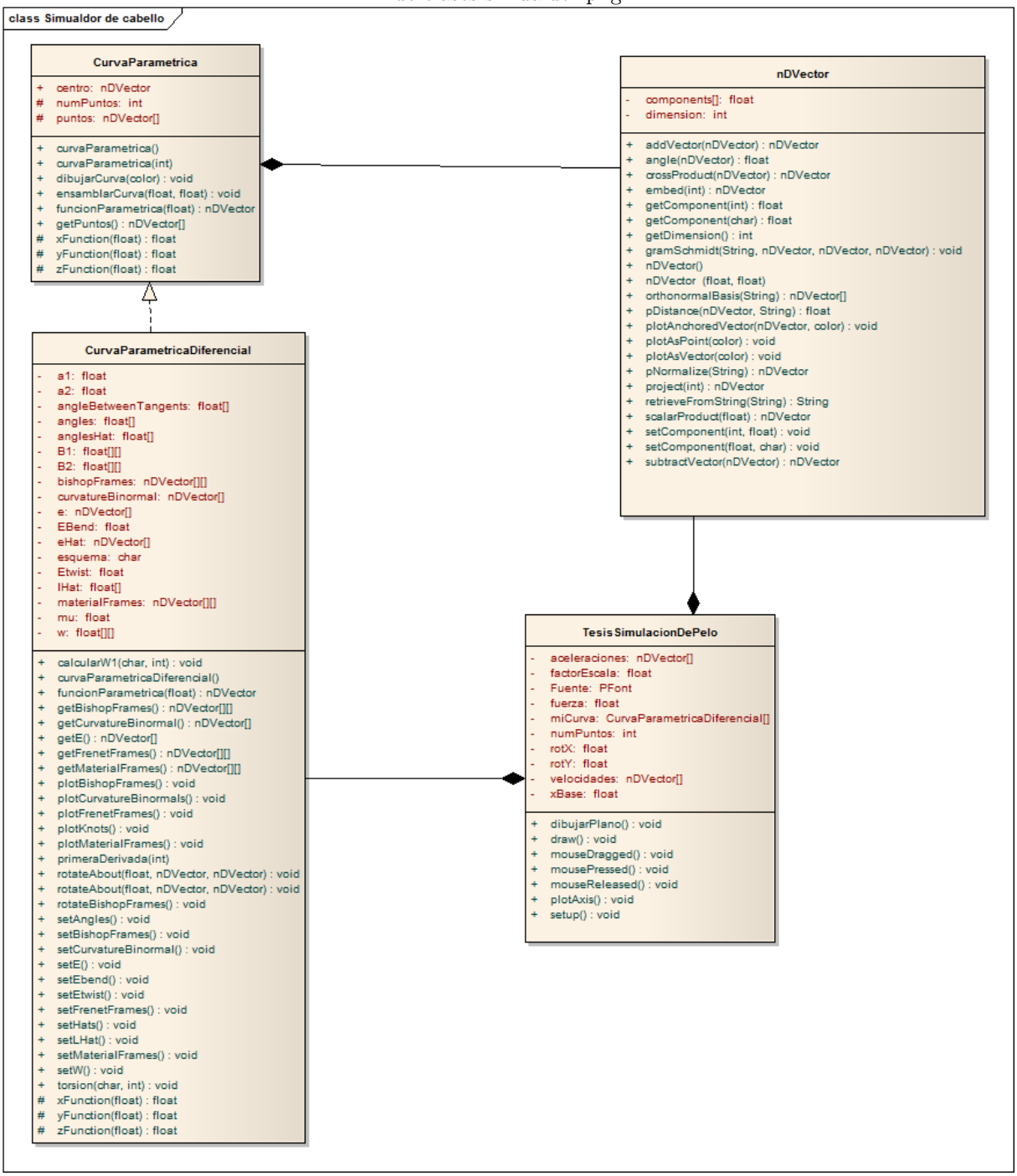


Figura 4.1: Diagrama de clases simulador de cabello implementado en procesing

7. `nDVector(float x, float y)` Constructor de la clase que inicializa un vector de dos dimensiones.
8. `nDVector(float x, float y, float z)` Constructor de la clase que inicializa un vector de tres dimensiones.
9. `void plotAnchoredVector(nDVector v, color colorVector)` Dibuja el vector que hace el llamado anclado al vector que se recibe como parámetro del color que recibe por parámetro.
10. `void plotAsPoint(color colorVector)` Dibuja el vector como un punto en el espacio.
11. `void plotAsVector(color colorVector)` Dibuja el vector como una línea en el espacio.
12. `float dotProduct(nDVector v)` este método retorna el producto punto entre el vector que hace el llamado al método y el vectorque se recibe como parámetro.
13. `nDVector scalarProduct(float i)` este método retorna el producto del vector que hace el llamado con el escalar que se pasa por parámetro.
14. `void setComponent (int i, float a )` este método establece el valor del componente que le diga el índice i con el valor recibido en a.
15. `void setComponent (char i, float a )` este método establece el valor del componente que le diga el caracter i con el valor recibido en a.

Gracias a esta clase podemos definir la estructura de datos más usada durante todo el programa, y gracias a ella las operaciones entre vectores son bastante sencillas y rápidas.

4.2.3. Clase Curva Parametrica

Esta clase modela las curvas paramétricas diferenciales de una forma general es decir define unos métodos como una clase abstracta pero implementa otros como una clase concreta:

Atributos Los atributos que posee la cuva paramétrica son:

1. `nDVector centro`: Esa variable representa el punto central de la curva paramétrica.
2. `int numPuntos`: Este entero establece el número de nodos que tendrá la curva. Recuérdese que entre más nodos se definan más preciso será el movimiento y la simulación pero así mismo el rendimiento de la aplicación baja considerablemente.
3. `nDVector[] puntos`: Este arreglo de tipo nDVector establece y almacena cada uno de los nodos que tendrá la curva. (véase seccion 3.2.3).

Los métodos definidos en esta clase implícitos, es decir, se encuentran las firmas de los métodos pero no hay ninguna implementación pero algunos son explícitos, a continuación describiremos la función que cumplen los métodos explícitos:

1. `curvaParamétrica(int numPuntos)`: Constructor que inicializa la curva paramétrica de acuerdo al número de puntos.
2. `nDVector funcionParametrica(float t)`: Esta función discretiza la curva paramétrica con los parámetros que son definidos en `xFunction(), yFunction(), zFunction()` de acuerdo al parámetro `float t`(véase sección siguiente).
3. `nDVector ensamblarCurva(float a, float b)`: Construye la curva paramétrica con los parámetros a y b 2.1.1 teniendo en cuenta la función paramétrica descrita en el ítem anterior.
4. `void dibujarCurva(color colorCurva)`: Dibuja la curva paramétrica del color del parámetro `colorCurva`.

De esta clase podemos heredar los métodos necesarios para convertirla en una curva paramétrica diferenciable y modelar las propiedades de las barras elásticas de Kirchhoff.

4.2.4. Clase Curva paramétrica diferencial

Esta clase es una especificación de la clase mencionada en la sección anterior, en la cual se modelan todas las propiedades de una barra de Kirchhoff.

Atributos La clase diferenciable posee un conjunto de atributos mucho más amplio que su clase padre, Estos atributos son:

1. `char esquema` Esta variable es utilizada para determinar el esquema con el cual se derivará la curva con respecto al tiempo.
2. `nDVector[][] bishopFrames` Para representar el marco de bishop planteado en 3.2.7 se usa una matríz de tipo nDVector, la cual almacena cada uno de los marcos de Bishop para los nodos presentes en la curva.
3. `nDVector[][] materialFrames` Para representar el marco de material planteado en 3.2.8 se usa una matríz de tipo nDVector, la cual almacena cada uno de los marcos de materiales para los nodos presentes en la curva.
4. `nDVector[][] materialFrames` Para representar el marco de frenet planteado en 2.3.8 se usa una matríz de tipo nDVector, la cual almacena cada uno de los marcos de materiales para los nodos presentes en la curva.
5. `nDVector[] curvatureBinormal` Para representar la curvatura del binormal planteada en 3.2.4 se usa una arreglo de tipo nDVector, la cual almacena cada uno de las curvaturas del binormal para los nodos presentes en la curva.
6. `nDVector[] e` Los segmentos presentes en cada una de las curvas son representados por un arreglo de tipo nDVector de n posiciones por cada $n+1$ nodos.
7. `float[] angleBeteweenTangents` este arreglo guarda el ángulo que existen entre las curvas tangentes de la curva paramétrica.
8. `float[] angles` este arreglo guarda los ángulos entre el marco de bishop y el marco de material 3.14.
9. `nDVector[] eHat` este arreglo almacena los segmentos en el estado inicial de la barra es decir, hacen referencia a los segmentos de tipo $\hat{e}^0, \ldots, \hat{e}^n$. Segmentos iniciales de la curva.
10. `nDVector[] anglesHat` este arreglo guarda los ángulos entre el marco de bishop y el marco de material iniciales.
11. `nDVector[] lHat` este arreglo guarda las longitudes iniciales de la curva antes de iniciar la simulación.
12. `float[][] w` matriz que guarda la discretización del vector de Darboux.
13. `float[][] B1` esta matriz representa la matriz definida por 3.3.5.
14. `float a1, a2, mu` recuérdese la proposición 3.3.5, donde se definen las variables α_1, α_2, μ.
15. `float Ebend` variable que almacena la energía debido a la flexión 3.2.10.
16. `float Etwist` variable que almacena la energía debido a la flexión 3.2.9.

De esta manera definimos todos los atributos de las curvas paramétricas diferenciables, sin olvidar aquellos heredeados desde la clase curva paramétrica. La lógica de la implementación del modelo planteado en el capítulo anterior esta en los *métodos* a continuación presentaremos los métodos propios de la curva diferenciable.

1. `curvaParametricaDiferencial()` constructor por defecto de la clase.
2. `curvaParametricaDiferencial(int numPuntos)` constructor que llama a la variable `numPuntos` desde su clase padre.
3. `primeraDerivada(int i, char esquema)` calcula la primera derivada de la curva estimando el esquema de derivación (progresivo, regresivo, central).
4. `segundaDerivada(int i, char esquemaDePrimeraDerivada)` calcula la segunda derivada de la curva estimando el esquema de derivación que se usó en la primera derivada para evitar inconsistencias (progresivo, regresivo, central).
5. `torsion(int i, char esquema)` retorna el producto cruz entre la primera derivada y la segunda derivada de la curva evaluada en la posición i con el esquema definido.
6. `calcularW1(int i, char esquema)` retorna el producto punto entre la primera derivada y la segunda derivada y se guarda en un flotante.
7. `setHats()` método que establece el valor de las variables `eHat` y `anglesHat`.
8. `setLHats()` método que establece el valor de las variable `lHat`.
9. `setE()` método que establece el valor del vector de segmentos e.
10. `getE()` retorna la variable `e`.
11. `setCurvatureBinormal()` método que establece el valor del arreglo `curvatureBinormal`.
12. `getCurvatureBinormal()` método que retorna el valor del arreglo `curvatureBinormal`.
13. `setMaterialFrames()` método que establece el valor de la matriz del marco material `materialFrames`.
14. `getMaterialFrames()` método que retorna el valor de la matriz del marco material `materialFrames`.
15. `setFrenetFrames()` método que establece el valor de la matriz del marco de frenet `frenetFrames`.
16. `getFrenetFrames()` método que retorna el valor de la matriz del marco de frenet `frenetFrames`.
17. `setBishopFames()` método que establece el valor de la matriz del marco de Bishop `bishopFrames`.
18. `getBishopFames()` método que retorna el valor de la matriz del marco de Bishop `bishopFrames`.
19. `rotateBishopFames()` método que rota el marco de bishop alrededor del marco material.
20. `setAngles()` método que estableceel arreglo de ángulos.
21. `nDVector rotateAbout(nDVector axis, nDVector v, float angle)` método que rota el marco de Bishop alrededor de un vector definido por parámetro.
22. `nDVector rotateAbout(nDVector v1,nDVector v2,nDVector v3, nDVector v, float angle)` método que rota el marco de Bishop alrededor de sus vectores definidos.
23. `setW()` método que establece el valor de la variable `W`.
24. `plotMaterialFrames()` dibuja el marco material sobre la curva paramétrica.
25. `plotFrenetFrames()` dibuja el marco de frenet sobre la curva paramétrica.
26. `plotCurvatureBinormals()` dibuja el marco de frenet sobre la curva paramétrica.
27. `setEbend()` método que calcula la energía debido a la flexión.

28. `setEtwist()` método que calcula la energía debido a la torsión.

29. `xFunction(float t)` recibe un parámetro t y define la función para la componente x de la curva paramétrica.

30. `xFunction(float t)` recibe un parámetro t y define la función para la componente y de la curva paramétrica.

31. `xFunction(float t)` recibe un parámetro t y define la función para la componente z de la curva paramétrica.

De esta forma el modelo de barras de Kirchhoff esta implementado en su totalidad. Cabe citar que por inconvenientes con las fórmulas de la holonomía y de la fuerza elástica no se logró terminar por completola implementación deceada, de igual forma se deja planteado toda la formulación matemática para una culminación exitosa de esta.

Capítulo 5

Resultados y Conclusiones

5.1. Problemas y éxitos a la hora de implementar la solución

Mientras nos encontrábamos implementando la solución al problema que planteamos para elaborar el proyecto de tesis nos encontramos con problemas variados, el primer obstáculo con el que nos encontramos fue lograr entender el modelo que estamos siguiendo, pues el léxico que se maneja en los diferentes documentos estudiados, analizados y que se tomarón como referencia son un poco complicados y confusos en algunas ecuaciones, logrando que la curva de aprendizaje y entendimiento fuera un poco mas lenta, ya que nos tocaba referirnos a muchas bibliografías en donde se encontraban las raíces de las ecuaciones que se estaban planteando en el momento y que no tenían una clara evolución en el documento, a medida que iba transcurriendo el tiempo logramos entender poco a poco cada unas de las ecuaciones que necesitamos para implementar la virtualización del cabello humano, así se logro poco a poco dar un resultado esperado por nosotros al trabajo de tesis de grado y lograr implementar con éxito las ecuaciones para poder modelar el cabello humano.

5.2. Lo que no se alcanzó hacer y propuestas para el futuro

Como dijimos anteriormente fue complicado en muchas ocasiones entender algunas de las ecuaciones que se nos eran planteadas dificultándonos un poco la implementación; uno de los problemas que tuvimos fue con el gradiente holonómico quien es el encargado de la rotación del marco material logrando así que no se pudiera discretizar la ecuación 3.29 la cual no logramos entender de donde surgía y por mas que buscamos en bibliografías no encontramos esta ecuación y mucho menos como lograr entenderla para poder implementarla.

Como propuesta para el futuro p12roponemos un estudio mas riguroso de cómo poder implementar diferentes tipos de cabellos, así poder simular cabellos lisos, rizados, crespos, utilizando ecuaciones que ya tenemos y modificando algunas variables, vectores, direcciones, para lograr el objetivo.

Bibliografía

[1] Courant R. Hilbert D. *Methods of Mathematical Physics*. Interscience, London, 1953.

[2] Pai D. Interactive simulation of thin solids using cosserat models. *Eurographics 02*, 2002.

[3] Bergou M. Wardetzky M. Robinson S. Audoly B. Grinspun E. Discrete elastic rods. *ACM Trans. Graph 27*, 2008.

[4] Demetri Terzopoulos Jhon Platt Alan Barr Kurt Fleischert. Elastically deformable models. *Computer Graphics*, 21(4), 1987.

[5] Hairer E. Lubich C. Wanner G. Geometric numerical integration: Structure-preserving algorithms for ordinary differential equations. *Series in Comp.*, 2006.

[6] Bercovier M. Grinspun E. Goildenthal R. Harmon D. Fattal R. Efficient simulation of extensible cloth. *ACM Trans.*, 2007.

[7] Choe B. Choi M. G. Ko H. Simulating complex hair with robust collision handling. *ACM Press*, 2005.

[8] Bertails F. Audoly B. Cani M. Querleux B. Leroy F. Lévêque J. Superhelices for predicting the dynamics of natural hair. *ACM Siggraph 2006 Papers*, 2006.

[9] Swift J. Some simple theoretical considerations. *J. Cosmet Sci*, 1995.

[10] Spillmann J. Teschner M. Cosserat rod elements for the dynamic simulation of onedimensional elastic objects. *Eurograpchis Association 07*, 2007.

[11] Ward K. Galoppo N. Lin M. Modeling hair influenced by water and styling products. *Computer animation and social agents (CASA)*, 2004.

[12] Ward K. Lin M. Adaptive grouping and subdivision for simulating hair dynamics. *IEEE Computer Society*, 2003.

[13] P. Kmoch U. Bonanni N. Magnenat-Thalmann. Hair simulation model for real-time environments. *ACM Trans. 2009*, 2009.

[14] Volino P. Manenat-Thalmann. Animating complex hairstyles in real-time. *ACM Press*, 2004.

[15] Joe F. Thompson Z. U. A. Warsi C. Wayne Mastin. *Numerical Grid Generation*. North Holland, 1985.

[16] Hadap S. Magnenat-Thalmann N. Modeling dynamic hair as continuum. *Computer Graphics Forum 20*, 2001.

[17] Barrett O'Neill. *Elementos de Geometría Diferencial*. Limusa Wiley, 1972.

[18] Bonnani U Kmoch P. Virtual hair handle: A model for haptic hairstyling. *Eurographics 2008 Short Papers*, 2008.

[19] Selle A. Lentine M. Fedkiw R. A mass spring model for hair simulation. *ACM Trans. Graph. 27*, 2008.

[20] Hadap S. Oriented strands: dynamics of stiff multi-body system. *Eurographcis Association*, 2006.

[21] Bando Y. Chen B. Nishita T. Animating hair with loosely connected partivles. *Eurographics 2003 Proc.*, 2003.

[22] Richard Szeliski David Tonnesen. *Surface modeling with oriented particle systems.* Cambridge Research Lab, 1991.

[23] Eitan Grinspun Max Wardetzky. Discrete differential geometry. *SIGGRAPH ASIA 2008 COURSE NOTES*, 2008.

[24] Keonwook Kang Chris Weinberger and Wei Cai. *A Short Essay on Variational Calculus.* Department of Mechanical Engineering Stanford University, 2006.

[25] David Baraff Andrew Witkin. Large steps in cloth simulation. *COMPUTER GRAPHICS Proceedings, Annual Conference Series*, 1988.