UNIVERSIDADE DA CORUÑA

Facultad de Informática
Departamento de Electrónica y Sistemas

Proyecto de Fin de Carrera
Ingeniería en Informática

Generación automática de secuencias en animación a partir de vídeo

Alumno: Diego Moreda Rodríguez
Director: Emilio José Padrón González
Fecha: 22 de septiembre de 2016

Título	**Generación automática de secuencias en animación a partir de vídeo**
	Xeración automática de secuencias en animación a partir de vídeo
	Automatic generation of animation sequences from video
Clase:	Proyecto clásico de Ingeniería
Autor:	Diego Moreda Rodríguez
Director:	Emilio José Padrón González
Tribunal:	
Fecha de defensa:	
Calificación:	

Resumen

En los últimos años, las técnicas de generación no realista de imágenes (*non-photorealistic rendering*) se han convertido en una línea de investigación y desarrollo muy activa. En el contexto de la generación de animaciones en dos dimensiones, uno de los aspectos más tediosos del proceso de animación tradicional es la necesidad de dibujar cada uno de los fotogramas que forman la animación de forma manual.

En esta memoria se presenta un sistema que facilita la generación de secuencias de animación creadas a partir de secuencias de vídeo utilizando la técnica conocida como rotoscopia. Esta técnica de animación consiste en la utilización de una secuencia de vídeo como base, sobre la que se dibujan una serie de trazos que destacan ciertas características elegidas por el artista. Este proceso facilita la labor del dibujante, pero cuenta con el inconveniente de que cada uno de los fotogramas debe ser dibujado de forma individual. En este proyecto se pretende desarrollar un sistema que resuelva este problema.

En concreto, este proyecto propone un método automático que, a partir de los trazos que dibuja el artista en dos fotogramas próximos en una secuencia de vídeo, realiza la interpolación de los trazos de los fotogramas intermedios de una forma suave y visualmente atractiva. Este método se basa en utilizar la información contenida en la imagen y en la forma de los trazos como guía para la interpolación, permitiendo también al usuario ajustar de forma manual los trazos en el caso de que los resultados no sean de su agrado.

Uno de los principales objetivos es minimizar la intervención por parte del usuario. Así, el usuario sólo debe preocuparse de dibujar los trazos en los fotogramas clave, definiendo alguna de las características de la imagen. Los trazos de los fotogramas intermedios son generados e interpolados de forma automática. Para que esto sea posible, en el primer paso de la interpolación es necesario establecer la correspondencia entre los dos trazos de los fotogramas clave de una manera automática. En este trabajo evaluaremos una serie de

algoritmos para tratar de resolver este problema.

Con el fin de dar soporte a los algoritmos implementados para resolver estos problemas, se ha desarrollado una aplicación gráfica de escritorio que permite a un usuario trabajar en una escena de animación. La aplicación está enfocada hacia su uso por parte de un animador, como uno de los últimos pasos en la producción de una secuencia de animación. El resultado de la combinación de estas cuestiones es la aplicación mutiplataforma ARAS.

Palabras clave

Rotoscopia, animación, *non-photorealistic rendering*, seguimiento, correspondencia, Bézier.

A mis seres queridos.

Agradecimientos

Un agradecimiento especial a Margarita Amor López y Emilio José Padrón González por darme la oportunidad de realizar este trabajo y por su ayuda, colaboración, y apoyo durante el desarrollo del mismo a todos los niveles, sin los cuales este trabajo no hubiera sido posible.

También quiero dar las gracias a Monsterrat Bóo Cepeda y a los miembros del Departamento de Electrónica y Sistemas que, directa o indirectamente, han contribuido a nivel técnico o personal.

Por último, pero no por ello menos importante, a mi familia, que me ha acompañado, apoyado, y en cierto modo sufrido a lo largo de todos estos años y en todas las circunstancias.

Índice general

1. Introducción **1**
1.1. Motivación 1
1.2. Objetivos 3
1.3. Organización de la memoria 4

2. Método de seguimiento basado en fotogramas clave **5**
2.1. Introducción 5
2.2. Estado del arte 7
2.3. Curvas Bézier 10
2.3.1. Curvas paramétricas 11
2.3.2. Definición de curvas Bézier 12
2.4. Método de seguimiento basado en fotogramas clave 15
2.4.1. Procedimiento 16
2.4.2. Función de energía 18
2.4.3. Método de optimización 23

3. Comparativa de diferentes algoritmos de correspondencia **27**
3.1. Introducción 27
3.2. Correspondencia entre dos curvas paramétricas 32
3.3. Método basado en correspondencia entre los vectores tangentes 34
3.4. Método basado en la similitud de las curvas 38
3.5. Método guiado por imagen 42
3.6. Método basado en el vector de desplazamiento 44

4. Pruebas y resultados de los algoritmos de correspondencia **47**
4.1. Secuencias utilizadas 47
4.2. Configuración de los algoritmos de correspondencia 51
4.3. Métricas 52

4.4. Resultados experimentales 54
4.4.1. Resultados de correspondencia y seguimiento 54
4.4.2. Resultados de tiempo 54
4.4.3. Resultados visuales y otros 61

5. Interpolación en ARAS **67**
5.1. Descripción de la aplicación ARAS 67
5.1.1. Interfaz gráfica 68
5.2. Diseño de la aplicación ARAS 70
5.2.1. Aplicación original 70
5.2.2. Diseño 72
5.2.3. Patrones utilizados 88
5.3. Implementación 94

6. Metodología de desarrollo y planificación **97**
6.1. Metodología de desarrollo 97
6.1.1. Metodologías consideradas 97
6.1.2. Metodología utilizada 100
6.2. Planificación 102
6.3. Análisis de coste 104

7. Conclusiones **111**

A. Trabajo futuro **115**
A.1. Método basado en propiedades geométricas 115
A.1.1. Análisis de la matriz de covarianza 116
A.1.2. Propiedades geométricas 117
A.1.3. Penalización de descartes de puntos característicos . . . 118
A.1.4. Algoritmo de correspondencia 119
A.2. División de curvas en igual número de segmentos 122
A.3. Simplificación de curvas con un número elevado de segmentos 123
A.4. Paralelización del proceso de seguimiento 124

B. Manual de usuario de la aplicación ARAS **125**
B.1. Guía rápida 125
B.2. Manual extendido 127

Glosario **137**

Bibliografía **141**

Índice de cuadros

4.1. Características de las curvas de la secuencia *Table-tennis* . . . 49
4.2. Características de las curvas de la secuencia *Amira* 50
4.3. Valores utilizados para los parámetros de los algoritmos 51
4.4. Tiempos (en ms) para la secuencia *Table-tennis* 62

6.1. Estimación de costes de tiempo (en días) 108
6.2. Estimación de costes económicos netos de personal 109
6.3. Estimación de costes económicos de recursos físicos 110

Índice de figuras

1.1. Películas que utilizan la técnica de rotoscopia 2

2.1. Fotogramas clave e intermedio 6
2.2. Descripción a alto nivel del sistema propuesto 7
2.3. Una curva Bézier y su polígono de control 13
2.4. Interpolación de fotogramas intermedios 15
2.5. Interacción del usuario con el sistema 16
2.6. Vector unitario normal utilizado en el cálculo de E_I 20
2.7. Cálculo del término E_C 22
2.8. Proceso de optimización 24
2.9. Algoritmo de optimización 25

3.1. Tipos de algoritmos de correspondencia 29
3.2. Correspondencia entre dos curvas paramétricas C_A y C_B . . . 33
3.3. Correspondencia entre los vectores tangentes de dos curvas . . 34
3.4. Posibles correspondencias anteriores a (C_A^i, C_B^j). 36
3.5. Cálculo de la correspondencia (programación dinámica) 37
3.6. Vectores tangentes y Ángulo 39
3.7. Influencia del término índice 41
3.8. Ventana $\mathbb{W}(C^i)$ 43
3.9. Vectores de desplazamiento e invarianza a traslación 46

4.1. Secuencia *Amira* 48
4.2. Secuencia *Table-tennis* 48
4.3. Curvas de la secuencia *Table-tennis* 49
4.4. Curvas de la secuencia *Amira* 50
4.5. Ventana $\mathbb{W}(C^i)$ orientada en la dirección de T^i 52
4.6. Error medio de las curvas de la secuencia *Table-tennis* 55
4.7. $|DM|$ y $PSNR$ de la secuencia *Table-tennis* 56

4.8. Error medio de las curvas de la secuencia *Amira* 57
4.9. $|DM|$ y $PSNR$ de la secuencia *Amira* 58
4.10. Tiempo por etapas para la secuencia *Table-tennis* 59
4.11. Tiempos medios para la secuencia *Table-tennis* 60
4.12. Ejemplo de los diferentes métodos de correspondencia 63
4.13. Estimación subjetiva de la secuencia *Table-tennis* 64
4.14. Curvas ideales de la secuencia *Table-tennis* 65

5.1. Casos de uso en ARAS 68
5.2. Proyecto con sus diferentes elementos 69
5.3. Ventana principal de la aplicación ARAS 70
5.4. Clases en la aplicación original 73
5.5. Diagrama de clases de ARAS 75
5.6. Paquete model 76
5.7. Paquete klt 77
5.8. Paquete commands 80
5.9. Paquete gui 81
5.10. Paquete gwidgets 83
5.11. Paquete match 84
5.12. Paquete util 86
5.13. Ejemplo de la utilización de señales y ranuras en ARAS 90
5.14. Clases que contienen señales y/o ranuras 91

6.1. Metodologías de desarrollo en cascada y ágil 99
6.2. Diagrama de Gantt 105

7.1. Comparación entre curvas ideales y generadas 113
7.2. Detalle la familia *Pierna_Dcha* en los fotogramas 0-10 113

A.1. Puntos característicos de una curva C 116
A.2. Regiones de confianza y autovectores 117
A.3. Ejemplo de camino completo 119
A.4. Correspondencia entre puntos característicos. 120

B.1. Ventana principal de la aplicación ARAS 127
B.2. Diálogo Exportar 130
B.3. Diálogo Configuración\Correspondencia 132
B.4. Diálogo Configuración\KLT 134
B.5. Diálogo Configuración\Debug 135

Capítulo 1

Introducción

En este proyecto se desarrolla una herramienta que asistirá al usuario en la aplicación de la técnica de rotoscopia para automatizar la generación de una secuencia de animación. En concreto, se plantea un nuevo método de seguimiento basado en fotogramas clave para la generación de los fotogramas intermedios en una secuencia de animación en dos dimensiones creadas a partir de una secuencia de vídeo.

1.1. Motivación

Las secuencias de animación se caracterizan por representar la realidad con un grado elevado de abstracción y un estilo determinado. La producción de una secuencia de animación en dos dimensiones es tradicionalmente un proceso laborioso, donde cada uno de los fotogramas que compone la secuencia de animación debe ser dibujado de forma manual. Este es un trabajo tedioso, y con elevados costes en tiempo y dinero. Además, es un proceso propenso a introducir elementos no deseados tales como parpadeos en la animación resultante, debido a la dificultad de mantener la continuidad en el tiempo de las curvas, al ser dibujadas a mano. El entorno de trabajo más adecuado es aquel en el que el artista se concentra en dibujar sólo una serie de fotogramas clave, que serán utilizados como referencia a la hora de dibujar los fotogramas intermedios. La automatización de la generación de estos fotogramas intermedios permitirá un ahorro en términos de tiempo de desarrollo, calidad y costes de producción de la animación, ya que los esfuerzos se centrarán en el dibujo de estos fotogramas clave.

(a) (b)

Figura 1.1: Películas que utilizan la técnica de rotoscopia (a) *A Scanner Darkly* [53], © Warner Independent Pictures (b) *Waking Life* [52], © Fox Searchlight Pictures

En nuestro trabajo nos centramos en el contexto de la técnica utilizada en animación conocida como rotoscopia [34]. Esta técnica consiste en la utilización de una secuencia de imágenes de vídeo real como base para el dibujado de los fotogramas de una animación. Así, el artista dibuja trazos que siguen las características de las formas que existen en dicha secuencia, lo que facilita su trabajo e incrementa la suavidad de la animación con respecto a la animación tradicional. A pesar de esto, el dibujante debe seguir trabajando con un fotograma de cada vez. La rotoscopia tiene gran relevancia en la generación de efectos especiales para el cine, a menudo representando cerca del veinte por ciento de recursos humanos utilizados en el proyecto [6]. Entre sus aplicaciones, además de la aplicación para la que fue inicialmente diseñada de generación de figuras animadas en sí, destaca la creación de regiones sobre las que aplicar un filtro a la imagen, o la delimitación de regiones que serán sustituidas por una imagen generada por ordenador. En la Figura 1.1 se muestran dos de las películas más destacadas en las que emplean técnicas de rotoscopia para dotarlas de un estilo característico.

En la actualidad no existe un método que permita generar los fotogramas intermedios de forma automática a partir de unos fotogramas clave. Por este motivo, en este trabajo proponemos un método de interpolación de los trazos de los fotogramas intermedios basado en técnicas de seguimiento (*tracking*), que automatiza la generación de dichos trazos en gran medida. El objetivo es permitir al usuario obtener los resultados deseados minimizando su interacción con el sistema. Así, a partir de los trazos dibujados por el artista en determinados fotogramas clave de una secuencia, el sistema interpolará

los trazos de los fotogramas intermedios utilizando para ello la información contenida en las imágenes y la información de la forma de los trazos. Si la información contenida en la imagen no es suficiente y los resultados no son los adecuados, el usuario podrá refinar los resultados añadiendo información en los fotogramas intermedios. El resultado final es una secuencia de formas temporalmente coherente, con una transición suave entre los fotogramas. En contraste con la aproximación tradicional consistente en editar cada fotograma individualmente, se espera que el sistema reduzca notablemente la cantidad de interacción y esfuerzo necesarios para la generación de una secuencia de animación. Además, se espera que la calidad de la secuencia de animación resultante sea mayor que la de una secuencia dibujada de forma manual, ya que no ocurren problemas debidos a la falta de coherencia temporal, como el parpadeo (*flickering*).

1.2. Objetivos

El objetivo de este proyecto es el desarrollo e implementación de un sistema que automatice la generación de los fotogramas situados entre dos fotogramas clave utilizando rotoscopia. Partiendo de esta idea, el sistema debe permitir en concreto:

- Realizar la interpolación de los trazos situados en los fotogramas intermedios de una secuencia, a partir de los trazos dibujados por el artista en dos fotogramas clave diferentes. Dados dos trazos dibujados por el artista en dos fotogramas clave, el sistema utilizará la información contenida en los fotogramas de la secuencia y la forma de los trazos para generar trazos intermedios que sigan el movimiento de la secuencia de una forma suave. Los trazos serán representados por curvas Bézier.

- Dado que se pretende dotar al usuario de la mayor flexibilidad posible a la hora de dibujar los trazos situados en los fotogramas clave, sus curvas Bézier pueden tener diferente número de puntos de control. El algoritmo de seguimiento utiliza las muestras de cada una de las curvas para realizar los cálculos, y en concreto, para una muestra determinada es necesario saber a qué muestra corresponde en el siguiente instante de tiempo. Por lo tanto, surge la necesidad de utilizar algoritmos de correspondencia para resolver el problema de la asignación de muestras entre dos curvas Bézier. En este trabajo se evaluarán e implementarán

diferentes algoritmos de correspondencia, con el fin de determinar el algoritmo más adecuado para resolver este problema.

- Teniendo en cuenta que el sistema estará orientado a su uso por parte de diseñadores y artistas, la aplicación que implementará estos algoritmos debe disponer de un interfaz sencillo a la vez que flexible, de manera que el usuario pueda explotar su funcionalidad de una manera intuitiva.

1.3. Organización de la memoria

Esta memoria está organizada siguiendo esta estructura:

En el Capítulo 2 se presenta en primer lugar el problema de la interpolación de curvas utilizando algoritmos de seguimiento dentro del contexto de la animación en dos dimensiones, así como un análisis de los trabajos previos realizados en este campo. Posteriormente se introducen las curvas Bézier como paso previo a la explicación del método de seguimiento que hemos tomado como base para nuestro trabajo. Por último, se presenta el método de seguimiento que hemos propuesto.

En el Capítulo 3 comienza con la introducción del problema de la correspondencia entre dos curvas Bézier, surgido por las modificaciones realizadas al método de seguimiento. Se analizan los trabajos más destacados relacionados con este problema y que se han implementado en nuestra herramienta.

En el Capítulo 4 se presentan las secuencias y métricas utilizadas para comparar los métodos de seguimiento descritos en el Capítulo anterior, y los resultados obtenidos en términos de bondad de los resultados y tiempos de ejecución.

En el Capítulo 5 se describe el diseño e implementación de la aplicación ARAS (Automatic Rotoscoping for Animated Sequences).

En el Capítulo 6 se describe la metodología de desarrollo elegida para nuestro trabajo, incluyendo la planificación y el análisis de coste.

En el Capítulo 7 se presentan las conclusiones y principales aportaciones de este trabajo.

Para concluir, los Apéndices A y B contienen, respectivamente, líneas de trabajo e ideas para continuar desarrollando el trabajo objeto de esta memoria; y el manual de usuario, en el que se describen las operaciones que permite la aplicación.

Capítulo 2

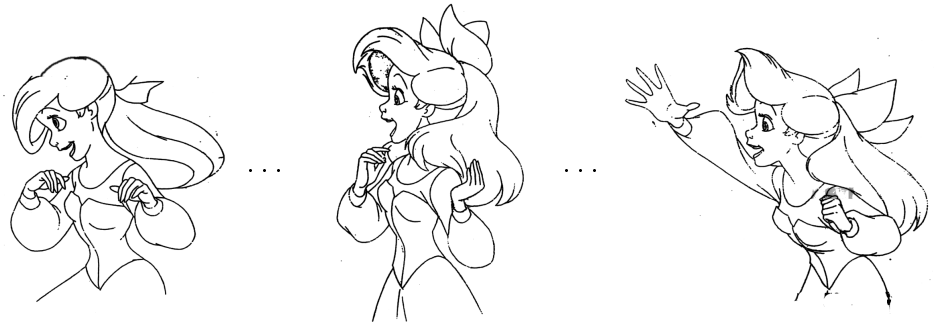
Método de seguimiento basado en fotogramas clave para rotoscopia

En el contexto de la rotoscopia, uno de los pasos más tediosos es el de la generación de los fotogramas situados entre dos fotogramas clave. Aplicando un método de seguimiento (*tracking*), es posible automatizar en gran medida la generación de fotogramas intermedios.

En este capítulo analizaremos un método de seguimiento basado en fotogramas clave para rotoscopia, así como las modificaciones que hemos propuesto con el fin de incrementar la automatización y generalidad del mismo. En primer lugar describiremos el problema del seguimiento y su relación con los objetivos de este proyecto, así como los trabajos previos realizados en este campo. La segunda parte de este capítulo detalla el método de seguimiento utilizado en nuestra aplicación, tras una explicación breve de las curvas Bézier, que son las utilizadas por este método.

2.1. Introducción

Tradicionalmente, la producción de animaciones en dos dimensiones implica una labor de diseño altamente costosa, donde las secuencias de animación tienen que ser elaboradas de una forma completamente manual. El dibujo de cada uno de los fotogramas que constituyen una secuencia de animación es un trabajo tedioso y costoso en tiempo y dinero. Además, este proceso introduce numerosos efectos no deseables tales como parpadeos en la animación resultante, al ser muchas veces imposible mantener la continuidad



(a) (b) (c)

Figura 2.1: (a) Fotograma clave 1 (b) Fotograma intermedio (c) Fotograma clave 2

temporal de las curvas dibujadas a mano entre fotogramas.

El entorno de trabajo más adecuado es aquel en el que el artista se concentra en la tarea creativa de dibujar una serie de fotogramas clave, pudiendo generar posteriormente y de la manera más automática posible los fotogramas intermedios. Así, el artista puede centrarse en el dibujo de los fotogramas importantes, reduciendo el tiempo de desarrollo y por lo tanto los costes de producción. En la Figura 2.1 se muestran dos fotogramas clave (Figuras 2.1a 2.1c) y un fotograma intermedio (Figura 2.1b) dibujado a mano a partir de dichos fotogramas clave siguiendo las técnicas de animación tradicional. En particular, en este proyecto nos centramos en el contexto de la rotoscopia (el artista dibuja sobre una película de vídeo, fotograma a fotograma, con la finalidad de crear una secuencia animada completa), donde también existen diversas técnicas para llevar a cabo el "interpolado" de los fotogramas intermedios. Nuestro objetivo, por lo tanto, es obtener una automatización del proceso de generación de los fotogramas intermedios en la tarea de creación de una animación, minimizando así la intervención del usuario.

En concreto, en este proyecto proponemos una interpolación de trazos que definen características de las imágenes que el artista dibuja en dos fotogramas cercanos dentro de una secuencia de animación. El método presentado en este trabajo realizará la interpolación de dichos trazos a lo largo de toda la secuencia, utilizando la información proporcionada por las imágenes y la correspondencia entre los trazos para guiar la interpolación, permitiendo al usuario refinar los resultados interactuando manualmente con el sistema hasta alcanzar el resultado deseado. La Figura 2.2 muestra una visión global

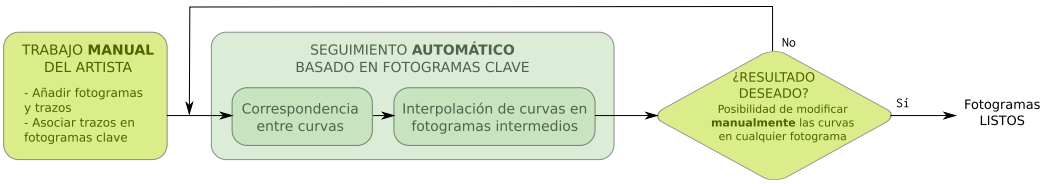


Figura 2.2: Descripción a alto nivel del sistema propuesto

del sistema propuesto en nuestro trabajo. En la resolución de este problema intervienen una serie de cuestiones fuertemente relacionadas entre sí, entre las que destacan el seguimiento automático de formas, la segmentación de imágenes, y el estilizado.

2.2. Estado del arte

El primero de los problemas es el seguimiento (*tracking*) de una forma a lo largo de una secuencia de fotogramas. El objetivo es que, dada una forma o una característica proporcionada en el fotograma inicial, sea posible localizar y seguir de forma automática esa misma característica a lo largo de una secuencia de fotogramas. También, en gran parte de los trabajos que tratan el problema del seguimiento, éste suele ser considerado junto a otro problema estrechamente relacionado: el problema de la segmentación de imágenes, que consiste en, dada una imagen determinada, obtener de forma automática una división de la misma en regiones semánticamente diferentes y significativas. Estas regiones obtenidas tras la segmentación son utilizadas para identificar las características que serán objeto del seguimiento.

En el contexto de la animación existe amplia bibliografía [26, 29, 28, 32, 48, 51, 61, 66, 69, 49]. Estos trabajos se centran en diferentes aspectos relacionados con la animación de figuras. Entre las diferentes propuestas para resolver el problema de la interpolación de los fotogramas intermedios de una secuencia de animación, destacamos la combinación de técnicas de modelado en 2.5 dimensiones y *skeletons* [29]; la utilización de árboles para representar la estructura de las figuras a interpolar [66]; la combinación de modelos en 3D con deformaciones en 2D para dar expresividad a los resultados [51]; y el empleo de vectores de desplazamiento para establecer jerarquías de características y guiar la interpolación [61]. Otros trabajos se centran en ofrecer soluciones completas para el proceso de animación bidimensional. En [32] se

describe una de las primeras aproximaciones utilizada de forma profesional, reemplazando cada uno de los pasos tradicionales por equivalentes electrónicos y realizando la interpolación a través de mínimos cuadrados y un paso de estilizado configurable por el usuario. En [48] se propone una herramienta centrada en la interpolación, optimizando una función de coste para la correspondencia y permitiendo aplicar efectos a los trazos asociando cada uno de ellos con un estilo particular. En [91] se describe una solución haciendo énfasis en resolver el problema de generación de trazos intermedios, empleando para ello un método semi-automático que combina un interfaz que permite al usuario guiar el proceso con una optimización basada en trayectorias logarítmicas espirales de los vertíces de las curvas y medidas de curvatura. En [14] se plantea una solución orientada a replicar el estilo de diferentes artistas en el resultado final, utilizando para ello información estadística sobre los trazos de dicho artista [16], mejorando la coherencia temporal del resultado gracias a una interpolación por pasos basada en algoritmos avaros de búsqueda sobre grafos. Otros trabajos se centran en aplicaciones o dominios específicos tales como el renderizado y la animación de cabellos para dibujos animados [26, 93], el coloreado automático de los fotogramas basándose en regiones [69], entre otros. Recientemente se ha intentado también aplicar técnicas que aprovechan el incremento en la capacidad computacional para abordar el problema desde otros puntos de vista. En [68] se utilizan técnicas de inteligencia artificial y visión artificial para extraer información sobre las imágenes que componen la secuencia y maximizar la cantidad de información compartida entre fotogramas para guiar el proceso de interpolación. En [95] se propone un método basado en aprendizaje máquina semi-supervisado, entrenado con una base de datos que contiene diversas métricas morfológicas de secuencias existentes de animación, para obtener mejores resultados en secuencias nuevas. Este trabajo es continuado y generalizado en [94]. En el contexto de la segmentación, en [89] se describe un sistema para la conversión de una secuencia de vídeo real en una secuencia de animación no fotorrealista. En un primer paso, se realiza una segmentación de la secuencia utilizando el algoritmo descrito en [88], tras lo que el usuario delimita en determinados fotogramas clave las regiones que serán objeto del seguimiento. Estas regiones son interpoladas y suavizadas utilizando la información proporcionada por la segmentación. Finalmente, el usuario puede añadir estilos y trazos a estas regiones para obtener el efecto deseado. Uno de los inconvenientes de este método es que el usuario debe indicar las regiones en fotogramas clave

predeterminados, además de indicar la correspondencia entre ellas de forma manual. En [24] se propone un sistema similar al descrito en [89], utilizando también una representación de la secuencia de vídeo como un volumen espacio-temporal para lograr una mayor coherencia temporal. A diferencia del anterior, la segmentación se realiza para cada fotograma individual, y se asocian las regiones en una fase posterior. El resultado es que se previene el sobre-segmentado que ocurría en el método anterior. También se permite al usuario un mayor control sobre el estilizado y se facilita su interacción en la fase de seguimiento. En [90] se describe un sistema, orientado a su uso con secuencias de vídeo no profesionales, utilizando para la segmentación un algoritmo diferente en el que cada fotograma es guiado por la propagación del flujo de movimiento estimado de las regiones de los fotogramas anteriores, reduciendo la complejidad computacional con respecto a [90]. En [46] se presenta un algoritmo en el que la representación como un volumen espacio-temporal de la secuencia, resultante de la segmentación, es parametrizada con valores que pueden ser ajustados por el usuario para obtener diferentes estilizaciones o efectos. La principal mejora con respecto a los trabajos anteriores es un método iterativo de suavizado entre fotogramas, que disminuye el parpadeo ("*flickering*") de las diferentes regiones durante la secuencia.

En general, los métodos basados en la segmentación tienen la desventaja de tener una complejidad computacional muy elevada, además de presentar problemas de coherencia temporal difíciles de resolver sin intervención por parte del usuario a lo largo de la secuencia. Una de las aproximaciones al problema sin realizar segmentación se encuentra en [4], en el que se utilizan curvas *snakes* [47], esto es, curvas spline bidimensionales que se ajustan a la imagen minimizando una función de energía basada en la intensidad de la imagen. El usuario dibuja los contornos de forma aproximada en el fotograma inicial, y son ajustados a la imagen utilizando las curvas *snakes*. La interpolación de estas curvas en los fotogramas intermedios se realiza utilizando el algoritmo de seguimiento descrito en [77]. El principal inconveniente de este trabajo es que las secuencias que se obtienen son demasiado esquemáticas y con poco detalle, debido a que el sistema fue diseñado para usuarios noveles y no orientado a uso profesional. También, los cambios realizados por el usuario en un fotograma se propagan sólo a los fotogramas posteriores, y al no tener en cuenta el fotograma final al realizar la interpolación, pueden aparecer parpadeos en la animación resultante. Para solucionar estos problemas, en [6] se mejora el método anterior [4] proponiendo un sistema más general

de seguimiento basado en fotogramas clave para rotoscopia. Para llevar a cabo la interpolación de las curvas utiliza una combinación de métodos de seguimiento automático con la intervención del usuario, en concreto, se basa en optimizar una función de energía para obtener las posiciones de los puntos de control de las curvas. El usuario parte de unas curvas denominadas roto-curvas dibujadas en los fotogramas inicial y final, pudiendo realizar modificaciones en cualquiera de las curvas de los fotogramas intermedios, que se propagarán a lo largo de toda la secuencia. El sistema propuesto persigue los mismos objetivos que el descrito en [89], pero es más sencillo y computacionalmente menos costoso al no realizar la segmentación de la secuencia de vídeo.

En nuestra aplicación hemos adaptado el método descrito en [6] con algunas modificaciones para adecuarlo a los objetivos de nuestro trabajo. En concreto, las modificaciones que hemos realizado permiten que las roto-curvas inicial y final sean dibujadas de forma independiente por el usuario, eliminando la restricción de que tengan el mismo número de puntos de control. Esto permite reducir y simplificar la interacción del dibujante con el sistema. Estas modificaciones realizadas sobre el método son completadas en el Capítulo 3. En la Sección 2.4 describimos con más detalle el algoritmo empleado. Antes de detallar el método, en la Sección 2.3 se describen las curvas utilizadas en la aplicación.

2.3. Curvas Bézier

En el contexto de nuestra aplicación, es importante que el usuario sea capaz de dibujar y manipular de una manera flexible los trazos que serán objeto del proceso de seguimiento. También, la representación utilizada por la aplicación y por la interfaz debe permitir al usuario modelar y controlar los trazos de una manera sencilla e intuitiva. Por último, desde el punto de vista computacional, la representación debe ser adecuada para el proceso de seguimiento y permitir un procesado eficiente de los trazos. Para lograr estos objetivos, nuestra aplicación considera cada uno de los trazos como una secuencia de curvas Bézier [71], permitiendo al usuario modificar cualquiera de sus puntos de control para lograr la forma deseada.

Una curva Bézier [71] es un tipo determinado de curva paramétrica. Las curvas y superficies Bézier fueron desarrolladas por el ingeniero francés Pierre Bézier en Renault. Bézier no sólo utilizó estas curvas y superficies para

el diseño de automóviles, sino que también las aplicó en el diseño de alas de aviones, cascos de yates e incluso productos de uso corriente tales como un asiento de la red ferroviaria francesa. Las curvas paramétricas son utilizadas comúnmente en aplicaciones informáticas, como por ejemplo aplicaciones de modelado, CAD, fuentes tipográficas o diseño gráfico, para aproximar modelos del mundo real. El problema en estas aplicaciones es similar al que se presenta en nuestro trabajo: la necesidad de una representación flexible y eficiente de algún contorno o superficie.

2.3.1. Curvas paramétricas

Una primera aproximación para modelar superficies o contornos es utilización de polígonos para aproximar el contorno. Sin embargo, para conseguir que la representación sea precisa suele ser necesario utilizar un gran número de segmentos, lo que hace que sea muy tedioso en el caso de que haya que ajustar alguno de los puntos de la curva. Por este motivo se plantea utilizar en la representación funciones de mayor orden que las funciones lineales, que aproximen el contorno con suficiente detalle y de una forma compacta. Se puede enfocar desde tres diferentes puntos de vista [1] :

1. Expresar las componentes y e z como funciones explícitas de x, esto es, $y = f(x)$, $z = f(x)$. Los principales problemas de esta representación es que no es una representación invariante a rotaciones, además de la dificultad de representar curvas con tangentes verticales, y la imposibilidad de representar elipses o círculos dado que no es posible obtener varios valores de y o de z para un mismo x.

2. Modelar las curvas como soluciones a ecuaciones implícitas de la forma $f(x, y, z) = 0$. Los inconvenientes son que la ecuación puede tener más soluciones de las que se pretenden, y la necesidad de añadir condiciones externas a la ecuación para conseguir evitar estas soluciones. Además, dados dos segmentos consecutivos, no es fácil determinar si sus direcciones tangentes son la misma en el punto en el que se unen.

3. Utilizar la representación paramétrica de las curvas, esto es, $x = x(t)$, $y = y(t)$, $z = z(t)$. En lugar de la utilización de las pendientes, que

[1] Aunque nuestro sistema trabaja con curvas bidimensionales, por completitud y generalidad presentamos las ecuaciones matemáticas en tres dimensiones

pueden ser infinitas, se utilizan vectores tangentes paramétricos, que son siempre finitos. Esto proporciona una serie de ventajas sobre las demás representaciones.

La representación paramétrica de las curvas es la más utilizada para aproximar un contorno. Las funciones x, y e z son polinomios de t, y suelen utilizarse polinomios de grado 3 salvo que sea necesario controlar las derivadas de mayor orden, como por ejemplo en el diseño de superficies aerodinámicas. En nuestro trabajo nos centraremos en las curvas paramétricas cúbicas.

Los polinomios cúbicos que definen un segmento de una curva $Q(t) = [x(t) \quad y(t) \quad z(t)]$ son de la forma:

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x, \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y, \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z, \end{aligned} \tag{2.1}$$

donde $0 \leq t \leq 1$. Si consideramos $T = [t^3 \quad t^2 \quad t \quad 1]$, y la matriz de coeficientes C como:

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \tag{2.2}$$

entonces se puede reescribir la Ecuación 2.1 de la forma:

$$Q(t) = [x(t) \quad y(t) \quad z(t)] = T \cdot C \tag{2.3}$$

2.3.2. Definición de curvas Bézier

Una curva Bézier (ver Figura 2.3), que es un tipo especial de curva NURBS [71], está determinada por su polígono de control (representado en líneas discontinuas en la Figura 2.3). Matemáticamente, una curva paramétrica Bézier se define por:

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t) \tag{2.4}$$

donde B_i son los vértices del polígono de control y $J_{n,i}(t)$ son sus funciones base, también llamadas funciones de Bernstein, que se definen como:

$$J_{n,i}(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i} \tag{2.5}$$

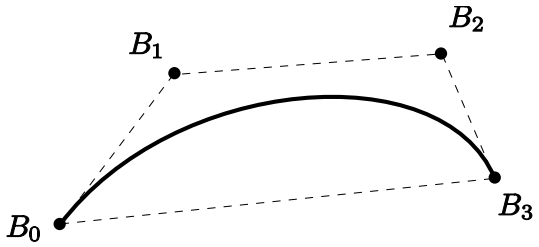


Figura 2.3: Una curva Bézier y su polígono de control

donde

$$\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!}, \qquad 0! \equiv 1 \tag{2.6}$$

y $J_{n,i}(t)$ es la i-ésima función base de orden n. Se puede observar que n, esto es, el grado de la función base de Bernstein y por lo tanto del segmento polinomial, es uno menos que el número de vértices del polígono de control.

Las curvas Bézier tienen las siguientes propiedades:

- Las funciones base son reales.
- El grado del polinomio que define el segmento de curva es uno menos que el número de puntos del polígono de control.
- La curva generalmente sigue la forma del polígono de control.
- El primer y último punto de la curva coinciden respectivamente con el primer y último punto del polígono de control.
- Los vectores tangentes en los extremos de la curva tienen la misma dirección que el primer y último segmentos del polígono de control, respectivamente.
- La curva está contenida dentro de la envoltura convexa del polígono de control, esto es, dentro del polígono convexo de mayor tamaño cuyos vértices son los vértices del polígono de control.
- La curva es invariante a transformaciones afines.

Otra propiedad que se cumple es que para cualquier valor del parámetro t, la suma de sus funciones base es precisamente 1, esto es:

$$\sum_{i=0}^{n} J_{n,i}(t) = 1 \tag{2.7}$$

En el caso de una curva Bézier cúbica, es decir, donde $n = 3$, el polígono de control tendrá cuatro vértices. En este caso, las funciones base que se obtienen son:

$$\begin{aligned} J_{3,0}(t) &= (1)t^0(1-t)^3 = (1-t)^3 \\ J_{3,1}(t) &= 3t(1-t)^2 \\ J_{3,2}(t) &= 3t^2(1-t) \\ J_{3,3}(t) &= t^3 \end{aligned} \tag{2.8}$$

Por lo tanto:

$$\begin{aligned} P(t) &= B_0 J_{3,0} + B_1 J_{3,1} + B_2 J_{3,2} + B_3 J_{3,3} \\ &= (1-t)^3 B_0 + 3t(1-t)^2 B_1 + 3t^2(1-t)B_2 + t^3 B_3 \end{aligned} \tag{2.9}$$

La ecuación de una curva Bézier puede ser expresada en forma matricial. En el caso particular de una curva Bézier cúbica, se puede expresar de la forma:

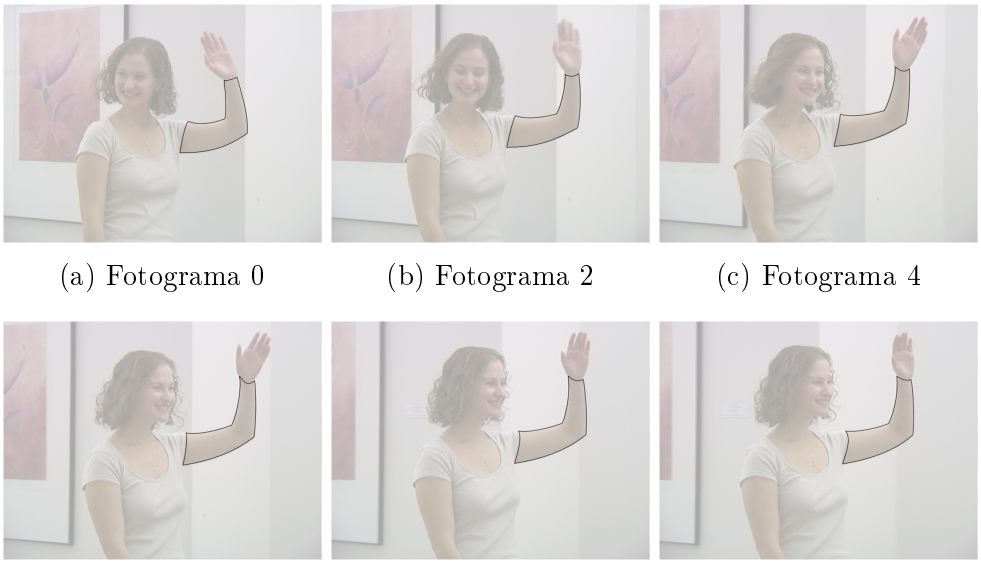
$$P(t) = \begin{bmatrix} (1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{2.10}$$

Agrupando los coeficientes de los términos del parámetro, se puede reescribir como:

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{2.11}$$

En nuestro trabajo, representamos los trazos utilizados por el usuario como una secuencia de curvas Bézier, en las que el punto de control final de un segmento es el punto de control inicial del segmento siguiente. El usuario puede modificar la forma de los trazos moviendo sus puntos de control, y puede también mover los trazos. En el caso de que el trazo sea abierto y tenga m segmentos, el número de puntos de control de ese trazo será de $3m + 1$.

Una explicación más detallada de los diferentes tipos de curvas paramétricas, así como de los cálculos de las matrices base, se puede consultar en [71].



(a) Fotograma 0 (b) Fotograma 2 (c) Fotograma 4

(d) Fotograma 6 (e) Fotograma 8 (f) Fotograma 9

Figura 2.4: Interpolación de fotogramas intermedios a partir de los fotogramas (a) inicial y (f) final

2.4. Método de seguimiento basado en fotogramas clave

El método descrito en [6] para el seguimiento de roto-curvas para rotoscopia combina técnicas de seguimiento automático con la interacción del usuario. En esta sección explicamos el método utilizado en nuestra aplicación, que está basado en [6], así como las modificaciones que proponemos para aumentar la automatización y generalidad del método. Partiendo de una secuencia de fotogramas y una serie de pares de roto-curvas, cada una en un fotograma inicial (ver Figura 2.4a) y un fotograma final (ver Figura 2.4f), el objetivo del sistema es conseguir interpolar las roto-curvas intermedias de manera que sigan las características y su movimiento en los fotogramas intermedios (ver Figuras 2.4b-2.4e). Para simplificar la explicación, a lo largo de este capítulo consideraremos que el usuario parte de un único par de roto-curvas, la primera de ellas situada en el fotograma inicial y la última situada en el fotograma final.

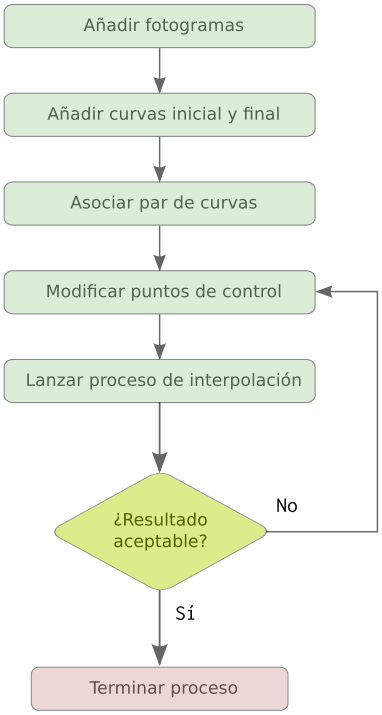


Figura 2.5: Interacción del usuario con el sistema

2.4.1. Procedimiento

En la Figura 2.5 se representa de una manera gráfica el proceso que sigue el usuario para interactuar con nuestra aplicación. La interacción entre el usuario y el sistema está basada en la propuesta presentada en [6], aunque con algunas diferencias sustanciales:

- En [6] se utilizan dos tipos de curvas para determinar el resultado final: roto-curvas y *strokes*. Por una parte, las roto-curvas son curvas que representan las características de la imagen que serán objeto del seguimiento, y son interpoladas en el proceso de optimización. La función de las roto-curvas es servir de referencia o guía en la interpolación de los *strokes*, que son curvas dibujadas por el usuario que determinan el aspecto final de la secuencia y que contienen información de estilo tal como color o ancho de línea. Estos *strokes*, a diferencia de las roto-curvas, no necesitan estar dibujados siguiendo una característica determinada de la imagen. En nuestra aplicación, sólo se considera un

tipo de curvas, que denominaremos trazos. Estos trazos son el equivalente a las roto-curvas empleadas en [6], con información de estilo adicional que representa el color y el ancho de línea.

- El proceso mediante el cual el usuario añade roto-curvas a la escena también difiere. En [6], el usuario parte de una roto-curva que dibuja en el fotograma inicial. Para generar la roto-curva correspondiente en el fotograma final, lo que hace es copiar la roto-curva inicial a ese fotograma, y modificar sus puntos de control para darle la forma adecuada. En nuestro caso, el usuario importa las curvas en los fotogramas clave previamente dibujadas por un artista. Esto facilita la separación de roles en el proceso de animación, permitiendo que un diseñador dibuje los trazos en los fotogramas clave utilizando una herramienta específica, y que posteriormente sean importados para su uso con nuestra herramienta.

- Debido al punto anterior, en nuestra aplicación se presenta un problema adicional que no era necesario considerar en [6]: una vez añadidas roto-curvas a los fotogramas clave, es necesario asignar cada una de las roto-curvas del fotograma inicial a una de las roto-curvas del fotograma final. En [6] la asignación se realiza de forma automática al copiar la roto-curva en el fotograma final. Como se pueden asignar curvas que tengan diferente número de puntos de control, surge a su vez el problema de la correspondencia entre las dos curvas, que consiste en hacer corresponder cada una de las muestras de la curva inicial con una de las muestras de la curva destino. Este problema es tratado con más detalle en el Capítulo 3, en el que se analizan, comparan y prueban los algoritmos más adecuados para establecer la correspondencia entre dos curvas.

Centrándose en la interacción entre el usuario y el sistema, el procedimiento tiene los siguientes pasos:

1. *Añadir fotogramas.* El usuario selecciona las imágenes para cada uno de los fotogramas de la secuencia.

2. *Añadir curvas en los fotogramas clave.* Una vez añadidos los fotogramas, el usuario añade las roto-curvas a la escena en los fotogramas clave. Cada una de las características que serán objeto del seguimiento está definida al menos por un par de roto-curvas: la primera de las roto-curvas en un fotograma s_i y la segunda en un fotograma s_f.

3. *Asociar un par de curvas.* Una vez añadidas las curvas, el usuario asocia una curva en el fotograma s_i con una curva en el fotograma s_f. Las roto-curvas de los fotogramas intermedios son generadas automáticamente.

4. *Modificar puntos de control.* Para cualquiera de las roto-curvas entre los fotogramas s_i y s_f, ambos incluídos, el usuario puede modificar cualquiera de los puntos de control para ajustar la forma de la curva. Además, puede marcar cualquier punto de control como "fijado", lo que hace que su posición permanezca constante durante la optimización. De esta forma el usuario puede influir en la interpolación.

5. *Lanzar proceso de interpolación.* En cualquier momento del proceso el usuario puede lanzar el proceso de interpolación. Este proceso se detalla en la Sección 2.4.3. Al concluir el proceso, si el usuario no está conforme con los resultados obtenidos, puede volver a modificar alguno de los puntos de control y lanzar de nuevo la optimización hasta que los resultados sean adecuados.

El objetivo es optimizar un conjunto de roto-curvas. Cada una de las roto-curvas está parametrizada por una serie de puntos de control, éstos son las incógnitas a resolver en la optimización. En concreto, escribimos cada una de las curvas como C_s, donde s es un índice sobre los fotogramas. Las roto-curvas dibujadas por el usuario están en los fotogramas s_i y s_f, por lo tanto las variables de la optimización son las posiciones de los puntos de control para todos los fotogramas s, donde $s_i < s < s_f$.

La interpolación de las curvas intermedias está basada en la optimización de una función de energía que describiremos en la subsección siguiente. Los términos de energía de la optimización están definidos en base a muestras tomadas en las curvas del fotograma clave inicial s_i, separadas una distancia de aproximadamente un píxel. En el dominio paramétrico, representamos estas muestras como $\{C_s(t_1), C_s(t_2), \ldots, C_s(t_N)\}$. En [6], dado que las roto-curvas de los fotogramas clave se generan a partir de una copia de la roto-curva en el fotograma inicial, se utilizan esas mismas muestras para el resto de los fotogramas.

2.4.2. Función de energía

La función de energía es utilizada para guiar el proceso de seguimiento. Esta función contiene dos tipos de términos: términos de imagen, que guían

las curvas hacia características de la imagen tales como zonas con un alto gradiente de intensidad, y términos de forma, que provocan una transición suave entre las curvas. Gracias a la combinación de ambos tipos de términos, el algoritmo produce resultados adecuados en situaciones en las que una interpolación basada solamente en la forma o en la imagen no sería adecuada.

La función de energía es una combinación lineal de cinco términos de energía:

$$E = \omega_L E_L + \omega_C E_C + \omega_V E_V + \omega_I E_I + \omega_G E_G \tag{2.12}$$

donde los tres primeros términos E_L, E_C y E_V son términos de forma, y los dos últimos E_I y E_G son términos de imagen. En primer lugar describiremos los cinco términos utilizados en [6], y a continuación describiremos los cambios realizados para adaptar estos términos a nuestro trabajo.

Términos de forma

Los términos de forma utilizan la información sobre la forma de las curvas para obtener una transición suave entre las mismas. El primero de los términos E_L se denomina término de longitud y penaliza los cambios en la longitud del vector que une dos muestras consecutivas de una curva:

$$E_L = \sum_{s=s_i}^{s_f-1} \sum_{i=0}^{N-1} \left(\|C_s(t_{i+1}) - C_s(t_i)\|^2 - \|C_{s+1}(t_{i+1}) - C_{s+1}(t_i)\|^2 \right)^2 \tag{2.13}$$

donde N es el número de muestras en cada curva.

El segundo de los términos E_C se denomina término de curvatura, y mide el cambio de curvatura de la curva utilizando para ello una aproximación al cálculo de la segunda derivada:

$$\begin{aligned} E_C = \sum_{s=s_i}^{s_f-1} \sum_{i=0}^{N-1} \| & \left(C_s(t_i) - 2C_s(t_{i+1}) + C_s(t_{i+2})\right) \\ & - \left(C_{s+1}(t_i) - 2C_{s+1}(t_{i+1}) + C_{s+1}(t_{i+2})\right) \|^2 \end{aligned} \tag{2.14}$$

Estos dos términos están basados en los términos propuestos para la interpolación de curvas en [73].

El tercer término E_V es el término de velocidad, y penaliza los cambios bruscos en la posición de las muestras:

$$E_V = \sum_{s=s_i}^{s_f-1} \sum_{i=0}^{N} \|C_s(t_i) - C_{s+1}(t_i)\|^2 \tag{2.15}$$

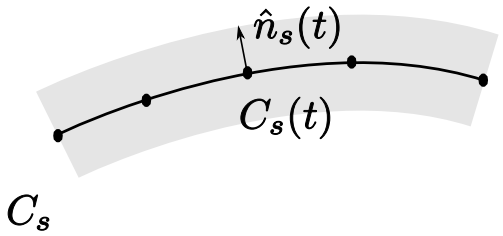


Figura 2.6: Vector unitario normal utilizado en el cálculo de E_I

Términos de imagen

Por su parte, los términos de imagen utilizan la información presente en las imágenes utilizadas como referencia.

El primer término E_I es el término de intensidad de la imagen, y está basado en el algoritmo de seguimiento propuesto en [62]. En ese trabajo se utilizan ventanas cuadradas de la imagen para comparar dos fotogramas, buscando la posición óptima de la ventana en el segundo fotograma que minimice la diferencia con la ventana del primer fotograma. Este concepto es utilizado en el término de intensidad E_I, comparando conjuntos de puntos de la imagen en dos fotogramas consecutivos. Los puntos utilizados para la comparación están situados en la dirección del vector unitario normal a la curva C_s en la muestra i, que denominamos $\hat{n}_s(t_i)$ (ver Figura 2.6). Entonces:

$$E_I = \sum_{s=s_i}^{s_f-1} \sum_{i=0}^{N} \sum_{k=-\alpha}^{\alpha} \|(I_s(C_s(t_i)) + k\hat{n}_s(t_i)) - (I_{s+1}(C_{s+1}(t_i)) + k\hat{n}_{s+1}(t_i))\|^2 \tag{2.16}$$

donde $I_s(p)$ es la intensidad de color en el punto p de la imagen del fotograma s, y k varía entre unos límites $[-\alpha, \alpha]$ definidos por el usuario.

El segundo de los términos E_G se denomina término de gradiente, y mide la diferencia en la magnitud del gradiente de la imagen en puntos de la curva. Sea $G(p)$ la magnitud del gradiente en el punto p de la imagen, calculado como la suma del módulo del vector gradiente de cada uno de los tres colores básicos. Dado que el objetivo es minimizar la función de energía, es necesario utilizar la diferencia entre el gradiente máximo K, definido como el vector gradiente con mayor módulo de todas las muestras de la curva, y el gradiente en el punto p; $G'(p) = K - G(p)$. También, el peso relativo que se le da al gradiente en una muestra i depende del gradiente de esa muestra en los fotogramas inicial y final. Si en los fotogramas inicial o final la muestra no está

situada sobre un punto en el que el gradiente sea elevado, E_G debería tener poca influencia en el cómputo global. Por eso, $G'(p)$ se normaliza utilizando el mínimo M_i de los gradientes en la muestra i en los fotogramas inicial y final:

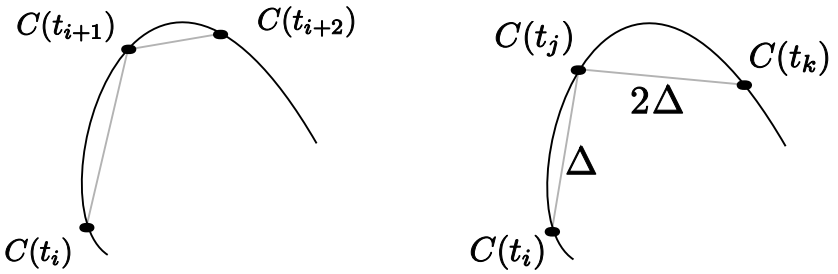
$$M(t_i) = min\left(G'(C_{s_i}(t_i)), G'(C_{s_f}(t_i))\right) \tag{2.17}$$

Entonces se define el término de gradiente como:

$$E_G = \sum_{s=s_i}^{s_f-1} \sum_{i=0}^{N} \left(\frac{G'(C_s(t_i))}{M(t_i)}\right)^2 \tag{2.18}$$

Método de seguimiento en muestreo no uniforme

Los términos de energía descritos anteriormente utilizan un mismo número de muestras que son obtenidas a partir de un muestreo uniforme de las curvas: en primer lugar se muestrea la curva inicial de forma uniforme con N muestras, y se añade por cada muestra de la curva inicial una muestra en la curva final en su misma posición paramétrica. Esta asunción es razonable en el contexto descrito en [6], donde la roto-curva final es creada a partir de una copia de la roto-curva inicial, y tiene por lo tanto el mismo número de puntos de control y esos puntos de control representan en general las mismas características de la curva. Sin embargo, en nuestra propuesta se trabaja directamente con los trazos, donde los trazos inicial y final tienen, en general, un número diferente de puntos de control, lo que hace que sea imposible aplicar el muestreo de la misma manera que en el artículo original. Por este motivo, en nuestro trabajo el muestreo de las curvas no tiene por qué ser uniforme, la distancia entre muestras consecutivas puede variar considerablemente y tampoco se tiene el mismo número de muestras. El problema del muestreo se trata con más detalle en el Capítulo 3. Debido a esta diferencia, el cálculo de los términos de forma se ve afectado ya que los términos de forma utilizan la distancia uniforme entre muestras como base para su cálculo. En concreto, el término de curvatura, E_C, descrito en la Ecuación 2.14 asume que la distancia entre las muestras es la misma para calcular la segunda derivada en un punto de la curva. Si la distancia entre las muestras no es uniforme, el término no representará el cambio de curvatura en ese punto. Para solucionar este problema, proponemos una formulación alternativa. En lugar de utilizar para calcular la curvatura en una muestra $C(t_i)$ la muestras siguientes $C(t_{i+1})$ y $C(t_{i+2})$, se utilizan dos muestras adicionales situadas a



(a) (b)

Figura 2.7: Cálculo del término E_C (a) en el algoritmo original (b) en nuestro trabajo

una distancia uniforme de $C(t_i)$:

$$E_C = \sum_{s=s_i}^{s_f-1} \sum_{i=0}^{N} \|(C_s(t_i) - 2C_s(t_j) + C_s(t_k)) - (C_{s+1}(t_i) - 2C_{s+1}(t_j) + C_{s+1}(t_k))\|^2 \tag{2.19}$$

donde $C(t_j)$ y $C(t_k)$ son dos muestras situadas a una distancia Δ y $2 \cdot \Delta$ respectivamente, medidas sobre la longitud de la curva de la muestra $C(t_i)$. De esta manera, el término estima de forma razonable la diferencia de curvatura entre las dos curvas, independientemente del muestreo. Para poder calcular las muestras $C(t_j)$ y $C(t_k)$ es necesario utilizar una reparametrización de la curva en función de la longitud de arco. Para ello hemos empleado el algoritmo descrito en [40], que se basa en realizar una subdivisión adaptativa gaussiana de la curva para facilitar la búsqueda de puntos determinados en la misma. En la Figura 2.7 se representa la diferencia entre el término de forma propuesto en [6] (ver Figura 2.7a) y el término de forma utilizado en nuestro trabajo (ver Figura 2.7b).

Algunos de los algoritmos de correspondencia implementados, en concreto el método basado en correspondencia entre los vectores tangentes (Sección 3.3), el método basado en la similitud de las curvas (Sección 3.4) y el método guiado por imagen (Sección 3.5) dan lugar a asignaciones uno a varios entre las muestras de cada una de las curvas en el caso general. El algoritmo de seguimiento utiliza para una muestra determinada su muestra anterior y posterior en el tiempo para realizar los cálculos, por lo tanto es necesario un paso posterior a la correspondencia en el que las asignaciones uno a varios son reducidas a asignaciones uno a uno. Para ello se pueden aplicar diferentes estrategias, basadas en el descarte de muestras o en la creación de nuevas

muestras. En la Sección B.2 se describen las posibles estrategias a utilizar.

2.4.3. Método de optimización

El objetivo del algoritmo de seguimiento es minimizar la función de energía definida en la Ecuación 2.12. Es decir, encontrar las posiciones de los puntos de control que minimizan esta función para todos los fotogramas. La función de energía tiene la forma general de un problema de mínimos cuadrados no lineales (*Non Linear Least Squares*, NLLS), y se puede expresar de la forma:

$$E = \sum_{K} \omega_k \left\| \mathbf{f}_k(\vec{x}) \right\|^2 \tag{2.20}$$

donde $\vec{x}$ es el vector de incógnitas. Por otra parte, cuando el usuario marca alguno de los puntos de control de las curvas como fijo, la posición de ese punto de control deja de ser una incógnita y pasa a ser una restricción a la hora de optimizar la función de energía.

En la Figura 2.8 se representa de una manera esquemática el proceso de optimización. Para la resolución del problema de optimización NLLS, uno de los métodos más conocidos es el método de Levenberg-Marquardt (LM) [65]. Este método es un algoritmo iterativo que calcula una solución numérica al problema de minimización de una función. El algoritmo requiere el cómputo de la matriz jacobiana de cada $\mathbf{f}_k(\vec{x})$. Se define el jacobiano de cada término de energía con respecto a todos los puntos muestreados $C_s(t_i)$ en todos los fotogramas como:

$$(\mathbf{J}_k)_j = \frac{\partial \mathbf{f}_x(\vec{x})}{\partial C_j} \tag{2.21}$$

donde j es un índice sobre todos las muestras de todas las curvas. Se utiliza una matriz de cambio de base $\mathbf{B}$ para pasar las muestras a puntos de control de las roto-curvas, siendo por lo tanto $\mathbf{J}_k\mathbf{B}$ la matriz jacobiana de $\mathbf{f}_k(\vec{x})$.

El cálculo del jacobiano es inmediato para los términos de forma, presentando dificultades sólo en el caso de los términos de imagen. Para el cálculo de la derivada de $I_t(\mathbf{p})$ se utiliza una aproximación similar al método de Lucas-Kanade [62]. El cálculo de la derivada de $G(p)$ también se realiza de una forma análoga. Para calcular la derivada del vector unitario normal $\hat{n}_s(t_i)$ es necesario tener en cuenta la normalización aplicada a ese vector. Como la longitud de ese vector tiende a variar lentamente de un fotograma al fotograma posterior debido al término de longitud, se simplifica la influencia de la

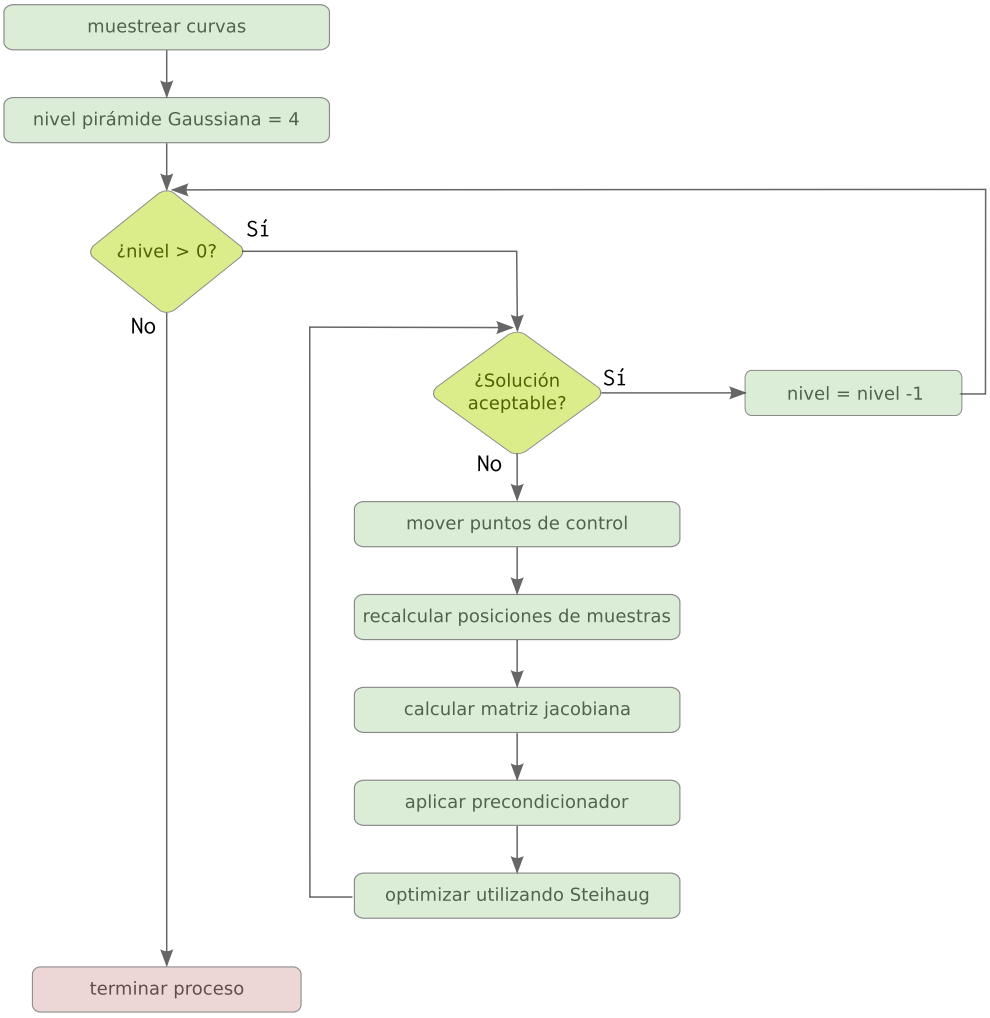


Figura 2.8: Proceso de optimización

normalización suponiendo que el factor de normalización es constante. Así, la derivada del vector normal normalizado es aproximada por la derivada del vector normal no normalizado, que es una combinación lineal de los cuatro puntos de control del segmento al que pertenece la muestra.

Debido a que la matriz jacobiana utilizada a lo largo del algoritmo es muy dispersa y el número de elementos es relativamente grande, se utiliza para la resolución del problema NLLS una variante más adecuada del método de Levenberg-Marquardt, que es el método de Steihaug [79]. Para evitar problemas si el sistema no está bien condicionado se utiliza el precondicionador descrito en [82].

1: Muestrear las curvas
2: **for** $i \leftarrow 4 \ldots 1$ **do**
3: $\Delta \leftarrow 10$, $\delta \leftarrow \infty$
4: **while** ($\Delta > 0{,}5$) $\wedge$ ($\delta > 0{,}1$) **do**
5: Recalcula posiciones de la muestras en función de Z_0
6: Calcular Jacobianos J en función de Z_0, nivel i
7: Aplicar precondicionador a J
8: Aplicar $Steihaug(J, \Delta)$ para obtener la solución Z_1
9: $\delta \leftarrow$ el máximo desplazamiento entre Z_0, Z_1
10: Aumentar / reducir Δ en función de Z_0, Z_1
11: **if** Z_1 mejora Z_0 **then**
12: $Z_0 \leftarrow Z_1$
13: **end if**
14: **end while**
15: **end for**

Figura 2.9: Algoritmo de optimización

Por último, la optimización se realiza desde un nivel grueso a un nivel fino utilizando una pirámide Gaussiana de cuatro niveles, de la manera descrita en [15], para evitar que la optimización caiga en mínimos locales.

En la Figura 2.9 se concretan los pasos que sigue el algoritmo en nuestra implementación. La solución al problema de la optimización es un vector que contiene las posiciones de los puntos de control de todas las curvas, representada en la figura como Z. Comenzando por la representación más gruesa de la imagen, en cada iteración se busca la solución que optimice la función de energía. En cada una de las iteraciones del bucle más interno (líneas 4-14), se parte de la última solución tomada como válida, representada por Z_0, siguiendo los pasos descritos en la figura. En primer lugar, se recalculan las posiciones de cada una de las muestras de acuerdo con los puntos de control de la última solución Z_0 (línea 5), para poder calcular la matriz jacobiana J (línea 6). Posteriormente se aplica el precondicionador de Szeliski a la matriz resultante (línea 7), y se buscan los puntos de control que optimizan la función de energía aplicando el método de Steihaug (línea 8). Esta matriz representa la solución candidata, y es representada por Z_1. Comparando la solución anterior Z_0 con la solución candidata Z_1 se realizan modificaciones sobre los parámetros δ, que es el desplazamiento máximo que sufre un pun-

to de control entre Z_0 y Z_1, y Δ, que es el radio del intervalo de confianza utilizado en el método de Steihaug (líneas 9-10). En el último paso del bucle (líneas 11-12), se comprueba si la solución Z_1 mejora la solución anterior Z_0, en cuyo caso se intercambian. El bucle más interno termina cuando se alcanza un radio de confianza Δ lo suficientemente pequeño, o el desplazamiento máximo δ de un punto de control entre las dos soluciones Z_0 y Z_1 es menor que un determinado umbral. En nuestra implementación, hemos utilizado para estos umbrales los valores $\Delta > 0,5$ y $\delta > 0,1$, también empleados en [6].

Capítulo 3

Comparativa de diferentes algoritmos de correspondencia

En el proceso de seguimiento (*tracking*), uno de los factores que determinan la calidad de la interpolación resultante es la forma de establecer la correspondencia entre las curvas inicial y final. Una correcta identificación y asignación de las características de las dos curvas ayudará a guiar el seguimiento hacia un resultado más natural y visualmente atractivo para el usuario.

En este capítulo analizaremos diferentes algoritmos para establecer la correspondencia entre dos curvas paramétricas, así como su rendimiento y aplicación a nuestro trabajo. En la primera parte analizaremos los trabajos previos realizados en este campo, continuando con una descripción formal del problema de la correspondencia, y concluyendo con aquellos métodos de correspondencia que hemos seleccionado para su implementación.

3.1. Introducción

La necesidad de hallar de forma automática (o con la menor intervención posible por parte del usuario) la correspondencia óptima entre dos curvas a priori diferentes surge al permitir que las curvas inicial y final tengan un número diferente de muestras y/o de puntos de control. En nuestro trabajo, las curvas inicial y final, si bien representan la misma característica a lo largo del tiempo, son curvas dibujadas por el usuario de forma independiente, cada una en fotogramas diferentes. El usuario tiene libertad para utilizar los puntos

de control que crea necesarios para definir la curva, independientemente de los que haya utilizado en la curva correspondiente, al contrario de lo que ocurre en el método de seguimiento original [6].

En el método de seguimiento original, el usuario realiza una copia del trazo inicial en uno de los fotogramas posteriores, y edita sus puntos de control para darle la forma deseada. En este caso, la correspondencia entre las dos curvas se establece muestreando de forma uniforme la curva inicial C_A y por cada muestra de la curva inicial se añade una muestra a la curva final C_B, asignando cada muestra C_A^i de la curva inicial a la muestra C_B^i de la curva final. La muestra añadida estará en la misma posición paramétrica que la muestra correspondiente en la curva inicial. Esta asignación sólo puede ser realizada en el caso de que las curvas tengan el mismo número de puntos de control. En nuestro trabajo hemos intentado hacer el método más general, eliminando esta restricción y permitiendo que las curvas inicial y final tengan un número diferente de puntos de control. Al introducir esta generalización, la aproximación descrita anteriormente ya no puede ser aplicada, puesto que incluso en el caso en que las curvas inicial y final tengan el mismo número de puntos de control no se puede asumir que el punto de control P_B^i represente al punto de control P_A^i interpolado.

La correspondencia entre curvas paramétricas es un problema importante en multitud de contextos, entre los que destacan *tracking* [6, 17, 38], *morphing* [8, 48, 55], animación o modelado. Diferentes autores han considerado esta cuestión y han intentado resolverla aplicando diferentes técnicas y enfoques. Atendiendo a la forma en que los algoritmos muestrean o procesan puntos de la curva, podemos clasificarlos en *dense matching*, *feature matching* y *skeleton matching*.

Los algoritmos *dense matching* se caracterizan por representar cada curva como una secuencia de muestras y formular el problema como la minimización de una función de coste. Esta función de coste representa el coste de hacer corresponder una muestra determinada C_A^i de la curva inicial con una muestra determinada C_B^j de la curva final. Uno de los primeros artículos [73] propone una aproximación en la que la función de coste es la representación matemática del trabajo de torsión y elongación que sería necesario para convertir un polígono en otro. Uno de los principales inconvenientes de este método es su dependencia de una serie de parámetros, que representan las propiedades físicas del material del que estarían hechos los polígonos, que el usuario debe ajustar para obtener el resultado adecuado. En [38] se intenta

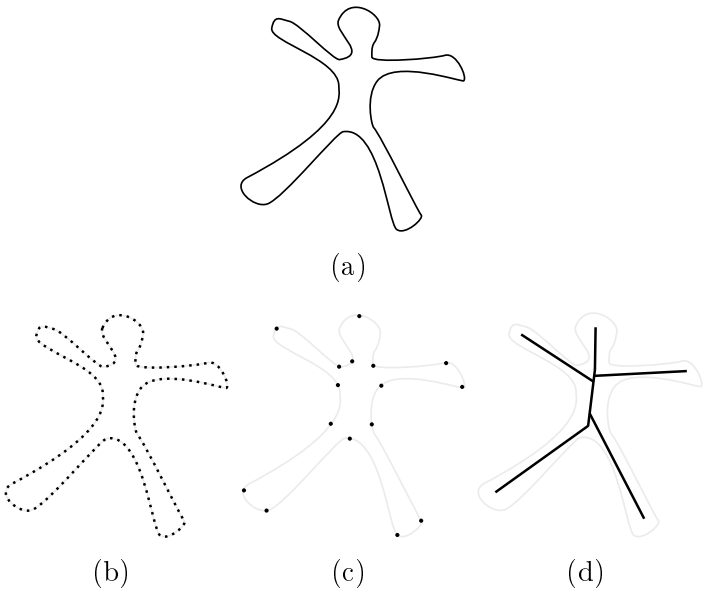


Figura 3.1: Representaciones de la curva original (a) utilizadas en *dense matching* (b), *feature matching* (c) y *skeleton matching* (d)

resolver el problema de la correspondencia entre dos curvas utilizando una función de coste diferente basada en la distancia entre las muestras y aplicando programación dinámica para resolver el problema de manera eficiente. El artículo considera la correspondencia como uno de los pasos del seguimiento automático de imágenes, lo que hace que los resultados del algoritmo sean muy dependientes de las características de las curvas consideradas. En [23] se extienden las ideas anteriores intentando incrementar la robustez del algoritmo. Para ello considera una nueva función de coste basada en propiedades geométricas de las curvas, en concreto, en la curvatura de las curvas, y aplica también programación dinámica. Sin embargo, no garantiza que el método sea invariante a rotaciones, y presenta problemas si la orientación de las curvas es diferente. En [96] se emplean técnicas de programación difusa para resolver el problema de correspondencia entre vértices de un polígono. La estrategia consiste en construir un grafo difuso en el que cada nodo es una variación del polígono inicial, y utilizar una función de similitud para evaluar el grado de semejanza de cada par de polígonos. Esta función de similitud relaciona las posiciones de los vértices de los polígonos y la longitud de sus lados, permitiendo al algoritmo seleccionar a los polígonos que son más parecidos entre sí. En [45] se mejora la invarianza a rotación y la

robustez del algoritmo propuesto en [23] utilizando una función de coste diferente que está definida no sólo en términos de la curvatura de las curvas, sino también en términos de la posición relativa de las muestras que se quieren hacer corresponder. En [50] se añade a la función de coste un término que evalúa la similitud de intensidad de color, utilizando la información proporcionada por las imágenes sobre las que fueron dibujadas las curvas. El coste computacional de esta aproximación es el más elevado de todos los métodos considerados, y presenta problemas similares a [23], ya que el término geométrico de la función de coste es el mismo.

Los algoritmos *feature matching* se caracterizan por intentar obtener la correspondencia entre algunos puntos característicos (*feature points*) de las dos curvas, en lugar de hacerlo sobre todos los puntos de la secuencia de muestras. Uno de los problemas asociados a éste es la obtención automática de los puntos característicos de las curvas. En [37] se construye una representación sintáctica de las curvas, en términos de los segmentos definidos entre dos puntos característicos, su orientación y su longitud. El problema de la correspondencia se formula como la búsqueda de la combinación óptima de operadores sintácticos necesarios para convertir la curva inicial en la final; por ejemplo, eliminación, sustitución, inserción, *gap insertion* y unión. En [55] se presenta un método basado en la minimización de una función de similitud entre los puntos característicos de cada curva. En un primer paso del algoritmo se extraen automáticamente los puntos característicos [21], y posteriormente se halla la correspondencia entre esos puntos aplicando programación dinámica. En [54] se describe una aproximación genérica para la extracción de puntos característicos basada en descriptores. Estos descriptores utilizan diferentes técnicas (puntos, ajuste lineal, media y desviación típica, momentos y coeficientes de Fourier) para identificar líneas, curvas y regiones utilizando las información de las imágenes como apoyo. En [7] se presenta un método de reconocimiento de curvas en imágenes invariante a traslación, dentro del cual se utiliza una aproximación basada en cálculos sobre los ángulos de las curvas para identificar los puntos característicos, almacenando además otras características de las mismas. Esta información es utilizada durante el establecimiento de la correspondencia, que es realizada con diferentes niveles de granularidad para hallar el resultado óptimo.

Tanto en los algoritmos *dense matching* como *feature matching* una línea de investigación destacable es la relacionada con la *distancia de Fréchet* [9], que en su definición original es la distancia mínima de una "cuerda" que une

los puntos de cada una de las curvas considerando que sólo se puede avanzar hacia adelante en ambas curvas desde su punto inicial al final. La sensibilidad de la distancia de Fréchet con respecto a la presencia de puntos atípicos es uno de sus principales inconvenientes. En [31] se presenta un método computacionalmente asequible (*"Dynamic Time Warping"*) para calcular la correspondencia entre dos curvas utilizando la distancia de Fréchet promediada en lugar de la definición tradicional, mejorando su sensibilidad con respecto a puntos atípicos. En [11] y [92] se proponen también aproximaciones para mejorar la robustez de la distancia de Fréchet, modificando el cálculo para permitir la utilización de "atajos" en el recorrido de las curvas que detectan e ignoran determinados segmentos conflictivos de las curvas empleando algoritmos de toma de decisiones y selección. En [33] se generaliza la distancia de Fréchet para poder ser calculada en curvas que presenten "huecos" y discontinuidades, utilizando un proceso de búsqueda para la construcción de un grafo que reconstruye aquellos tramos de la curva no presentes. La complejidad del cálculo de la distancia de Fréchet es un tema recurrente en la literatura: en [19] se demuestra que el cálculo de la distancia de Fréchet permitiendo discontinuidades es un problema *NP-completo*, y propone dos aproximaciones computacionalmente razonables para resolverlo. En otros artículos [75] [63] [42] se describen las complejidades de algunas de las variaciones de la distancia de Fréchet y diversas técnicas y algoritmos para aproximar la solución de forma eficiente.

En los algoritmos *skeleton matching* uno de los primeros pasos es extraer el *skeleton* de cada una de las curvas, que es una representación esquemática de la topología de la misma. En [20] se introduce el concepto de *skeleton* como una técnica complementaria a la animación tradicional basada en fotogramas clave. El usuario dibuja representaciones muy esquematizadas de cualquiera de los fotogramas indicando sus puntos clave, y el sistema utiliza esa información para mejorar la calidad de los fotogramas obtenidos al interpolar. En [76] la estructura de las curvas se representa mediante una estructura denominada *star-skeleton*. Cada uno de los polígonos originales es dividido de forma automática en un conjunto de polígonos que tienen al menos un vértice visible desde todos los demás. Este conjunto de polígonos y la forma en que se relacionan entre sí constituyen el *star-skeleton*. Uno de los inconvenientes de este método es que el cálculo de la correspondencia es semi-automático, necesitando que el usuario establezca la correspondencia entre algunos de los vértices de los polígonos iniciales. En [64] se presenta un

sistema que comprende desde la digitalización de las imágenes dibujadas por un artista hasta la generación de la interpolación. Tras vectorizar las figuras en un primer paso, se analizan las curvas que las componen para extraer un grafo que contiene sólo los trazos que definen la figura, basándose en su interacción con los puntos característicos de la misma. La correspondencia entre las dos representaciones la obtiene aplicando un algoritmo general de correspondencia entre dos grafos [25]. De forma similar, en [70] se segmenta cada imagen en regiones, y se obtiene la correspondencia entre las figuras utilizando una heurística basada en la longitud y los puntos característicos de las curvas que la componen, además de las relaciones de cada curva con sus curvas vecinas.

Por lo general, los algoritmos *skeleton* no son adecuados a nuestro problema, ya que suelen estar orientados a la correspondencia entre curvas cerradas y necesitan en general una gran cantidad de intervención por parte del usuario para obtener los resultados deseados.

En este trabajo hemos implementado y analizado varios de estos algoritmos que etiquetamos según su característica de correspondencia principal. Entre los algoritmos de *dense matching* estudiaremos un método basado en correspondencia entre los vectores tangentes [23], un método basado en la similitud de las curvas [45], un método guiado por imagen [50], y un método basado en el vector de desplazamiento [38]. Así mismo, en la Sección A.1 hemos descrito también un algoritmo del tipo *feature matching*: un método basado en propiedades geométricas de un conjunto característico de puntos [55].

A continuación se define la correspondencia entre dos curvas paramétricas, y posteriormente presentamos un breve resumen de los métodos implementados.

3.2. Correspondencia entre dos curvas paramétricas

Hallar la correspondencia entre dos curvas paramétricas $C_A(u)$ y $C_B(v)$ consiste en definir una función biyectiva $\mathbb{V} : u \rightarrow v$ de la curva C_B tal que cada punto de la curva inicial $C_A(u)$ se corresponda con un punto de la curva final $C_B(\mathbb{V}(u))$. Este problema se representa en la Figura 3.2. Desde el punto de vista de nuestro trabajo, las características ideales de la reparametrización

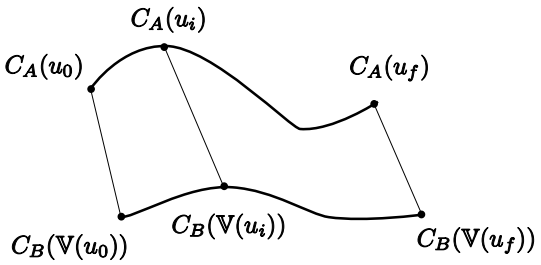


Figura 3.2: Correspondencia entre dos curvas paramétricas C_A y C_B

$\mathbb{V}(u)$ son:

1. **Monotonía**: La reparametrización debe conservar el orden y la orientación de los elementos asignados. Se debe cumplir que $\mathbb{V}(u_i) \leq \mathbb{V}(u_j)$, $\forall u_i < u_j$.

2. **Coincidencia de extremos**: Si la curva es abierta, debe cumplirse la restricción $\mathbb{V}(u_0) = v_0$ y $\mathbb{V}(u_f) = v_f$. En caso de que la curva sea cerrada, la restricción que debe cumplirse es $\mathbb{V}(u_0) = \mathbb{V}(u_f)$.

3. **Preservación de las características de la curva**: La reparametrización debe hacer corresponder las características de interés de las dos curvas tales como puntos de máxima curvatura, miembros, y otros puntos o zonas de la curva destacadas.

4. **Invariante a transformaciones geométricas**: Idealmente, la reparametrización debe ser invariante a traslación, homotecia y rotación. También, en el caso de curvas cerradas, debe ser invariante a la elección de los puntos iniciales y finales de las curvas, así como a la orientación de las curvas.

Es importante destacar que la calidad de la correspondencia entre dos curvas no puede medirse de forma "definitiva" utilizando métricas, en muchos casos intervienen criterios subjetivos o depende de la aplicación que estemos considerando. En lugar de una correspondencia óptima, suele haber un conjunto de correspondencias "aceptables" más o menos amplio dependiendo de las curvas consideradas.

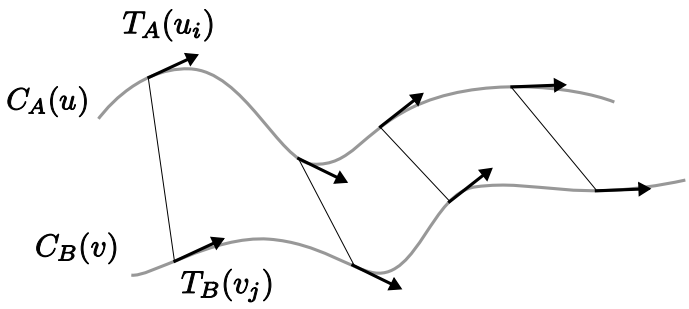


Figura 3.3: Correspondencia entre los vectores tangentes de dos curvas

3.3. Método basado en correspondencia entre los vectores tangentes

En este método [23] se establece la correspondencia entre dos curvas paramétricas utilizando una función de coste que considera la diferencia de los cambios de curvatura. Al contrario que en aproximaciones anteriores [73, 38], se considera que para conseguir una correspondencia adecuada no pueden utilizarse solamente propiedades geométricas básicas tales como la distancia entre los puntos de las dos curvas, y es necesario recurrir a otras propiedades de la forma con una continuidad de mayor orden. Su propuesta consiste en utilizar la relación entre los vectores tangentes normalizados de cada una de las curvas, ya que representan los cambios en la curvatura.

Dada una curva paramétrica $C(u)$ regular, esto es, una curva para la que existe la derivada, ésta es continua y verifica que $\|C'(u)\| > 0$, $\forall u$, condición que equivale geométricamente al hecho de que exista el vector tangente a la curva en el punto $C(u)$, en $\mathbb{R}^n, n > 0$, se define el vector tangente normalizado como:

$$T(u) = \frac{\frac{dC(u)}{du}}{\left\|\frac{dC(u)}{du}\right\|} = \frac{C'(u)}{\|C'(u)\|} \tag{3.1}$$

De esta forma, la correspondencia entre dos curvas paramétricas se enfoca en cómo establecer la correspondencia entre los vectores tangentes de las curvas, utilizando para ello el producto escalar $\langle T_A(u_i), T_B(v_i)\rangle$. Entonces si $\langle T_A(u_i), T_B(v_i)\rangle$ es 1 implica que los vectores son paralelos y tienen la misma dirección (ver Figura 3.3). Basándose en esta propiedad, el problema de encontrar la correspondencia se convierte en buscar la reparametrización

$\mathbb{V}$ que relacione los campos tangentes de las curvas de forma que cada uno de los productos escalares sea lo más cercano posible a 1.

En el caso ideal en el que $\langle T_A(u), T_B(\mathbb{V}(u))\rangle = 1, \forall u$ se considera que se ha establecido una correspondencia completa entre las dos curvas. Por lo general, no es posible establecer una correspondencia completa entre dos curvas cualquiera, pero esto nos permite obtener una idea de la calidad de la correspondencia, siendo mejor cuanto más se aproxime a la correspondencia completa. Entonces podemos definir una correspondencia óptima entre dos curvas como la que maximiza la siguiente función de coste:

$$\mathbb{F} = max_{\mathbb{V}(u)} \int_{u_0}^{u_f} \langle T_A(u), T_B(\mathbb{V}(u))\rangle du \tag{3.2}$$

con $\mathbb{V}(u_0) = v_0$ y $\mathbb{V}(u_f) = v_f$, de forma que las muestras iniciales y finales de cada curva se corresponden entre sí. Como solución se obtiene la reparametrización $\mathbb{V}(u)$. Resolver esta ecuación de forma analítica no es fácil para el caso general. Una solución es la utilización de técnicas de programación dinámica [74] sobre una representación discreta de las dos curvas con las que se puede calcular una solución aproximada con complejidad temporal polinómica.

Sean C_A^i, $0 \leq i < m$, y C_B^j, $0 \leq j < m$ dos secuencias de m muestras uniformes de $C_A(u)$ y $C_B(v)$, respectivamente. Sean T_A^i, $0 \leq i < m$, y T_B^j, $0 \leq j < m$ dos secuencias de vectores tangentes en las posiciones de las muestras anteriores. La Ecuación 3.2 se puede expresar de forma discreta por la siguiente ecuación:

$$\mathbb{F} = max_{\mathbb{V}(i)} \sum_{i=0}^{m-1} \langle T_A^i, T_B^{\mathbb{V}(i)}\rangle \tag{3.3}$$

con $\mathbb{V}(0) = 0$, $\mathbb{V}(m-1) = m-1$ y $\mathbb{V}(i) \leq \mathbb{V}(i+1)$.

Sea $\mathbb{F}_{i,j}$ el coste óptimo de hacer corresponder las primeras i muestras de C_A con las primeras j muestras de C_B. El valor de $\mathbb{F}_{i,j}$ se puede definir en términos del coste necesario para hacer corresponder las muestras anteriores y del coste de establecer la correspondencia entre las dos muestras C_A^i y C_B^j. El coste de establecer la correspondencia entre dos muestras C_A^i y C_B^j lo denotaremos por:

$$f_{i,j} = \langle T_A^i, T_B^{\mathbb{V}(i)}\rangle \tag{3.4}$$

Puesto que no se permiten saltos en la asignación de muestras, hay tres opciones para seleccionar la asignación anterior a (i,j): $(i-1, j-1)$, $(i-$

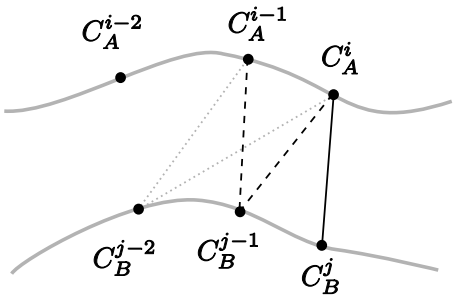


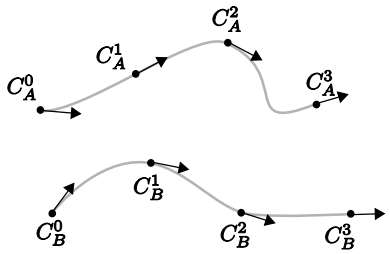
Figura 3.4: Posibles correspondencias anteriores a (C_A^i, C_B^j).

$1, j)$ ó $(i, j-1)$, entre las cuales se elige la que tenga el coste mínimo. Esta situación se representa en la Figura 3.4: dada una correspondencia entre dos muestras C_A^i y C_B^j, representadas con trazo continuo en la figura, se representan las correspondencias anteriores posibles con trazo discontinuo. Las correspondencias no válidas se muestran con línea punteada. Entonces,

$$\mathbb{F}_{i,j} = min\left\{\mathbb{F}_{i-1,j-1}, \mathbb{F}_{i-1,j}, \mathbb{F}_{i,j-1}\right\} + f_{i,j} \tag{3.5}$$

siendo $\mathbb{F}_{i,j} = 0$ para $i = j = -1$. La propuesta es aplicar programación dinámica para calcular $\mathbb{F}_{m-1,m-1}$ de un modo eficiente, obteniendo así la correspondencia de menor coste. En representación matricial, con los índices de las muestras de la curva C_A en el eje x y los índices de las muestras de la curva C_B en el eje y, es posible alcanzar la solución en tiempo $\mathcal{O}(m^2)$ calculando la matriz fila por fila, de izquierda a derecha, puesto que $\mathbb{F}_{i,j}$ depende solamente de tres de sus vecinos en la matriz, $\mathbb{F}_{i-1,j-1}$, $\mathbb{F}_{i-1,j}$ y $\mathbb{F}_{i,j-1}$, que ya han sido calculados. Una vez calculado $\mathbb{F}_{m-1,m-1}$, la correspondencia se obtiene hallando el camino óptimo recorriendo la matriz en sentido inverso. El número de nodos de este camino tiene como límite inferior m (sería lo correspondiente a recorrer la diagonal) y como límite superior $2m-1$.

Se puede ver un ejemplo y la correspondencia resultante en la Figura 3.5. En este ejemplo se muestran dos curvas muestreadas con $m = 4$ para las que se calcula su correspondencia. Para mayor claridad, se muestra la matriz resultante de calcular todos los productos de vectores tangentes (ver Figura 3.5b), en la que cada producto $f_{i,j} = \langle T_A^i, T_B^j \rangle$ puede ser calculado independientemente de los demás. El algoritmo de programación dinámica comienza calculando $\mathbb{F}_{0,0}$, y termina cuando se calcula el coste del par $\mathbb{F}_{m-1,m-1} = \mathbb{F}_{3,3}$. Como se indica en la Figura 3.5c, para que la muestra inicial de la curva C_A se asigne siempre a la muestra inicial de la curva C_B, para todos aquellos



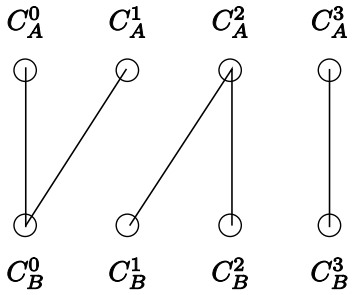
(a)

	0	1	2	3
0	**0.7**	**0.2**	0.4	0.5
1	0.3	0.8	**0.5**	0.6
2	0.5	0.4	**0.5**	0.7
3	0.2	0.6	0.7	**0.4**

(b)

	-1	0	1	2	3
-1	0	8	8	8	8
0	8	**0.7**	**0.9**	1.6	2.1
1	8	1.0	1.5	**1.3**	1.9
2	8	1.5	1.4	**1.5**	2.0
3	8	1.7	2.0	2.1	**1.9**

(c)



(d)

Figura 3.5: Ejemplo del cálculo de la correspondencia aplicando programación dinámica. (a) dos curvas muestreadas con m=4 (b) $f_{i,j}$ (c) matriz calculada mediante programación dinámica (d) correspondencia óptima.

pares de muestras en los que $i = -1$ ó $j = -1$ se les asigna el valor $2 * m = 8$, salvo en el caso en el que $i = j = -1$, en cuyo caso se le asigna el valor 0. El algoritmo va construyendo la matriz primero por filas y después por columnas, seleccionando para cada nodo $\mathbb{F}_{i,j}$ el nodo de menor valor de sus tres vecinos $\mathbb{F}_{i-1,j-1}$, $\mathbb{F}_{i-1,j}$ y $\mathbb{F}_{i,j-1}$ y sumándole el valor de la matriz $f_{i,j}$. Al terminar el algoritmo, partiendo del nodo $\mathbb{F}_{m-1,m-1}$, se construye el camino óptimo que va hasta el nodo $\mathbb{F}_{0,0}$ seleccionando el nodo vecino más pequeño. En la Figura 3.5c se muestra en negrita el camino óptimo resultante, y en la Figura 3.5d se muestran las asignaciones resultantes.

Se observa que bajo determinadas condiciones, es demasiado restrictivo intentar conseguir correspondencias completas, esto es, $\langle T_A(u), T_B(\mathbb{V}(u))\rangle = 1, \forall u$. En lugar de ello, se intenta conseguir una correspondencia válida, en la que $\langle T_A(u), T_B(\mathbb{V}(u))\rangle > 0, \forall u$.

Una de las características del resultado es la no existencia de autointersecciones en el caso de una correspondencia válida. Sin embargo, en la búsqueda de una mayor generalidad los autores proponen una modificación del méto-

do. Esta modificación persigue que la correspondencia asegure que el plano definido por la sucesión de curvas esté libre de autointersecciones. Entonces, la Ecuación 3.2 se redefine de la forma:

$$\mathbb{F} = max_{\mathbb{V}(u)} \int_{u_0}^{u_f} \frac{\langle T_A(u) \times \{C_A(u) - C_B(\mathbb{V}(u))\}, T_B(\mathbb{V}(u)) \times \{C_A(u) - C_B(\mathbb{V}(u))\}\rangle}{\|C_A(u) - C_B(\mathbb{V}(u))\|^2} du \tag{3.6}$$

La Ecuación 3.3 pasa a ser:

$$\mathbb{F} = max_{\mathbb{V}(i)} \sum_{i=0}^{m-1} \frac{\langle T_A^i \times (C_A^i - C_B^{\mathbb{V}(i)}, T_B^{\mathbb{V}(i)} \times (C_A^i - C_B^{\mathbb{V}(i)}\rangle}{\|(C_A^i - C_B^{\mathbb{V}(i)})\|^2} \tag{3.7}$$

Y la Ecuación 3.4 se convierte en:

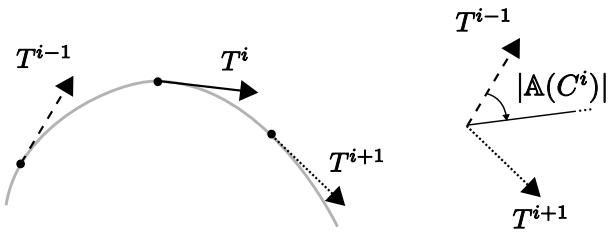
$$f_{i,j} = \frac{\langle T_A^i \times (C_A^i - C_B^j), T_B^{\mathbb{V}}(i) \times (C_A^i - C_B^j)\rangle}{\|(C_A^i - C_B^{\mathbb{V}(i)})\|^2} \tag{3.8}$$

En nuestro trabajo hemos implementado ambas variantes, comprobando que la primera de ellas es la que presenta mejores resultados en nuestra aplicación.

Uno de los problemas del método descrito es que los campos de vectores tangentes son invariantes sólo a traslación, y no a rotación. Así, la correspondencia obtenida al aplicar el algoritmo a las curvas C_A y C_B será diferente a la obtenida al aplicarlo a las curvas C_A y C_{B^R} (donde C_{B^R} es la curva C_B tras haberle sido aplicada una rotación). Este problema puede ser remediado aplicando en un paso previo una rotación a una de las curvas de tal forma que los vectores T_A^i y T_B^j queden alineados para dos valores determinados i y j que se sabe que pertenecerán a la correspondencia resultante. De todas maneras, la independencia con respecto a rotación sólo puede ser conseguida cuando se establece una correspondencia completa entre las curvas. También, se propone una segunda mejora al algoritmo cuando es aplicado a curvas cerradas que hace que presente invarianza a la selección de los puntos inicial y final de las curvas. Sin embargo, esto hace que la complejidad computacional aumente a $\mathcal{O}(m^3)$.

3.4. Método basado en la similitud de las curvas

En este método [45] se propone mejorar los resultados obtenidos al aplicar el método basado en la correspondencia entre los vectores tangentes [23],



(a) (b)

Figura 3.6: (a) Vectores tangentes de la muestra C^i y de las muestras anterior y siguiente. (b) Ángulo de la muestra C^i

modificando la función de coste considerando dos términos: ángulo e índice. En particular, el método aumenta la robustez del algoritmo con respecto a rotaciones sin incrementar la complejidad computacional del algoritmo ni añadir un paso de preprocesado para alinear las dos curvas. También, en la formulación del problema se permite que las curvas sean muestreadas con un número diferente de muestras cada una, lo que es particularmente interesante desde el punto de vista de nuestro trabajo.

Dadas dos curvas C_A y C_B, sea m el número de muestras de la curva inicial, y n el número de muestras de la curva destino, donde $n \geq m$. Para una muestra C^i de alguna de las curvas, su ángulo $\mathbb{A}(C^i)$ se define como:

$$\mathbb{A}(C^i) = \frac{1}{2} arcos(\langle T^{i-1}, T^{i+1} \rangle) \cdot sign_z(T^{i+1} \times T^{i-1}) \tag{3.9}$$

donde $sign_z(x)$ es el signo de la coordenada z. Como puede observarse en la Figura 3.6, en la que se representa gráficamente el ángulo de una muestra C^i, el cálculo se basa en el ángulo formado por los vectores tangentes de las muestras anterior y siguiente (T^{i-1} y T^{i+1} respectivamente).

El segundo de los términos, el índice $\mathbb{P}(C^i)$ de una muestra C^i de una curva, se define como el cociente entre el índice i de la muestra y el número total de muestras de esa curva:

$$\mathbb{P}(C^i) = \frac{i}{m}$$

donde m es el número de muestras de la curva.

Para medir la similitud entre una muestra C_A^i de la curva inicial y una muestra C_B^j de la curva final, se propone utilizar la diferencia entre el ángulo y el índice de cada una de las dos muestras. El primero de los términos mide

el grado de semejanza entre las curvas desde el punto de vista de la curvatura en los puntos considerados. Por su parte, el segundo de los términos permite penalizar aquellos pares de muestras que estén situados en puntos demasiado apartados de las curvas. Así, la ecuación que sustituye a la Ecuación 3.4 es:

$$f_{i,j} = \omega_1 |\mathbb{A}(C_A^i) - \mathbb{A}(C_B^j)| + \omega_2 |\mathbb{P}(C_A^i) - \mathbb{P}(C_B^j)| \tag{3.10}$$

donde los pesos ω_1 y ω_2 son los pesos de los términos ángulo e índice, respectivamente, que deben ser introducidos por el usuario.

Para obtener la solución al problema de optimización se aplican técnicas de programación dinámica, de forma análoga a la descrita en el apartado 3.3, que permiten resolver el problema en $\mathcal{O}(mn)$, siendo m y n el número de muestras de la curva origen y de la curva destino, respectivamente. Si se aplica el algoritmo sobre curvas cerradas, los resultados obtenidos dependerán de cuáles muestras han sido seleccionadas como muestras iniciales de cada una de las curvas. Para que el algoritmo sea invariante a la selección de estas muestras, en [45] se propone utilizar programación dinámica siguiendo el algoritmo descrito en [36]. Esto permite hallar la correspondencia óptima entre dos curvas cerradas en $\mathcal{O}(log_2 nn^2)$, independientemente de cuáles muestras son consideradas como las muestras iniciales.

Comparando los términos de la función de coste utilizada en este algoritmo (Ecuación 3.10) con la función de coste utilizada en el método descrito en la sección anterior (Ecuación 3.4 ó 3.8), se observa que el término ángulo de este algoritmo tiene el mismo objetivo que toda la función de coste del algoritmo de la sección anterior, que es buscar la similitud entre muestras de acuerdo con la curvatura de la curva en esos puntos. La principal diferencia entre los dos métodos es la inclusión del término índice. En el algoritmo descrito en la Sección 3.3 no se tiene en cuenta la posición relativa de las muestras en cada una de las curvas, lo que puede dar lugar a la asignación de puntos próximos en una de las curvas a puntos muy lejanos en la otra curva, y esto provocar a su vez asignaciones entre muestras en posiciones muy distantes entre sí. Estos efectos no siempre implican una mala asignación, ya que para obtener una asignación válida puede ser necesario que se produzcan en mayor o menor medida este tipo de asignaciones. Por su parte, en [45] se parte de la asunción de que si dos curvas con el mismo número de muestras m son consideradas similares por el usuario, con frecuencia la correspondencia entre las características de las mismas tenderá a asignar la muestra C_A^i a la muestra C_B^j, con $|i - j| \leq \delta$ siendo δ un valor proporcional a m; e incluye

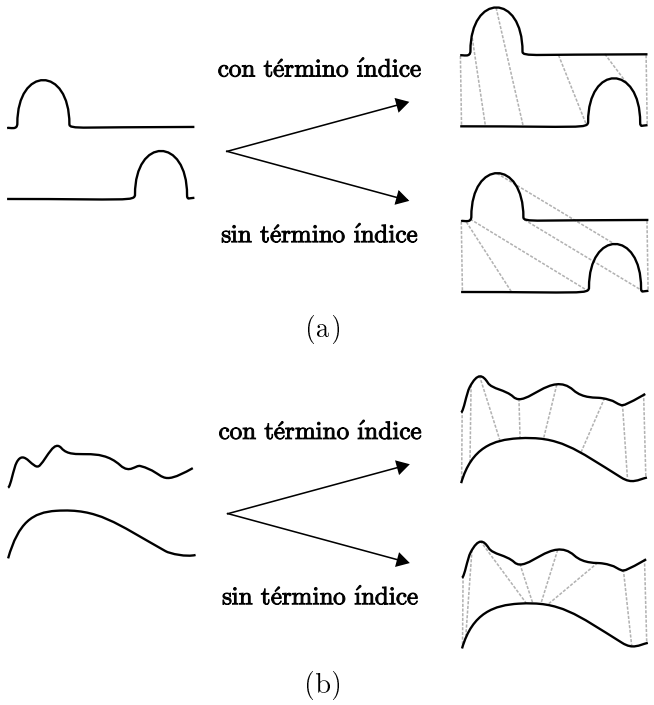


Figura 3.7: Influencia del término índice. (a) Curva con una característica que se desplaza. (b) Curva con características que desaparecen.

el término índice para incorporar esta restricción sin definir δ de una forma explícita.

La calidad asociada a la inclusión del término índice depende de las curvas entre las que se establece la correspondencia. El efecto del término es balancear la correspondencia resultante, lo que puede ser contraproducente en algunos casos, por ejemplo, en curvas que presentan características que se desplazan. En la Figura 3.7a, el término índice impide que se asigne correctamente la correspondencia entre las regiones curvas, por estar demasiado separadas entre sí. Por otra parte, en curvas que presentan un elevado número de detalles pero que tienen una estructura similar a más alto nivel, o curvas en las que alguna de las características desaparece, se consigue una mejor asignación (ver Figura 3.7b).

Una de las mejoras de este algoritmo es la mejora de la varianza a rotación sin que esto implique aumentar la complejidad computacional. En el cálculo de $\mathbb{A}(C^i)$ (Ecuación 3.9) se utilizan operaciones entre vectores tangentes de una misma curva: el producto escalar para calcular la magnitud del ángulo

y el producto vectorial para calcular el signo. Al rotar una curva, se le aplica la misma rotación a todos los vectores tangentes, por lo que el ángulo que forman los dos vectores T^{i-1} y T^{i+1} utilizados en el cálculo de cada $\mathbb{A}(C^i)$ es el mismo. Además, como las magnitudes de los vectores son siempre 1 al ser vectores normalizados, tanto el producto vectorial como el producto escalar utilizados son invariantes a rotaciones. Debido a esto, el término ángulo tendrá el mismo valor independientemente de la rotación aplicada a la curva, y como el cálculo del término índice tampoco se ve afectado por la rotación al ser un cociente entre dos enteros, los resultados obtenidos al aplicar el algoritmo no dependen de la rotación aplicada a las curvas.

3.5. Método guiado por imagen

En este método se propone utilizar la información proporcionada por las imágenes como guía para la correspondencia, además de la información proporcionada por las propias curvas. Así, la evaluación de la similitud entre dos muestras estará guiada por dos criterios: un criterio geométrico, que utiliza las propiedades de la curva, y un criterio de intensidad de la imagen.

En [50] se enfoca el problema de la correspondencia como parte del seguimiento automático de imágenes en el ámbito médico. En este contexto, se parte de una serie de imágenes en escala de grises que representan secciones transversales de un mismo objeto a diferente profundidad obtenidas por aplicaciones médicas. El problema que se plantea resolver es la reconstrucción automática de la estructura tridimensional de los componentes del objeto, para lo que es necesario en primer lugar identificar los contornos que definen los elementos en cada imagen individual, y posteriormente establecer la correspondencia entre contornos para cada par de imágenes consecutivas. Se utiliza la relación entre los contornos y las imágenes para obtener resultados más acertados. De forma análoga, en nuestro trabajo cada una de las curvas entre las que se pretende establecer una correspondencia pertenecen a diferentes fotogramas, y representan la misma característica en diferentes instantes de tiempo.

$I_K(x, y)$ es el píxel situado en las coordenadas (x, y) de la imagen correspondiente al fotograma K. Para medir la intensidad $\mathbb{L}_K(x, y)$ del píxel $I_K(x, y)$, como no existe la restricción de que las imágenes sean en escala de grises, utilizamos la luminancia relativa [80], que se calcula a partir de los

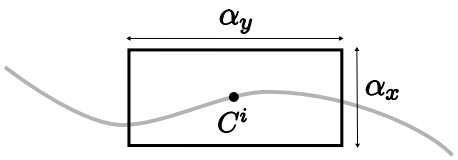


Figura 3.8: Ventana $\mathbb{W}(C^i)$

componentes de color del píxel:

$$\mathbb{L}_K(x,y) = 0,299 \cdot I_K^{\mathcal{R}}(x,y) + 0,587 \cdot I_K^{\mathcal{G}}(x,y) + 0,114 \cdot I_K^{\mathcal{B}}(x,y)$$

donde $I_K^{\mathcal{R}}(x,y)$, $I_K^{\mathcal{G}}(x,y)$, $I_K^{\mathcal{B}}(x,y)$ representan respectivamente los componentes rojo, verde y azul del píxel correspondiente.

A la hora de comparar dos muestras desde el punto de vista de su intensidad, utilizar únicamente el píxel situado en la posición de cada muestra no es adecuado, puesto que la comparación sería demasiado sensible al ruido. Por este motivo se propone utilizar no sólo el píxel situado en la posición de la muestra, sino una región de píxeles que representarán con más robustez la intensidad de esa posición. Dada una muestra C^i, se define su ventana $\mathbb{W}(C^i)$ como los píxeles contenidos en un rectángulo de dimensiones $\alpha_x \alpha_y$, centrado en la posición de la muestra. En la Figura 3.8 se representa una ventana centrada en C^i: los píxeles encerrados en el rectángulo son los puntos que pertenecen a $\mathbb{W}(C^i)$. Dependiendo del tamaño de la ventana y de la variabilidad en la curvatura de la curva, es posible que la ventana no sea una muestra representativa de la característica a la que pertenece la muestra, por ejemplo en muestras que coincidan con picos y valles con cambios de curvatura muy pronunciados.

Para evaluar la similitud de intensidad entre dos muestras C_A^i y C_B^j se utiliza su correlación de imagen, que se calcula utilizando la fórmula:

$$Corr(C_A^i, C_B^j) = \frac{\sigma^2(C_A^i, C_B^j)}{\sqrt{\sigma^2(C_A^i) \cdot \sigma^2(C_B^j)}} \tag{3.11}$$

donde $\sigma^2(C_A^i)$ y $\sigma^2(C_B^j)$ son las varianzas de la intensidad en las ventanas $\mathbb{W}(C_A^i)$ y $\mathbb{W}(C_B^j)$ respectivamente, y $\sigma^2(C_A^i, C_B^j)$ es la covarianza entre las dos ventanas. La varianza se calcula como:

$$\sigma^2(C_K^i) = \sum_{x=-\frac{\alpha_x}{2}}^{\frac{\alpha_x}{2}} \sum_{y=-\frac{\alpha_y}{2}}^{\frac{\alpha_y}{2}} \frac{\{\mathbb{L}_K(a+x, b+y) - \mu_{\mathbb{W}(C_K^i)}\}^2}{\alpha_x \alpha_y} \tag{3.12}$$

donde (a, b) son las coordenadas del píxel situado en la posición de la muestra C_K^i, y $\mu_{\mathbb{W}(C_K^i)}$ es la media de la intensidad de todas las muestras que pertenecen a la ventana $\mathbb{W}(C_K^i)$. Por su parte, la covarianza se calcula utilizando la expresión:

$$\sigma^2(C_A^i, C_B^j) = \sum_{x=-\frac{\alpha_x}{2}}^{\frac{\alpha_x}{2}} \sum_{y=-\frac{\alpha_y}{2}}^{\frac{\alpha_y}{2}} \frac{\left\{\mathbb{L}_A(a+x, b+y) - \mu_{\mathbb{W}(C_A^i)}\right\} \cdot \left\{\mathbb{L}_B(c+x, d+y) - \mu_{\mathbb{W}(C_B^i)}\right\}}{\alpha_x \alpha_y} \tag{3.13}$$

donde (a, b) son las coordenadas del píxel situado en la posición de la muestra C_A^i, y (c, d) las coordenadas del píxel situado en la posición de la muestra C_B^j.

En cuanto al término geométrico, se utiliza el producto de los vectores tangentes de las muestras, como en [23] (ver Ecuación 3.4). Desde el punto de vista geométrico, el razonamiento es análogo al descrito en la Sección 3.3: la correspondencia óptima es aquella para la que la suma de los productos de los vectores tangentes que se corresponden tiene el máximo valor. De esta manera, la ecuación que evalúa la similitud entre dos muestras y que que sustituye a la Ecuación 3.4 es:

$$f_{i,j} = \omega_1 \cdot \langle T_A^i, T_B^j \rangle + \omega_2 \cdot Corr(C_A^i, C_B^j) \tag{3.14}$$

donde ω_1 y ω_2 son los pesos del término geométrico y del término intensidad respectivamente.

La correspondencia óptima se formula como un problema de optimización, de manera análoga a la descrita en la Sección 3.3. Para hallar el máximo de la función de coste se utiliza el mismo algoritmo de programación dinámica.

3.6. Método basado en el vector de desplazamiento

En [38] se propone resolver el problema de la correspondencia entre dos curvas utilizando la distancia euclídea entre las muestras y las posiciones relativas de las mismas como base de la función de coste.

Sean dos curvas C_A y C_B muestreadas con m y n muestras, respectivamente. Dada una muestra C_A^i de una curva, y su muestra correspondiente en la otra curva C_B^j, se define el de desplazamiento entre ambas muestras como:

$$\vec{D}_i = \left(x_j^2 - x_i^2, y_j^2 - y_i^2\right) \tag{3.15}$$

donde (x_i, y_i) son las coordenadas correspondientes de ambas muestras. Teniendo en cuenta esto, se propone la siguiente función de energía:

$$\mathbb{F} = \sum_{i=1}^{n} \left[\omega_1 \left\| \vec{D}_i - \vec{D}_{i-1} \right\| + \omega_2 \left(i - \mathbb{V}(i) \right)^2 \right] \tag{3.16}$$

El primero de los términos de la función de energía favorece aquellas correspondencias en las que el vector de desplazamiento cambia suavemente de una muestra a la muestra siguiente, mientras que el segundo término penaliza aquellas correspondencias en las que los índices de las muestras están demasiado alejados entre sí. Los parámetros ω_1 y ω_2 son los pesos de cada uno de los términos, y en [38] se muestran los resultados obtenidos variando el parámetro ω_2. Cuanto más se aproxime ω_2 a 0, menos serán penalizados los saltos en la asignación de las muestras.

La correspondencia es obtenida al minimizar la función de energía (Ecuación 3.16). Definimos el coste de establecer la correspondencia entre dos muestras C_A^i y C_B^j como:

$$f_{i,j} = \omega_1 \left\| \vec{D}_i - \vec{D}_{i-1} \right\| + \omega_2 (i - j)^2 \tag{3.17}$$

donde $f_{0,0} = 0$. El coste de hacer corresponder las primeras i muestras de la primera curva con las primeras j muestras de la segunda se representa por $\mathbb{F}_{i,j}$ (ver Ecuación 3.5), y el problema se resuelve aplicando programación dinámica tal y como se explica en la Sección 3.3.

Para evitar la existencia de saltos demasiado grandes en la correspondencia, además de utilizar el segundo término de la función de energía se añade una restricción adicional al algoritmo. En cada iteración del algoritmo de programación dinámica, si la posición relativa con respecto a la longitud de la curva de las muestras C_A^i y C_B^j que se quieren hacer corresponder es mayor que un determinado valor ϵ, se establece que el coste de hacer corresponder esas dos muestras es ∞:

$$\left| \frac{i}{n} - \frac{j}{m} \right| \geq \epsilon \Rightarrow f_{i,j} = \infty \tag{3.18}$$

Al contrario que en los demás métodos analizados en este trabajo, la principal característica de este método es que tiene en cuenta la distancia entre las dos muestras que se hacen corresponder, al utilizar el vector de desplazamiento $\vec{D}$. La invarianza frente a traslaciones se mantiene, puesto que, si desplazamos una de las curvas una cantidad $\vec{T}$, al calcular la diferencia entre

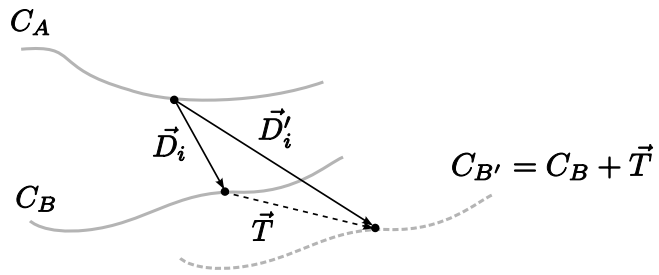


Figura 3.9: Vectores de desplazamiento e invarianza a traslación

los vectores de desplazamiento $\vec{D}'_i$ y $\vec{D}'_{i-1}$ de la curva trasladada, el resultado es el mismo que al calcular la diferencia entre los vectores de desplazamiento $\vec{D}_i$ y $\vec{D}_{i-1}$ de la curva no trasladada (ver Figura 3.6):

$$\left.\begin{array}{l}\vec{D}'_i = \vec{D}_i + \vec{T} \\ \vec{D}'_{i-1} = \vec{D}_{i-1} + \vec{T}\end{array}\right\} \Rightarrow \vec{D}'_i - \vec{D}'_{i-1} = \left(\vec{D}_i + \vec{T}\right) - \left(\vec{D}_{i-1} + \vec{T}\right) = \vec{D}_i - \vec{D}_{i-1}$$

Sin embargo, esta representación hace que el algoritmo no presente invarianza a rotaciones, ya que si una de las curvas es rotada, los vectores de desplazamiento de cada par de muestras variarán de forma no uniforme. Para resolver este problema, en [38] se propone utilizar los vectores tangentes en la función de coste, tal y como se realiza en los métodos comentados en las Secciones 3.3 y 3.4, pero no llega a desarrollar esta idea.

Capítulo 4

Pruebas y resultados de los diferentes algoritmos de correspondencia

En este capítulo se describen los resultados experimentales obtenidos al utilizar los diferentes algoritmos descritos en las Secciones 3.3 - 3.5 en nuestra aplicación. En primer lugar, se describen las secuencias de prueba que hemos considerado. En el apartado siguiente, se detallan las características utilizadas en nuestra implementación, para posteriormente definir las métricas que hemos empleado para evaluar la calidad de los resultados. Finalmente se presentan los resultados obtenidos.

4.1. Secuencias utilizadas

Para la evaluación y comparación de los diferentes algoritmos se han utilizado dos secuencias: la secuencia *Table-tennis* (ver Figura 4.2), y la secuencia *Amira* (ver Figura 4.1), cada una de ellas con curvas con diferentes características.

La secuencia *Table-tennis* está compuesta por 25 fotogramas, con unas dimensiones de 720x576 píxeles y cuenta con 11 curvas. En las Figuras 4.2a y 4.2b se muestran las curvas superpuestas al fotograma inicial y final respectivamente. En estas dos últimas figuras el brillo de las imágenes ha sido modificado, para facilitar la visualización de las curvas. La Figura 4.3 muestra gráficamente las curvas pertenecientes a esta secuencia en el fotograma

(a) (b)

(c) (d)

Figura 4.1: Secuencia *Amira*. (a)-(b) Fotogramas inicial y final, sin las curvas. (c)-(d) Fotogramas inicial y final, con las curvas superpuestas

(a) (b)

(c) (d)

Figura 4.2: Secuencia *Table-tennis*. (a)-(b) Fotogramas inicial y final, sin las curvas. (c)-(d) Fotogramas inicial y final, con las curvas superpuestas

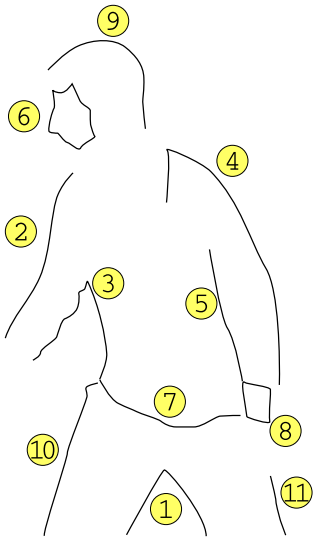


Figura 4.3: Curvas de la secuencia *Table-tennis*

		Nºde segmentos		
		Fotograma 0	Fotograma 24	Cerrada
1.	*Arco_piernas*	3	3	○
2.	*Brazo_dcho_exterior*	3	3	○
3.	*Brazo_dcho_interior*	9	6	○
4.	*Brazo_izqdo_exterior*	4	4	○
5.	*Brazo_izqdo_interior*	2	6	○
6.	*Cara*	12	9	●
7.	*Cintura*	7	6	○
8.	*Muñeca*	4	5	●
9.	*Pelo*	5	12	○
10.	*Pierna_dcha*	4	5	○
11.	*Pierna_izqda*	3	2	○

Cuadro 4.1: Características de las curvas de la secuencia *Table-tennis*

inicial de forma más detallada. El Cuadro 4.1 detalla el número de segmentos de cada una de las curvas en el fotograma inicial y en el final. Esta secuencia

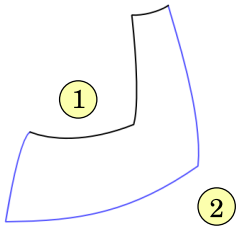


Figura 4.4: Curvas de la secuencia *Amira*

		Nºde segmentos		
		Fotograma 0	Fotograma 9	Cerrada
1.	*Brazo_Superior*	3	3	○
2.	*Brazo_Inferior*	3	3	○

Cuadro 4.2: Características de las curvas de la secuencia *Amira*

está tomada del banco de datos ITU-R BT.601 para la evaluación de medios de tratamiento de vídeo [86] y pertenece al dominio público.

La secuencia *Amira* está compuesta por 10 fotogramas, con unas dimensiones de 640x480 píxeles y cuenta con 2 curvas. En las Figuras 4.1a y 4.1b se muestran las curvas superpuestas al fotograma inicial y final respectivamente. La Figura 4.4 muestra gráficamente las curvas pertenecientes a esta secuencia en el fotograma inicial de forma más detallada. El Cuadro 4.2 detalla el número de segmentos de cada una de las curvas en el fotograma inicial y en el final. Esta secuencia está tomada directamente de uno de los ejemplos incluidos incluida con el código fuente del artículo original [6] en el que está basado este trabajo, aunque para nuestro trabajo hemos optado por dividir la curva inicial en dos curvas.

Método basado en la similitud de las curvas	
ω_1	1.0
ω_2	3.0
Método guiado por imagen	
α_x	9
α_y	6
ω_1	0.7
ω_2	0.3
Método basado en el vector de desplazamiento	
ω_1	1.0
ω_2	0.1
ϵ	0.3

Cuadro 4.3: Valores utilizados para los parámetros de los algoritmos

4.2. Configuración de los algoritmos de correspondencia

En el Cuadro 4.3 se muestran los valores utilizados en nuestro trabajo para cada uno de los parámetros de los diferentes algoritmos de correspondencia.

Para el método basado en la similitud de las curvas [45], hemos utilizado $\omega_1 = 1{,}0$ y $\omega_2 = 3{,}0$, ya que son los valores sugeridos en el artículo original y de acuerdo con las pruebas realizadas son valores que funcionan razonablemente bien para todos los tipos de curvas consideradas. La Ecuación 4.1 muestra la fórmula resultante.

$$f_{i,j} = |\mathbb{A}(C_A^i) - \mathbb{A}(C_B^j)| + 3 \cdot |\mathbb{P}(C_A^i) - \mathbb{P}(C_B^j)| \tag{4.1}$$

En el método guiado por imagen [50], el tamaño de la ventana considerado ha sido de $\alpha_x = 9$ y $\alpha_y = 6$, ambos medidos en píxeles. Además, en nuestro trabajo hemos utilizado ventanas orientadas en la dirección del vector tangente de la muestra C^i en la que están centradas, como puede verse en la Figura 4.5. Para el establecimiento de los pesos, hemos considerado las particularidades de nuestro trabajo y las observaciones del artículo original. El [50] se sugiere utilizar $\omega_1 = 0{,}7$ y $\omega_2 = 0{,}3$ si las imágenes sobre las que se aplica el algoritmo son imágenes reales; reduciendo el peso del término de intensidad y aumentando el peso del término geométrico si las imágenes han sido generadas por ordenador o previamente tratadas. En nuestro trabajo las

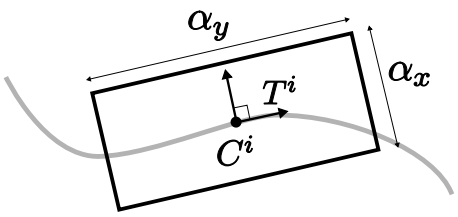


Figura 4.5: Ventana $\mathbb{W}(C^i)$ orientada en la dirección de T^i

imágenes utilizadas en las secuencias de prueba, al ser obtenidas a partir de secuencias de vídeo, son más nítidas que las utilizadas en el artículo original, puesto que son procedentes de instrumental médico, y las curvas se ajustan mejor a las características. Hemos optado por mantener los pesos originales propuestos en el artículo, con el fin de observar mejor el efecto producido por el término de intensidad. La fórmula resultante se muestra en la Ecuación 4.2.

$$f_{i,j} = 0{,}7 \cdot \langle T_A^i, T_B^j \rangle + 0{,}3 \cdot Corr(C_A^i, C_B^j) \tag{4.2}$$

En el método basado en el vector de desplazamiento [38] hemos utilizado $\omega_1 = 1$ y $\omega_2 = 0{,}1$, basándonos en los resultados comentados en [38]. En cuanto a la restricción impuesta sobre los índices relativos de las muestras, hemos utilizado $\epsilon = 0{,}3$ siguiendo las recomendaciones. La Ecuación 4.3 muestra la fórmula resultante.

$$f_{i,j} = \left\| \vec{D}_i - \vec{D}_{i-1} \right\| + 0{,}1 \cdot (i-j)^2 \tag{4.3}$$

4.3. Métricas

Para la evaluación de la calidad de los resultados obtenidos hemos utilizado diferentes métricas. Estas métricas comparan las curvas obtenidas tras el proceso de interpolación con unas curvas de referencia, que representan el resultado óptimo. El proceso para obtener estas curvas de referencia es lanzar un proceso de seguimiento inicial, y posteriormente ajustar manualmente aquellos puntos de control que no sigan el contorno de las imágenes de referencia. Estos puntos de control ajustados manualmente son marcados como puntos fijos, y el proceso de seguimiento es lanzado de nuevo, repitiendo el proceso hasta garantizar que todas las curvas en todos los fotogramas siguen correctamente las características de las imágenes.

Consideramos las dos curvas a comparar C_A y C_B como dos secuencias que tienen el mismo número m de muestras. La distancia euclídea entre dos muestras la representamos como $d(C_A^i, C_B^j)$, donde las muestras pertenecientes al conjunto compuesto por todas las muestras $\{C_A^0, \ldots, C_A^m\} \cup \{C_B^0, \ldots, C_B^m\}$ están contenidas en un rectángulo delimitado por los puntos min y max, que representan los puntos cuyas coordenadas x e y son las mínimas y las máximas de todo el conjunto de muestras, respectivamente.

Las métricas utilizadas son las siguientes:

1. Distancia media

$$\mathbf{DM} = \frac{1}{m} \sum_{i=0}^{m-1} d(C_A^i, C_B^i) \tag{4.4}$$

2. Distancia media normalizada

$$|\mathbf{DM}| = \frac{1}{m} \sum_{i=0}^{m-1} d(C_A^i, C_B^i) \cdot Diag \tag{4.5}$$

donde la diagonal del rectángulo es la distancia euclídea entre min y max, y la representaremos como $Diag$:

$$Diag = d(min, max) \tag{4.6}$$

3. Signal-to-noise ratio [78] (SNR)

$$\mathbf{SNR} = 10 \ log_{10} \left(\frac{\frac{1}{m} \sum_{i=0}^{m-1} \left(d(C_A^i, min) \right)^2}{ECM} \right) \tag{4.7}$$

donde el error cuadrático medio es:

$$ECM = \frac{1}{m} \sum_{i=0}^{m-1} (d(C_A^i, C_B^i))^2 \tag{4.8}$$

4. Peak signal-to-noise ratio (PSNR)

$$\mathbf{PSNR} = 10 \ log_{10} \left(\frac{Diag^2}{ECM} \right) = 20 log_{10} \left(\frac{Diag}{\sqrt{ECM}} \right) \tag{4.9}$$

4.4. Resultados experimentales

Las figuras y tablas en esta sección contienen los resultados obtenidos utilizando un procesador *Intel®Core™2 i7-4500U* a 1.80GHz bajo un entorno *GNU/Linux (Debian 8)* con kernel *3.16.0-4-amd64*. Los valores de las tablas y las figuras que representan intervalos de tiempo corresponden a la media obtenida tras realizar 10 ejecuciones de las pruebas, con el objetivo de minimizar el efecto de factores ajenos.

4.4.1. Resultados de correspondencia y seguimiento

La Figura 4.6 muestra los valores medios de cada una de las métricas (promediada sobre los fotogramas) para cada una de las familias de la secuencia *Table-tennis*. Los valores de la distancia media normalizada ($|DM|$) y el $PSNR$ para cada uno de los fotogramas y curvas por separado se detallan en la Figura 4.7. En el caso de las dos primeras métricas (DM y $|DM|$) valores más bajos indican mejores resultados, y en el caso de las últimas dos métricas (SNR y $PSNR$) valores más elevados indican mejores resultados.

De manera análoga, la Figura 4.8 muestra los valores medios de cada una de las métricas para cada una de las familias de la secuencia *Amira*. Los valores de la distancia media normalizada ($|DM|$) y el $PSNR$ para cada uno de los fotogramas y curvas por separado se detallan en la Figura 4.9. En las figuras de esta sección se incluyen únicamente los resultados de aquellos métodos de correspondencia que producen resultados numéricamente válidos para todas las familias, de acuerdo con la Figura 4.13.

4.4.2. Resultados de tiempo

Los tiempos invertidos en el procesado de cada una de las familias de la secuencia *Table-tennis* se muestran en el Cuadro 4.4. Esta tabla se divide en cuatro bloques: el bloque *Correspondencia* detalla el tiempo invertido en calcular la correspondencia entre las curvas inicial y final, realizado al comienzo del proceso de seguimiento. El bloque *Seguimiento* contiene el tiempo invertido en completar proceso de seguimiento. El bloque *Pirámides* contiene el tiempo invertido en el preprocesado previo de las imágenes que componen la secuencia para representarlas como pirámides de Gauss [15] [1]. Finalmente,

[1]Durante la ejecución normal de la aplicación *ARAS* este preprocesado sólo se realiza una vez por secuencia, ya que las pirámides son independientes de las roto-curvas y

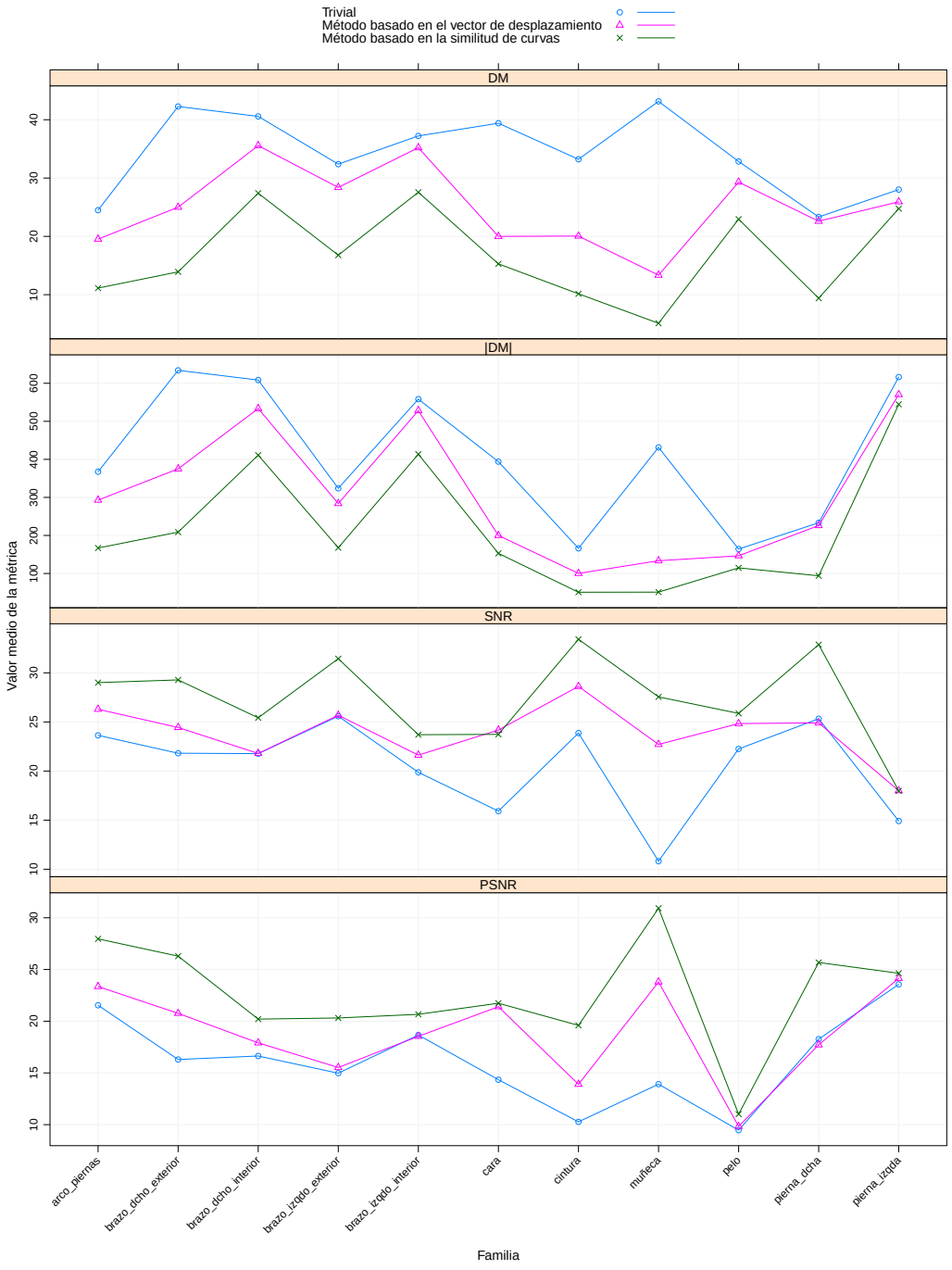
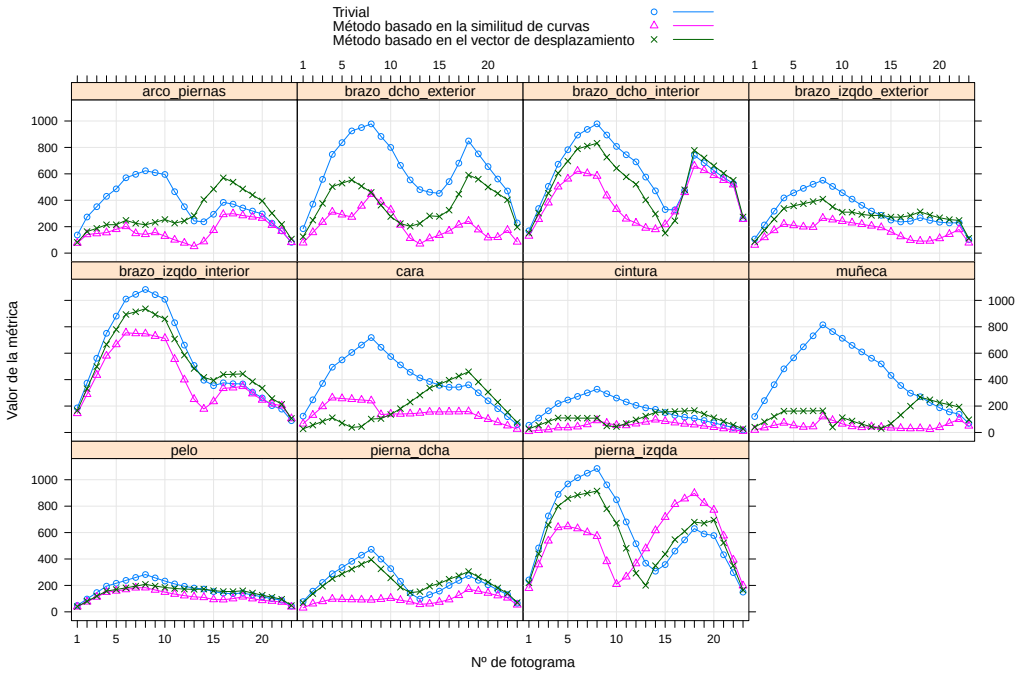
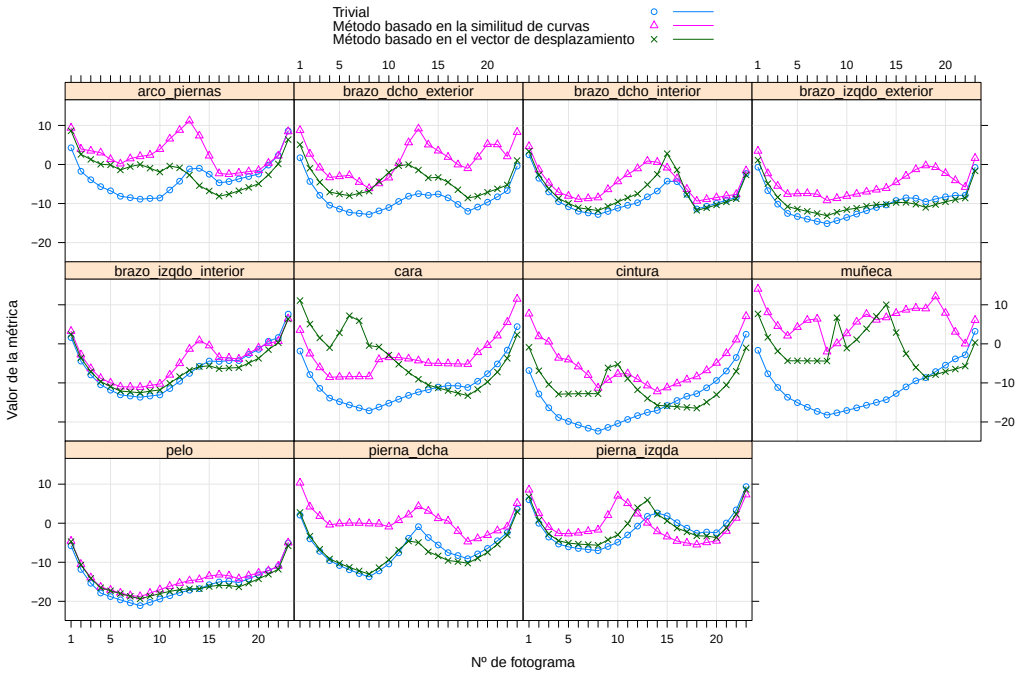


Figura 4.6: Error medio de las curvas de la secuencia *Table-tennis* con los diferentes algoritmos de correspondencia



(a) $|DM|$



(b) $PSNR$

Figura 4.7: Distancia media normalizada (a) y *Peak signal-to-noise ratio* (b) para cada una de las curvas de la secuencia *Table-tennis*

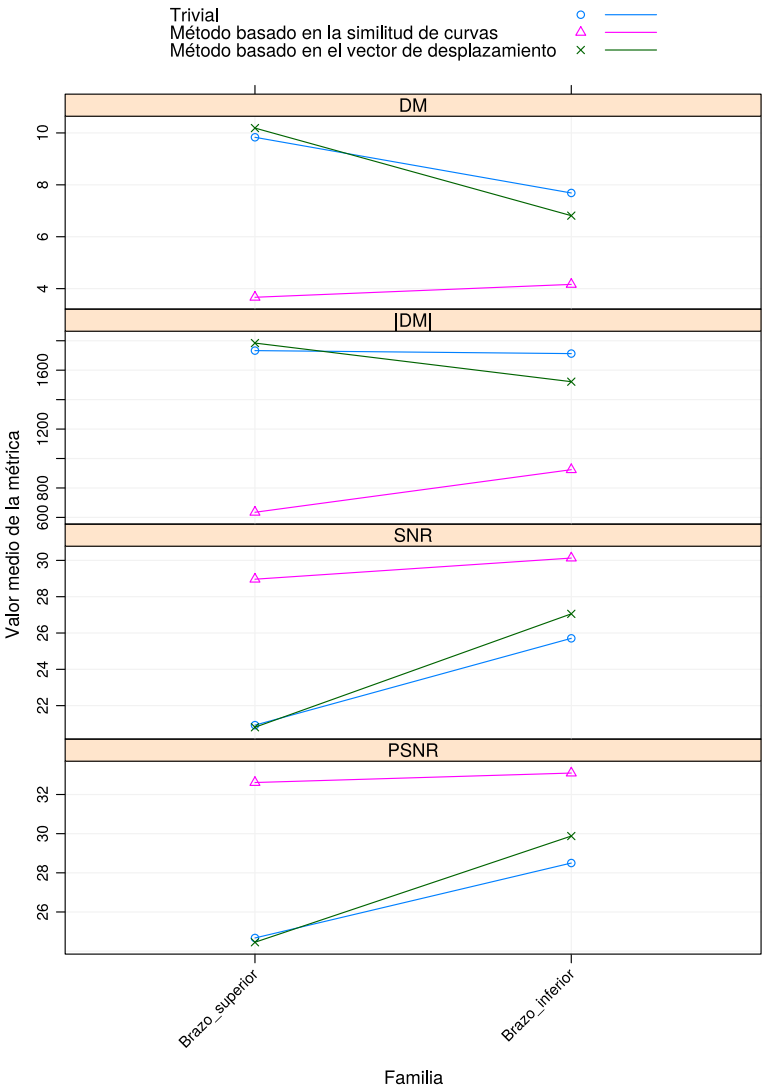
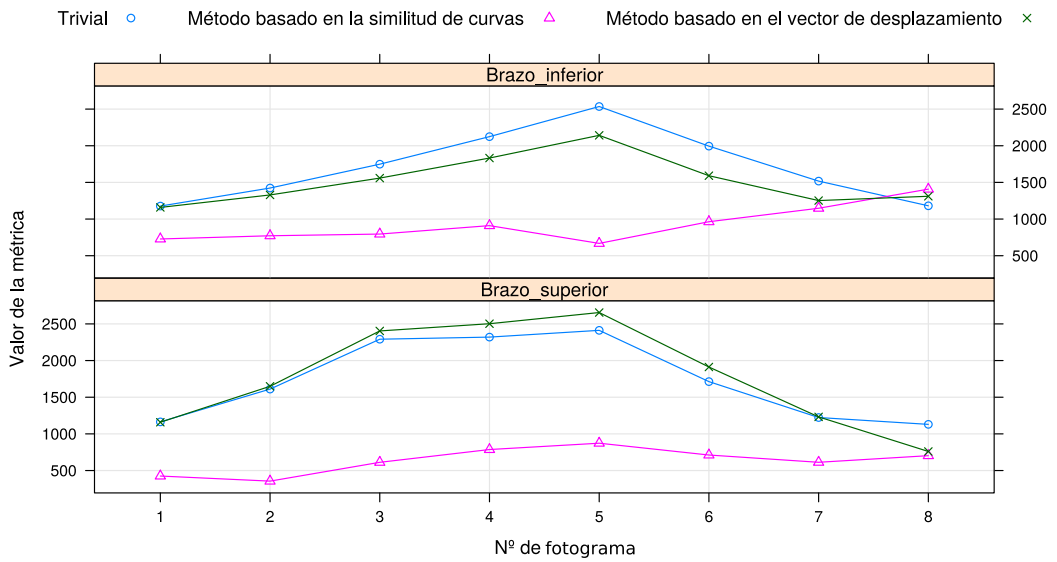
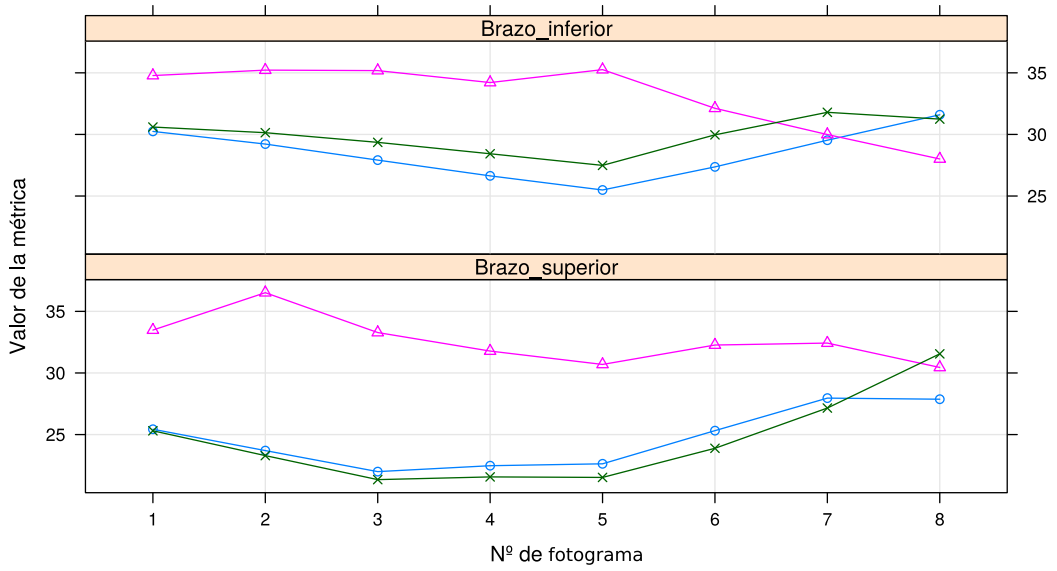


Figura 4.8: Error medio de las curvas de la secuencia *Amira* con los diferentes algoritmos de correspondencia



(a) $|DM|$



(b) $PSNR$

Figura 4.9: Distancia media normalizada (a) y *Peak signal-to-noise ratio* (b) para cada una de las curvas de la secuencia *Amira*

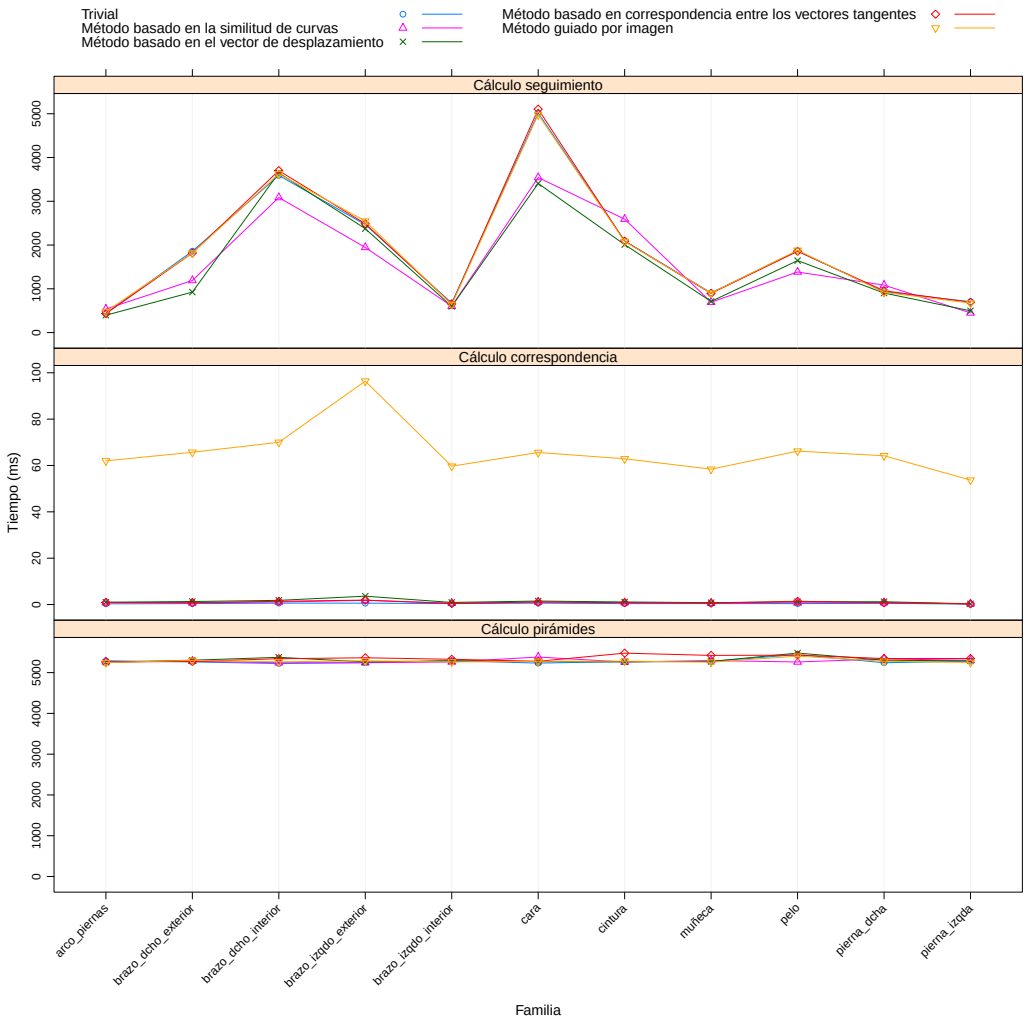
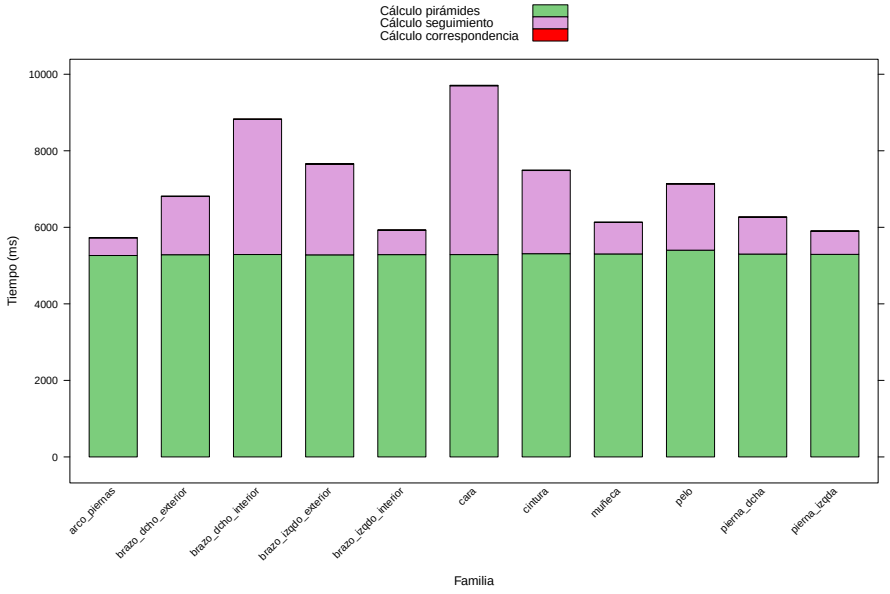
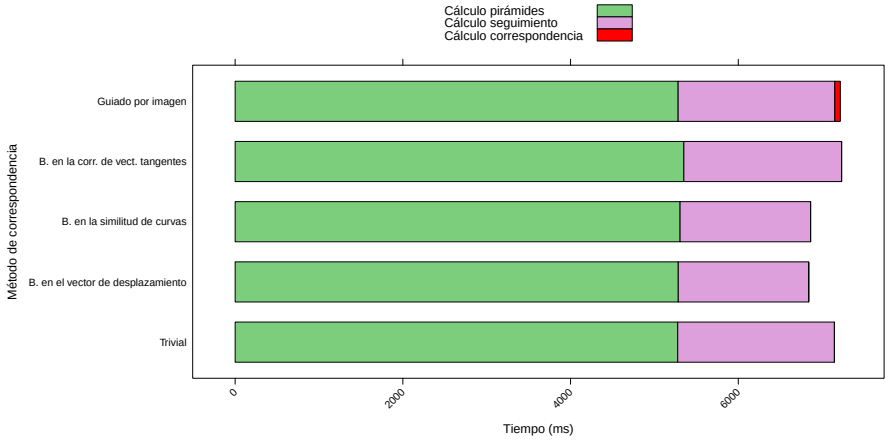


Figura 4.10: Tiempo empleado en cada una de las etapas para las familias de la secuencia *Table-tennis*



(a) Media de tiempos (por familias)



(b) Media de tiempos (por método de correspondencia)

Figura 4.11: Tiempos medios por familias (a) y por método de correspondencia (b) de la secuencia *Table-tennis*

el bloque *Total* contiene el tiempo total invertido en cada curva, siendo éste la suma de los tiempos de cada uno de los bloques anteriores.

La Figura 4.10 representa los tiempos listados en el cuadro anterior divididos por los diferentes bloques y utilizando una escala en el eje Y diferente para cada bloque. Con el objetivo de proporcionar una visión agrupada de los datos temporales, la Figura 4.11 muestra los tiempos medios, promediados por familias (Figura 4.11a) y por método de correspondencia (Figura 4.11b).

4.4.3. Resultados visuales y otros

La Figura 4.12 ilustra la diferencia entre los métodos de correspondencia descritos en esta sección e implementados en nuestro trabajo, representando gráficamente la correspondencia entre dos curvas con diferente número de segmentos. Los puntos de control situados sobre la curva se muestran con un rombo de color oscuro, y las rectas entre curvas representan el punto de la curva final al que corresponde cada muestra de la curva inicial.

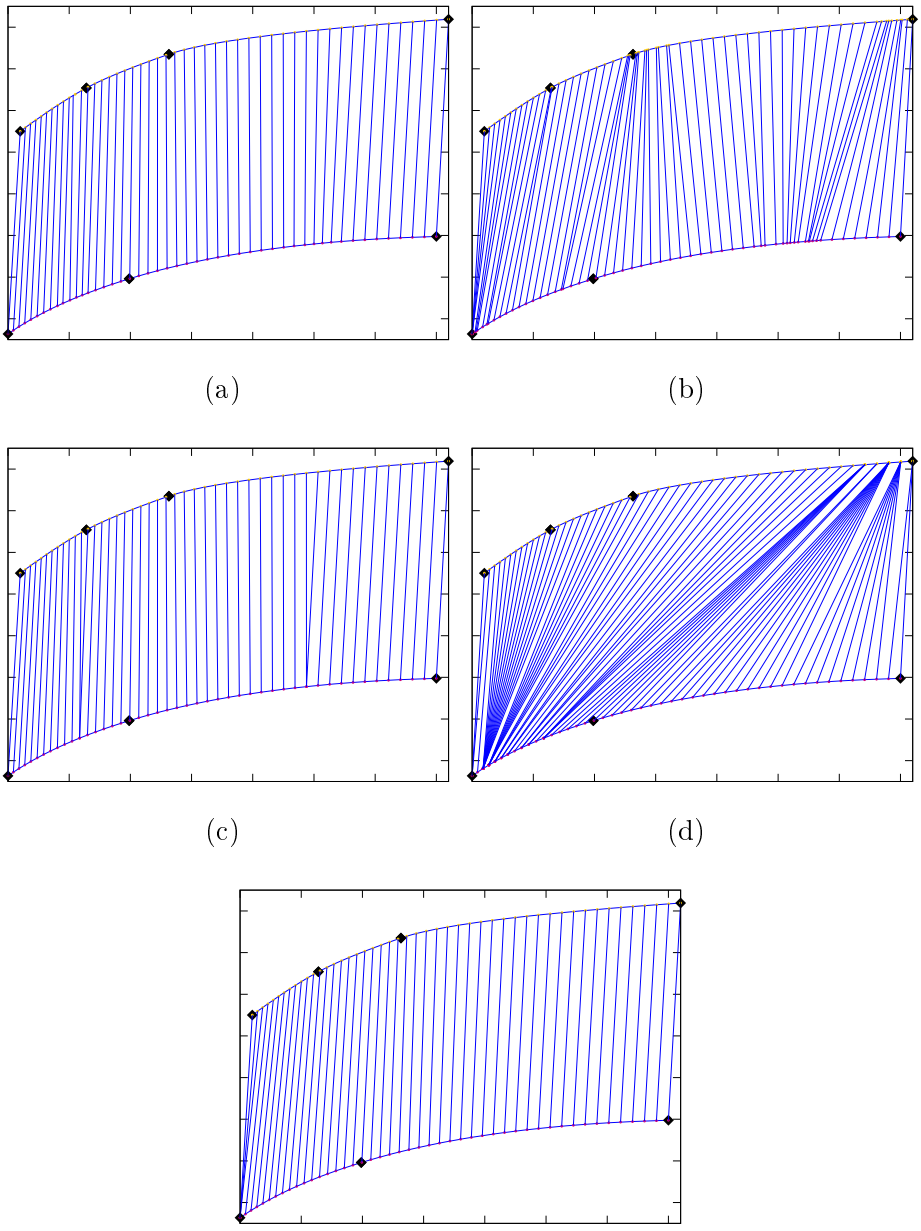
La Figura 4.13 muestra la estimación subjetiva de la bondad de los resultados obtenidos al realizar el proceso de seguimiento sobre las diferentes familias de curvas de la secuencia *Table-tennis* utilizando los diferentes métodos de correspondencia. Debido a la complejidad de alguna de las curvas y a detalles de la implementación, algunas de las curvas no producen resultados aceptables, presentando defectos tales como lazos o puntos situados fuera de los límites de la imagen.

La Figura 4.14 muestra las curvas que han sido utilizadas como conjunto ideal de curvas para la secuencia *Table-tennis*, obtenidas mediante un proceso iterativo de ajuste manual y lanzamiento de seguimiento, como se describe en la Sección 4.3. Los puntos de control situados sobre las curvas se muestran en color azul, y los demás puntos de control en color gris, para facilitar la visualización de los distintos segmentos que componen cada curva.

pueden ser reutilizados para cada una de las familias. Sin embargo, durante las pruebas el preprocesado se ha realizado de forma independiente en cada ejecución, con el fin de representar el tiempo invertido en el proceso de una secuencia partiendo de un entorno de trabajo recién inicializado.

	Correspondencia					**Seguimiento**				
Nº curva	Trivial	Bas. vector de desplazamiento	Bas. similitud de curvas	Bas. correspondencia v. tangentes	Guiado por imagen	Trivial	Bas. vector de desplazamiento	Bas. similitud de curvas	Bas. correspondencia v. tangentes	Guiado por imagen
1	0.3	0.8	1	0.9	62	439	538	397	433	481
2	0.4	0.7	1.3	0.8	65.7	1859	1190	924	1817	1816
3	0.6	1.1	1.8	1.4	70	3590	3086	3648	3700	3625
4	0.6	1.8	3.6	1.9	96.4	2468	1946	2374	2487	2548
5	0.4	0.6	0.9	0.4	59.7	675	602	599	656	644
6	0.6	1.1	1.5	1.1	65.6	5017	3543	3407	5105	4980
7	0.4	0.8	1.1	0.8	62.9	2094	2592	2010	2089	2082
8	0.4	0.6	0.8	0.6	58.4	908	694	711	895	904
9	0.4	0.7	1.2	1.4	66.2	1865	1384	1645	1857	1881
10	0.5	0.8	1.2	0.8	64.2	934	1084	903	956	919
11	0.2	0.3	0.3	0.3	53.7	705	449	497	692	673
	Pirámides					**Total**				
1	5290	5266	5254	5260	5247	5729.3	5804.8	5652	5693.9	5790
2	5262	5279	5304	5274	5300	7121.4	6469.7	6229.3	7091.8	7181.7
3	5223	5243	5375	5341	5264	8813.6	8330.1	9024.8	9042.4	8959
4	5244	5242	5262	5364	5290	7712.6	7189.8	7639.6	7852.9	7934.4
5	5292	5261	5290	5327	5262	5967.4	5863.6	5889.9	5983.4	5965.7
6	5233	5382	5271	5277	5276	10250.6	8926.1	8679.5	10383.1	10321.6
7	5271	5260	5264	5477	5282	7365.4	7852.8	7275.1	7566.8	7426.9
8	5275	5295	5274	5422	5256	6183.4	5989.6	5985.8	6317.6	6218.4
9	5443	5261	5479	5428	5404	7308.4	6645.7	7125.2	7286.4	7351.2
10	5242	5337	5304	5345	5279	6176.5	6421.8	6208.2	6301.8	6262.2
11	5281	5302	5288	5346	5249	5986.2	5751.3	5785.3	6038.3	5975.7

Cuadro 4.4: Tiempos (en ms) para cada una de las familias de la secuencia *Table-tennis*



(a) (b)

(c) (d)

(e)

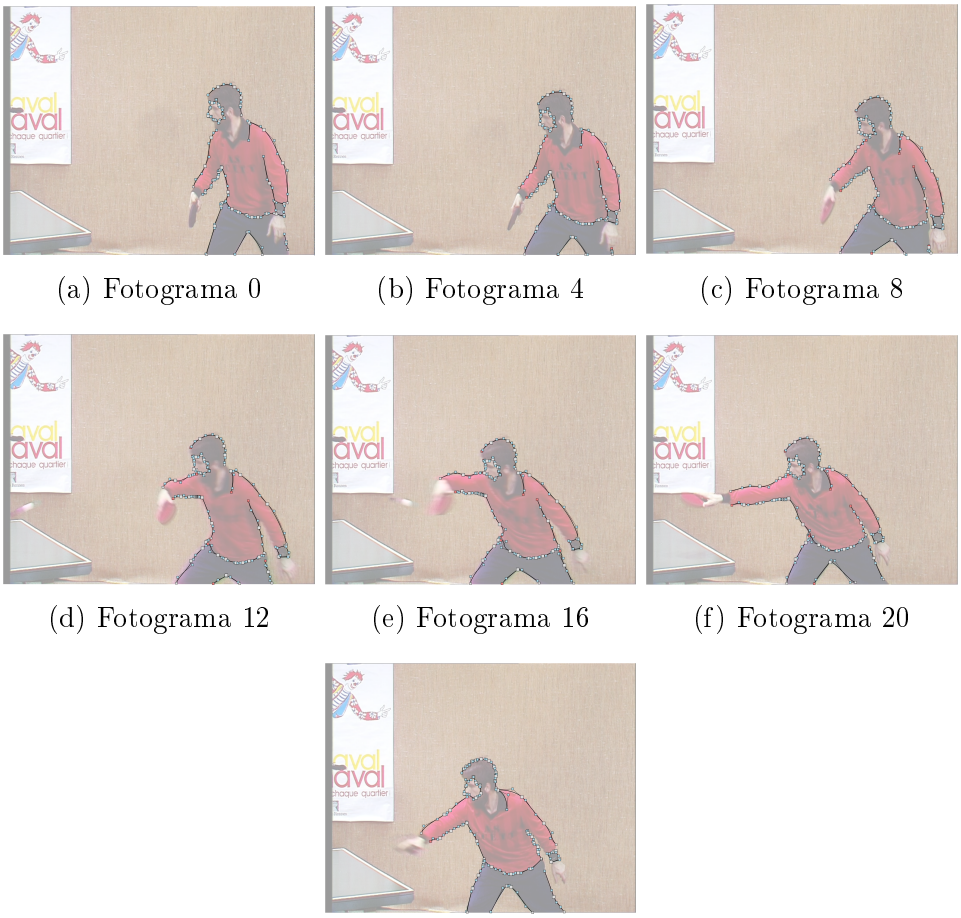
Figura 4.12: Ejemplo de la correspondencia entre dos curvas utilizando el método trivial (a), el método basado en correspondencia entre los vectores tangentes (b), el método basado en la similitud de las curvas (c), el método guiado por imagen (d), y el método basado en el vector de desplazamiento (e)

		Trivial	Bas. vector de desplazamiento	Bas. similitud de las curvas	Bas. correspondencia vectores tangentes	Guiado por imagen
1.	*Arco_piernas*	○	○	●	○	
2.	*Brazo_dcho_exterior*	●	●	●	○	
3.	*Brazo_dcho_interior*	○	●	●		
4.	*Brazo_izqdo_exterior*	●	●	●		
5.	*Brazo_izqdo_interior*	●	●	●		
6.	*Cara*	●	●	●	○	○
7.	*Cintura*	●	●	●	●	●
8.	*Muñeca*	○	●	●		
9.	*Pelo*	●	●	●	●	●
10.	*Pierna_dcha*	●	●	●	●	●
11.	*Pierna_izqda*	○	○	○		

●: resultado visualmente aceptable

○: resultado visualmente razonable, pero incorrecto

Figura 4.13: Estimación subjetiva de la bondad de los resultados obtenidos para la secuencia *Table-tennis*



(a) Fotograma 0 (b) Fotograma 4 (c) Fotograma 8

(d) Fotograma 12 (e) Fotograma 16 (f) Fotograma 20

(g) Fotograma 24

Figura 4.14: Curvas ideales de la secuencia *Table-tennis* con los puntos de control de las curvas

Capítulo 5

Interpolación en ARAS

En esta sección describimos la herramienta ARAS (*Automatic Rotoscoping for Animated Sequences*), que implementa el algoritmo semiautomático de seguimiento descrito en el Capítulo 2. Para uno de los pasos de la interpolación, se necesitan los algoritmos de correspondencia entre dos curvas descritos en el Capítulo 3, que también son implementados en nuestra herramienta.

En la primera parte de este capítulo describiremos las principales características y funcionalidades de la aplicación, para en la siguiente sección presentar los detalles de diseño e implementación. En el Apéndice B se presenta el manual de la aplicación, detallando el modo de operación de la misma.

5.1. Descripción de la aplicación ARAS

La aplicación ARAS fue diseñada en respuesta a la necesidad de una herramienta que facilitara la automatización del proceso de animación para conseguir una reducción del tiempo de diseño y aumentar la calidad de las imágenes finales. En concreto, este proyecto se centra en el paso de la generación automática de los trazos de los fotogramas intermedios a partir de los trazos en dos fotogramas clave. Estos trazos son los elementos que utiliza el artista para definir el estilizado final de la secuencia, destacando características importantes según su criterio artístico. En este contexto, el objetivo planteado al diseñar la herramienta fue la combinación de una interfaz gráfica sencilla a la vez que flexible, en el que un usuario disponga de la funcionalidad necesaria para ajustar y editar los trazos de una secuencia,

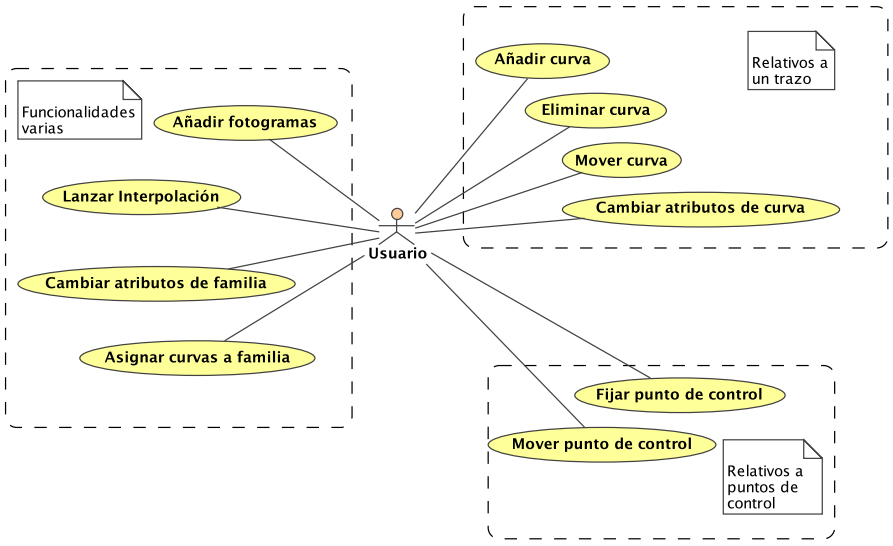


Figura 5.1: Casos de uso en ARAS

y también que la aplicación permita utilizar los algoritmos necesarios para realizar la interpolación de una manera intuitiva.

La funcionalidad de ARAS se muestra en la Figura 5.1, en la que se puede ver el diagrama de casos de uso de esta aplicación. El usuario interactúa con el sistema en varios grupos de casos de uso, según su funcionalidad: los casos de uso relacionados con la modificación de una de las curvas (añadir curva, eliminar curva, mover curva, cambiar atributos de curva), casos de uso relacionados con sus puntos de control (mover punto de control, fijar punto de control), y funcionalidades diversas (lanzar interpolación, añadir fotogramas al proyecto, asignar familia de curvas).

5.1.1. Interfaz gráfica

En la Figura 5.2 se representa de manera esquemática un proyecto, con los diferentes elementos que lo componen. La interfaz gráfica de la herramienta ha sido desarrollada utilizando para ello los componentes de las librerías *Qt*, que proporcionan una serie de componentes gráficos estándar que simplifican el desarrollo de la aplicación. El aspecto general de la aplicación se muestra en la Figura 5.3. Como se puede observar en la figura, en la ventana principal destacan varias secciones diferenciadas:

1. Árbol de proyecto: el objetivo de este componente es proporcionar al

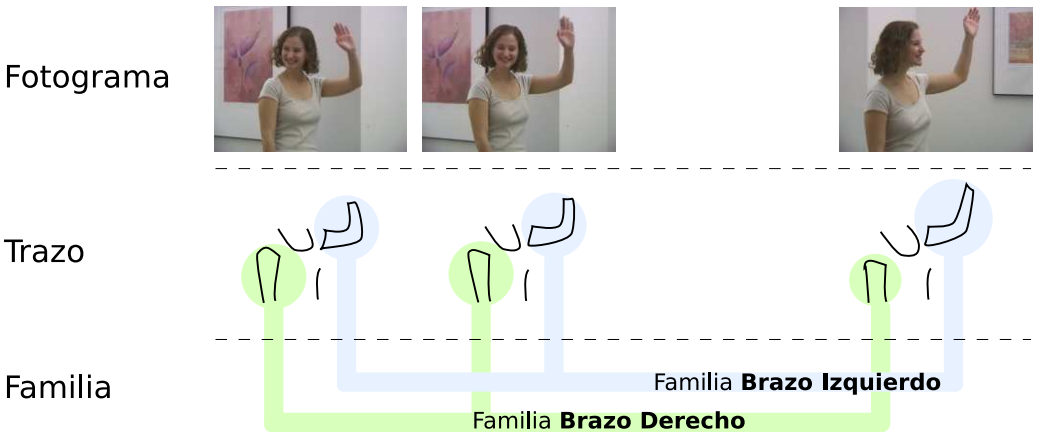


Figura 5.2: Proyecto con sus diferentes elementos

usuario una vista de la jerarquía de curvas del proyecto. En el árbol se muestran todos los fotogramas, y las curvas y familias que pertenecen a cada una de ellas. De esta manera, el usuario puede seleccionar una curva en particular pulsando sobre su identificador.

2. Ventanas individuales de fotograma: cada una de estas ventanas representa un fotograma individual, con sus trazos. La edición de los trazos se realiza en estas ventanas de forma gráfica, arrastrando los trazos o los puntos de control.

3. Barra de herramientas de la aplicación: mediante esta barra de herramientas, el usuario puede conmutar entre los diferentes modos de la aplicación (edición de trazos y asignación de curvas a familias), así como lanzar el proceso de seguimiento.

4. Barra de menús y funcionalidades comunes: la barra de menús y la barra de herramientas de funcionalidades comunes permite un acceso rápido a las funciones de la herramienta, además de tareas relacionadas con la gestión de proyectos tales como abrir o crear un nuevo proyecto.

Los diferentes componentes de la interfaz son utilizados por el usuario para llevar a cabo el proceso de interpolación y manipular de diferentes maneras los trazos. La interfaz está pensada para permitir al usuario el tratamiento de las secuencias de vídeo de una forma sencilla e intuitiva. De esta manera, típicamente el usuario comienza una sesión de trabajo cargando un proyecto anteriormente creando o construyendo un nuevo proyecto, añadiendo los

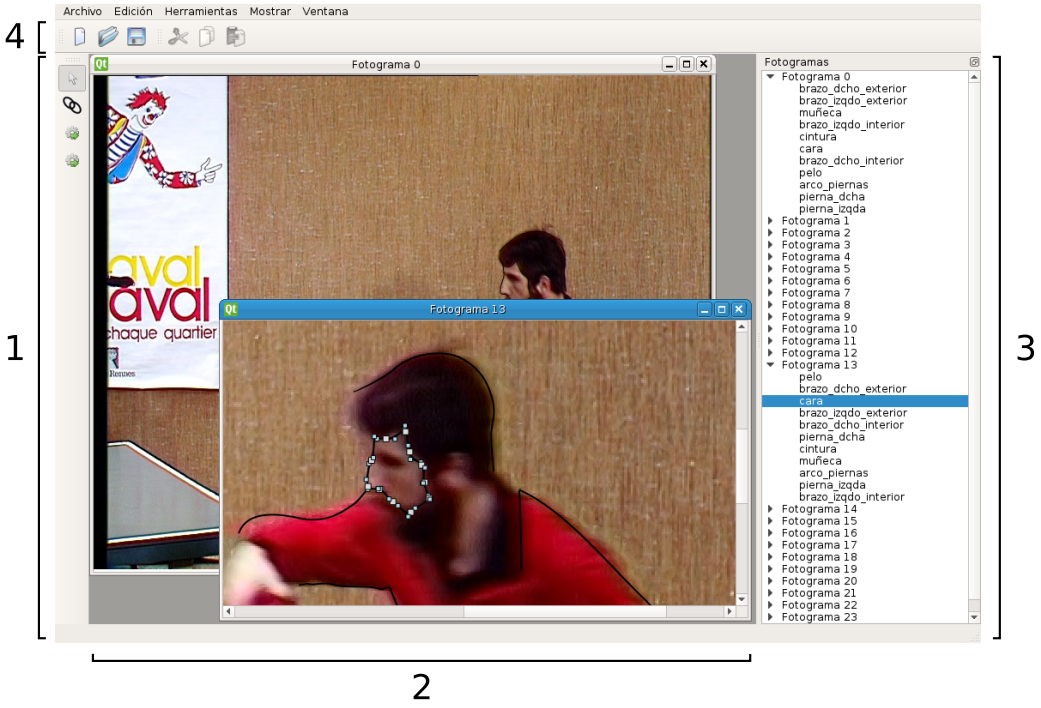


Figura 5.3: Ventana principal de la aplicación ARAS

fotogramas y los trazos iniciales. El ciclo más repetido durante una sesión es el de la edición de los puntos de control de los trazos y lanzamiento del proceso de interpolación, ajustando algunos de los puntos e iterando de nuevo hasta que el resultado sea satisfactorio. Una explicación más detallada del funcionamiento de la aplicación se puede encontrar en el Apéndice B.

5.2. Diseño de la aplicación ARAS

En esta sección describiremos el diseño e implementación de la aplicación ARAS, describiendo los diferentes paquetes que componen el modelo y los patrones utilizados durante el diseño.

5.2.1. Aplicación original

El método de seguimiento empleado en nuestra aplicación está basado en el algoritmo desarrollado en [6], cuya implementación está disponible en Internet [5]. En las fases iniciales del trabajo se evaluó la posibilidad de

reutilizar completamente la aplicación original, manteniendo la estructura y la parte relativa al interfaz. Sin embargo, tras analizar el código fuente se identificaron una serie de inconvenientes:

- A pesar de que el código fuente está en lenguaje *C++*, muchas partes del código no aprovechan las ventajas de la orientación a objetos. Muchos de los archivos contienen un número de funciones demasiado grande para ser manejable y sin una semántica definida, más propio de programación declarativa.

- La ausencia de documentación o soporte por parte del autor hizo que fuera necesario un esfuerzo de ingeniería inversa para poder inferir la estructura y funcionalidad del sistema y de sus componentes.

- El diseño no sigue una determinada metodología o enfoque que permita la mantenibilidad del mismo. Por ejemplo, en diversas clases se declaran los atributos como públicos, violando el principio de encapsulación.

- El diseño de la aplicación tiene problemas especialmente graves en cuanto a la modularidad del mismo y la cohesión de sus componentes. En particular, las clases que representan a las curvas combinan su representación abstracta (modelo) con funciones para representarlas en pantalla (vista) y funcionalidades para modificarlas.

- Una parte de las clases presentes en la aplicación original estaban relacionadas con funcionalidades ya proporcionadas por las librerías estándar de C++ *STL* (*Standard Template Library*).

- Las clases que proporcionaban la funcionalidad necesaria para añadir el paso de estilizado del algoritmo estaban implementadas de forma incompleta.

- El código fuente estaba pensado para la versión 3 de las librerías *Qt*, anterior a la actual y con numerosos componentes obsoletos. En concreto, las clases utilizadas para representar los elementos de la interfaz han sido declaradas obsoletas en la versión 4 de las librerías, por motivos de diseño.

La combinación de estos inconvenientes se traduce en la dificultad de añadir nueva funcionalidad a la aplicación sin modificar considerablemente

el diseño de la misma. Además, nuestra aplicación presenta diferencias importantes con respecto a la herramienta original, entre ellas, las modificaciones en el algoritmo de seguimiento o la utilización de un sólo tipo de curvas (trazos) en lugar de los dos tipos de curvas propuestos en la aplicación original. Por estos motivos, se decidió utilizar solamente el conjunto mínimo de código necesario para lanzar el proceso de seguimiento. Para ello, se han adaptado una serie de clases del código original en un subsistema de nuestra aplicación (en concreto, en el paquete klt, como se detalla en secciones posteriores). Este conjunto de clases fue utilizado como base para desarrollar nuestro algoritmo de seguimiento. En nuestra aplicación este subsistema es accedido a través de una clase que transforma los datos de nuestra aplicación en los datos que el algoritmo original utiliza. Este paso es necesario para poder desarrollar en resto de los componentes de nuestra aplicación independientemente de la implementación de este subsistema. Mediante la utilización de una clase que comunique los dos sistemas, el diseño está libre de las restricciones que pudiera suponer el reutilizar las clases originales.

En la Figura 5.4 se representa el diagrama de clases de la aplicación original. Las clases en color blanco fueron clases eliminadas en la aplicación final. Las clases representadas en azul son clases en las que hubo que realizar cambios sustanciales en su estructura para poder ser reutilizadas, mientras que las clases en naranja representan aquellas clases que no sufrieron cambios. En el diagrama no se muestran aquellas clases que se corresponden con la funcionalidad relativa al segundo paso del algoritmo descrito en el artículo original (el estilizado final de las curvas utilizando trazos), ni aquellas que representaban funcionalidades ya proporcionadas por la *STL*, por motivos de espacio. También, en este diagrama no se consideran como cambios aquellas modificaciones en las que se elimina la funcionalidad no necesaria para nuestra aplicación (por ejemplo, funciones para dibujar los elementos en pantalla, o reaccionar a acciones en la interfaz antigua).

5.2.2. Diseño

En esta sección presentamos los diagramas de cada uno de los paquetes que componen la aplicación. En la Figura 5.5 se muestra el diagrama completo de clases de la aplicación, detallando cada uno de los paquetes en las subsecciones correspondientes. En este diagrama, las clases coloreadas en naranja son aquellas clases que han podido ser reutilizadas con modificaciones

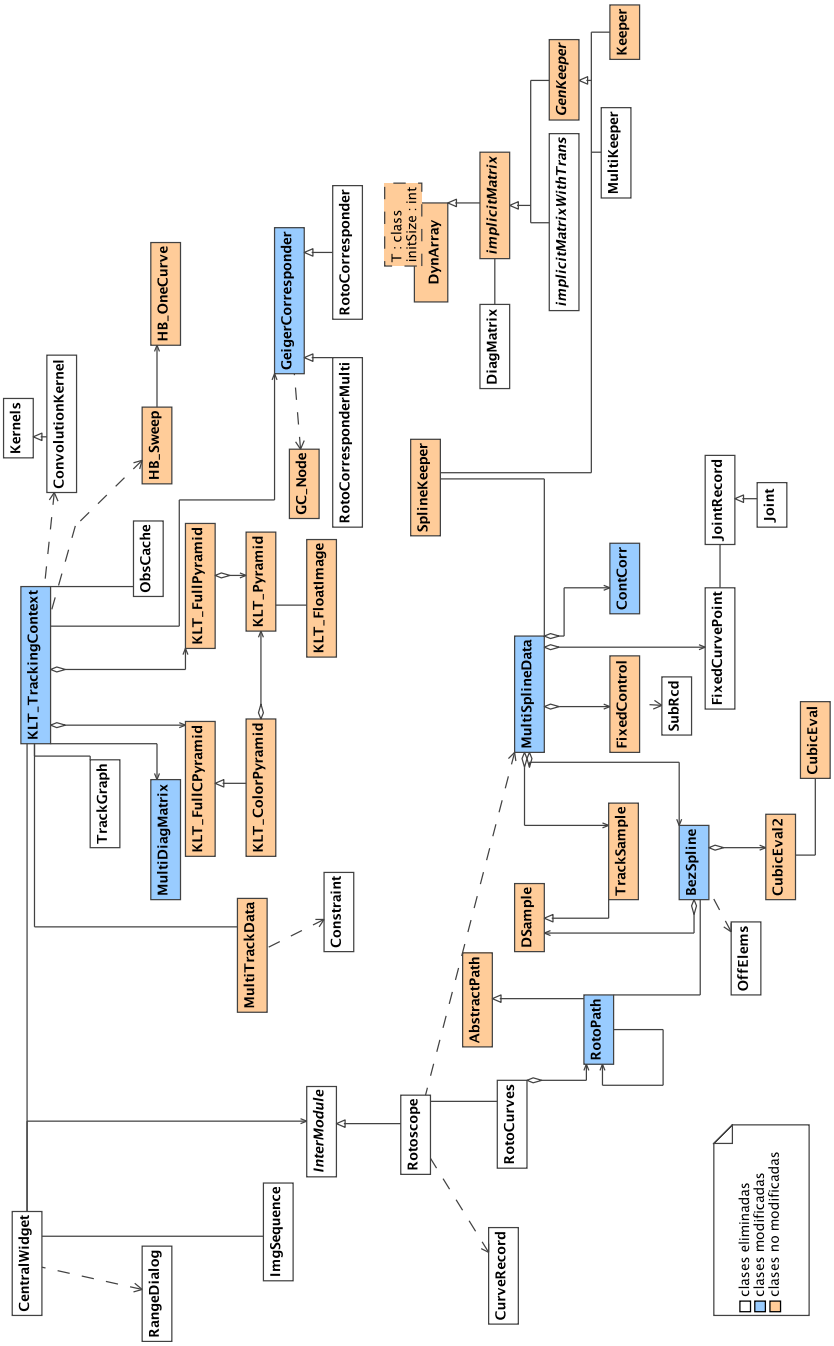


Figura 5.4: Clases en la aplicación original

de la aplicación original. Como puede verse en el diagrama, ha sido necesario diseñar la aplicación prácticamente por completo debido a los problemas de diseño de la aplicación original comentados en la sección anterior.

Paquete model

En la Figura 5.6 se puede observar las diferentes clases que componen el paquete model. El paquete model engloba aquellas clases que representan el modelo de datos con el que trabaja nuestra aplicación. Las principales clases que componen el modelo son las que representan los conceptos con los que trabaja nuestra aplicación: Frame, que representa un fotograma de la secuencia, Curve, que representa un trazo, y CurveFamily, que representa una familia de trazos. Siguiendo una estructura jerárquica, cada objeto Curve está compuesto por un objeto de la clase BezierCurve, que representa la forma de la curva, y ésta a su vez está compuesta por una serie de objetos de la clase ControlPoint, que representan cada uno de los puntos de control de esa curva. La clase RotoscopeProject funciona como una Façade, almacenando las referencias a la lista de curvas, fotogramas y familias de curvas.

Por otra parte, tanto los objetos de la clase Curve como los objetos de la clase CurveFamily pueden tener una serie de atributos, tales como color o grosor de línea. Estos atributos son modelados como un mapa de la *STL* (std::map) que relaciona el nombre del atributo con su valor, lo que facilita la futura adición de atributos. Los atributos de un trazo siempre tienen prioridad sobre los atributos de su familia. En los diagramas de esta memoria, para simplificar la notación, estas relaciones se representan como relaciones de agregación o composición, aún cuando su implementación se lleve a cabo utilizando los contenedores de la *STL* tales como vectores o mapas.

Paquete klt

En la Figura 5.7 se muestran las clases que componen el paquete klt. En este paquete están contenidas aquellas clases necesarias para implementar una parte del algoritmo de seguimiento. Así, muchas de las clases que componen este paquete han sido adaptadas de la implementación original, a la que se han añadido los cambios introducidos en nuestro trabajo comentados en el Capítulo 3.

Las clases principales relacionadas con el almacenamiento de los datos están representadas en la Figura 5.7 de color naranja. La clase RotoPath re-

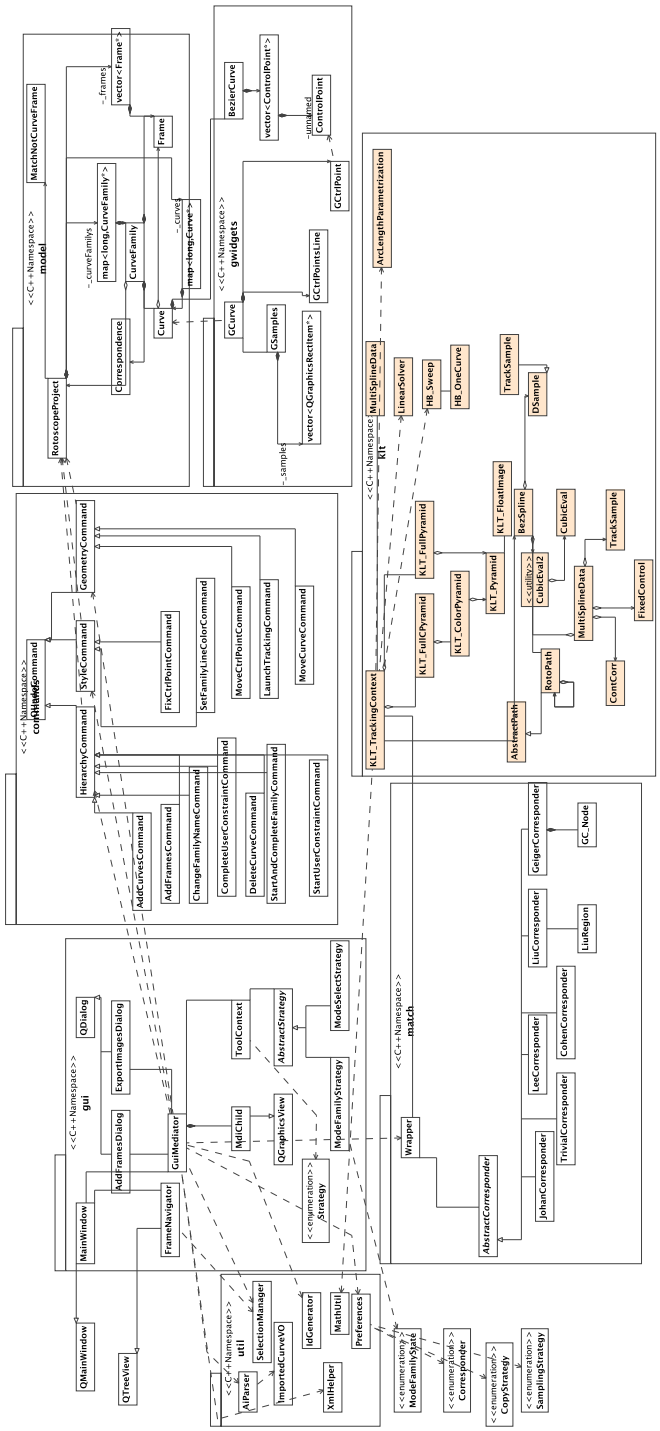


Figura 5.5: Diagrama de clases de ARAS

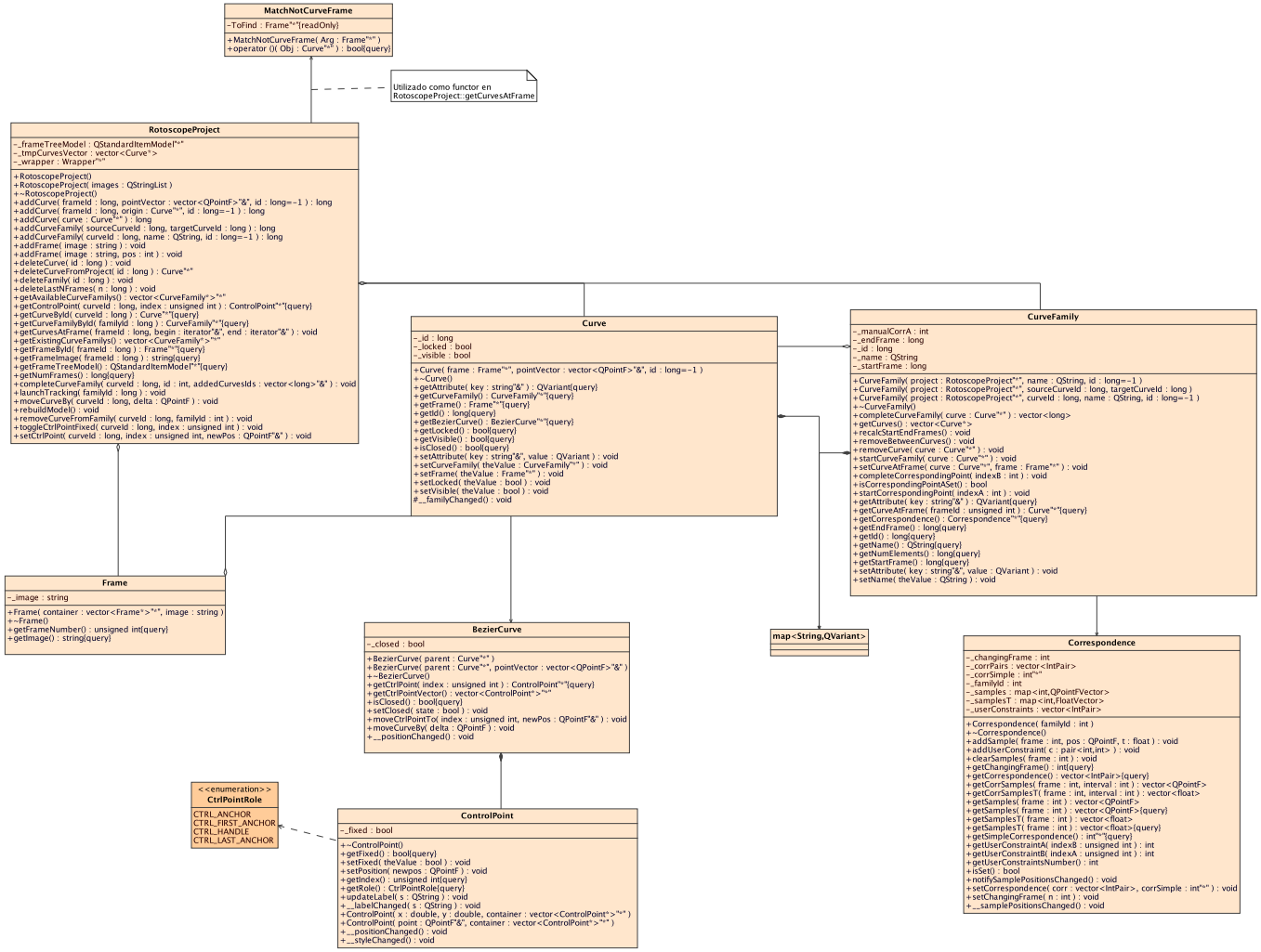


Figura 5.6: Paquete model

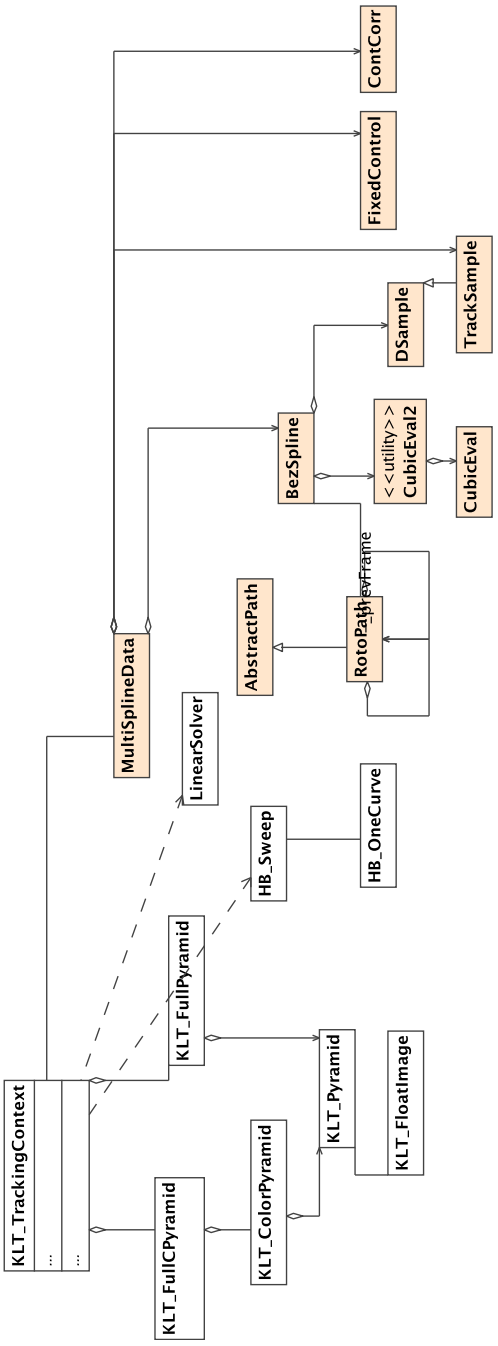


Figura 5.7: Paquete klt

presenta una curva individual, de forma semánticamente similar a la clase model::Curve. Para representar las curvas que forman parte de su familia, se almacenan referencias a los objetos RotoPath anterior y siguiente. Cada una de estas curvas contiene una referencia a un objeto BezSpline, que sería el equivalente a la clase model::BezierSpline. La clase MultiSplineData es la clase que es utilizada durante el proceso de interpolación, y consiste en una colección de objetos BezSpline, uno por cada una de las curvas de la secuencia, así como de la correspondencia entre los mismos representada por un objeto de la clase ContCorr, la información sobre el estado de sus puntos de control, en la clase FixedControl, y las muestras correspondientes, en la clase TrackSample.

Las clases que intervienen en el proceso de seguimiento propiamente están representadas en color blanco. La clase principal es KLT_TrackingContext, que controla el algoritmo de seguimiento relacionando la información contenida por las imágenes de los fotogramas, almacenada en una serie de objetos KLT_Pyramid y KLT_FullPyramid, que representan los diferentes niveles de la pirámide Gaussiana [15]. También, esta clase tiene una referencia a un objeto MultiSplineData, que almacena información sobre las curvas. Las demás clases son utilizadas durante el proceso de seguimiento; en concreto, las clases HB_Sweep y HB_OneCurve son las que implementan la funcionalidad relacionada con el precondicionador de Szeliski [82].

Paquete commands

En la Figura 5.8 se muestran las clases que constituyen el paquete commands. En este paquete están contenidas aquellas clases que representan un comando de la aplicación. Existen tres tipos de comandos, representados como clases abstractas:

- GeometryCommand: comandos que cambian la geometría de alguno de los elementos que componen la escena, tales como la modificación de alguno de los trazos.
- HierarchyCommand: comandos que afectan a la composición (jerarquía) de elementos que componen la escena, tales como la inserción de nuevos elementos.
- StyleCommand: comandos que cambian las propiedades de estilo, tales como el color de alguno de los trazos.

Estas clases heredan de la clase QtGui::QUndoCommand, que junto con la clase QtGui::QUndoStack representan el patrón Comando en las librerías *Qt*. De esta manera, cada uno de los comandos tiene dos operaciones definidas: undo() y redo(), que implementan la funcionalidad del comando. Todos los comandos tienen una referencia a un objeto RotoscopeProject, que es el objeto contra el que realizan las operaciones necesarias.

Paquete gui

El paquete Gui contiene aquellas clases que representan la interfaz de la aplicación, como se puede comprobar en la Figura 5.9. La aplicación consta de una ventana principal, representada por la clase MainWindow. Esta ventana tiene una serie de componentes:

- árbol del proyecto: representado por la clase FrameNavigator
- barra de herramientas: representada por una instancia de la clase QtGui::QActionGroup
- facilidad para deshacer: representada por una instancia de la clase QtGui::QUndoStack
- ventanas de edición de fotogramas: instancias de la clase MdiChild
- diálogos varios: AddFramesDialog, ExportImagesDialog

La clase MainWindow contiene estos elementos, pero la funcionalidad de la interfaz está implementada en la clase GuiMediator. El motivo de la inclusión de esta clase es facilitar el diseño de los componentes, de manera que la comunicación entre los diferentes elementos que componen la interfaz se realiza a través de algún *slot* de esta clase, utilizando el patrón Observer descrito anteriormente. Estos *slots* al ser llamados crean un objeto del paquete commands correspondiente, y delegan en él. Por ejemplo, si en la aplicación el usuario mueve una de las curvas, el *slot* _moveCurveBySlot(...) es llamado, lo que hace que se cree un objeto de la clase commands::MoveCurveCommand, que es el que implementa la funcionalidad.

Por otra parte, es necesario que la aplicación tenga diferentes "modos" de funcionamiento, entendiendo por "modo" una manera específica de responder al mismo evento. Por ejemplo, la respuesta de la aplicación al evento de pulsar sobre uno de los trazos depende de si se pretende asignar una curva a una

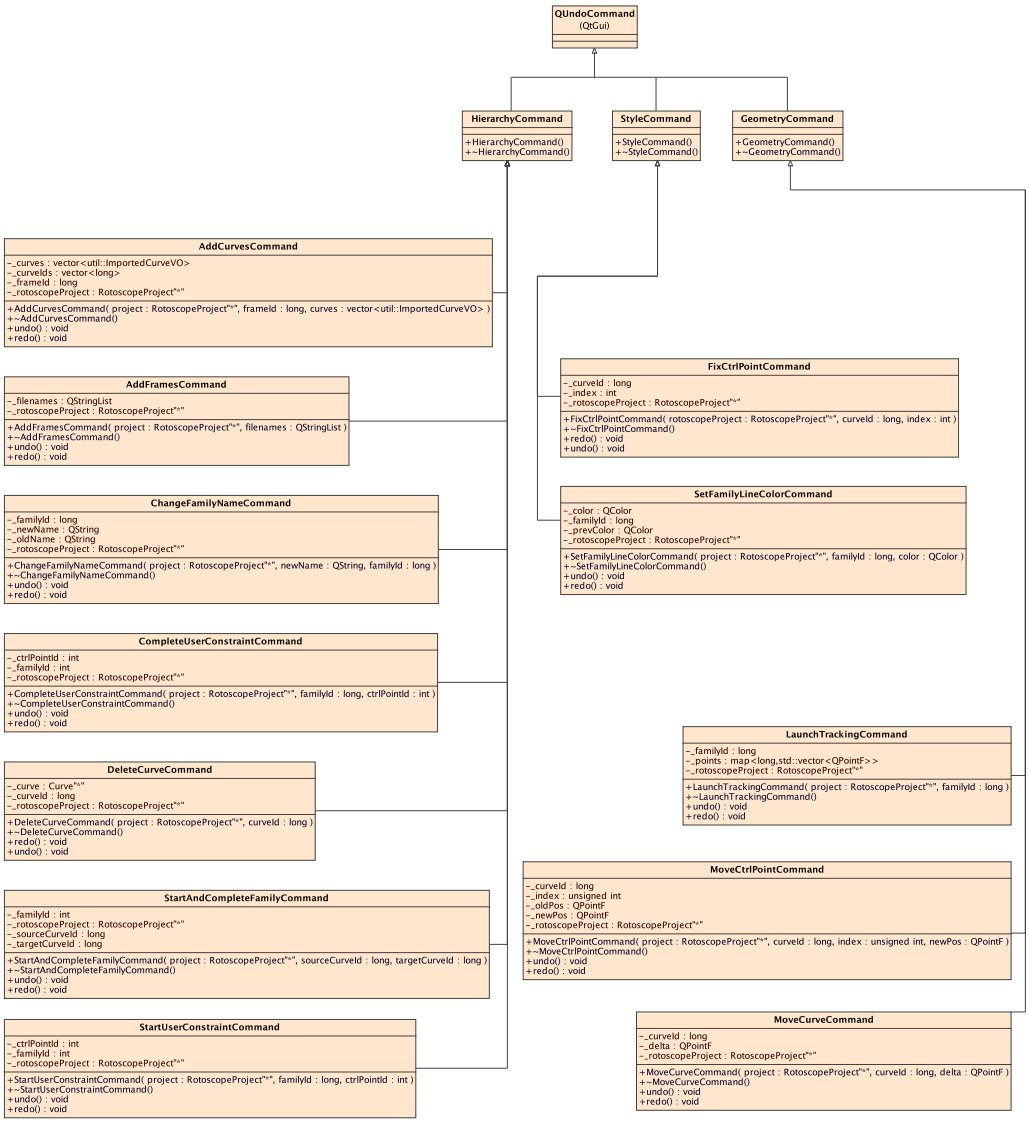


Figura 5.8: Paquete commands

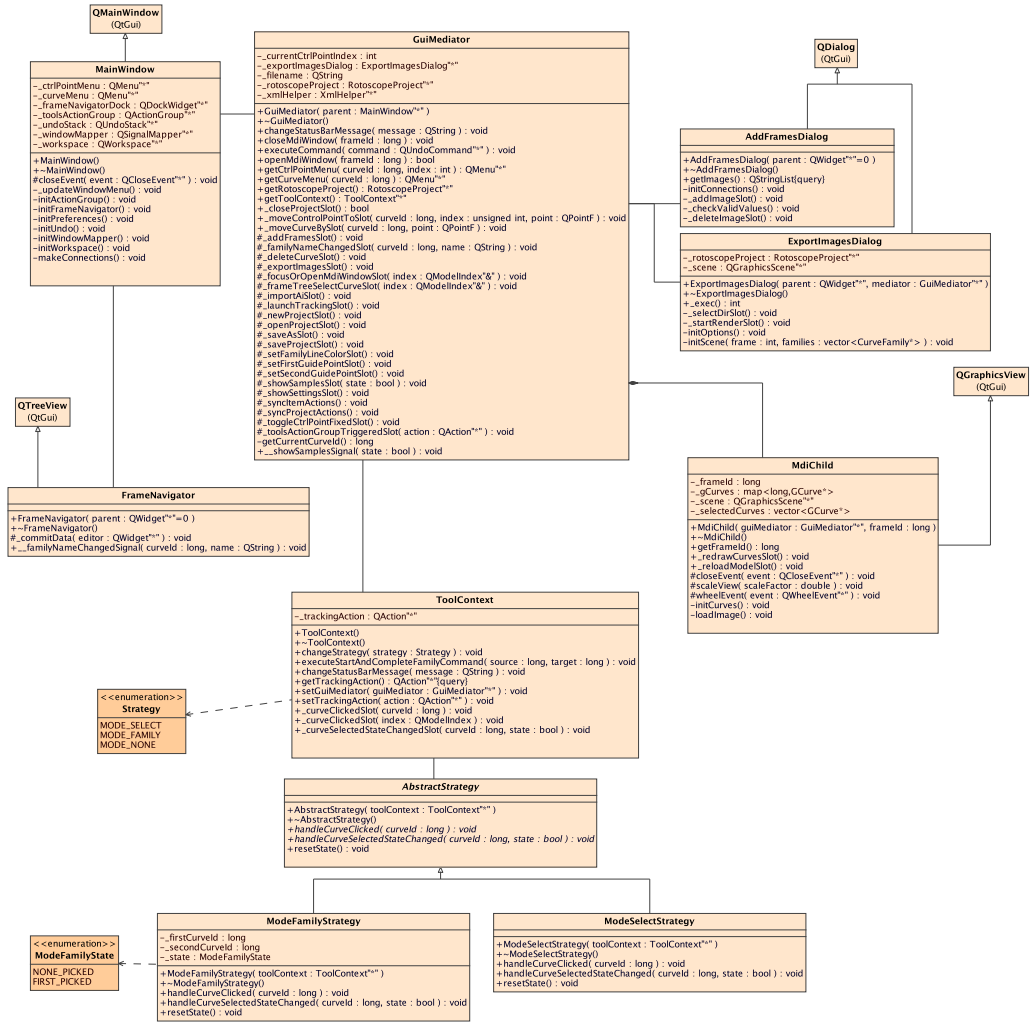


Figura 5.9: Paquete gui

familia, o de si se quiere editar la curva. Para implementar esta funcionalidad hemos utilizado el patrón Strategy, que se explicará con más detalle en la siguiente subsección. La clase ToolContext actúa como un objeto cuyos *slots* delegan el comportamiento en la estrategia concreta, ModeFamilyStrategy o ModeSelectStrategy. Estas clases, que especializan la clase AbstractStrategy, implementan la manera de reaccionar a los distintos eventos.

Paquete gwidgets

El paquete gwidgets almacena los elementos gráficos que pueden ser manipulados por el usuario, donde cada uno de ellos representa uno de los elementos del modelo. Como se puede apreciar en la Figura 5.10, la clase GCurve es la representación de uno de los objetos model::Curve, y está compuesta por una serie de objetos de la clase GCtrlPoint, asociados a su vez con un objeto model::ControlPoint. Las otras dos clases que pertenecen a este paquete representan elementos gráficos adicionales: la clase GCtrlPointsLine representa las líneas que se dibujan entre los puntos de control de una curva, y la clase GSamples es la representación gráfica del conjunto de muestras de una curva.

Las clases contenidas en este paquete aprovechan la funcionalidad ofrecida por el marco de desarrollo *Graphics View Framework* [58] de las librerías *Qt*. Éste proporciona un conjunto de clases genéricas con el propósito de representar elementos gráficos en la pantalla, donde cada uno de esos elementos es un objeto de una clase que recibe directamente los eventos del usuario y pueden aplicársele transformaciones y modificaciones de forma eficiente. Esto facilita el desarrollo de la interfaz gráfica en gran medida. En particular, cada una de las ventanas de un fotograma heredan de la clase QtGui::QGraphicsView, y cada uno de los elementos individuales (GCurve, GCtrlPoint o GCtrlPointsLine) especializan la clase QtGui::QGraphicsItem.

Paquete match

Este paquete almacena las clases utilizadas para el cálculo de la correspondencia entre dos curvas (ver Figura 5.11). Los diferentes algoritmos de correspondencia están representados por una de las clases de este paquete:

- CohenCorresponder: Método basado en correspondencia entre los vectores tangentes (ver Sección 3.3)

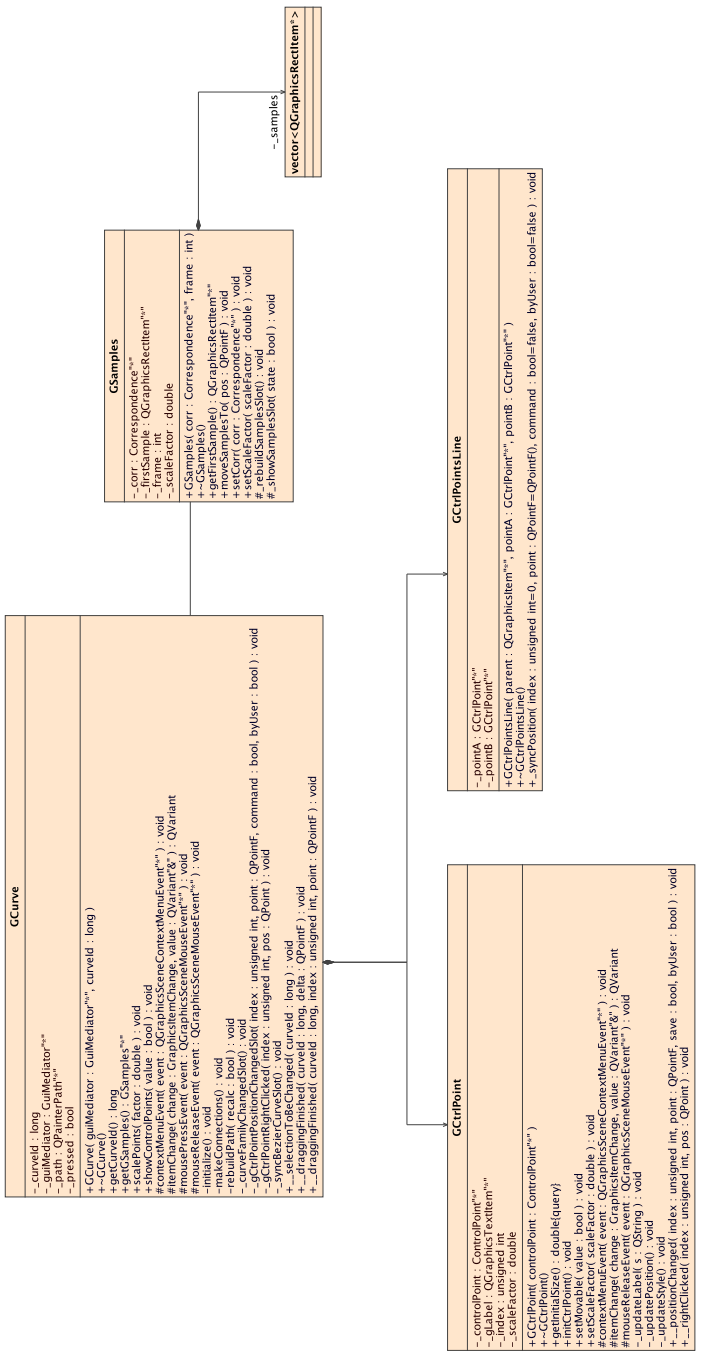


Figura 5.10: Paquete gwidgets

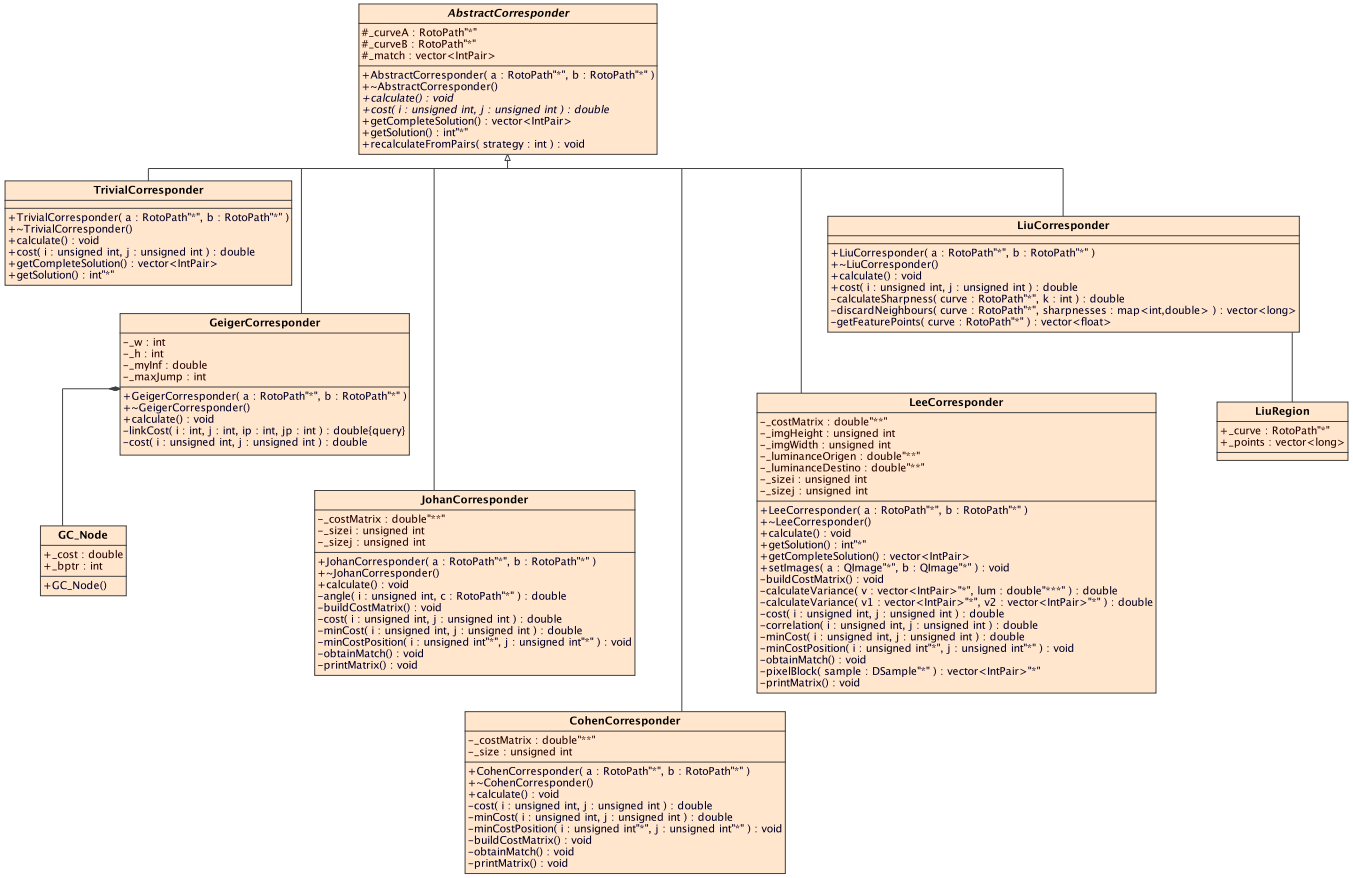


Figura 5.11: Paquete match

- GeigerCorresponder: Método basado en el vector de desplazamiento (ver Sección 3.6)
- JohanCorresponder: Método basado en la similitud de las curvas (ver Sección 3.4)
- LeeCorresponder: Método guiado por imagen (ver Sección 3.5)
- LiuCorresponder: Método basado en propiedades geométricas de un conjunto característico de puntos (ver Sección A.1)
- TrivialCorresponder: método trivial, que hace corresponder las muestras una a una, sin aplicar ningún algoritmo.

Todos estos algoritmos heredan de la clase AbstractCorresponder. Es de destacar que estas clases no trabajan sobre elementos del paquete model, sino con las representaciones de las curvas utilizadas en la implementación original (paquete klt). Esto es así puesto que el cálculo de la correspondencia es uno de los pasos internos del algoritmo de interpolación, y durante todo el proceso se trabaja con las curvas ya representadas de esta manera.

Paquete util

En el paquete util están contenidas aquellas clases que son utilizadas en diferentes partes de la aplicación para realizar la entrada y salida de datos o implementar operaciones matemáticas en general. En la Figura 5.12 se muestran las clases que componen este paquete: Las clases AiParser, SvgParser y XmlHelper (ver Figura 5.12a) son clases utilizadas para la entrada y salida de datos. La clase XmlHelper se encarga de la transformación de los datos del modelo al formato utilizado por la aplicación y viceversa. La clase AiParser es un analizador sintáctico (*parser*) de ficheros en formato *Adobe Illustrator* (.ai), que reconoce una serie de primitivas *PostScript* en esos ficheros y los convierte en curvas utilizadas por nuestro modelo. La clase SvgParser, con el mismo diseño que la clase AiParser, convierte un conjunto parcial de primitivas de los ficheros en formato *SVG* (.svg) en curvas utilizadas por nuestro modelo.

Las clases ArcLengthParametrizer y MathUtil (ver Figura 5.12b) son clases que implementan algoritmos u operaciones matemáticas en general. En MathUtil se implementan el método de Newton-Raphson [65], y el cálculo de la integral de una función utilizando el método de Gauss [65] y el método

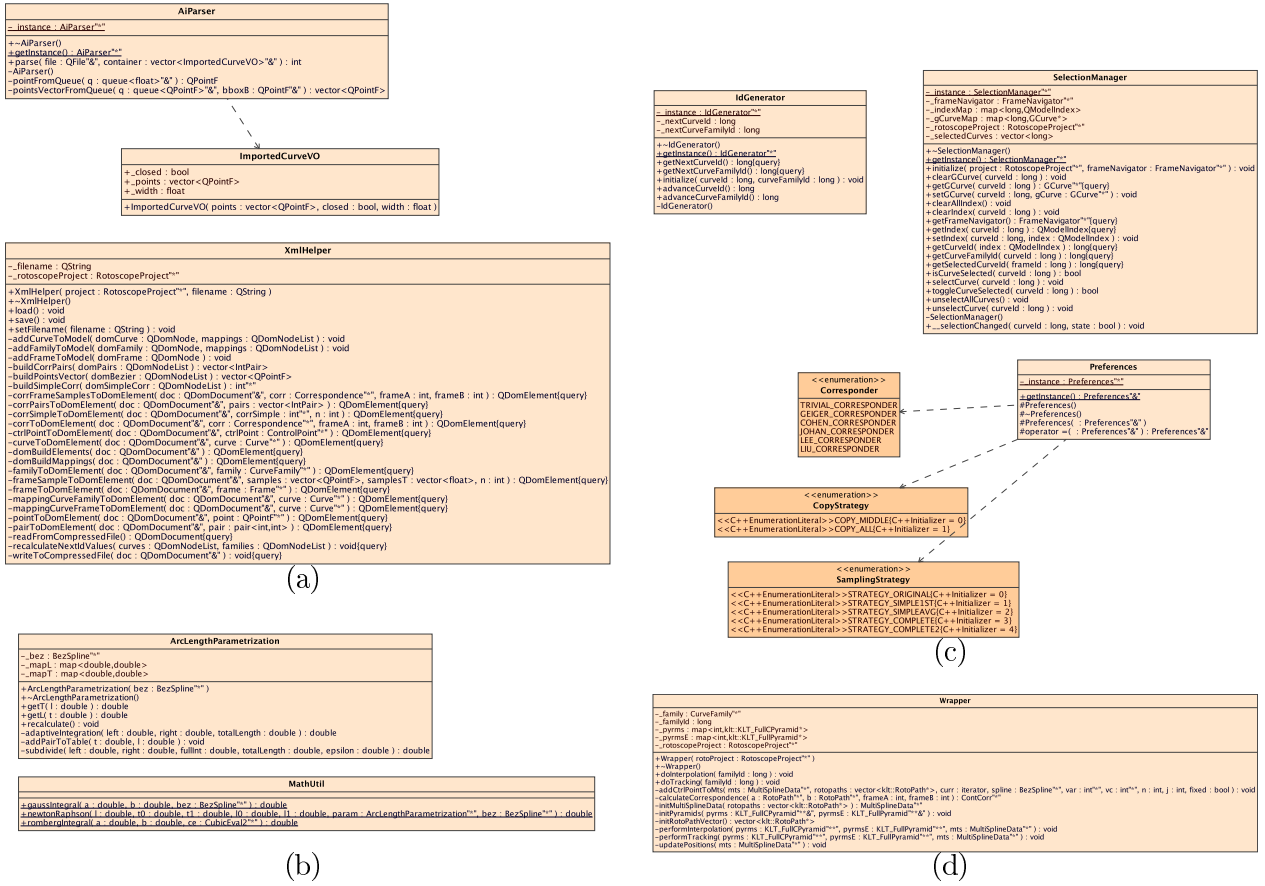


Figura 5.12: Paquete util

de Romberg [65]. Por su parte, la clase ArcLengthParametrizer implementa el método descrito en [40] para calcular la reparametrización de una curva en función de su longitud de arco.

Las clases Preferences, SelectionManager y IdGenerator (ver Figura 5.12c) son clases *Singleton* con diferente propósito. En general, estas clases proporcionan funcionalidades comunes a diferentes partes de la aplicación, por eso hemos decidido implementarlas siguiendo este patrón. La clase IdGenerator permite la generación de identificadores únicos para cada uno de los elementos (curvas y familias) que componen la escena. La clase SelectionManager es utilizada para sincronizar la selección de elementos en la interfaz: como es posible seleccionar las curvas desde varios elementos de la interfaz diferentes, esto es, el usuario puede seleccionar una curva pulsando sobre su representación gráfica, o sobre el elemento correspondiente en el árbol del proyecto. Por tanto, es necesario mantener sincronizados ambos elementos. Además, la gran mayoría de comandos necesita saber cuál es la curva o familia seleccionada para actuar sobre ella. La clase SelectionManager es la que implementa esta funcionalidad, desacoplando el estado de la selección de las ventanas u objetos individuales y simplificando el diseño de los diferentes componentes. De la misma manera, la clase Preferences representa las preferencias utilizadas en la aplicación. El usuario puede configurar estas preferencias, tales como el método de correspondencia empleado, los parámetros utilizados en ese método, o diversas opciones de visualización, desde la ventana principal. Los valores de estas preferencias son utilizados en diferentes partes de la aplicación, por lo que almacenarlos en un objeto independiente favorece el bajo acoplamiento entre los distintos componentes.

Por último, la clase Wrapper (ver Figura 5.12d) es la encargada de realizar la conversión entre los datos utilizados en nuestra aplicación y los datos utilizados por la implementación original en el subsistema klt, y viceversa. La utilización de esta clase permite que la aplicación fuera diseñada sin preocuparse por los detalles de implementación utilizados en la aplicación original, puesto que sólo es necesaria cuando se lanza el proceso de seguimiento. Cuando esto ocurre, los datos almacenados en las clases del paquete model son procesados por esta clase y pasados al algoritmo de seguimiento, siendo transformados de nuevo cuando éste concluye.

5.2.3. Patrones utilizados

En esta sección describiremos brevemente los patrones aplicados durante el diseño de la aplicación.

Observer

El patrón de diseño Observer define una dependencia del tipo uno a muchos entre un objeto sujeto (observable) y una serie de objetos observadores, que reaccionan cuando el objeto observable notifica un cambio de estado. Con la aplicación de este patrón se consigue desacoplar ambas clases de objetos, y en la implementación el objeto observable no sabe qué clases lo observan.

Las librerías *Qt* proporcionan un mecanismo denominado "*signals and slots*"[60] que implementa el patrón Observer utilizando el preprocesador de C++. En *Qt*, se puede definir para cualquier clase una serie de señales (*signals*) y de ranuras (*slots*) utilizando unas macros específicas. En tiempo de ejecución, utilizando también una macro específica, se puede conectar una señal determinada de un objeto A con una ranura determinada de otro objeto B, de manera que cuando el objeto A emita la señal, se ejecute automáticamente la función correspondiente en el objeto B. Las relaciones entre señales y ranuras pueden ser modificadas en cualquier momento durante la ejecución del programa. Este mecanismo es particularmente útil en el desarrollo de los componentes de la interfaz de la aplicación, ya que simplifica la interacción con el controlador y entre los diferentes elementos que componen la GUI.

En nuestra aplicación, el patrón de diseño Observer se utiliza en los siguientes contextos:

1. **Sincronizar la interfaz y el controlador**: el usuario interacciona con los distintos elementos de la interfaz, lo que da lugar a la generación de señales. Estas señales son recogidas por el mediador GuiMediator, que a su vez lleva a cabo las acciones oportunas.

2. **Sincronizar los diferentes GWidgets con el modelo**: cuando las posiciones o propiedades de algunos de los componentes del modelo son cambiadas, se emite una señal que es recogida por su representación gráfica correspondiente. De esta manera es posible mantener las diferentes vistas sincronizadas, sin tener que redibujar toda la escena cada vez que se produce un cambio en alguno de sus elementos. Por

ejemplo, si se lanza el proceso de interpolación sobre una curva en particular, lo que hace que se modifiquen las posiciones de sus puntos de control, su objeto BezierCurve emite una señal positionChanged, que hace que el objeto gráfico GCurve correspondiente se redibuje, en lugar de redibujar todos los elementos de la ventana.

3. **Sincronizar los diferentes componentes de una curva**: para implementar las operaciones en las que el usuario mueve un punto de control o una curva, se utilizan señales y ranuras para que todos los componentes de una curva (el trazo, los puntos de control y los posibles decoradores) estén sincronizados.

En la Figura 5.13 se muestra un ejemplo de la utilización de este mecanismo en nuestra aplicación. En ella se representa de manera esquemática la utilización de señales y ranuras desde que el usuario pulsa el botón de "Lanzar interpolación" hasta la actualización de la posición de uno de los GCurve de un fotograma de la familia de curvas. En el ejemplo se supone que sólo uno de los fotogramas está abierto en la aplicación. En el caso en el que hubiera más de un fotograma abierto, la parte inferior de la figura (la recepción de la señal positionChanged por parte de la GCurve) se repetiría para cada uno de los fotogramas abiertos. En el caso en el que no hubiera ningún fotograma abierto, la señal es emitida por las BezCurve pero no es recibida por ningún objeto. En la Figura 5.14 se muestran todas las clases que contienen señales y/o ranuras, organizadas por paquetes: el paquete gui (ver Figura 5.14a), el paquete model (ver Figura 5.14b), el paquete util (ver Figura 5.14c) y el paquete gwidgets (ver Figura 5.14d). En los diagramas de esta memoria, con el fin de diferenciar las señales y ranuras de los métodos de las clases, utilizaremos los estereotipos «signal» y «slot», además de el prefijo "__" para las señales y "_" para las ranuras.

Singleton

Diversas funcionalidades de la aplicación han sido implementadas utilizando el patrón Singleton. El patrón Singleton se utiliza para restringir la instanciación de una clase determinada a sólo un objeto, de manera que un único objeto sea utilizado en diversas partes de la aplicación. La clase es responsable de asegurar que sólo se utiliza una instancia de la misma. En C++, la implementación de un Singleton se realiza definiendo una clase con

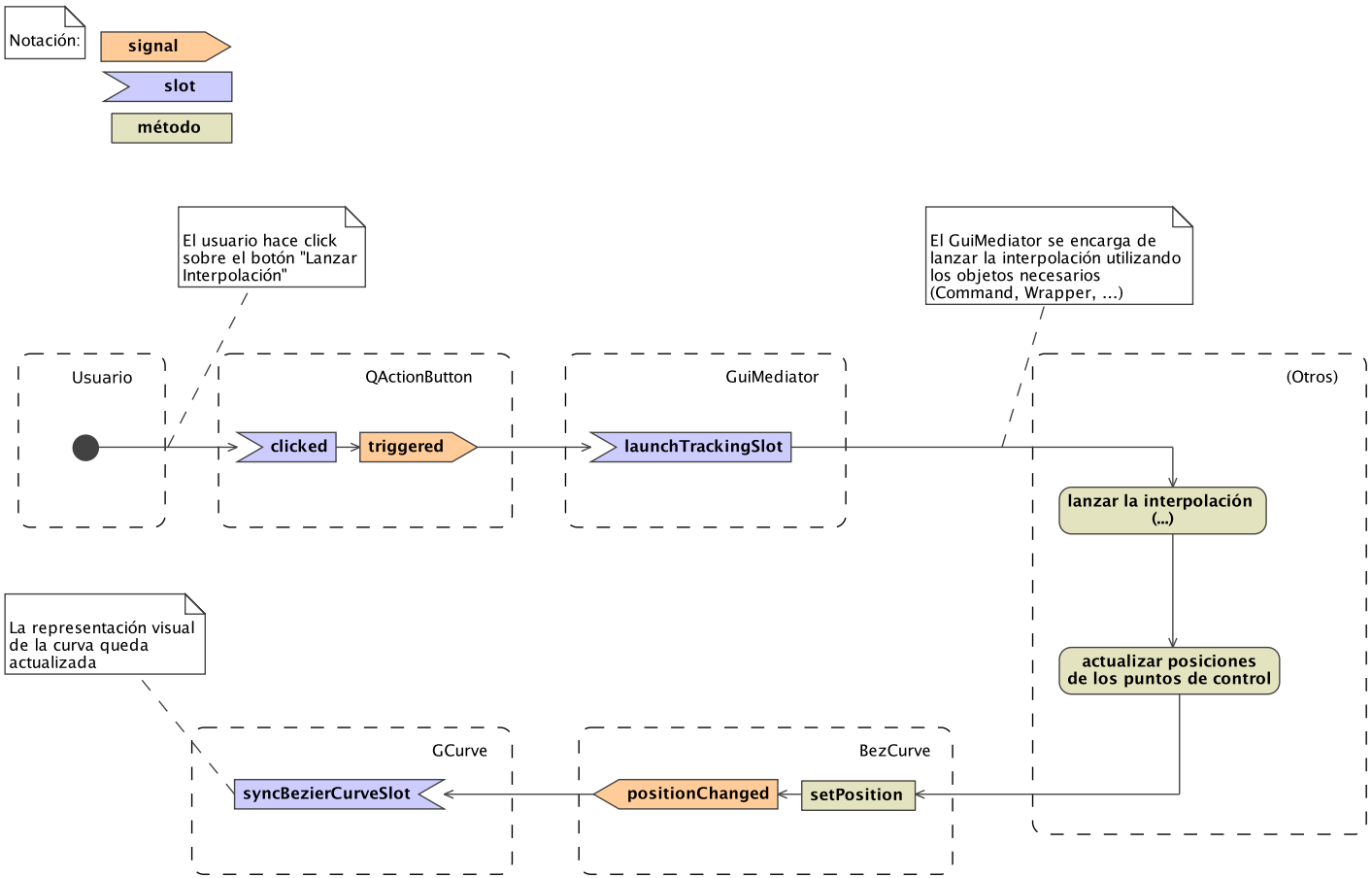


Figura 5.13: Ejemplo de la utilización de señales y ranuras en ARAS

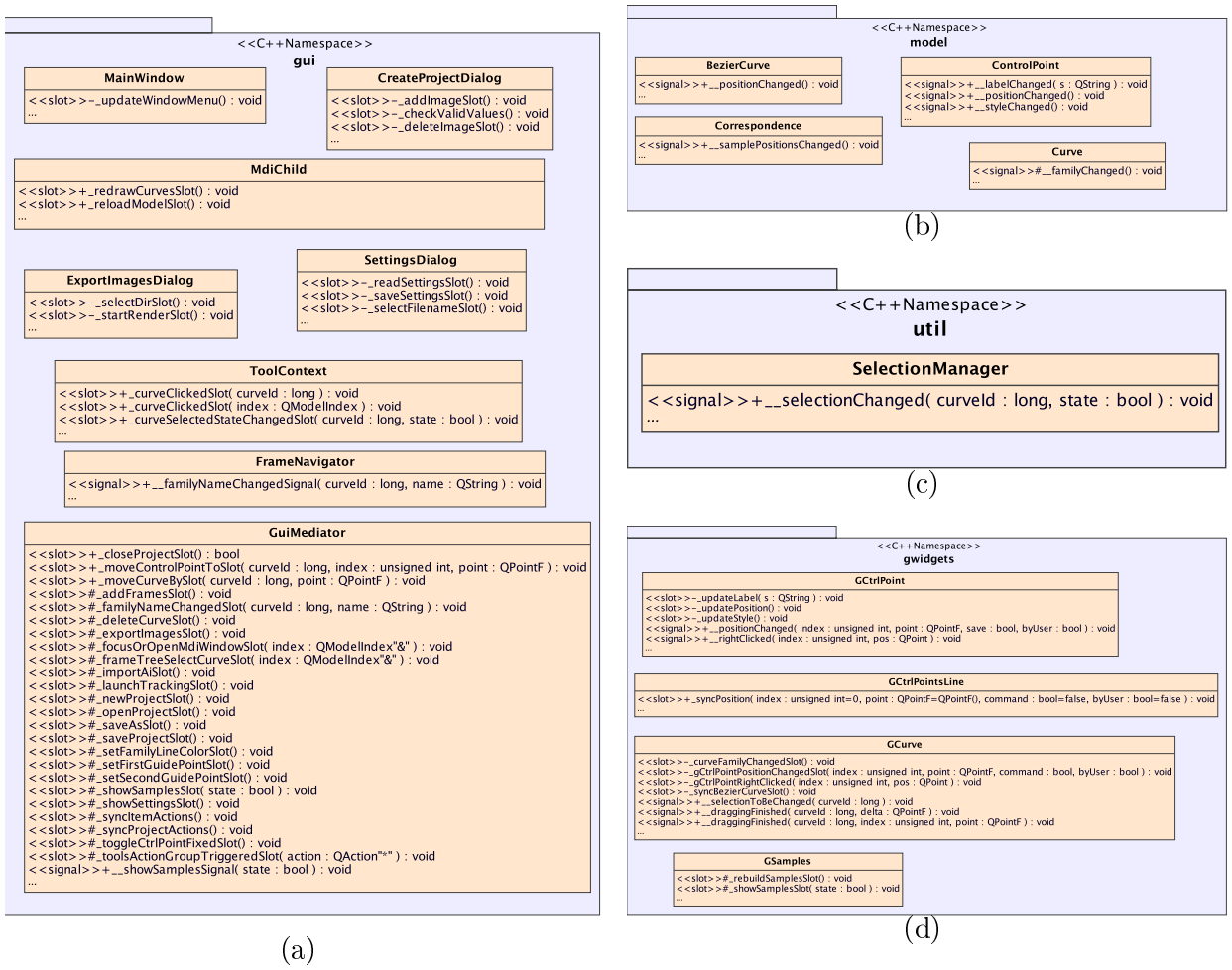


Figura 5.14: Clases que contienen señales y/o ranuras

un método estático getInstance(), que en la primera llamada crea un objeto de esa clase, y en llamadas siguientes devuelve una referencia a ese objeto. El constructor es definido como privado, de manera que el objeto no puede ser creado desde ningún otro punto de la aplicación.

En nuestra aplicación, varias clases del subsistema util son definidas utilizando un Singleton: las clases AiParser, IdGenerator, Preferences, SelectionManager. La funcionalidad de las mismas está explicada en el apartado dedicado al paquete util.

Command

El patrón de diseño Command está basado en la utilización de objetos comando, que encapsulan una acción o comportamiento. El objeto comando es responsable de llevar a cabo las acciones necesarias, permitiendo que el objeto emisor quede desvinculado de ellas. En las librerías *Qt* este patrón está implementado utilizando las clases QtGui::QUndoCommand y QtGui::QUndoStack. De esta manera, un objeto comando es una instancia de la clase QtGui::QUndoCommand que mediante la implementación de dos operaciones, undo y redo, definen la acción a realizar.

En la aplicación de ARAS, cada una de las clases del paquete commands implementa una acción de la aplicación, especializando varios tipos de comandos según su semántica. De esta manera se localiza la implementación del comportamiento en estas clases, independientemente de los demás componentes de la interfaz.

Strategy

El patrón strategy pretende encapsular un comportamiento o algoritmo perteneciente a una familia de algoritmos, permitiendo que éstos sean intercambiados de forma independiente. De esta manera, un objeto y su comportamiento son divididos en dos clases diferentes y la estrategia a aplicar puede ser cambiada en tiempo de ejecución.

En nuestra aplicación, el patrón strategy se utiliza en dos escenarios diferentes. En primer lugar, en la interfaz de nuestra aplicación es necesario utilizar diferentes "modos" de interacción con el sistema. La aplicación responde de manera diferente a los eventos generados por el usuario dependiendo de si se encuentra en un modo u otro, por ejemplo, el modo de selección de curvas, y el modo de asignación de curvas a familias. Las clases del paquete

gui, ModeFamilyStrategy y ModeSelectStrategy aplican este patrón en relación con la clase ToolContext.

También, la utilización de algoritmos de correspondencia utiliza este patrón. Cada uno de los algoritmos de correspondencia está definido en una clase diferente del paquete match, siendo seleccionada en tiempo de ejecución la estrategia a aplicar.

Mediator

El patrón mediator sirve para encapsular la interacción de varios a varios entre diversos objetos. Este patrón permite que el acople entre los diferentes objetos que interactúan sea bajo, de manera que los objetos se comuniquen con un objeto mediador en lugar de comunicarse entre sí. Mediante la utilización del objeto mediador, es posible controlar más fácilmente la manera que tienen de interactuar estos elementos.

En nuestra aplicación, se utiliza una instancia de la clase GuiMediator para definir la interacción entre los diferentes elementos que componen la interfaz. En concreto, se combina este patrón con el patrón observador para centralizar la interacción en esta clase: los diferentes componentes de la interfaz generan señales que son capturadas por el mediador, que actúa en consecuencia.

Model-View-Controller

El patrón arquitectónico Model-View-Controller (MVC) permite desacoplar el modelo de la vista, introduciendo un elemento intermedio que es el controlador. La utilización de este patrón está favorecida además por el soporte que las librerías *Qt* tienen del mismo. Así, en nuestra aplicación se utiliza este patrón en dos contextos diferentes:

- Separación de la interfaz de la aplicación con el modelo: los diferentes elementos que componen la interfaz de nuestra aplicación se pueden modificar de forma independiente de los componentes que implementan la funcionalidad del sistema. En concreto, los componentes gráficos están agrupados en los paquetes gui y gwidgets, utilizando el objeto Gui::GuiMediator (mediante el mecanismo de señales y ranuras anteriormente descrito) para comunicarse entre sí y con el resto del sistema.

- Implementación del árbol de proyecto: las librerías *Qt* proporcionan una implementación del patrón MVC con las familia de clases relacionadas

con [59] Qt::QAbstractItemModel y Qt::QAbstractItemView. Utilizando estas clases se puede asociar un modelo personalizado o uno de los modelos predefinidos en las librerías a un número indefinido de vistas, y la librería proporciona las facilidades necesarias para mantener las vistas y el modelo sincronizados de manera automática. En nuestra aplicación hemos definido un modelo que representa la jerarquía del proyecto, y utilizado una representación genérica en forma de árbol para implementar la funcionalidad correspondiente al árbol del proyecto.

Façade

El patrón de diseño Façade proporciona una interfaz unificada sencilla que hace de intermediaria entre un cliente y un conjunto de clases, a menudo con representación semántica similar. Mediante la utilización del patrón Façade, se reduce el acoplamiento entre las diferentes clases que componen un sistema.

En nuestra aplicación, se utiliza el patrón de diseño Façade para proporcionar un sólo punto de acceso a un determinado paquete, facilitando así el desarrollo de cada uno de los paquetes individuales y reduciendo el acoplamiento en el diseño. En concreto, en el paquete gui se utiliza el patrón Façade de manera que la clase RotoscopeProject centraliza el acceso a los diferentes componentes que componen un proyecto de ARAS.

5.3. Implementación

La aplicación ha sido implementada en el lenguaje *C++* [81], haciendo uso de la *Standard Template Library* en la medida de lo posible. El lenguaje *C++* combina las facilidades de la orientación a objetos con la flexibilidad de manejo de las estructuras mediante la utilización de punteros y otros recursos. Además, existe un conjunto de librerías estándares, la *STL*, que utilizan programación genérica para implementar una serie de funcionalidades comunes optimizadas y probadas tales como contenedores (vectores, mapas, listas) o algoritmos genéricos de búsqueda. En nuestra aplicación hemos utilizado los contenedores y las facilidades proporcionadas por la *STL* siguiendo el principio de reutilización de código.

Para el desarrollo de la interfaz se ha utilizado la librería *Qt* [57], en su versión 5.3.2. Esta librería no se limita a facilitar el desarrollo de aplicaciones Gui e incorporar una serie de facilidades como el mecanismo de señales y

ranuras, sino que también proporcionan funcionalidades orientadas al desarrollo de aplicaciones en general. Además, una de sus ventajas principales es que es una librería multiplataforma.

También se hace uso de las librerías *boost* [18] en la implementación del algoritmo de seguimiento. En particular, se utilizan las implementaciones de matrices dispersas de esta librería, que optimizan la utilización de memoria durante el algoritmo de seguimiento.

Para la importación de las imágenes que componen los fotogramas de las animaciones se ha utilizado la librería *ImageMagick* [56], que proporciona funciones y utilidades para importar y manipular una gran cantidad de formatos de imagen.

Durante el desarrollo de la aplicación, se ha utilizado la librería *log4cxx* [35] para proporcionar facilidades de registro de la aplicación con diferentes niveles de detalle.

Por último, se ha utilizado el generador de máquinas de estados finitos *ragel* [84, 85] para la implementación del analizador sintáctico parcial de archivos *Adobe Illustrator*.

Capítulo 6

Metodología de desarrollo y planificación

En este capítulo se describe el método elegido para el desarrollo de este trabajo, justificando la decisión de utilizar un sistema ágil. Posteriormente se expone la planificación propuesta utilizando un diagrama de Gantt con los plazos e hitos, y se concluye con un análisis de los costes del proyecto.

6.1. Metodología de desarrollo

En esta sección se describe la metodología de desarrollo elegida para nuestro trabajo, describiendo en primer lugar las alternativas consideradas, y posteriormente describiendo la metodología elegida y justificando su adecuación a nuestra aplicación.

6.1.1. Metodologías consideradas

Durante la fase de gestación del proyecto se han analizado las diferentes metodologías más populares para el desarrollo, con el fin de evaluar los beneficios e inconvenientes a priori y conseguir que el desarrollo de nuestro trabajo se realice de la manera más eficiente posible. Una metodología es un conjunto de métodos, técnicas y herramientas que se utilizan en el desarrollo de software para determinar los pasos a seguir a lo largo de todo el ciclo de vida del proyecto. Del conjunto de metodologías más utilizadas hemos evaluado para nuestro proyecto la aplicación de la metodología de desarrollo en cascada, y de la metodología de desarrollo ágil.

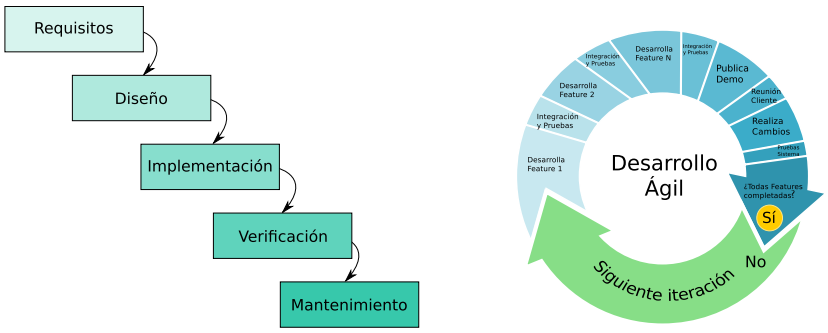
Desarrollo en cascada

La metodología de desarrollo en cascada [72] es una de las metodologías de desarrollo clásicas, adaptada de otras disciplinas con mayor tradición. Se trata de un diseño secuencial no iterativo, en el que sus diferentes fases (requisitos, diseño, implementación, verificación y mantenimiento) se encadenan, cayendo "en cascada" sucesivamente. Es un marco de trabajo altamente rígido en el que, al final de cada una de las fases, se lleva a cabo una revisión formal a través de la cual se determina si el proyecto está listo para avanzar a la siguiente. El desarrollo en cascada pone también mucho énfasis en la documentación de cada uno de los pasos y es muy exhaustivo, típicamente empleando alrededor del 20-40 % del tiempo en las fases dos primeras fases, 30-40 % en el desarrollo y el tiempo restante en la verificación y mantenimiento. Existen diferentes variaciones del desarrollo en cascada que intentan solventar estas limitaciones y proporcionar un sentido más dinámico a la metodología manteniendo sus virtudes, modificando el número de fases o permitiendo una interacción ente ellas mayor.

La rigurosidad inherente a esta metodología hace posible la identificación de defectos en las fases iniciales del desarrollo, reduciendo el tiempo y coste necesarios para corregirlos. Además, la claridad y la revisión de cada una de las fases está integrada en el modelo, favoreciendo la optimización de recursos y el seguimiento de cada una de las fases de desarrollo. Sin embargo, para que el modelo en cascada funcione correctamente es necesario un gran esfuerzo y concreción para definir los objetivos de cada una de las fases, lo que en la práctica es difícil de conseguir por diversos motivos (imposibilidad de definir completamente todos los requisitos al comienzo de un proyecto, evolución de la tecnología durante el ciclo de desarrollo, aparición de factores imprevistos, etc), resultando en un incremento de los costes y tiempos estimados inicialmente.

Desarrollo ágil

El término "metodologías de desarrollo ágiles" engloba a una serie de metodologías de desarrollo que siguen un conjunto de principios que intentan reflejar y dar soporte a la evolución de los requisitos y de las soluciones a lo largo del ciclo de vida del producto mediante la colaboración de grupos auto-organizados y multidisciplinarios. Son metodologías basadas en el desarrollo iterativo e incremental, repitiendo sucesivamente las diferentes fases



(a) Desarrollo en cascada (b) Desarrollo ágil

Figura 6.1: Procesos generales de las metodologías de desarrollo en cascada (a) y ágil (b)

que conforman el desarrollo en cascada pero orientadas a unidades de pequeño tamaño, de forma que el impacto provocado por defectos en alguna de esas unidades afecte lo mínimo posible al proyecto en general. También hace hincapié en la interacción con el cliente, a través de la generación de un producto o prototipo al final de cada iteración que permite encontrar defectos y recibir información de forma más temprana.

La definición original de la metodología [13] describe doce principios que plasman cuatro ideas principales:

- **Individuos e interacciones**: la auto-organización y la motivación son importantes. Otros valores promovidos son la co-ubicación y la programación por parejas.
- **Software funcional**: un software funcional (prototipo) es más útil y mejor recibido por los clientes que la presentación tan sólo de documentos y especificaciones.
- **Interacción con el cliente**: los requisitos no pueden ser definidos por completo al comienzo del ciclo de desarrollo, por lo que es crucial la participación continua del cliente.
- **Respuesta ante el cambio**: las metodologías ágiles se centran en respuestas rápidas ante los cambios, y desarrollo continuo.

Dentro de las metodologías ágiles existen diversas variaciones (entre otras, *Scrum* [83], *Kanban* [10], *Extreme Programming [12]*, o *Feature-driven deve-*

lopment [67]) que consiguen los objetivos propuestos por los principios generales a través del empleo de diversas técnicas.

En contraposición al desarrollo en cascada, las metodologías de desarrollo ágiles permiten un margen de maniobra mayor ante imprevistos y un coste mucho menor cuando las especificaciones son cambiantes o no pueden ser definidas al principio del proyecto. Su naturaleza ágil se ajusta mejor a las tendencias recientes en los proyectos de desarrollo de software, en los que habitualmente los grupos de trabajo son dinámicos y posiblemente geográficamente dispersos, la comunicación es constante y a través de diferentes vías con el cliente, la necesidad de disponer prototipos funcionales en todo momento es deseable, y las tecnologías empleadas evolucionan y son reemplazadas a un ritmo elevado. Por el contrario, entre sus desventajas se encuentra la relativa falta de documentación ya que no se contempla como una parte tan integral de la metodología y los costes derivados de ello. La falta de eficiencia en proyectos a gran escala o en organizaciones de gran tamaño, derivada de la laxitud en comparación con otros diseños tradicionales, o en proyectos en los que los requisitos pueden ser determinados con claridad es también frecuentemente citada como uno de sus puntos débiles.

6.1.2. Metodología utilizada

Para decidir la metodología utilizada para nuestro trabajo, se han identificado las siguientes características del mismo:

- **La naturaleza abierta de la parte de investigación**: si bien la estrategia general (análisis de los algoritmos de seguimiento, utilización de una aplicación existente [6] como base, y aplicación de algoritmos de correspondencia) es clara e identificable desde la gestación del proyecto, existe una cierta incertidumbre inherente a las labores de investigación, que se traduce en una dificultad para definir completamente todos los requisitos de manera formal y completa. En particular, la decisión sobre cuáles y cuántos algoritmos de correspondencia serán finalmente evaluados e implementados es difícil de estimar, ya que depende de los resultados de los análisis de los artículos y su rendimiento exacto se desconoce hasta el momento de la implementación, que a su vez depende del desarrollo de la herramienta *ARAS*.

- **Evaluación de la implementación existente**: como se describe en

la Sección 5.2.1, la complejidad del código fuente original es difícil de estimar sin un análisis detallado, debido a la falta de documentación y referencias. Esto dificulta el establecimiento de unos criterios de tiempo y esfuerzo necesarios claros para la etapa de ingeniería inversa, y el grado en que la aplicación original puede ser reutilizada o descompuesta en componentes de menor granularidad, traduciéndose en dificultades a la hora de decidir el diseño de la aplicación *ARAS* al comienzo del proceso. También se tiene en cuenta la familiaridad del equipo con las tecnologías empleadas (lenguaje de programación, librerías), que al comienzo del proyecto es relativamente baja y puede tener impacto durante las fases de diseño e implementación.

- **Identificación de características de la aplicación**: en las primeras reuniones previas al comienzo del trabajo se esbozó un análisis de las características que debería presentar la aplicación desde el punto de vista del usuario (casos de uso, funcionalidades, integración con otros sistemas), pero se reconoce y admite que la factibilidad de algunas características no puede ser estimada hasta conocer el comportamiento de la aplicación y el trabajo de investigación, dejando también abierta la posibilidad de incorporar nuevas características en función de los avances en la investigación.

- **Equipo de trabajo**: en el momento de comenzar el trabajo, se asume que el equipo que trabajará en el proyecto está compuesto por un alumno y dos directoras de proyecto con posibilidad de reuniones cara a cara al menos semanales, y en comunicación electrónica constante, pero dejando abierta la posibilidad de que estas circunstancias cambien a lo largo de la duración del trabajo.

En base a estas características, se ha considerado que existen demasiados factores que no pueden ser plasmados formalmente al comienzo del proyecto e interacciones entre los mismos que desaconsejan la utilización de la metodología de desarrollo en cascada, ya que los posibles costes temporales debidos a más que previsibles modificaciones del plan inicial y los costes de documentación serían muy elevados. El tamaño y disposición del equipo de trabajo podría ser considerado un argumento a favor de la utilización de la metodología de desarrollo en cascada, pero se ha considerado que su influencia en la decisión es mínima, optando finalmente utilizar una metodología de desarrollo ágil.

Una vez decidida la utilización de metodología de desarrollo ágil, se han evaluado las principales variaciones con el objetivo de buscar cuál de ellas sería más adecuada para nuestro trabajo. Sin embargo, la mayoría de las metodologías presentan alguno de los siguientes inconvenientes: están basadas en el trabajo en equipo, que en nuestro trabajo es reducido; asumen que el cliente tendrá una presencia en el proceso difícil de traducir a nuestras circunstancias; establecen una periodicidad en las reuniones que no se ajusta a nuestro equipo; y/o establecen límites temporales demasiado estrictos en la duración de los esfuerzos o fases que no se ajusta al carácter de investigación de nuestro trabajo.

Por este motivo, finalmente hemos optado por aplicar **una metodología ágil, siguiendo el espíritu de este tipo de desarrollo, pero sin adscribirnos a una variación en concreto** y utilizando en su lugar las técnicas individuales de cada una de ellas que mejor se adecúan a nuestro trabajo. En particular, se han seguido las siguientes prácticas:

- Desarrollo iterativo con "sprints" asociados a "features" relevantes en cada fase del proceso.
- Flexibilidad en la duración máxima de los "sprints" relacionados con la parte de investigación.
- Utilización de hitos ("milestones") para guiar el ciclo de desarrollo y mantener la visión general.
- División de los requerimientos en subtareas de menor tamaño.
- Corrección de errores antes de continuar con la siguiente iteración.
- Generación de un entregable al terminar cada iteración.
- Utilización de un sistema de control de código facilitar la comparación de revisiones y el trabajo en múltiples características y pruebas.

6.2. Planificación

A la hora de realizar la planificación se ha tratado de modelar la aplicación de los principios de las metodologías ágiles, permitiendo cierta flexibilidad durante los "sprints" y ciclos iterativos, pero organizando y guiando el ciclo

de desarrollo a través del establecimiento de unos hitos con unos límites más estrictos.

Estos hitos representan las fases que se han considerado más significativas e importantes durante el ciclo de desarrollo:

1. **Refinamiento de la labor de investigación y requisitos**: en esta primera etapa, (a) se profundiza en las labores de investigación para concretar el alcance y el conocimiento sobre los algoritmos, con el objetivo de adquirir la mayor cantidad de información posible sobre el dominio que será utilizada durante el resto del desarrollo; (b) se acomete el análisis y recogida de requisitos para la aplicación a desarrollar, identificando aquellos que pueden ser cumplidos independientemente de los resultados de la investigación. Al cumplimiento de este hito, se espera poder tener una idea más clara del proceso de desarrollo y utilizar la información obtenida durante el resto del trabajo.

2. **Creación del prototipo básico**: esta segunda etapa termina con la elaboración del primer prototipo de la aplicación, en el que se implementan las funcionalidades básicas, centrándose en conseguir una versión funcional sin centrarse en los requisitos de usabilidad o funcionalidades extendidas. Para ello se realiza un análisis de la aplicación original, resultando en el diseño e implementación del "esqueleto" de la aplicación, enfocando dicho diseño a la extensibilidad y reescribiendo o reutilizando aquellos componentes originales que se estimen oportuno. También incluye la familiarización del equipo con las tecnologías empleadas.

3. **Creación del segundo prototipo**: en esta etapa se mejora el prototipo básico enfocándose en los requisitos de usabilidad y funcionalidades avanzadas independientes de la correspondencia, tales como la integración con otros sistemas, los enfocados a la facilidad de manejo y aspecto visual, o los relacionados con la facilidad de configuración. Incluye también la creación de las escenas sobre las que se desarrollará el trabajo como parte final de la etapa.

4. **Desarrollo de los algoritmos de correspondencia**: esta etapa está centrada en refinar las labores de investigación relativas a los algoritmos de correspondencia, desarrollando un marco que permita implementarlos en la aplicación. Además del trabajo en la aplicación y en la

investigación, incluye el desarrollo de una herramienta de uso interno para facilitar el análisis (visual y numérico) de los algoritmos de correspondencia en relación con el resultado final, y la realización de pruebas. Concluye con la entrega de un prototipo que permita realizar el proceso de seguimiento aplicando cualquiera de los algoritmos de seguimiento de forma completa.

5. **Consolidación del conocimiento adquirido y documentación**: en esta última etapa se prevé la aplicación del conocimiento adquirido durante el desarrollo previo para realizar ajustes y mejoras a la aplicación y decidir acerca de los requisitos no tratados previamente, implementándolos. También se incluye la realización del manual de usuario y últimos ajustes y retoques para concluir con la entrega de la versión final de la aplicación, junto con la información técnica correspondiente.

En la Figura 6.2 se muestra el diagrama de Gantt de nuestro trabajo, de acuerdo con los hitos anteriormente descritos. Para la planificación temporal se ha optado por establecer una duración total estimada de 8 meses, utilizando un calendario laboral a jornada parcial (4 horas laborables diarias, 5 días laborables por semana, con un total de 20 horas por semana), y realizando unas estimaciones muy conservadoras. En la práctica se trata de una aproximación, dado que durante el desarrollo de este proyecto ha habido cambios significativos en las circunstancias que lo rodearon (en particular, se han modificado los directores hacia el final del proyecto, y la disponibilidad del autor ha sido muy variable por obligaciones laborales, académicas o de diversa índole), que provocaron numerosas y variadas etapas, incluyendo extensas pausas en el trabajo. En el diagrama se ha optado por intentar respetar la planificación original en su versión más sencilla, ya que consideramos que es la que mejor refleja la intención y circunstancias iniciales del trabajo.

6.3. Análisis de coste

Para estimar el coste en términos de tiempo y dinero de este trabajo se ha utilizado como referencia el diagrama de Gantt descrito en la Sección 6.2, haciendo un ejercicio de aproximación para extrapolar las circunstancias comentadas en dicha sección a un entorno profesional.

Se han considerado los siguientes recursos humanos:

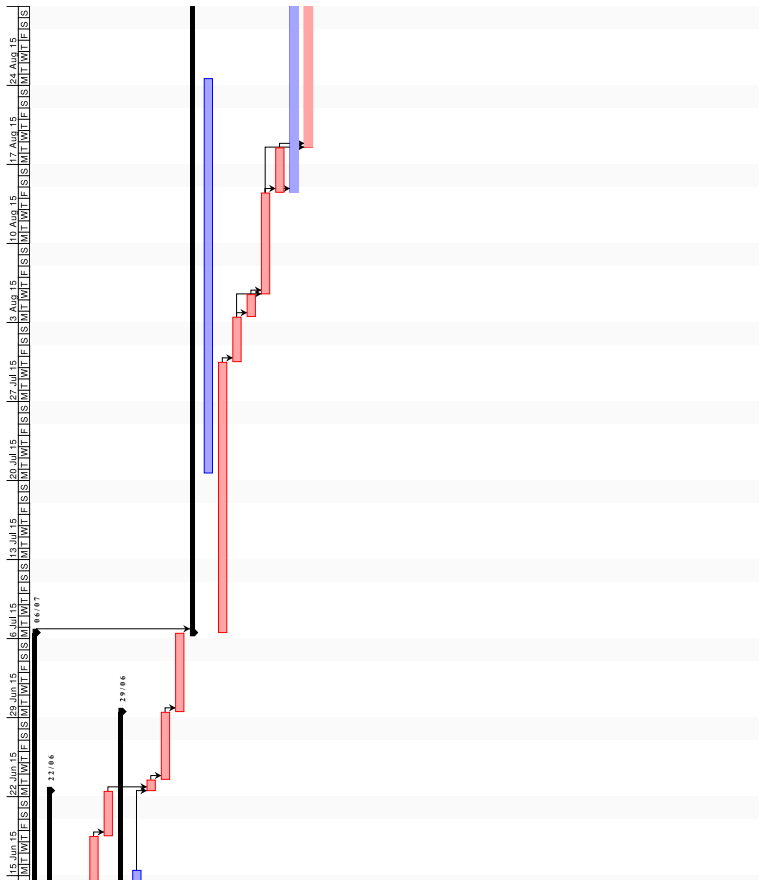
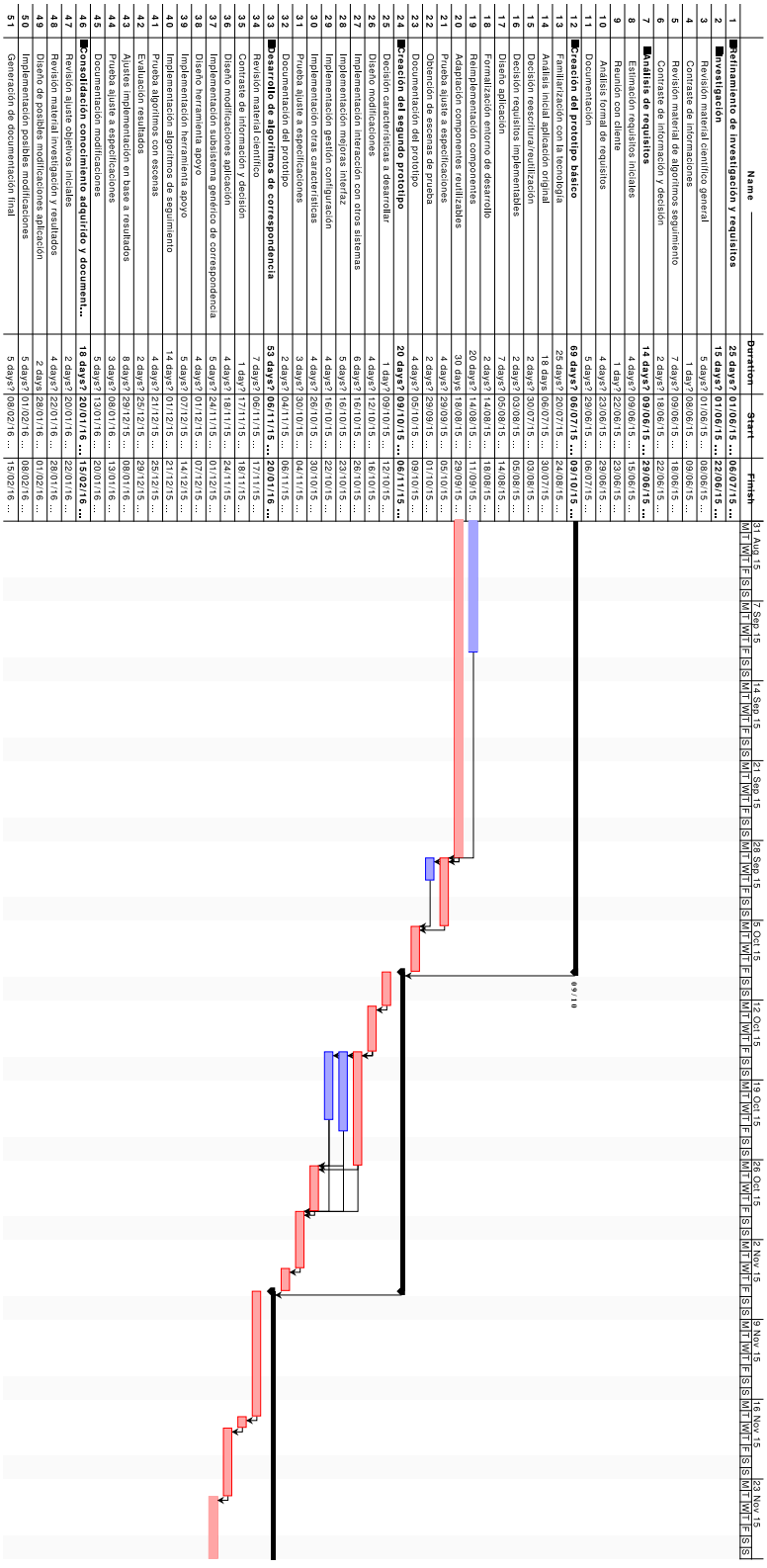
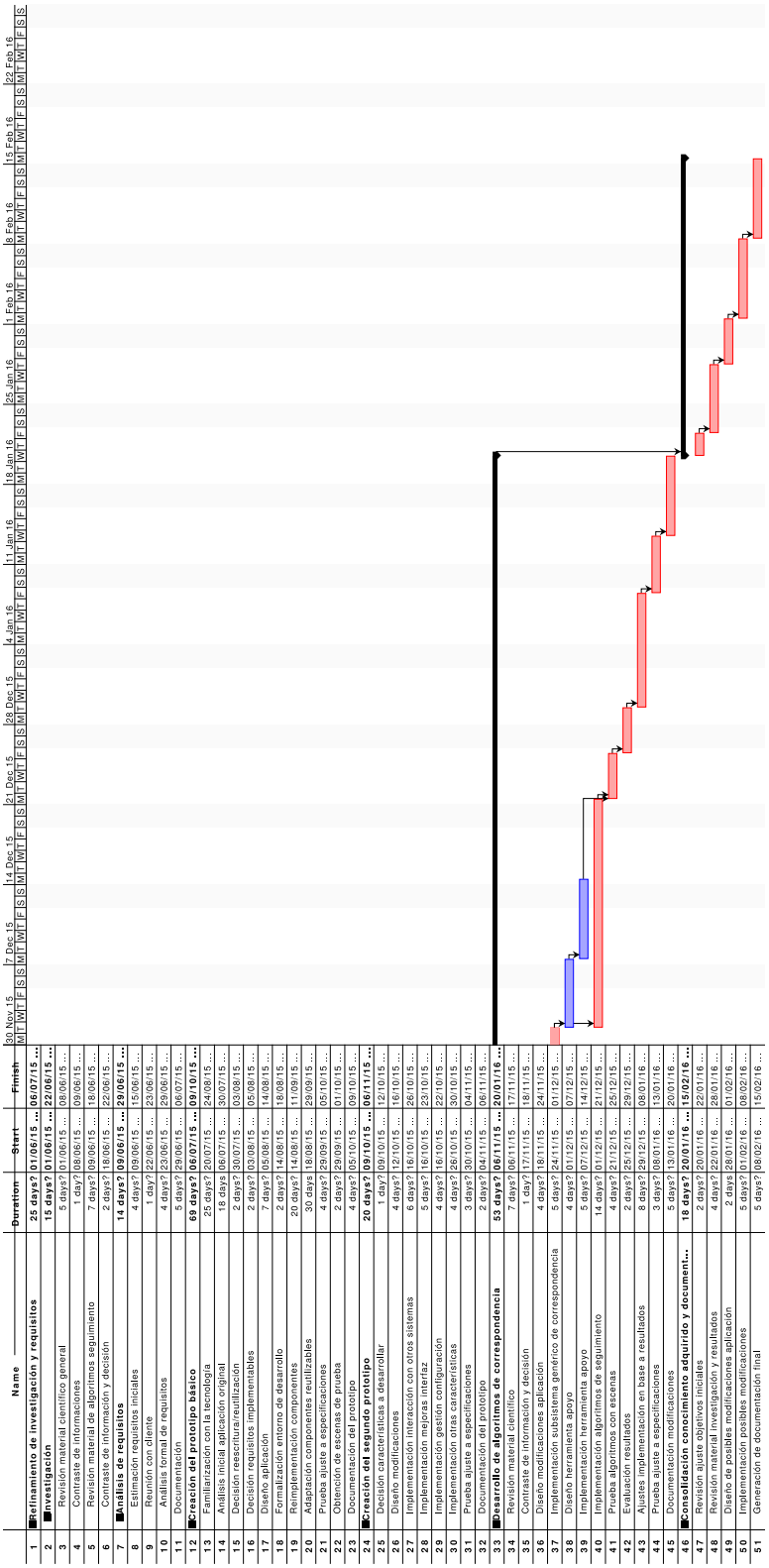


Figura 6.2: Diagrama de Gantt



	Name	Duration	Start	Finish
1	**Refinamiento de investigación y requisitos**	**25 days?**	**01/06/15 ...**	**06/07/15 ...**
2	**Investigación**	**15 days?**	**01/06/15 ...**	**22/06/15 ...**
3	Revisión material científico general	5 days?	01/06/15 ...	08/06/15 ...
4	Contraste de informaciones	1 day?	08/06/15 ...	09/06/15 ...
5	Revisión material de algoritmos seguimiento	7 days?	09/06/15 ...	18/06/15 ...
6	Contraste de información y decisión	2 days?	18/06/15 ...	22/06/15 ...
7	**Análisis de requisitos**	**14 days?**	**09/06/15 ...**	**29/06/15 ...**
8	Estimación requisitos iniciales	4 days?	09/06/15 ...	15/06/15 ...
9	Reunión con cliente	1 day?	22/06/15 ...	23/06/15 ...
10	Análisis formal de requisitos	4 days?	23/06/15 ...	29/06/15 ...
11	Documentación	5 days?	23/06/15 ...	06/07/15 ...
12	**Creación del prototipo básico**	**69 days?**	**06/07/15 ...**	**09/10/15 ...**
13	Familiarización con la tecnología	25 days?	20/07/15 ...	24/08/15 ...
14	Análisis inicial aplicación original	18 days	06/07/15 ...	30/07/15 ...
15	Decisión reescritura/reutilización	2 days?	30/07/15 ...	03/08/15 ...
16	Decisión requisitos implementados	2 days?	03/08/15 ...	05/08/15 ...
17	Diseño aplicación	7 days?	05/08/15 ...	14/08/15 ...
18	Formalización entorno de desarrollo	2 days?	14/08/15 ...	18/08/15 ...
19	Reimplementación componentes	20 days?	14/08/15 ...	11/09/15 ...
20	Adaptación componentes reutilizables	30 days	18/08/15 ...	29/09/15 ...
21	Prueba ajuste a especificaciones	4 days?	29/09/15 ...	05/10/15 ...
22	Obtención de escenas de prueba	2 days?	29/09/15 ...	01/10/15 ...
23	Documentación del prototipo	4 days?	05/10/15 ...	09/10/15 ...
24	**Creación del segundo prototipo**	**20 days?**	**09/10/15 ...**	**06/11/15 ...**
25	Decisión características a desarrollar	1 day?	09/10/15 ...	12/10/15 ...
26	Diseño modificaciones	4 days?	12/10/15 ...	16/10/15 ...
27	Implementación interacción con otros sistemas	6 days?	16/10/15 ...	26/10/15 ...
28	Implementación mejoras interfaz	5 days?	16/10/15 ...	23/10/15 ...
29	Implementación gestión configuración	4 days?	16/10/15 ...	22/10/15 ...
30	Implementación otras características	4 days?	26/10/15 ...	30/10/15 ...
31	Prueba ajuste a especificaciones	3 days?	30/10/15 ...	04/11/15 ...
32	Documentación del prototipo	2 days?	04/11/15 ...	06/11/15 ...
33	**Desarrollo de algoritmos de correspondencia**	**53 days?**	**06/11/15 ...**	**20/01/16 ...**
34	Revisión material científico	7 days?	06/11/15 ...	17/11/15 ...
35	Contraste de información y decisión	1 day	17/11/15 ...	18/11/15 ...
36	Diseño modificaciones aplicación	4 days?	18/11/15 ...	24/11/15 ...
37	Implementación subsistema genérico de correspondencia	5 days?	24/11/15 ...	01/12/15 ...
38	Diseño herramienta apoyo	4 days?	01/12/15 ...	07/12/15 ...
39	Implementación herramienta apoyo	5 days?	07/12/15 ...	14/12/15 ...
40	Implementación algoritmos de seguimiento	14 days?	01/12/15 ...	21/12/15 ...
41	Prueba algoritmos con escenas	4 days?	21/12/15 ...	25/12/15 ...
42	Evaluación resultados	2 days?	25/12/15 ...	29/12/15 ...
43	Ajustes implementación en base a resultados	8 days?	29/12/15 ...	08/01/16 ...
44	Prueba ajuste a especificaciones	3 days?	08/01/16 ...	13/01/16 ...
45	Documentación modificaciones	5 days?	13/01/16 ...	20/01/16 ...
46	**Consolidación conocimiento adquirido y document...**	**18 days?**	**20/01/16 ...**	**15/02/16 ...**
47	Revisión ajuste objetivos iniciales	2 days?	20/01/16 ...	22/01/16 ...
48	Revisión material investigación y resultados	4 days?	22/01/16 ...	28/01/16 ...
49	Diseño de posibles modificaciones aplicación	2 days	28/01/16 ...	01/02/16 ...
50	Implementación posibles modificaciones	5 days?	01/02/16 ...	08/02/16 ...
51	Generación de documentación final	5 days?	08/02/16 ...	15/02/16 ...



	Name	Duration	Start	Finish
1	**Refinamiento de investigación y requisitos**	**25 days?**	**01/06/15 ...**	**06/07/15 ...**
2	**Investigación**	**15 days?**	**01/06/15 ...**	**22/06/15 ...**
3	Revisión material científico general	5 days?	01/06/15 ...	08/06/15 ...
4	Contraste de informaciones	1 day?	08/06/15 ...	09/06/15 ...
5	Revisión material de algoritmos seguimiento	7 days?	09/06/15 ...	18/06/15 ...
6	Contraste de información y decisión	2 days?	18/06/15 ...	22/06/15 ...
7	**Análisis de requisitos**	**14 days?**	**09/06/15 ...**	**29/06/15 ...**
8	Estimación requisitos iniciales	4 days?	09/06/15 ...	15/06/15 ...
9	Reunión con cliente	1 day?	22/06/15 ...	23/06/15 ...
10	Análisis formal de requisitos	4 days?	23/06/15 ...	29/06/15 ...
11	Documentación	5 days?	29/06/15 ...	06/07/15 ...
12	**Creación del prototipo básico**	**69 days?**	**06/07/15 ...**	**09/10/15 ...**
13	Familiarización con la tecnología	25 days?	20/07/15 ...	24/08/15 ...
14	Análisis inicial aplicación original	18 days	06/07/15 ...	30/07/15 ...
15	Decisión reescritura/reutilización	2 days?	30/07/15 ...	03/08/15 ...
16	Decisión requisitos implementables	2 days?	03/08/15 ...	05/08/15 ...
17	Diseño aplicación	7 days?	05/08/15 ...	14/08/15 ...
18	Formalización entorno de desarrollo	2 days?	14/08/15 ...	18/08/15 ...
19	Reimplementación componentes	20 days?	14/08/15 ...	11/09/15 ...
20	Adaptación componentes reutilizables	30 days	18/08/15 ...	29/09/15 ...
21	Prueba ajuste a especificaciones	4 days?	29/09/15 ...	05/10/15 ...
22	Obtención de escenas de prueba	2 days?	29/09/15 ...	01/10/15 ...
23	Documentación del prototipo	4 days?	05/10/15 ...	09/10/15 ...
24	**Creación del segundo prototipo**	**20 days?**	**09/10/15 ...**	**05/11/15 ...**
25	Decisión características a desarrollar	1 day?	09/10/15 ...	12/10/15 ...
26	Diseño modificaciones	4 days?	12/10/15 ...	16/10/15 ...
27	Implementación interacción con otros sistemas	6 days?	16/10/15 ...	26/10/15 ...
28	Implementación mejoras interfaz	5 days?	16/10/15 ...	23/10/15 ...
29	Implementación gestión configuración	4 days?	16/10/15 ...	22/10/15 ...
30	Implementación otras características	4 days?	26/10/15 ...	30/10/15 ...
31	Prueba ajuste a especificaciones	3 days?	30/10/15 ...	04/11/15 ...
32	Documentación del prototipo	2 days?	04/11/15 ...	06/11/15 ...
33	**Desarrollo de algoritmos de correspondencia**	**53 days?**	**06/11/15 ...**	**20/01/16 ...**
34	Revisión material científico	7 days?	06/11/15 ...	17/11/15 ...
35	Contraste de información y decisión	1 day?	17/11/15 ...	18/11/15 ...
36	Diseño modificaciones aplicación	4 days?	18/11/15 ...	24/11/15 ...
37	Implementación subsistema genérico de correspondencia	5 days?	24/11/15 ...	01/12/15 ...
38	Diseño herramienta apoyo	4 days?	01/12/15 ...	07/12/15 ...
39	Implementación herramienta apoyo	5 days?	07/12/15 ...	14/12/15 ...
40	Implementación algoritmos de seguimiento	14 days?	01/12/15 ...	21/12/15 ...
41	Prueba algoritmos con escenas	4 days?	21/12/15 ...	25/12/15 ...
42	Evaluación resultados	2 days?	25/12/15 ...	29/12/15 ...
43	Ajustes implementación en base a resultados	8 days?	29/12/15 ...	08/01/16 ...
44	Prueba ajuste a especificaciones	3 days?	08/01/16 ...	13/01/16 ...
45	Documentación modificaciones	5 days?	13/01/16 ...	20/01/16 ...
46	**Consolidación conocimiento adquirido y document...**	**18 days?**	**20/01/16 ...**	**15/02/16 ...**
47	Revisión ajuste objetivos iniciales	2 days?	20/01/16 ...	22/01/16 ...
48	Revisión material investigación y resultados	4 days?	22/01/16 ...	28/01/16 ...
49	Diseño de posibles modificaciones aplicación	2 days	28/01/16 ...	01/02/16 ...
50	Implementación posibles modificaciones	5 days?	01/02/16 ...	08/02/16 ...
51	Generación de documentación final	5 days?	08/02/16 ...	15/02/16 ...

Fase	Ing. de software	Gerente	Diseñador
Refinamiento de investigación y requisitos	29	6	1
Creación del prototipo básico	69	7	1
Creación del segundo prototipo	20	2	3
Desarrollo algoritmos de correspondencia	53	4	1
Consolidación conocimiento adquirido	18	6	0
Total	**189**	**25**	**6**

Cuadro 6.1: Estimación de costes de tiempo (en días)

- **1 ingeniero de software** que será el responsable de llevar a cabo el grueso del trabajo de investigación, análisis, diseño e implementación, actuando como ingeniero principal.
- **1 gerente de proyecto** con conocimientos más especializados y mayor experiencia que será el responsable de coordinar y validar en general el desarrollo del proyecto.
- **1 diseñador gráfico** que actuará como consultor puntual en cuanto a las decisiones relacionadas con la vertiente artística del proyecto.

Y los siguientes recursos físicos:

- **1 equipo informático** sin características especiales a nivel de hardware o software, en el que se instalarán las herramientas de código libre utilizadas durante el proceso y se realizarán las labores de desarrollo e investigación.
- **1 licencia de la librería *Qt***, necesaria salvo que el producto final se distribuya bajo una licencia LGPL/GPL.

Bajo estas premisas, el Cuadro 6.1 muestra el número de días empleados por cada miembro del equipo de acuerdo con la distribución de tareas ilustrada en la Figura 6.2. El tiempo total del ingeniero de software se corresponde con el camino crítico, debido a que está presente en todas las tareas, distribuyendo su tiempo entre aquellas concurrentes y utilizando los principios de la metodología ágil dentro de ellas. Para estimar el tiempo del gerente de proyecto se han identificado aquellas tareas en las que prima la coordinación (revisiones, cumplimiento de requisitos, reuniones, etc), y el tiempo del diseñador gráfico se ha evaluado en base a las tareas en las que es necesario tomar decisiones artísticas y evaluar los prototipos.

Recurso	#	Horas	Coste/h.	Horas/día	Total
Ingeniero de software	1	135	20€	4	**10800€**
Gerente de proyecto	1	17.85	30€	4	**2142€**
Diseñador gráfico	1	4.28	15€	4	**257€**
					13199€

Cuadro 6.2: Estimación de costes económicos netos de personal

A partir del número de días empleados por cada miembro del equipo se puede realizar la estimación del número de horas, que a su vez se utilizará como base para calcular el coste económico de los recursos humanos. Una aproximación sencilla es utilizar el número de horas de cada jornada de trabajo (4), ajustando por el número de días laborales por semana, y directamente multiplicar los valores de la tabla anterior, incluso aunque algunas de las tareas no involucren la totalidad del tiempo de cada uno de los participantes. Esta estimación al alza refleja, especialmente en el caso del gerente de proyecto y del diseñador gráfico, el tiempo invertido en interactuar con el resto del equipo y su participación puntual en las tareas de acuerdo con el principio de interacción de la metodología ágil empleada.

La aproximación del coste económico presenta gran variabilidad debido a la cantidad de factores involucrados (salarios, composición del equipo, márgenes de beneficio, tipo de empresa, tipo de contrato, o tarificación, entre otros). Se ha optado por aproximar el coste económico que supondrían, para una empresa típica de desarrollo de software a medida, las horas de sus empleados y el gasto en recursos en el caso en que el producto sea entregado con una licencia comercial y privativa; sin entrar en el coste para el cliente final del producto. La Tabla 6.2 muestra una simulación basada en cálculos del coste por hora de cada miembro del equipo conservadores y netos.

En cuanto a los costes económicos debidos a los recursos físicos, la consideración de que el entregable final tenga una licencia privativa hace necesaria la adquisición de una licencia comercial de la librería *Qt*. El coste de la licencia varía de acuerdo con las condiciones de la misma (modelo de subscripción, uso del producto, número de desarrolladores, plataformas soportadas, etc), y en nuestra estimación se ha optado por elegir una licencia comercial mensual (300€ por mes de desarrollo activo) por presentar el mejor precio dadas las condiciones (sólo un desarrollador utilizando la librería durante 6 meses, ya que no es necesaria durante la primera fase). El coste mensual del equipo

Recurso	Cantidad	Meses	Coste mensual	Total
Equipo informático	1	8	55.5€	**444€**
Licencia *Qt*	1	6	300€	**1800€**
				2244€

Cuadro 6.3: Estimación de costes económicos de recursos físicos

informático se ha calculado considerando la amortización del mismo a 5 años fiscales. El Cuadro 6.2 resume los resultados.

Así, el coste total del proyecto asciende a 15443€ para la empresa. Este valor proporciona una aproximación que puede ser ajustada a las condiciones particulares del escenario en el que se aplique el proyecto (por ejemplo, los costes de licencia pueden ser asumidos por el cliente, y los costes por hora pueden ser ajustados con más precisión estimándolos para cada una de las fases) y utilizada como base para determinar el coste final para el cliente.

Capítulo 7

Conclusiones

En nuestro trabajo hemos revisado y mejorado un método de seguimiento para generación de las curvas intermedias entre dos fotogramas clave, utilizando diferentes algoritmos de correspondencia como el principal método para incrementar la calidad visual de los resultados, e implementado una aplicación que permite al usuario realizar todo el proceso: desde la importación de las imágenes y las curvas al lanzamiento del proceso de seguimiento de forma iterativa, pasando por la modificación de los puntos de control de las curvas y exportación de imágenes, de una forma intuitiva y con una granularidad muy fina a la hora de controlar el resultado final.

Combinando el desarrollo teórico de los algoritmos y la implementación de la aplicación se han conseguido solventar varias limitaciones del método de seguimiento original, mejorando también la flexibilidad en general y el tiempo empleado por el usuario en las distintas etapas del proceso de rotoscopia: en concreto, se ha eliminado la necesidad de modificar manualmente las curvas finales a partir de las iniciales, permitiendo el uso de curvas importadas con distinto número de segmentos; se ha mejorado notablemente la facilidad de modificar los puntos de control y las restricciones adicionales a gusto del usuario; y como consecuencia del incremento en la calidad de los resultados se ha permitido al usuario incrementar el intervalo entre fotogramas clave manteniendo resultados aceptables, haciendo que el número de modificaciones y ajustes manuales de las curvas por parte del usuario sea en conjunto significativamente menor.

En relación a los resultados descritos en la Sección 4.4: analizando la influencia del método de correspondencia en el resultado final, representada en las Figuras 4.6-4.9, se observa que el método basado en la similud de curvas

(descrito en la Sección 3.4) es el que produce mejores resultados, presentando de forma consistente mejores valores en las métricas con respecto al método basado en el vector de desplazamiento y al método trivial, confirmando la importancia de una correcta asignación de la correspondencia en el resultado final del proceso de seguimiento. El método basado en el vector de desplazamiento (descrito en la Sección 3.6) también mejora en general los resultados del método trivial, aunque en menor medida y con menos consistencia entre las familias de curvas, produciendo puntualmente peores resultados que el método trivial en tramos de alguna de las familias.

Las diferencias entre los métodos de seguimiento entre cada una de las familias de la secuencia *Table-tennis* revelan una cierta variabilidad. Se observa que en el caso de las familias con una diferencia entre la curva inicial y final menor (como por ejemplo las familias *Cintura*, *Pelo* o *Pierna_dcha*), el impacto del método de seguimiento es pequeño, produciendo resultados visuales muy similares. También se observa que en algunas de las familias más complejas ninguno de los métodos de seguimiento consigue resolver satisfactoriamente del todo algunas regiones de las curvas.

En particular, la familia *Brazo_dcho_interior* presenta cambios bastante bruscos tanto en la longitud de los tramos que la componen como en las posiciones y sentidos de sus puntos de control. Los métodos de correspondencia no son capaces de influir suficientemente en la resolución de esas características, en particular a la hora de determinar la longitud del tramo cuya longitud varía más a lo largo de la secuencia (en la región del antebrazo), dando como resultado un empeoramiento de las métricas a partir del fotograma 16, tal y como se muestra en la Figura 7.1. Por su parte, la familia *Pierna_izqda* también presenta retos que los métodos de correspondencia no son capaces de resolver. En este caso la familia fue incluida para analizar el comportamiento del seguimiento cuando la imagen no tiene la suficiente continuidad temporal: a lo largo de la secuencia, la mano interfiere con el trazo de la pierna izquierda, como se muestra en la Figura 7.2. Esta oclusión provoca problemas a la hora de realizar el seguimiento, ya que a su vez provoca cambios bruscos en el tamaño de los segmentos y su posición, y un cierto grado de conflicto entre la información generada por los términos de forma y los términos de imagen. Estas incidencias también afectan, en menor medida, a otras curvas que presentan características similares.

En la secuencia *Amira* los resultados en general son más homogéneos, debido a que las dos curvas de la secuencia son relativamente sencillas y si-

(a) 16 (b) 17 (c) 18 (d) 19 (e) 20 (f) 21 (g) 22 (h) 23

Figura 7.1: Comparación entre las curvas ideales y las generadas para la familia *Brazo_ dcho_ interior* en los fotogramas 16-23

(a) 0 (b) 2 (c) 4 (d) 6 (e) 8 (f) 10

Figura 7.2: Detalle la familia *Pierna_ Dcha* en los fotogramas 0-10

milares, y las imágenes presentan unas características muy favorables para los términos de imagen de la función de energía, con un contraste muy definido.

La interpretación de estas observaciones es, por un lado, que la implementación cumple con uno de los objetivos de nuestro trabajo, que es la mejora de los resultados finales visuales gracias a la aplicación de métodos de correspondencia independientemente del número de segmentos de las curvas inicial y final de las familias. Por otra parte, parece claro que el método de seguimiento sólo se ve influido hasta cierto punto por la correspondencia, y que los términos de imagen de la función de energía (Ecuación 2.12) diluyen el efecto debido a la correspondencia si la imagen presenta características muy marcadas. También se observa que el seguimiento no es capaz de resolver algunas situaciones independientemente de la bondad de la correspondencia, siendo necesario intervención por parte del usuario si las curvas presentan ciertas características.

En relación a los tiempos de ejecución, descritos en el Cuadro 4.4 y las Figuras 4.10 y 4.11, se observa que para todas las familias el paso en el que se invierte más tiempo es el cálculo de las pirámides de Gauss, representando de media un 77.14 % del tiempo total; y está directamente relacionado por las dimensiones de las imágenes de la secuencia y el número de fotogramas de la misma. La complejidad de los cálculos de las pirámides es constante para todos los métodos de correspondencia ($\mathcal{O}(w \times h \times n)$, siendo w y h la anchura y altura de las imágenes, y n el número de fotogramas), y su fuerte

influencia en el tiempo total de ejecución es debida a la naturaleza iterativa de los cálculos sobre los píxeles de las imágenes.

El impacto del tiempo empleado en el cálculo de la correspondencia sobre el tiempo total de ejecución es muy bajo, representando de media un 0.19 % del tiempo total. El método de correspondencia guiado por imagen es el único método en el que tiempo de cálculo de la correspondencia supera los 5 ms, siendo la familia *Brazo_izqdo_exterior* la que presenta el mayor impacto en el tiempo total (1.21 %). La diferencia con respecto al resto de métodos de correspondencia que explica que se invierta de forma consistente un tiempo una orden de magnitud mayor es debida a que en el cálculo de la función de coste (Ecuación 3.14) se utiliza un término que realiza operaciones sobre las imágenes.

El tiempo empleado en el cálculo del seguimiento es el que presenta mayor variabilidad, representando desde un 7.02 % (familia *Arco_piernas*, método basado en la similitud de curvas) a un 48.94 % (familia *Cara*, método trivial), debido a la naturaleza de proceso de minimización de energía y a la influencia de las características de las curvas (número de puntos de control y longitud de la curva) en el tamaño de la matriz de resultados.

Apéndice A

Trabajo futuro

En este capítulo describimos algunas líneas de trabajo e ideas que podrían mejorar los resultados obtenidos por el proceso de seguimiento. En particular, describimos un método de correspondencia perteneciente a la categoría *feature matching* así como sus parámetros, y posteriormente resaltamos algunas características de nuestro trabajo que pueden ser exploradas para mejorar el rendimiento.

A.1. Método basado en propiedades geométricas de un conjunto característico de puntos

Al contrario que los demás métodos estudiados en el Capítulo 3, que son todos ellos algoritmos de correspondencia *dense matching*, este método [55] pertenece a los métodos de correspondencia *feature matching*. En lugar de establecer la correspondencia teniendo en cuenta todas las muestras de las curvas, sólo se consideran una serie de puntos característicos de la misma. Los puntos característicos son seleccionados basándose en criterios geométricos [21].

En el primer paso del algoritmo se selecciona un conjunto de puntos potencialmente característicos. Así, partiendo de una curva C, muestreada de forma densa y uniforme con m muestras C^i con $0 \leq i \leq m$, al aplicar el algoritmo de identificación de puntos característicos se obtiene un subconjunto de M puntos característicos $\mathbb{C}^I$, con $0 \leq I \leq M$ y $M < m$. La función $\wp_K$

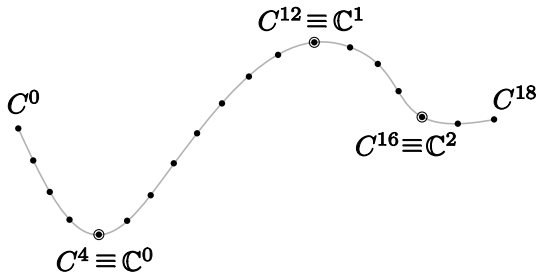


Figura A.1: Puntos característicos de una curva C, en la que $\wp(0) = 4$, $\wp(1) = 12$ y $\wp(2) = 16$

relaciona el índice de un punto característico $\mathbb{C}^I$ de una curva con el índice que tendría en la secuencia de muestras de esa misma curva:

$$\wp(I) = j$$

En la Figura A.1 se muestra ilustran los puntos característicos obtenidos para una curva de ejemplo con $m = 19$ y $M = 3$.

Antes de enunciar la solución propuesta para el problema de la correspondencia, es necesario definir una serie de conceptos. En los tres primeros apartados de esta sección se explican estos conceptos, mientras que en el último apartado se expone la solución al problema de la correspondencia.

A.1.1. Análisis de la matriz de covarianza

Definimos la región de apoyo $\mathcal{R}$ para un punto característico $\mathbb{C}^I$ como:

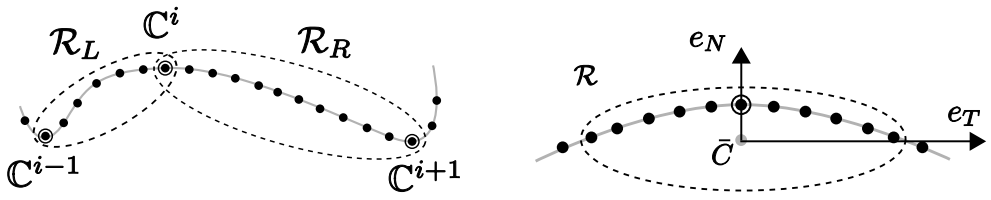
$$\mathcal{R}(\mathbb{C}^I) = \left\{ C^j \;\middle|\; |j - \wp(I)| \leq h \right\}$$

donde $h \in \mathbb{N}$ es un parámetro del algoritmo. Esta región contiene las h muestras siguientes y las h muestras anteriores al punto característico $\mathbb{C}^I$. Además de definir la región $\mathcal{R}$, es necesario definir las regiones derecha $\mathcal{R}_R$ e izquierda $\mathcal{R}_L$, que contienen las muestras situadas entre los puntos característicos $[\mathbb{C}^{I-1} \ldots \mathbb{C}^I]$ y $[\mathbb{C}^I \ldots \mathbb{C}^{I+1}]$ respectivamente (ver Figura A.2a):

$$\mathcal{R}_L(\mathbb{C}^I) = \left\{ C^j \;\middle|\; \wp(I-1) \leq j \leq \wp(I) \right\}$$

$$\mathcal{R}_R(\mathbb{C}^I) = \left\{ C^j \;\middle|\; \wp(I) \leq j \leq \wp(I+1) \right\}$$

Utilizaremos la notación $|\mathcal{R}|$ para referirnos al número de muestras que contiene la región $\mathcal{R}$. De las definiciones anteriores, se deduce que para un



(a) (b)

Figura A.2: (a) Definición de $\mathcal{R}_L$ y $\mathcal{R}_R$ para un punto $\mathbb{C}^i$. (b) Autovectores de la matriz de covarianza.

punto característico $\mathbb{C}^I$, el número de muestras que contienen sus regiones es $|\mathcal{R}(\mathbb{C}^I)| = 2h$, $|\mathcal{R}_L(\mathbb{C}^I)| = \wp(I) - \wp(I-1)$, y $|\mathcal{R}_L(\mathbb{C}^I)| = \wp(I+1) - \wp(I)$.

Si consideramos $\bar{C} = (\bar{x}, \bar{y})$ el punto situado en el centro de una región de apoyo $\mathcal{R}$, entonces la matriz de covarianza de la región $Cov(\mathcal{R})$ se define como:

$$Cov(\mathcal{R}) = \frac{1}{2h+1} \sum_{C^j \in \mathcal{R}} (C^j - \bar{C}^i)(C^j - \bar{C}^i)^T \tag{A.1}$$

Los autovectores $\{e_0, e_1\}$ de la matriz de covarianza $Cov(\mathcal{R}(\mathbb{C}^I))$ junto con los autovalores $\{\lambda_0, \lambda_1\}$ definen una elipse, en la que el autovector e_0 es un vector tangente y el vector e_1 es un vector normal al punto característico $\mathbb{C}^I$. Por este motivo, emplearemos $\{\lambda_T, \lambda_N\}$ para referirnos a $\{\lambda_0, \lambda_1\}$ respectivamente, y $\{e_T, e_N\}$ para referirnos a $\{e_0, e_1\}$ (ver Figura A.2b).

A.1.2. Propiedades geométricas

Las propiedades geométricas que van a guiar la correspondencia entre los puntos característicos son las siguientes:

(1) σ: Variación de la característica que mide la desviación del entorno de $\mathbb{C}^I$ con respecto a la dirección tangente a $\mathbb{C}^I$:

$$\sigma(\mathbb{C}^I) = \xi \frac{\lambda_N}{\lambda_N + \lambda_T} \tag{A.2}$$

donde $\xi = 1$ si el punto característico $\mathbb{C}^I$ es convexo, y $\xi = -1$ si es cóncavo; y λ_N y λ_T son los autovectores de la matriz de covarianza. El valor está en el intervalo $[-1, 1]$, siendo cercano a 0 cuando la región de $\mathbb{C}^I$ se aproxima a una línea recta, y tendiendo a -1 ó 1 cuando la región presenta una alta curvatura. También definimos esta propiedad con respecto a una región $\mathcal{R}$:

$$\sigma(\mathcal{R}) = \frac{\lambda_N}{\lambda_N + \lambda_T} \tag{A.3}$$

(2) τ: Variación lateral de la característica, este término pretende estimar la medida en que varía la curvatura de los segmentos situados a la izquierda y la derecha del punto característico $\mathbb{C}^I$:

$$\tau(\mathbb{C}^I) = \frac{\sigma(\mathcal{R}_L(\mathbb{C}^I)) + \sigma(\mathcal{R}_R(\mathbb{C}^I))}{2} \tag{A.4}$$

(3) ρ: Tamaño de la característica, que se utiliza para estimar la importancia de la característica en la curva, utilizando como estimador el tamaño de la misma:

$$\rho(\mathbb{C}^I) = \frac{1}{2}\frac{|\mathcal{R}_L(\mathbb{C}^I)| + |\mathcal{R}_R(\mathbb{C}^I)|}{m} \tag{A.5}$$

Para el cálculo del tamaño de la característica se utilizan los tamaños relativos de sus regiones izquierda y derecha con respecto al tamaño total de la curva, m.

Las tres propiedades geométricas que evalúan un punto característico $\mathbb{C}^I$ son invariantes a escala, rotación, y densidad de muestreo. Se propone utilizar las propiedades geométricas para medir la similitud entre dos puntos característicos. Las características de las dos curvas que sean similares entre sí deberían tener similares variaciones de la característica, variaciones laterales, y tamaños. Siguiendo ese razonamiento, se define el coste de similitud entre dos puntos característicos $\mathbb{C}^I_A$ y $\mathbb{C}^J_B$ como:

$$S(\mathbb{C}^I_A, \mathbb{C}^J_B) = \Psi(\mathbb{C}^I_A, \mathbb{C}^J_B) \sum_{q=\sigma,\tau,\rho} \omega_q \Delta_q(\mathbb{C}^I_A, \mathbb{C}^J_B) \tag{A.6}$$

donde ω_q son pesos (cuya suma es 1), cada uno de los términos Δ_q con $q = \sigma, \tau\, \rho$ es el coste de la propiedad q correspondiente, y el coeficiente $\Psi(\mathbb{C}^I_A, \mathbb{C}^J_B)$ es un peso relativo a la importancia de esa característica en las curvas. Los términos $\Delta_q(\mathbb{C}^I_A, \mathbb{C}^J_B)$ relacionan los dos puntos característicos utilizando la propiedad q correspondiente (en [55] se puede encontrar un análisis más detallado de los mismos).

A.1.3. Penalización de descartes de puntos característicos

También se considera la posibilidad de descartar alguno de los puntos característicos, si su entorno es lo suficientemente pequeño y con poca curvatura. Definimos el coste de descarte de un punto característico $\mathbb{C}^I$ como:

$$D(\mathbb{C}^I) = \rho(\mathbb{C}^I) \sum_{q=\sigma,\tau,\rho} \omega_q \left| q(\mathbb{C}^I) \right| \tag{A.7}$$

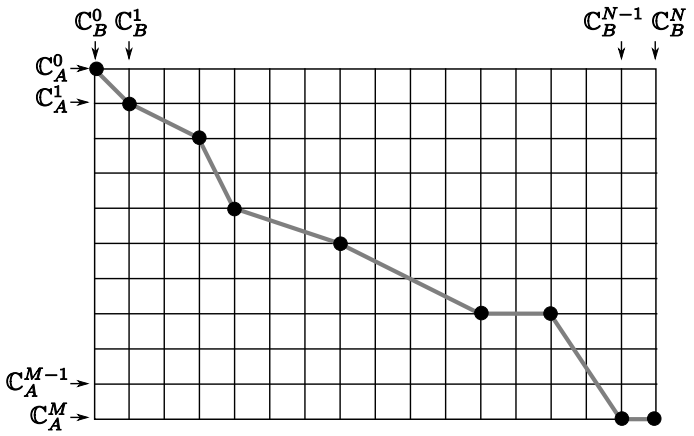


Figura A.3: Ejemplo de un camino completo en el grafo de programación dinámica

donde $q(\mathbb{C}^I)$, con $q = \sigma, \tau, \rho$, es cada una de las propiedades geométricas, y los pesos ω_q son los mismos que en la Ecuación A.6. El coeficiente $\rho(\mathbb{C}^I)$ se utiliza con la misma intención con la que se utiliza el coeficiente Ψ en la Ecuación A.6, que es evaluar la importancia relativa de la característica en la curva en función de su tamaño.

A.1.4. Algoritmo de correspondencia

La correspondencia entre dos curvas propuesta en el método basado en propiedades geométricas de un conjunto característico de puntos busca encontrar una reparametrización $\mathbb{V} : \mathbb{C}_A^I \rightarrow \mathbb{C}_B^J$ que minimice la función:

$$\mathbb{F} = min_{\mathbb{V}(I)} \sum_{I=0}^{M-1} S(\mathbb{C}_A^I, \mathbb{C}_B^{\mathbb{V}(I)}) \tag{A.8}$$

Para resolver la ecuación anterior eficientemente se aplican técnicas de programación dinámica. Si se representa cada correspondencia entre puntos característicos $(\mathbb{C}_A^I, \mathbb{C}_B^J)$ como el nodo de un grafo $M \times N$ en el que los puntos característicos $\mathbb{C}_B^J$ se representan en el eje x y los puntos característicos $\mathbb{C}_A^I$ se representan en el eje y, una correspondencia completa será el camino que recorre el grafo desde $(0, 0)$ hasta (M, N) (ver Figura A.3). A diferencia de los algoritmos descritos en las secciones anteriores, no existe la restricción de que los puntos característicos que se hacen corresponder tengan que ser puntos consecutivos. En la Figura A.4 se puede apreciar la correspondencia

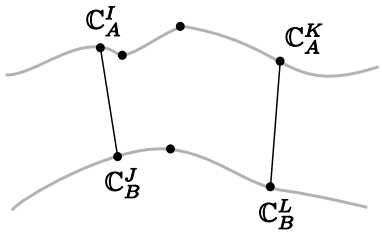


Figura A.4: Correspondencia entre puntos característicos.

entre los puntos $(\mathbb{C}_A^I, \mathbb{C}_B^J)$ y $(\mathbb{C}_A^K, \mathbb{C}_B^L)$ de dos curvas C_A y C_B. Los puntos intermedios que no están unidos por una línea continua representan los puntos característicos descartados por el algoritmo.

El algoritmo de programación dinámica busca encontrar el camino en el que el coste de hacer corresponder los puntos característicos que lo componen sea mínimo. Sea $\mathbb{C}^{I|J} = \left\{\mathbb{C}^I, \mathbb{C}^{I+1}, \ldots, \mathbb{C}^{J-1}, \mathbb{C}^J\right\}$ la secuencia de puntos característicos comprendidos entre el punto $\mathbb{C}^I$ y el punto $\mathbb{C}^J$, donde $I \leq J$. El coste de correspondencia entre dos secuencias de puntos característicos $\mathbb{C}_A^{I|J}$ y $\mathbb{C}_B^{K|L}$ se define como:

$$f(\mathbb{C}_A^{I|J}, \mathbb{C}_B^{K|L}) = \sum_{\alpha=I+1}^{J-1} D(\mathbb{C}_A^{\alpha}) + \sum_{\beta=K+1}^{L-1} D(\mathbb{C}_B^{\beta}) + \lambda \cdot S(\mathbb{C}_A^J, \mathbb{C}_B^L) \qquad \text{(A.9)}$$

donde los dos primeros términos evalúan el coste de descartar los puntos característicos que pertenecen al intervalo, y la constante λ permite ajustar la importancia relativa que tienen los descartes frente a la función de similitud. Un valor bajo de λ hará que tiendan a producirse pocos descartes, y un valor elevado los favorecerá.

El coste de hacer corresponder los primeros I puntos característicos de la curva C_A con los primeros J puntos característicos de la curva C_B se representa como $\mathbb{F}_{I,J}$:

$$\mathbb{F}_{I,J} = min_{K,L}\left[\mathbb{F}_{I-K,J-L} + f(\mathbb{C}_A^{I-K|I}, \mathbb{C}_B^{J-L|J})\right] \qquad \text{(A.10)}$$

donde el mínimo es sobre todos los pares posibles (K, L), con $K, L \geq 0$. Una vez calculado cada $\mathbb{F}_{I,J}$, es necesario almacenar los índices $I-K$ y $J-L$ del nodo en el que se ha hallado el coste mínimo, puesto que indica cuál es el nodo anterior a (I, J). Una vez calculado $\mathbb{F}_{M,N}$, estos índices serán utilizados para reconstruir el camino óptimo.

En el cálculo de cada uno de los términos de la función de coste se utilizan las propiedades geométricas de los puntos característicos de cada una de las curvas por separado. Como en el cálculo de éstas se utilizan sólo el número de puntos y los autovalores de las regiones de apoyo, que no varían al ser aplicada una rotación o una traslación a la curva, el método es invariante a rotación y traslación. La complejidad del algoritmo, en el caso en el que no se permita ningún salto en el camino, sería de $\mathcal{O}(MN)$, al igual que el algoritmo comentado en la Sección 3.3. Si se restringe el número máximo p de puntos característicos que pueden ser descartados, la complejidad aumenta a $\mathcal{O}(p^2MN)$.

A.2. División de curvas en igual número de segmentos

Tanto el algoritmo que ha sido tomado como punto de partida para desarrollar este trabajo [6] como la aplicación original basada en dicho artículo parten de la asunción de que la roto-curva del fotograma final C_F es una copia directa de la roto-curva del fotograma inicial C_0, modificada manualmente por el usuario tras realizar la copia. Por lo tanto ambas curvas tendrán el mismo número de puntos de control, y en consecuencia el mismo número de segmentos.

Si bien nuestro trabajo no presenta esa limitación y permite que las curvas inicial y final tengan un número diferente de segmentos, en las pruebas experimentales hemos podido comprobar que algunos de los métodos de correspondencia se comportan mejor en aquellos casos en los que las curvas inicial y final tienen el mismo número de segmentos. Por este motivo, la aplicación de un paso de preprocesado que iguale el número de segmentos de las curvas antes de lanzar el seguimiento podría afectar positivamente a los resultados. Este paso podría realizarse de forma transparente al usuario, o como un paso explícito durante la importación inicial de las curvas a un fotograma.

La división de una curva Bèzier en dos curvas es un problema resuelto en la literatura [39, 22], siendo a priori el método más adecuado en nuestro caso el algoritmo de de Casteljau [27], por ser el método más estable numéricamente a pesar de no ser el más veloz. Con este algoritmo es posible subdividir una curva Bèzier por cualquier valor arbitrario de su parámetro t. La idea principal del algoritmo de de Casteljau ha sido también empleada en nuestro trabajo para la implementación de la función que calcula la longitud de un segmento de la roto-curva, utilizando el método descrito en [40], de manera que es posible reutilizar parte de ese código para la división de curvas.

Una primera aproximación al problema consistiría en dividir el segmento más largo de la curva con menor número de segmentos por su punto medio, repitiendo el proceso hasta igualar el número de segmentos de las dos curvas. La desventaja de esta aproximación es que no se tiene en cuenta la información que proporciona la correspondencia entre las dos curvas. Dado que los algoritmos de correspondencia descritos en el Capítulo 3 utilizan las curvas como secuencias de muestras independientemente de sus puntos de control, esta información puede ser utilizada para decidir los segmentos que serán

divididos y el punto por el que realizar la división: por ejemplo, dividiendo en primer lugar aquel segmento de C_A cuyas muestras se corresponden con muestras pertenecientes al mayor número de segmentos de C_B.

A.3. Simplificación de curvas con un número elevado de segmentos

En las pruebas experimentales, hemos podido comprobar que determinadas curvas con un número muy elevado de segmentos de pequeño tamaño provocan comportamientos incorrectos durante el proceso de seguimiento con algunos métodos de correspondencia debido a cuestiones de implementación. En particular, la importación de trazos desde otras aplicaciones después de dibujar los trazos a mano alzada puede dar como resultado un excesivo número de segmentos en alguna de las curvas que no necesariamente se traducen en una mejor representación visual de la característica que el artista esté deseando resaltar, lo que puede provocar que el seguimiento tenga problemas con los mínimos locales o a la hora de realizar el muestreo de dichos segmentos, que son propagados hacia los demás pasos del proceso. Por ello, es posible que la aplicación de un paso de preprocesado en el que la curva sea simplificada con la intención de reducir el número de segmentos se traduzca en un mejor rendimiento y resultados del algoritmo.

Existen numerosos trabajos acerca de la simplificación de curvas y superficies [43]. Uno de los métodos más utilizados por su sencillez [30] consiste en un algoritmo recursivo basado en la reducción de la distancia entre la curva original y la curva simplificada. Otros trabajos [44] formulan el problema como encontrar el camino más corto entre dos nodos de un grafo, que se corresponden con los vértices de la curva original, consiguiendo preservar las características destacables de la curva tales como ángulos. En [3] se aborda el problema considerando que el conjunto de puntos de control de la curva simplificada sea siempre un subconjunto de los puntos de control de la curva original: en nuestro caso, este enfoque podría ser interesante si suponemos que los puntos de control elegidos por el usuario están situados sobre características destacables de la curva.

Dadas las características de las curvas de nuestro trabajo, la aplicación de un método no interactivo de simplificación de curvas podría dar lugar a curvas demasiado simplificadas o que no conserven las características que el

artista deseaba resaltar. De manera similar al problema de establecer cuál es la correspondencia correcta entre dos curvas descrito en la Sección 3.2, hay un alto componente subjetivo a la hora de decidir si la simplificación de una curva es adecuada. Una alternativa puede ser ofrecer al usuario la posibilidad de controlar de forma interactiva el proceso de simplificación. En [41] se describe una aplicación de simplificación de curvas en la que el usuario puede controlar los distintos parámetros de la misma, lo que podría ser combinado con las herramientas de edición de la curva para permitir al usuario obtener una curva simplificada que se ajuste a la curva original de acuerdo con su percepción.

A.4. Paralelización del proceso de seguimiento

Dadas las características del proceso de seguimiento, sería posible aplicar algoritmos de paralelización que utilicen las capacidades de la *GPU* de la tarjeta gráfica para mejorar el rendimiento de la aplicación.

Apéndice B

Manual de usuario de la aplicación ARAS

En este manual se explica como utilizar la herramienta *ARAS*. En la Sección B.1 se describen los pasos más comunes de una sesión de trabajo con la herramienta, para posteriormente detallar las funciones del programa con más detalle.

B.1. Guía rápida

El proceso normal de una sesión de trabajo con la herramienta sigue los siguientes pasos:

1. El primer paso después de abierta la ventana de *ARAS* es abrir un proyecto previamente creado y guardado en disco, o bien crear un proyecto nuevo. Ambas opciones pueden ser seleccionadas desde el menú Archivo.

2. Una vez cargado el proyecto se añaden las imágenes de los fotogramas al proyecto. Seleccionando la opción del menú Herramientas ▷ Añadir fotogramas, aparece una ventana que permite añadir una lista de imágenes al proyecto. Cada una de estas imágenes será un nuevo fotograma de la secuencia.

3. Se añaden al proyecto los trazos previamente dibujados por un artista con una herramienta de diseño externa. Seleccionando la opción de

menú Herramientas ▷ Añadir trazos el usuario elige el archivo que contiene las curvas a añadir. Los trazos son añadidos al fotograma que esté abierto en ese momento, y en caso de que no haya ninguna ventana abierta, se presenta un diálogo que permite al usuario seleccionar a qué fotograma se quieren añadir los trazos.

4. Para poder lanzar el seguimiento, es necesario identificar dos trazos de diferentes fotogramas. Para ello se utiliza el modo de asignación de curvas pulsando sobre el icono de Asignación en la barra auxiliar (segundo icono de la barra vertical izquierda). Una vez activado el modo de asignación de curvas, el proceso se realiza en dos partes:

   - **Seleccionar el trazo inicial**: Se selecciona el trazo que se convertirá en el trazo inicial en el fotograma que será el inicio de la secuencia, pinchando con el botón izquierdo sobre el trazo o en el árbol de proyecto.
   - **Seleccionar el trazo final**: Se selecciona el trazo que se convertirá en el trazo final pinchando con el botón izquierdo en el fotograma que será el final de la secuencia o en el árbol de proyecto.

   Los trazos de los fotogramas intermedios se generan de forma automática copiando el trazo inicial y son añadidos a la escena. Esta secuencia de trazos tendrá un nombre genérico que es el nombre del trazo inicial, que es posible cambiar haciendo doble clic sobre el identificador en el árbol de proyecto.

5. El proceso de seguimiento se lanza seleccionando un trazo y pulsando el botón Lanzar proceso de seguimiento (tercer botón de la barra auxiliar) o seleccionando el elemento del menú Herramientas ▷ Lanzar seguimiento.

6. Una vez lanzado el proceso de seguimiento se interpolan las curvas de los fotogramas intermedios. Algunas veces es necesario ajustar las curvas intermedias e introducir restricciones. Los puntos de control se pueden mover arrastrándolos con el botón izquierdo del ratón. Para introducir una restricción, se pulsa con el botón derecho sobre el punto de control que se quiere marcar como fijo y se selecciona la opción Bloquear, lo que hace que no pueda ser desplazado por el usuario y que permanezca en la misma posición durante el proceso de seguimiento.

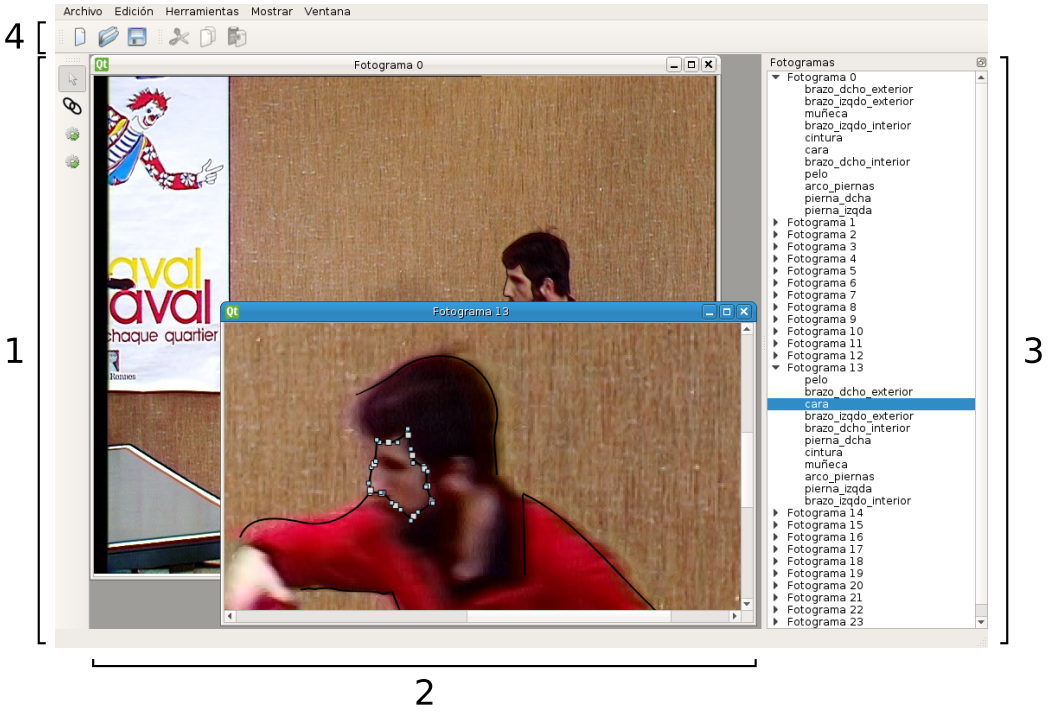


Figura B.1: Ventana principal de la aplicación ARAS

B.2. Manual extendido

La ventana de la herramienta (Figura B.1) se divide en cuatro zonas diferentes:

1. **Barra de herramientas de la aplicación**: mediante esta barra de herramientas, el usuario puede conmutar entre los diferentes modos de la aplicación (edición de trazos y asignación de trazos a secuencias), así como lanzar el proceso de seguimiento. Esta barra de herramientas consta de los siguientes elementos:

   
   Activa el modo selección, en el que se pueden editar los trazos y sus puntos de control.

   
   Activa el modo asignación de trazos a secuencias, en el que se pueden asignar los trazos a una secuencia de trazos.

   
   Lanza el proceso de seguimiento sobre la secuencia de curvas seleccionada actualmente.

2. **Ventanas individuales de fotograma**: cada una de estas ventanas representa un fotograma individual, con sus trazos. La edición de los

trazos se realiza en estas ventanas de forma gráfica, arrastrando los trazos o los puntos de control.

En el modo selección, el usuario puede realizar las siguientes acciones en esta ventana:

- **seleccionar trazo**: el usuario selecciona un trazo pinchando con el ratón sobre él. Un trazo debe estar seleccionado para poder ser manipulado (desplazarlo, modificar sus puntos de control).
- **mover trazo**: el trazo seleccionado se desplaza pinchando sobre él con el botón izquierdo y arrastrándolo a la nueva posición.
- **mover puntos de control**: el usuario mueve un punto de control pinchando sobre él y arrastrándolo a la nueva posición.
- **fijar punto de control**: el usuario puede fijar o liberar un punto de control pinchando con el botón derecho sobre él y seleccionando la opción "bloquear" o "desbloquear" en el menú emergente. Las consecuencias de marcarlo como fijo es que no puede ser desplazado por el usuario, y que permanecerá en la misma posición durante el proceso de seguimiento.
- **cambiar color de una secuencia**: para cambiar el color de una secuencia de trazos, se pulsa con el botón derecho sobre cualquiera de los trazos que componen la secuencia y se selecciona la opción Cambiar color en el menú desplegable.
- **eliminar trazo**: para eliminar un trazo, se pulsa con el botón derecho sobre él y se selecciona la opción Suprimir en el menú desplegable.

En el modo asignación, el usuario puede realizar las siguientes acciones en esta ventana:

- **asignar trazo a secuencia de trazos**: el usuario asigna un trazo a una secuencia de trazos pulsando sobre el trazo inicial, y a continuación pulsando sobre el trazo final. Al concluir esta operación, los trazos intermedios son generados automáticamente.

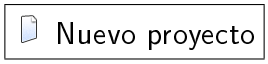
Tanto en el modo asignación como en el modo selección, el usuario puede realizar las siguientes acciones:

- ***zoom***: para hacer zoom se utiliza la rueda del ratón. Desplazar la rueda del ratón hacia arriba hace *zoom in* sobre el documento, y desplazarla hacia abajo realiza un *zoom out*.

3. **Árbol de proyecto**: este componente presenta al usuario una vista jerárquica de los trazos que componen el proyecto. En el árbol se muestran todos los fotogramas, con las secuencias de trazos que pertenecen a cada uno de los fotogramas. El usuario puede seleccionar un trazo en particular pulsando sobre su identificador.

4. **Barra de menús y funcionalidades comunes**: la barra de menús y la barra de herramientas de funcionalidades comunes permite un acceso rápido a las funciones de la herramienta, además de tareas relacionadas con la gestión de proyectos tales como abrir o crear un nuevo proyecto.

   El menú Archivo contiene los siguientes elementos:

   Inicia un nuevo proyecto vacío.

   Carga un proyecto almacenado en disco. Los ficheros utilizados para guardar la información de la aplicación son ficheros basados en *XML*, con la idea de facilitar la inclusión de características en nuevas versiones de la aplicación manteniendo la compatibilidad. Estos ficheros constan de dos partes: en la primera de ellas se definen los elementos que componen la escena, esto es, los trazos, secuencias de trazos y fotogramas con sus características. En la segunda parte se definen las asociaciones existentes entre ellos, el fotograma y la secuencia de trazos al que pertenece cada trazo.

   Cierra el proyecto actual.

   Guarda el proyecto actual con el mismo nombre de archivo.

   Guarda el proyecto actual con un nombre de archivo diferente.

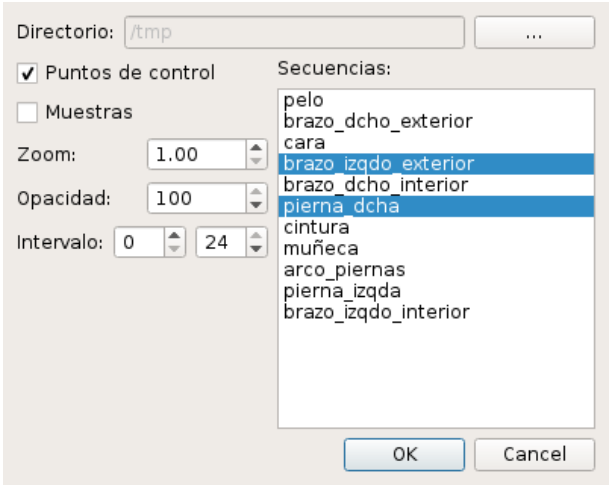


Figura B.2: Diálogo Exportar

Exportar imágenes

Esta opción permite exportar (como imágenes en formato *PNG*) las imágenes que componen el proyecto. En la Figura B.2 se muestra el diálogo a partir del cual el usuario puede exportar las imágenes, configurando las distintas opciones:

- **Directorio**: directorio en el que se crearán los archivos de imagen.
- **Puntos de control**: si está activada, los puntos de control de todas las curvas se representarán en las imágenes generadas.
- **Muestras**: si está activada, las muestras de todas las curvas se mostrarán en las imágenes generadas.
- **Zoom**: nivel de zoom de la imagen generada.
- **Opacidad**: nivel de opacidad de las imágenes que son utilizadas como fondo en los fotogramas.
- **Intervalo**: intervalo de fotogramas que se desea exportar.
- **Secuencias**: el usuario selecciona las secuencias que quiere que sean visibles en las imágenes generadas.

Salir

Cierra la aplicación.

El menú Herramientas contiene los siguientes elementos:

Añadir fotogramas

Esta opción permite al usuario añadir fotogramas, que se situarán tras el último fotograma existente. Al seleccionar esta opción aparece un diálogo de selección de archivo en el que el usuario puede seleccionar las imágenes que se utilizarán como fondo en los fotogramas.

Añadir trazos

Mediante esta opción el usuario añade trazos al proyecto. Al seleccionar esta opción, aparece un diálogo de selección de archivo en el que el usuario selecciona el archivo que contiene los trazos. Los trazos almacenados en ese fichero son añadidos al fotograma actual en caso de que haya alguno abierto. En caso contrario, se pide al usuario que indique a qué fotograma desea añadir los trazos.

La importación de curvas diseñadas utilizando otra aplicación soporta dos formatos: *Adobe Illustrator* 6 [2] (.ai) y *SVG* 1.1 [87] (.svg). Los archivos *Adobe Illustrator* 6.0 son documentos en lenguaje *PostScript* que siguen una estructura y convenciones determinadas, que se pueden consultar con más detalle en su especificación [1]. Un documento *PostScript* es en realidad un programa que contiene una secuencia de operadores que indican qué y cómo se debe construir la página, por lo que es necesario utilizar un intérprete *PostScript* para reconstruir la página. El formato *Adobe Illustrator* puede verse como un subconjunto del lenguaje *PostScript*, ya que no utiliza todas las características del lenguaje. El programa *Adobe Illustrator* define una serie de operadores nuevos utilizando los operadores estándares, y los utiliza para construir los documentos.


Lanzar seguimiento

Lanza el seguimiento de la secuencia de trazos seleccionada.

Configuración

Abre el diálogo que permite configurar las diferentes opciones de la aplicación. Es una ventana pensada para usuarios avanzados, con la que poder probar de forma sencilla los diferentes algoritmos implementados así como sus parámetros. En la ventana existen tres pestañas, que agrupan las opciones de configuración relacionadas con la correspondencia, con el motor de seguimiento, y con opciones de depuración

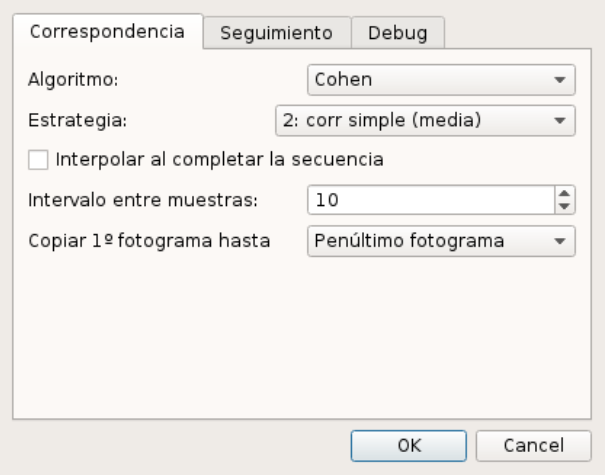


Figura B.3: Diálogo Configuración\Correspondencia

y diagnóstico.

La pestaña de Correspondencia se muestra en la Figura B.3. En esta ventana se eligen las opciones relacionadas con los algoritmos de correspondencia entre dos curvas. Estos algoritmos son utilizados durante el proceso de interpolación para identificar y asignar características entre dos curvas. En concreto, se pueden establecer los siguientes parámetros:

- **Algoritmo**: algoritmo de correspondencia a aplicar, con las siguientes opciones:
  - *Cohen*: método basado en correspondencia entre los vectores tangentes
  - *Geiger*: método basado en el vector de desplazamiento
  - *Johan*: método basado en la similitud de las curvas (por defecto)
  - *Lee*: método guiado por imagen
  - *Liu*: método basado en propiedades geométricas de un conjunto característico de puntos
- **Estrategia**: al realizar la correspondencia, es posible que se produzcan asignaciones uno a varios entre las muestras de los dos trazos. Este parámetro controla la estrategia utilizada para reducir

las asignaciones uno a varios a asignaciones uno a uno, igualando el número de muestras de cada uno de los trazos.

- *Original*: en cada iteración se remuestrean todos los trazos con el mismo número de muestras, situadas en las posiciones paramétricas de las muestras del trazo inicial. Al seleccionar esta estrategia, se utiliza el algoritmo de Geiger independientemente del que esté seleccionado. De esta manera, el comportamiento del método de seguimiento es igual al utilizado en la aplicación original.
- *Correspondencia simple (1ª)*: en caso de que una muestra del trazo A se asigne a varias muestras del trazo B, se escoge la primera muestras del grupo, y se descartan las demás.
- *Correspondencia simple (media)*: en caso de que una muestra del trazo A se asigne a varias muestras del trazo B, se escoge la muestra intermedia del grupo, y se descartan las demás.
- *Correspondencia completa*: en caso de que una muestra del trazo A se asigne a varias muestras del trazo B, se crea una nueva muestra en el trazo A por cada muestra del trazo B. Para ello se interpola su posición entre la muestra anterior y la muestra siguiente de A.
- *Correspondencia completa (deltas)*: en caso de que una muestra del trazo A se asigne a varias muestras del trazo B, se crea una nueva muestra en el trazo A por cada muestra del trazo B. Estas muestras están situadas aproximadamente en la misma posición que la muestra de A, pero se les suma un desplazamiento muy pequeño para diferenciarlas a la hora de realizar los cálculos.

- **Interpolar al completar la secuencia**: si esta casilla está activada, cuando se crea una nueva secuencia de trazos se lanza automáticamente el algoritmo de seguimiento con sólo los términos de forma activados. En caso contrario, los trazos intermedios son copias del trazo inicial o final, y no son interpolados de forma automática.
- **Intervalo entre muestras**: número de muestras que se visualizarán en la interfaz.

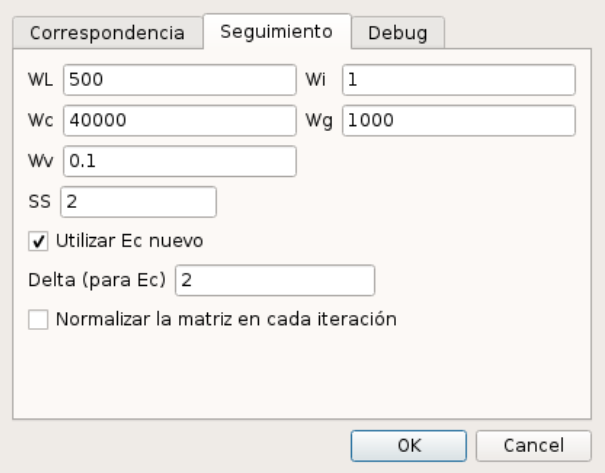


Figura B.4: Diálogo Configuración\KLT

- **Copiar 1º fotograma hasta**: al asignar el trazo final de una secuencia de trazos, cada uno de los trazos intermedios es generado copiando el trazos inicial o final. Esta opción indica hasta qué fotograma se copia el trazo inicial, siendo el resto de trazos copias del trazo final.

En la Figura B.4 se muestra la pestaña Seguimiento con los valores por defecto. Esta pestaña agrupa las opciones que afectan al algoritmo de seguimiento.

- W_L, W_C, W_V, W_I, W_G: pesos de los diferentes términos del algoritmo de seguimiento:
  - W_L: peso para el término de longitud de los vértices E_L
  - W_C: peso para el término de curvatura E_C
  - W_V: peso para el término de velocidad E_V
  - W_I: peso para el término de intensidad de la imagen E_I
  - W_G: peso para el término de gradiente E_G
- **Utilizar E_C nuevo**: determina las muestras utilizadas en el cálculo del término E_C. Si está activado, se utilizan las muestras situadas a una distancia (medida sobre la curva) $-\delta$ y $+\delta$ de la muestra actual, como hemos propuesto en las modificaciones al método de seguimiento. En caso contrario se utiliza el término E_C empleado en el algoritmo original.

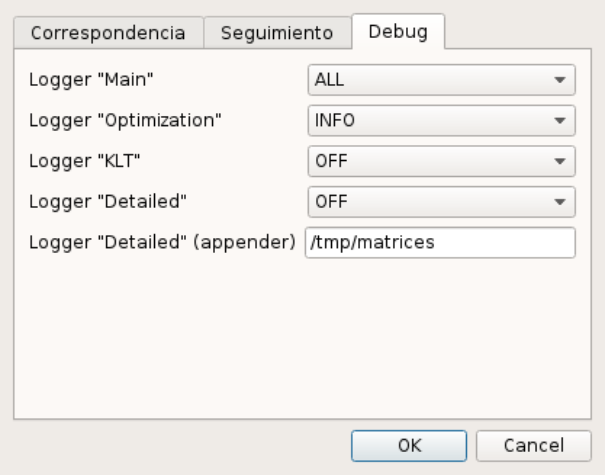


Figura B.5: Diálogo Configuración\Debug

- **Delta (para E_C)**: valor para la δ del apartado anterior (medido en píxeles).
- **Normalizar la matriz en cada iteración**: si está activado, normaliza la matriz de jacobianos teniendo en cuenta el número de muestras teórico que deberían afectar a cada uno de los puntos de control (en base a la longitud del segmento y el intervalo de muestreo), y el número de muestras que realmente afecta (en base al número de iteraciones realizadas).

En la Figura B.5 se muestra la pestaña Debug, que contiene opciones para depurar y mostrar información adicional tanto en la consola como en un archivo, dependiendo de la opción activada. De esta manera se pueden exportar los datos obtenidos durante los cálculos intermedios del algoritmo de seguimiento a aplicaciones externas.

- **Logger "Main"**: determina el nivel de detalle del "logger" principal, responsable de mostrar información general acerca de la aplicación y el estado de la misma.
- **Logger "Optimization"**: determina el nivel de detalle del "logger" de optimización, responsable de mostrar información durante el proceso de seguimiento.
- **Logger "KLT"**: determina el nivel de detalle del "logger" del módulo KLT, responsable de mostrar información acerca de las partes

de la aplicación reutilizadas de la aplicación original.

- **Logger "Detailed"**: determina el nivel de detalle del "logger" detallado, responsable de mostrar información de muy bajo nivel para el proceso de seguimiento.
- **Logger "Detailed" (appender)**: ruta del archivo en el que se guardará alguna información durante el proceso de seguimiento (un resumen al finalizar cada iteración y su matriz de jacobianos) para su análisis posterior. El archivo sólo se crea y se escribe si el nivel de detalle del "logger" "Detailed" está activado.

Glosario

ARAS *Automatic Rotoscoping for Animated Sequences*, acrónimo de la herramienta desarrollada en este trabajo.

correspondencia (*matching*) relación entre dos curvas paramétricas que hace corresponder a cada punto de una de ellas un punto de la otra teniendo en cuenta la morfología de las mismas.

curva Bézier curva paramétrica utilizada frecuentemente en el diseño asistido por ordenador, en el que la forma de la curva está determinada por el grado de su polinomio y las posiciones de sus puntos de control.

dense matching familia de algoritmos de correspondencia que representa las curvas como una secuencia de muestras y formula el problema como la minimización de una función de coste.

familia conjunto de roto-curvas pertenecientes a fotogramas contiguos que representan a la misma característica en diferentes instantes del tiempo.

feature matching familia de algoritmos de correspondencia que enfoca el problema como una búsqueda de correspondencia entre puntos específicos de las curvas que representan características destacables de las mismas.

fotograma clave en una secuencia, aquellos fotogramas que representan el comienzo o final de una roto-curva, de las que las posiciones de sus puntos de control se consideran fijos y sobre las que se calcula la correspondencia.

función de coste en el contexto de la correspondencia, función que representa el coste de hacer corresponder una muestra de la curva inicial con una muestra de la curva final.

función de energía Función utilizada para guiar el proceso de seguimiento en nuestro trabajo, utilizando términos de imagen basados en la información de los fotogramas y términos de forma basados en la información de las roto-curvas.

interpolación de curvas en el contexto de nuestra aplicación, generación de las curvas intermedias de una familia de curvas de una manera suave y visualmente adecuada.

Levenberg-Marquardt método de resolución de problemas *NLLS* que calcula una solución numérica utilizando la matriz jacobiana.

muestra (*sample*) punto discreto de una curva paramétrica, que puede ser representado por su valor paramétrico t o sus sus coordenadas, utilizado durante el establecimiento de la correspondencia y el proceso de seguimiento.

muestreo (*sampling*) proceso de obtención de un conjunto de puntos (muestras) de una curva paramétrica, no necesariamente uniforme, transformando su representación continua en una representación discreta.

NLLS *Non Linear Least Squares*, familia de problemas de mínimos cuadrados no lineales.

pirámide Gaussiana representación múltiple de una señal o imagen utilizando varios niveles de precisión en base a la aplicación de técnicas de suavizado y sub-muestreo.

punto de control cada uno de los puntos que forman el polígono de control de una curva Bézier y que determinan la forma de la curva.

roto-curva curva Bézier situada sobre un fotograma que representa alguna característica que el usuario considera de interés, y que es objeto de interpolación durante el proceso de seguimiento.

rotoscopia técnica utilizada en animación consistente en la utilización de una secuencia de imágenes de vídeo real como base para el dibujado de los fotogramas de una animación.

secuencia conjunto ordenado de fotogramas y las familias de roto-curvas relacionadas, que representan una unidad de tratamiento completa desde el punto de vista del usuario.

seguimiento (*tracking*) proceso de localización y búsqueda de una característica o forma a través de una secuencia de fotogramas.

skeleton matching familia de algoritmos de correspondencia que enfoca el problema extrayendo en primer lugar un *skeleton* de las curvas, que es una representación esquemática de su topología.

trazo utilizado como sinónimo de roto-curva en los contextos en los que la significación artística es relevante.

ventana (de imagen) en el contexto del procesado de imagen, región de una imagen que es utilizada para los cálculos, centrada en una muestra.

Bibliografía

[1] Adobe Systems Incorporated. *PostScript Language Document Structuring Conventions Specification*, 1992. URL `http://partners.adobe.com/public/developer/en/ps/5001.DSC_Spec.pdf`.

[2] Adobe Systems Incorporated. *Adobe Illustrator File Format Specification*, 1998. URL `http://partners.adobe.com/public/developer/en/illustrator/sdk/AI7FileFormat.pdf`.

[3] Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3-4):203–219, 2005.

[4] A. Agarwala. SnakeToonz: a semi-automatic approach to creating cel animation from video. *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, 2002.

[5] A. Agarwala. Keyframe-Based Tracking for Rotoscoping and Animation, 2004. URL `http://grail.cs.washington.edu/projects/rotoscoping/`.

[6] A. Agarwala, A. Hertzmann, D.H. Salesin, and S.M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (TOG)*, 23(3):584–591, 2004.

[7] R. Agushinta. Image recognition based on similarity of curve characteristics invariant to translation. *Journal of Theoretical and Applied Information Technology*, 74(1), 2015.

[8] T. Akenine-Moller, T. Moller, and E. Haines. *Real-Time Rendering*. AK Peters, Ltd. Natick, MA, USA, 2002.

[9] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.

[10] David J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[11] Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. The discrete Fréchet distance with shortcuts via approximate distance counting and selection. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 377. ACM, 2014.

[12] Kent Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999.

[13] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, and others. Manifesto for agile software development. 2001.

[14] Nir Ben-Zvi, Jose Bento, Moshe Mahler, Jessica Hodgins, and Ariel Shamir. Line-Drawing Video Stylization. In *Computer Graphics Forum*. Wiley Online Library, 2015.

[15] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. *European Conference on Computer Vision*, 588:237–252, 1992.

[16] Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Carter, and Jessica Hodgins. Style and abstraction in portrait sketching. *ACM Transactions on Graphics (TOG)*, 32(4):55, 2013.

[17] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1998.

[18] Boost. Boost C++ Libraries, 2016. URL `http://www.boost.org`.

[19] Maike Buchin, Anne Driemel, and Bettina Speckmann. Computing the Fréchet distance with shortcuts is NP-hard. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 367. ACM, 2014.

[20] N. Burtnyk and M. Wein. Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation. *Communications*, 1976.

[21] D. Chetverikov and Z. Szabo. A simple and efficient algorithm for detection of high curvature points in planar curves. *Proc. 23rd Workshop of the Austrian Pattern Recognition Group*, pages 175–184, 1999.

[22] Reed D. Clay and Henry P. Moreton. Efficient adaptive subdivision of Bézier surfaces. In *Eurographics*, volume 88, pages 357–371, 1988.

[23] S. Cohen, G. Elber, and R. Bar-Yehuda. Matching of freeform curves. *Computer-Aided Design*, 29(5):369–378, 1997.

[24] J.P. Collomosse, D. Rowntree, and P.M. Hall. Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):540–549, 2005.

[25] L.P. Cordella, P. Foggia, C. Sansone, F. Tortorella, and M. Vento. Graph matching: a fast algorithm and its evaluation. *Proc. of the 14th International Conference on Pattern Recognition*, pages 1582–1584, 1998.

[26] M. Cote, P.M. Jodoin, C. Donohue, and V. Ostromoukhov. Non-Photorealistic Rendering of Hair for Animated Cartoons. *Proceedings of GRAPHICON'04*, 2004.

[27] Paul de Faget De Casteljau. *Shape mathematics and CAD*, volume 2. Kogan Page, 1986.

[28] F. Di Fiore and F. Van Reeth. A Multi–Level Sketching Tool for "Pencil–and–Paper" Animation. *Sketch Understanding: Papers from the 2002 American Association for artificial Intelligence (AAAI 2002) Spring Symposium. Technical Report SS–02*, 8:32–36, 2002.

[29] F. Di Fiore, P. Schaeken, K. Elens, and F. Van Reeth. Automatic inbetweening in computer assisted animation byexploiting 2.5 D modelling techniques. *Computer Animation, 2001. The Fourteenth Conference on Computer Animation. Proceedings*, pages 192–200, 2001.

[30] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[31] Alon Efrat, Quanfu Fan, and Suresh Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal of Mathematical Imaging and Vision*, 27(3):203–216, 2007.

[32] J.D. Fekete, É. Bizouarn, É. Cournarie, T. Galas, and F. Taillefer. TicTacToon: a paperless system for professional 2D animation. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 79–90, 1995.

[33] Omrit Filtser and Matthew J. Katz. The Discrete Fréchet Gap. *arXiv preprint arXiv:1506.04861*, 2015.

[34] Kax Fleischer. Method of producing moving picture cartoons, October 9 1917. US Patent 1,242,674.

[35] Apache Foundation. Apache Log4cxx logging framework for C++, 2014. URL `http://logging.apache.org/log4cxx/index.html`.

[36] H. Fuchs, Z.M. Kedem, and S.P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, 1977.

[37] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21 (12):1312–1328, 1999.

[38] D. Geiger, A. Gupta, L.A. Costa, and J. Vlontzos. Dynamic programming for detecting, tracking, and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):294–302, 1995.

[39] C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75, 2001.

[40] B. Guenter and R. Parent. Computing the arc length of parametric curves. *Computer Graphics and Applications, IEEE*, 10(3):72–78, 1990.

[41] Sumanta Guha, Paul Janecek, and Nguyen Duc Cong Song. SIMPLIPOLY: Curvature-based polygonal curve simplification. In *GRAPP (GM/R)*, pages 166–171. Citeseer, 2007.

[42] Sariel Har-Peled, Amir Nayyeri, Mohammad Salavatipour, and Anastasios Sidiropoulos. How to walk your dog in the mountains with no magic leash. *Discrete & Computational Geometry*, 55(1):39–73, 2016.

[43] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, DTIC Document, 1997.

[44] Hiroshi Imai and Masao Iri. An optimal algorithm for approximating a piecewise linear function. *Journal of information processing*, 9(3):159–162, 1986.

[45] H. Johan, Y. Koiso, and T. Nishita. Morphing using curves and shape interpolation techniques. *Proceedings of Pacific Graphics 2000, 8th Pacific Conference on Computer Graphics and Applications*, pages 348–358, 2000.

[46] Mizuki Kagaya, William Brendel, Qingqing Deng, Todd Kesterson, Sinisa Todorovic, Patrick J. Neill, and Eugene Zhang. Video painting with space-time-varying style parameters. *IEEE transactions on visualization and computer graphics*, 17(1):74–87, 2011.

[47] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[48] A. Kort. Computer aided inbetweening. *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 125–132, 2002.

[49] Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the"Art&# x201D;: A Taxonomy of Artistic Stylization Techniques for Images and Video. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):866–885, 2013.

[50] T.Y. Lee and C.H. Lin. Feature-guided shape-based image interpolation. *Medical Imaging, IEEE Transactions on*, 21(12):1479–1489, 2002.

[51] Y. Li, M. Gleicher, Y.Q. Xu, and H.Y. Shum. Stylizing motion with drawings. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 309–319, 2003.

[52] Richard Linklater. Waking Life [Motion Picture]. *Fox Searchlight Pictures*, 2001.

[53] Richard Linklater. A Scanner Darkly [Motion Picture]. *Warner Independent Pictures*, 2005.

[54] Hong-Min Liu, Zhi-Heng Wang, and Chao Deng. Extend point descriptors for line, curve and region matching. In *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*, volume 1, pages 214–219. IEEE, 2010.

[55] L. Liu, G. Wang, B. Zhang, B. Guo, and H.Y. Shum. Perceptually based approach for planar shape morphing. *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*, pages 111–120, 2004.

[56] ImageMagick Studio LLC. ImageMagick Libraries, 2016. URL `http://www.imagemagick.org`.

[57] The Qt Company Ltd. Qt 5, 2016. URL `http://doc.qt.io/qt-5/index.html`.

[58] The Qt Company Ltd. Qt 5: The Graphics View Framework, 2016. URL `http://doc.qt.io/qt-5/graphicsview.html`.

[59] The Qt Company Ltd. Qt 5: Model View Programming, 2016. URL `http://doc.qt.io/qt-5/model-view-programming.html`.

[60] The Qt Company Ltd. Qt 5: Signals and Slots, 2016. URL `http://doc.qt.io/qt-5/signalsandslots.html`.

[61] J. Lu, H.S. Seah, and F. Tian. Computer-assisted cel animation: post-processing after inbetweening. *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Austalasia and South East Asia*, 2003.

[62] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Understanding Workshop*, 121:130, 1981.

[63] Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer. Approximating the Integral Fréchet Distance. *arXiv preprint arXiv:1512.03359*, 2015.

[64] K. Melikhov, F. Tian, H.S. Seah, Q. Chen, and J. Qiu. Frame skeleton based auto-inbetweening in computer assisted cel animation. *Proceedings of the 2004 International Conference on Cyberworlds (CW'04)-Volume 00*, pages 216–223, 2004.

[65] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 1999.

[66] M. Owen and P. Willis. Modelling and interpolating cartoon characters. *Computer Animation'94., Proceedings of*, pages 148–155, 1994.

[67] Steve R. Palmer and Mac Felsing. *A Practical Guide to Feature-Driven Development*. Pearson Education, 2001.

[68] Brian L. Price, Bryan S. Morse, and Scott Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *2009 IEEE 12th International Conference on Computer Vision*, pages 779–786. IEEE, 2009.

[69] J. Qiu, H.S. Seah, F. Tian, Q. Chen, and Z. Wu. Enhanced auto coloring with hierarchical region matching. *Computer Animation and Virtual Worlds*, 16(3/4):463, 2005.

[70] J. Qiu, H.S. Seah, F. Tian, Z. Wu, and Q. Chen. Feature-and region-based auto painting for 2D animation. *The Visual Computer*, 21(11): 928–944, 2005.

[71] D.F. Rogers. *An Introduction to NURBS: With Historical Perspective*. Morgan Kaufmann Publishers, 2001.

[72] Winston W. Royce. Managing the development of large software systems. In *Proceedings of IEEE WESCON*, volume 26, pages 328–338. Los Angeles, 1970.

[73] T.W. Sederberg and E. Greenwood. A physically based approach to 2–D shape blending. *ACM SIGGRAPH Computer Graphics*, 26(2):25–34, 1992.

[74] B. Serra and M. Berthod. Subpixel contour matching using continuous dynamic programming. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 202–207, 1994.

[75] Kaveh Shahbaz. Applied similarity problems using Fréchet distance. *arXiv preprint arXiv:1307.6628*, 2013.

[76] M. Shapira and A. Rappoport. Shape blending using the star-skeleton representation. *Computer Graphics and Applications, IEEE*, 15(2):44–50, 1995.

[77] J. Shi and C. Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[78] M. Sonka, V. Hlavac, and R. Boyle. Image processing, analysis, and machine vision second edition. *International Thomson*, 1999.

[79] T. Steihaug. The Conjugate Gradient Method and Trust Regions in Large Scale Optimization. *SIAM Journal on Numerical Analysis*, 20(3): 626–637, 1983.

[80] M. Stokes, M. Anderson, S. Chandrasekar, and R. Motta. A Standard Default Color Space for the Internet–sRGB. *Microsoft and Hewlett-Packard Joint Report, Version*, 1, 1996.

[81] B. Stroustrup et al. *The C++ programming language*. Addison-Wesley Reading, MA, 1997.

[82] R. Szeliski et al. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528, 1990.

[83] Hirotaka Takeuchi and Ikujiro Nonaka. The new new product development game. *Harvard business review*, 64(1):137–146, 1986.

[84] A. Thurston. Parsing Computer Languages with an Automaton Compiled from a Single Regular Expression. *CIAA 2006*, 4094:285–286, 2006.

[85] A. Thurston. Ragel State Machine Compiler, 2014. URL `http://www.colm.net/open-source/ragel`.

[86] International Telecommunications Union. BT.802 : Test pictures and sequences for subjective assessments of digital codecs conveying signals produced according to Recommendation ITU-R BT.601, 1994. URL `https://www.itu.int/rec/R-REC-BT.802/en`.

[87] W3C. Scalable Vector Graphics (SVG) 1.1 (Second Edition), 2011. URL `https://www.w3.org/TR/SVG11/`.

[88] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. *Proc. European Conference on Computer Vision*, 1, 2004.

[89] J. Wang, Y. Xu, H.Y. Shum, and M.F. Cohen. Video tooning. *ACM Transactions on Graphics (TOG)*, 23(3):574–583, 2004.

[90] Tinghuai Wang, John Collomosse, David Slatter, Phil Cheatle, and Darryl Greig. Video stylization for digital ambient displays of home movies. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 137–146. ACM, 2010.

[91] Brian Whited, Gioacchino Noris, Maryann Simmons, Robert W. Sumner, Markus Gross, and Jarek Rossignac. Betweenit: An interactive tool for tight inbetweening. In *Computer Graphics Forum*, volume 29, pages 605–614. Wiley Online Library, 2010.

[92] Timothy Randall Wylie. *The discrete Fréchet distance with applications*. PhD thesis, Montana State University Bozeman, 2013.

[93] Chih-Kuo Yeh, Pradeep Kumar Jayaraman, Xiaopei Liu, Chi-Wing Fu, and Tong-Yee Lee. 2.5 D Cartoon Hair Modeling and Manipulation. *IEEE transactions on visualization and computer graphics*, 21(3):304–314, 2015.

[94] Jun Yu and Dacheng Tao. *Modern Machine Learning Techniques and their Applications in Cartoon Animation Research*, volume 4. John Wiley & Sons, 2013.

[95] Jun Yu, Meng Wang, and Dacheng Tao. Semisupervised multiview distance metric learning for cartoon synthesis. *IEEE Transactions on Image Processing*, 21(11):4636–4648, 2012.

[96] Y. Zhang. A fuzzy approach to digital image warping. *Computer Graphics and Applications, IEEE*, 16(4):34–41, 1996.