Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

AMH VIDA

Autor:
José Luis Martínez Jiménez

Tutores:
D. Francisco Javier Finat Codes
D. Valentín Cardeñoso Payo

AMH VIDA

Resumen

El reconocimiento de patrones en el movimiento humano a partir de secuencias de video tomadas por una cámara móvil es un problema clásico en Visión Computacional. La dificultad del mismo radica en la complejidad de la geometría del cuerpo humano y la de los movimientos, incluso bajo condiciones ideales de iluminación. La Coreografía muestra un amplio abanico de casos difíciles de seguir y analizar. Conseguir desarrollar un sistema que permita reconocer pasos de baile, daría pie a resolver otros muchos casos más sencillos de reconocimiento de patrones de movimiento humano con infinidad de aplicaciones prácticas en multitud de tareas relacionadas con movimientos humanos complejos. Para ello se utilizarán videos facilitados por el Museo de Mariema.

El trabajo propuesto pretende únicamente centrarse en desarrollo del sistema de segmentación y filtrado, dejando como trabajo futuro el desarrollo de herramientas de aprendizaje.

Abstract

Pattern recognition in human movement from video footage taken by a mobile camera is a classical problem in Computer Vision. The trouble lies in the complexity of the geometry of the human body and movement, even under ideal conditions of illumination. The choreography shows a wide range of cases difficult to follow and analyze. Getting develop a system to recognize dance moves, would lead to solving many simple cases in pattern recognition of human movement with many practical applications in a multitude of tasks related to complex human movements. It will be used videos provided by the Mariema Museum.

The proposed work focus only on the development of segmentation and filtering system, leaving as future work the development of learning tools.

Agradecimientos

Con estas líneas deseamos mostrar un más que merecido agradecimiento a todas aquellas personas que durante este tiempo han estado ahí ayudando a conseguir este trabajo.

En primer lugar a mis tutores, don Javier Finat y don Valentín Cardeñoso, por su paciencia y por darme la oportunidad de trabajar en este proyecto; a don Rubén Martínez y a don Francisco Delgado, por su apoyo y sus consejos; a don Eduardo Rodríguez Alija por todos esos buenos momentos que hemos pasado durante la carrera; y como no, a toda mi familia, por su paciencia y comprensión.

Sin vosotros, este trabajo nunca hubiese sido posible.

Gracias.

Índice general

Índice de ilustraciones

Índice de tablas

Bloque I: Introducción

Capítulo 1. Introducción.

1.1. Motivación.

Uno de los temas de mayor interés en el área de la computación es el de crear sistemas capaces de hacer tareas similares a las que realizan los humanos. El reconocimiento de la postura que adopta un individuo a lo largo del tiempo es uno de estos tópicos (muy de actualidad en áreas como los videojuegos), que permite definir un nuevo tipo de interacción entre la persona y la máquina.

Hoy en día hay varios trabajos orientados hacia este tópico, como por ejemplo Kinect, sin embargo, dada la enorme complejidad geométrica del cuerpo humano, aún es un reto encontrar un metodología que permita aportar una solución en tiempo real a la identificación de posiciones complejas de un cuerpo humano en movimiento con una única cámara en movimiento no controlado en condiciones no ideales de iluminación y sin ningún tipo de información adicional.

Con este trabajo pretendemos desarrollar un sistema que permita reconocer los movimientos que realiza un bailarín en un video facilitado por el Museo de Mariema donde se presenta una abundante cantidad de casos complejos de estudiar derivados de los rápidos desplazamientos y rotaciones del bailarín, ya que resolviendo este problema para un caso de esta complejidad, se podrían resolver otros de menor dificultad.

1.2. Pruebas de concepto.

El *ambiente para la prueba de concepto* que ha dado lugar a esta Memoria corresponde a un escenario de danza que se recoge en el video adjunto a esta memoria, en el que un bailarín ejecuta movimientos de danza sobre un escenario real mientras la cámara lo graba, con condiciones cambiantes de iluminación, que complica seriamente el análisis del video.

Se supone que la cámara de video es sujetada por un individuo que está presente en la escena, moviendo la cámara para intentar seguir al bailarín mientras éste se desplaza por el escenario.

1.3. Estructura.

Esta memoria está agrupada en bloques, cada uno de los cuales recoge una serie de capítulos que están, de una forma u otra, relacionados entre sí.

El primer bloque, titulado Introducción, en el cual nos encontramos actualmente, pretende mostrar un acercamiento sobre los motivos que han impulsado la creación de este proyecto. El segundo bloque está dedicado a exponer las diversas metodologías seguidas para resolver los problemas de visión computacional necesarias para el desarrollo del sistema que cumpla con los objetivos expuestos, sin olvidarnos de analizar las diversas dificultades encontradas, y que en definitiva, han condicionado las técnicas empleadas para resolverlas. Después profundizaremos en los métodos de preparación del proyecto en su etapa más básica, la planificación, explicada en el bloque tercero. El bloque cuarto y quinto corresponde al análisis y diseño del sistema que está siendo expuesto en este documento respectivamente. Explicaremos en el bloque sexto las diversas pruebas realizadas en el sistema construido para evaluar y documentar los resultados obtenidos en este trabajo. El séptimo bloque enlazará con el anterior recopilando que objetivos hemos cumplido y cuáles han sido los resultados de nuestro trabajo, haciendo un resumen general de las conclusiones obtenidas a lo largo del proyecto. El bloque octavo, está dedicado a exponer las referencias y otros

trabajos sobre los que nos hemos basado, y por lo tanto, sin los cuales este proyecto nunca hubiese podido realizarse. Y finalmente concluimos con un capítulo de anexos.

Capítulo 2. Estado del arte.

2.1. Marco teórico del Proyecto

Un problema que ha sido necesario resolver previamente es la *elección del modelo más apropiado* para el tratamiento de la información. A la vista de las dificultades relacionadas con la captura y procesamiento en tiempo real de un modelo de perspectiva y las dificultades para actualizarlo (hay un gran número de oclusiones parciales fijas y móviles), se ha optado por modelos "menos estructurados" asociados a la detección, clasificación y etiquetado de "eventos". No obstante, se han evaluado diferentes posibilidades que se comentan en esta subsección correspondiente al marco teórico.

El marco teórico del Proyecto corresponde al *Reconocimiento de hechos relativos a la estructura del cuerpo humano a partir de Video* en Visión Computacional. Suponiendo conocidas las herramientas básicas de Procesamiento y Análisis de Imagen, al tratarse de escenarios arquitectónicos, hay un solapamiento parcial con la Reconstrucción 3D a partir de video y, de una forma más específica con la obtención de la Estructura a partir del Movimiento (SFM: Structure From Motion).

2.1.1. Estructura a partir del Movimiento (SfM)

El propósito de SfM es obtener información sobre la escena ó sobre un objeto a partir del movimiento de una cámara móvil; la *estructura de la escena* se refiere a la captura de información 3D sobre "hechos" de la escena. Ello requiere una estimación previa de la localización (posición y orientación) y a continuación del movimiento rígido de una cámara; en la escena dicho movimiento está compuesto de una colección de transformaciones rígidas elementales -rotaciones y traslaciones- que generan el grupo euclídeo en el espacio *SE(3; R)*. Una vez establecida las correspondencias entre hechos, se calculan las matrices de proyección y, a partir de ellas la estructura de la escena mediante triangulación. En ausencia de información sobre otros sensores ó de conocimiento previo sobre la calibración de la cámara, hay que deducir la transformación euclídea a partir de la información contenida en cada par de vistas que corresponde a una transformación afín que relaciona dos vistas:

A) La vista V_t en el instante t está representada por una proyección de la escena sobre el plano de la cámara para la localización considerada (posición y orientación de la cámara).
B) La comparación entre elementos homólogos de diferentes vistas tiene como objetivo la recuperación de la "estructura" de la escena, incluyendo localización de la propia cámara (parámetros extrínsecos) y de "hechos significativos".
C) La Geometría Epipolar proporciona un modelo estructural para la puesta en correspondencia entre elementos homólogos de diferentes vistas próximas, pero tiene un coste computacional elevado.
D) Para escenas con un modelo de perspectiva previamente calculado, la puesta en correspondencia se realiza a partir de la identificación del desplazamiento aparente de los elementos de perspectiva (líneas de perspectiva, puntos de fuga, línea del horizonte) con un coste sensiblemente inferior, sobre todo si se utiliza una aproximación lineal al modelo de perspectiva. Este modelo de perspectiva proporciona el primer paso para una Reconstrucción Proyectiva (con una línea proyectiva como línea del horizonte).

E) La *auto-calibración* (calibración de los parámetros extrínsecos) a partir de varias vistas proporciona un modelo métrico, correspondiente a la reconstrucción euclídea; ésta es la parte más complicada del proceso.

El objetivo final de la Reconstrucción 3D es la producción de un modelo 3D, incluyendo modelos realistas que incluyen mallas y re-mapeo de texturas. A pesar de la gran cantidad de avances que han tenido lugar desde principios de los noventa y de la brillantez de resultados, esta metodología tiene un elevado coste computacional y requiere actualmente una gran cantidad de retoques manuales, por lo que no se desarrolla en la Memoria; para detalles ver R.Hartley y A.Zisserman (2000).

Los problemas más significativos para SfM conciernen a la puesta en correspondencia automática mediante el cálculo robusto de las restricciones bilineales (matriz fundamental/esencial) ó multilineales (incluyendo tensor trifocal para el caso de 3 vistas) y la estimación de las matrices de proyección.

2.1.2. Movimiento a partir de la estructura (MfS)

A diferencia del SfM, el MfS parte del conocimiento de (alguna parte de) la estructura correspondiente a la escena ó al objeto y trata de evaluar características cinemáticas de objeto(s) en movimiento dentro de la escena; el movimiento puede ser real ó aparente. Esta aproximación resulta especialmente útil cuando se dispone de información métrica sobre la escena (procedente p.e. de un escaneo láser 3D) y se desea proporcionar un asistente a la navegación real a partir de la navegación visual capturada por una cámara. Dos casos importantes son:

- Para una *cámara fija*, el objeto debe tener un movimiento propio y, en este caso, hay que resolver un problema de *seguimiento* a partir de la detección y puesta en correspondencia de hechos homólogos que se suponen bien conocidos previamente; el caso más "simple" corresponde a objetos con marcas que, recientemente, aparecen asociados a sensores ópticos (tipo LED, p.e.).
- Para una *cámara móvil* y en una escena estática (se supone que no hay otros agentes en movimiento), el problema a resolver es la estimación del movimiento propio ó *egomotion* a partir del "desplazamiento aparente" de los hechos geométricos ó de los elementos estructurales (ligados a un modelo de perspectiva, p.e.).

En cualquier caso, como el movimiento es continuo, en ausencia de marcas ó de hechos significativos que puedan ser seguidos fácilmente, interesa generar un pegado estéreo denso} que permita recuperar una aplicación densa de profundidad. Para alcanzar un pegado denso, hay que diseñar e implementar herramientas que faciliten una interpolación entre vistas 2D ó bien entre los modelos 3D toscos generados a partir del conocimiento de la estructura de la escena ó del objeto. El pegado denso tiene un elevado coste computacional y, actualmente, no se conocen soluciones que proporcionen resultados robustos en tiempo real. Por ello, la estrategia más apropiada consiste en utilizar una cantidad suficiente de marcas sobre la escena que faciliten localización, seguimiento y soporte para el guiado a partir de las 4 más próximas localizadas.

Los *problemas más significativos para MfS* conciernen a la detección de las 4 marcas "más próximas", las oclusiones parciales, el tratamiento redundante de información, y el cálculo del egomotion (movimiento propio) a partir del seguimiento de hechos ó bien del flujo óptico 'denso".

2.1.3. Más allá del MfS y SfM

La descripción anterior pone de manifiesto la existencia de una realimentación entre la estructura S y el movimiento M que se puede representar simbólicamente mediante un esquema secuencial que tiene lugar a lo largo del tiempo:

$$\ldots \rightarrow S \rightarrow M \rightarrow S \rightarrow M \rightarrow S \ldots$$

Existen soluciones alternativas que tratan de estimar estructura y movimiento de forma simultánea, pero ninguna de ellas proporciona una solución en tiempo real. Al analizar las metodologías utilizadas para las dos realimentaciones consideradas (SfM y MfS) se observa que las fases con mayor complejidad computacional corresponden a la estimación de la Estructura. Por ello, cualquier aplicación que está enfocada hacia una respuesta on-line debe poner el acento en la estimación del movimiento. Más abajo se revisan algunos aspectos técnicos relacionados con los problemas más significativos señalados para SfM y MfS, en relación con su utilidad para el Reconocimiento a bajo nivel a partir de secuencias de video. A continuación se aborda el problema de las características significativas para los hechos 0D con objeto de facilitar detección, localización y seguimiento.

2.1.4. Detectores y descriptores

Un *detector de hechos* es un proceso que permite detectar hechos significativos en imágenes digitales. La información más importante del detector es la que corresponde a su localización y la escala (tamaño relativo) a la que aparece detectado. Dos características importantes para tener un "buen detector de hechos" son la *repetitividad* (el mismo hecho puede ser detectado en diferentes imágenes de una secuencia de video) y la *fiabilidad* (el punto detectado debe ser lo suficientemente distintivo para que el número de posibles pegados sea pequeño).

Un *descriptor* es un proceso que a partir de la información de hecho y de imagen genera información descriptiva que, habitualmente, se presenta en forma de "vectores de hechos". Estas descripciones se usan para pegar hechos entre diferentes imágenes. Por ello, un descriptor debe ser *invariante* con respecto a transformaciones euclídeas (rotación, traslación), de semejanza (euclídeas salvo escala) ó afines (deformaciones aparentes debidas a efectos de perspectiva), según el marco geométrico elegido. Además de las características dadas para los detectores, un descriptor debe estar caracterizado por presentar "casi" el mismo valor en diferentes imágenes.

A la vista del tipo de imágenes y de hechos que utilizaremos para secuencias de video, en el resto de esta subsección nos limitamos a detectores y descriptores para puntos 0D y en menor medida para regiones 2D; para los elementos estructurales vinculados a modelos de perspectiva son más significativos los hechos 1D, pero el alineamiento de minisegmentos, la supresión de outliers y la generación de modelos robustos presenta un mayor coste computacional. El prerrequisito correspondiente a un rápido procesamiento (al menos on-line) motiva que nos hayamos restringido a la captura y procesamiento y análisis de hechos 0D y 2D que presentan un menor coste computacional.

2.1.4.1. Detectores para puntos y regiones

En [Sch00] se presenta una clasificación de detectores de puntos en tres categorías: basadas en contornos, intensidad y en modelos paramétricos:

Detectores basados en contornos: En primer lugar se extraen contornos y se identifican os puntos que tienen características geométricas especiales como p.e. junturas, extremos ó máximos de

curvatura. Una aproximación multiescala (diferentes tipos de pirámides) se puede aplicar para obtener resultados más robustos.

- *Detectores basados en intensidad:* Se utilizan (diferentes formas y combinaciones de) las derivadas de primer y segundo orden para detectar el cambio de intensidad en el entorno de cada punto.
- *Detectores basados en modelos paramétricos:* Se obtienen utilizando plantillas para detectar puntos característicos con "formas especiales" como las junturas dobles (en forma de L ó de T) ó triples (en forma de Y ó de ↑), p.e.

Un *detector repetible* debe ser invariante por transformaciones geométricas (euclídeas, de semejanza, afines) ó radiométricas (variaciones en la intensidad). A continuación se presenta una relación de los detectores de puntos más utilizados para Reconstrucción 3D:

- El *detector de esquinas de Harris* [Har88] es invariante con respecto a rotaciones y, parcialmente, con respecto a cambios de intensidad, pero no con respecto a cambios de escala.
- El operador SIFT [Low99] busca los máximos locales correspondientes a las diferencias de Gaussianos (DoG) en el espacio y la escala
- En [Mik01] se utilizan las esquinas detectadas por Harris para identificar los hechos en el dominio espacial y, a continuación se usa una *pirámide laplaciana* (Laplaciano en escala) para seleccionar los hechos que son invariantes por transformaciones de escala.
- Un detector *invariante por transformaciones afines* se desarrolla en [Tuy00]: A partir de un máximo de intensidad local se buscan entre las rectas que pasan a través de dicho punto para encontrar extremos locales para la función intensidad; enlazando los extremos se obtiene una región de interés que se aproxima a continuación por una elipse. Cambiando las rectas y usando elipses para representar las regiones, se obtiene una colección de regiones que son invariantes por transformaciones afines y que resultan estables por cambios de orientación de hasta 60º. El criterio para evaluar la repetitividad es la ratio del número de puntos repetidos sobre el total de puntos detectados en la parte común de dos imágenes.
- El *detector de Harris mejorado* (IHD) utiliza un jet de orden 2 para caracterizar los puntos de interés; la fiabilidad se mide en [Sch00] por la difusión de los jets locales; cuanto más "difusivos" sean los valores descriptivos más capacidad discriminatoria (y, por tanto, más fiabilidad) tiene el detector. La difusión se mide usando la entropía.

En la práctica es poco viable (salvo para puntos correspondientes a marcas con propiedades reflectantes) el seguimiento de puntos particulares a lo largo de una minisecuencia de movimiento; por ello, es preferible realizar un seguimiento de regiones 2D con un tamaño por encima de un umbral. Las *regiones* en imagen *2D* se obtienen por técnicas de agrupamiento de puntos que presentan características similares estáticas (geométricas, radiométricas) ó cinemáticas (tasas de variación de las propiedades geométricas ó radiométricas por debajo de un umbral).

Conclusión: Aunque los resultados de IHD son los mejores, es difícil identificar el detector más apropiado para utilizar, en general, debido a condiciones cambiantes. Asimismo, tampoco hay datos sobre la velocidad de procesamiento ni sobre un análisis comparativo para la velocidad entre diferentes soluciones. En el apartado siguiente se muestra que la tasa de pegado final es aparentemente independiente del detector elegido. Por ello, un detector medio pero rápido y un buen descriptor son probablemente una elección razonable.

2.1.4.2. Descriptores para puntos

En este apartado, presentamos una *clasificación de los descriptores para puntos* y su aplicación al "pegado" para la reconstrucción *3D* según [Mik01]:

- *Descriptores basados en distribuciones:* Los histogramas se utilizan para representar las características de cada región. Las características más relevantes son la intensidad en píxeles, las distancias desde el centroide [Laz03], la ordenación relativa de la intensidad [Zab94] ó el gradiente [Low04].
- *Descriptores basados en Frecuencia Espacial:* Estas técnicas se utilizan para la clasificación y descripción de texturas. La descripción de texturas utilizando filtros de Gabor se ha estandarizado en MPEG7 [Ro51]
- *Descriptores basados en diferenciales* para reforzar la fiabilidad a partir de un jet local (colección de derivadas locales) para identificar las regiones de interés [Sch00]
- *Descriptores basados en Momentos* asociados a una región *2D*. El momento central de una región es una combinación de formas de grado bajo invariantes con respecto a las transformaciones de semejanza y afines usuales.

La *invariancia para descriptores* se obtiene de muchas formas para diferentes factores que intervienen en los trabajos que acaban de ser citados. Así, p.e.,

- Los máximos de gradientes locales con diferentes direcciones se usan en [Low04], [Mik01] para identificar la orientación.
- La escala y el factor de sesgo ("oblicuidad" correspondientes a deformaciones aparentes) determinados en la fase de detección se utilizan para normalizar los pegados de imagen [Bau00]
- En [Mik01] se presenta una evaluación de los descriptores con una tasa de detección ROC (Receiver Operating Characteristic) con respecto a falsos positiva.
- En [Sch00] se presenta una evaluación de los descriptores usando la ratio de pegados correctos sobre posibles pegados.
- El descriptor SIFT que es invariante con respecto a escala y parcialmente con respecto al cambio de vista [Low99] y los métodos basados en dicho operador presentan los mejores resultados.

Diferentes evaluaciones recientes ponen de manifiesto que la utilización de *detectores basados en regiones* y, más particularmente, en detectores que son invariantes con respecto a escala y transformaciones afines da mejores resultados que otros.

Como *conclusión*, la elección de un buen descriptor es más importante que la elección del detector. Sin embargo, nótese que la mayor parte utilizan información producida por detectores complejos (salvo escala, p.e.), lo cual puede ralentizar la realización de la tarea. En general, los descriptores SIFT y basados en SIFT ([Ke04], [Mik05]) tienen un buen rendimiento para puntos en secuencias de video.

2.2. Reconocimiento a bajo nivel a partir de video

En esta sección se abordan los problemas y resultados más significativos para el Proyecto en relación con problemas de Reconstrucción y Reconocimiento a bajo nivel a partir de *video* digital.

2.2.1. Estrategias habituales para seguimiento

Esta subsección tiene dos apartados correspondientes al Flujo Óptico (como marco estructural), y los modelos de partículas. Se muestran las limitaciones de los dos primeros y la necesidad de contar con información adicional para proporcionar modelos más estables.

2.2.1.1. Flujo Óptico

El modelo físico-matemático de más bajo nivel para evaluar el desplazamiento de una distribución de puntos δ**q** está basado en el flujo Óptico. El *flujo Óptico* a lo largo de una secuencia de vistas representa la orientación local "en cada punto" del movimiento en imagen, *suponiendo que la intensidad en la escala de grises es constante*. Este modelo proporciona una aproximación "suave" a la proyección sobre el plano de imagen del campo Φ asociado al movimiento (real ó aparente) de objetos en la escena 3D en un intervalo "infinitesimal" en el que "no hay" cambio en la función de intensidad en la escala de grises $I_g(x,y)$; por ello, φ está dado por $grad(I_g) = 0$.

Intuitivamente, se desea una estimación del "campo del movimiento" en la escena 3D ó bien al menos una estimación en la imagen, es decir, el *objetivo general* es la estimación de una distribución de vectores que represente el campo de movimiento mediante un campo vectorial ξ que a cada punto $\underline{x}$ le asocie un vector $\mathbf{v} = \xi(\underline{x})$.

Si podemos "pegar" los datos locales correspondientes a los vectores $\xi(\underline{x})$ en un objeto global, se obtiene un "campo vectorial" ξ_α que representa las características del movimiento para cada objeto móvil B_α a lo largo de una secuencia de imágenes en términos de una "función" f_α. En este caso, cada campo ξ_α expresa la variación de unas funciones f_α (ó variables dependientes) con respecto a los "incrementos" en las variables independientes x_j (posición e intensidad para un "hecho significativo" del objeto móvil, p.e.). La variación de la función f_α correspondiente al incremento con respecto a x_j está dada por $\frac{\partial f_\alpha}{\partial x_j}$; la versión discreta de esta derivada parcial está dada por $f_\alpha(x_1, \dots, x_j, \dots, x_n) - f_\alpha(x_1, \dots, x_j - 1, \dots, x_n)$, es decir, el valor que toma la función de intensidad en el píxel anterior en la dirección de la coordenada x_j. Por ello, el problema inicial a resolver es el "agrupamiento espacial" que etiquetamos como una *distribución* D_α correspondiente a un "tipo de movimiento" a lo largo de la secuencia de imágenes muestreadas. En el caso más simple, cuando la distribución D_α presenta un "patrón regular" asociado a un movimiento elemental (un generador del álgebra de Lie correspondiente a las transformaciones afines), podemos asociar un campo ξ_{D_α} a la distribución que representará las características cinemáticas del movimiento. La regularidad que se pretende identificar para D_α requiere resolver problemas tales como:

- detectar movimientos a un nivel muy tosco (sustracción del fondo, p.e.);
- estimar características de mini-movimientos para desagregar movimientos;
- agrupar según valores de las características (modulo un cierto umbral);
- etiquetar las regiones (con forma eventualmente variable) que presenten "cierta homogeneidad cinemática"

El output debe ser una segmentación dinámica dada por un *mapa cinemático*. Este mapa es una unión disjunta de regiones "homogéneas" desde el punto de vista cinemático, es decir, regiones "homogéneas" en el espacio-tiempo, es decir, regiones que presenten diferencias cinemáticas por debajo de un umbral seleccionado por el usuario a lo largo de una secuencia de video. Lamentablemente, las hipótesis precedentes son ideales y, en la práctica (salvo casos sencillos correspondientes a escenas de tráfico con cámara fija, p.e.) el desacoplamiento de movimientos complejos en "patrones de movimiento" está aún muy lejos de haber sido resuelto; como banco de

pruebas se puede ensayar con el seguimiento asociado a coreografías del folklore tradicional español, p.e.

Una vez inicializado el proceso a partir de una elección apropiada de detector y descriptor (ver más arriba), la estimación del flujo óptico resulta más sencilla si se añaden *modelos previos relativos al seguimiento*. El seguimiento presupone algún tipo de modelado tanto para el objeto, como para las características cinemáticas del movimiento; CGAL ha incorporado las KDS que resultan de utilidad para este problema). El modelado cinemático a alto nivel de objetos articulados (el cuerpo humano, p.e.) presenta cierta complejidad y no es un objetivo del Proyecto, por lo que no se aborda en esta Memoria. Para simplificar, suponemos que los objetos a seguir son rígidos y corresponden a "hechos" detectados en la escena. En este caso, el seguimiento no-lineal de mínimos cuadrados minimiza una función de coste sobre el espacio de estados del objeto en movimiento; un modelo localmente lineal se obtiene a partir del gradiente del error residual. Para construir el gradiente se requiere que la función de error residual sea diferenciable, pero las oclusiones dan lugar a discontinuidades que es preciso rellenar para resolver el problema del seguimiento.

Además, en un modelo "suave" ó diferenciable el espacio de estados debe ser completamente observable; la observabilidad se pierde cuando la matriz jacobiana no tiene rango máximo (singularidades cinemáticas) en correspondencia con el alineamiento de componentes funcionalmente independientes para estados próximos. Este problema puede resolverse usando varias cámaras, pero para una silla de ruedas es inviable y, además, caso de haber varias cámaras, a menudo la intersección de los dominios enfocados por diferentes cámaras no tiene rango suficiente para proporcionar información robusta. Por ello, la persistencia espacio-temporal de datos es una hipótesis a añadir, modelar y resolver computacionalmente mediante algoritmos de restauración dinámica de imagen.

2.2.1.2. Seguimiento visual

El *seguimiento visual* (tracking) tiene como objetivo la construcción on-line de una trayectoria que permita actualizar, predecir y validar el modelo propuesto mediante alguna variante del "campo del movimiento" (en términos del campo gradiente, flujo óptico, etc). Incluye la caracterización de los *eventos* asociados a la (des)aparición de objetos móviles en las vistas y la actualización de la información a lo largo de la secuencia. Para ello, hay *tres estrategias* basadas en puntos, kernel y siluetas. En presencia de trayectorias previsibles, es deseable que el seguimiento proporcione una representación simbólica en la escena en términos de un grafo dinámico; las operaciones básicas permitidas (contracciones y expansiones) sobre el grafo se introducen para preservar la topología del grafo punteado (eventualmente desconectado) asociado a la supresión del objeto, en previsión de nuevos eventos dinámicos en la escena que no afectan al seguimiento de un objeto previamente seleccionado.

Herramientas para el seguimiento visual

Según [McL97], hay 4 tipos de herramientas para el seguimiento visual:

- Utilizar marcas ó hechos bien conocidos en la escena asociados a detectores y descriptores, diseñar e implementar "seguidores" de dichos hechos y utilizarlos para actualizar la pose del objeto. Esta estrategia es adaptable a modelos geométricos rígidos [Har92] y deformables (usando snakes, p.e.) [Bla93].
- *Sustracción del fondo:* Los seguidores de blobs definen un objeto de interés como el que puede ser "separado" con respecto al resto de la escena utilizando un algoritmo tipo

"figura/fondo" tales como la detección de un movimiento con respecto a un fondo estacionario [Kol94], [Let93], [McL92].

- *Medida directa de la velocidad* usando flujo de imagen, correlación ó métodos similares ([Bra93], [Bur89], [Pah93]). En este caso, se supone que el esquema de correlación se actualiza a lo largo del tiempo. Para ello, se debe mantener fija la plantilla, en cuyo caso el "seguidor de correlación' devuelve una realimentación de la posición, en lugar de una realimentación de la velocidad. Esta metodología es muy sensible a pequeños cambios en la orientación, aunque se puede remediar correlacionando sólo regiones pequeñas en imagen y permitiendo deformaciones, aunque en este caso el algoritmo es una variante del seguidor de hechos descrito en varios lugares [Shi94].
- Seguimiento de hechos que registra hechos dados típicamente por puntos y "líneas" a lo largo de una secuencia de imágenes ([Sha95], [Smi95] [Zha92]), que en ocasiones se presentan de forma conjunta con restricciones de rigidez para imponer una consistencia global ([Bea97], [McL95]).

2.3. El movimiento humano

El análisis del movimiento humano basado en Visión Computacional es un tópico que ha recibido una atención creciente desde principios de los noventa. Este tópico tiene un gran número de aplicaciones en cuestiones tales como:

- ***Seguimiento y vigilancia visual,*** incluyendo reconocimiento de cara, forma de andar, gestos faciales, monitorización de congestiones pedestres, patologías ó asistencia en lugares públicos;
- ***Desarrollo de interfaces avanzadas para usuarios***, incluyendo lenguajes de signos, comunicación inteligente entre hombres y máquinas, ó asistencia a personas dependientes ó discapacitadas;
- ***Identificación y asistencia al diagnóstico*** para mejorar el rendimiento de actividades deportivas, culturales ó de entretenimiento, detección de patologías asociadas a tareas cotidianas

La complejidad de las cuestiones anteriores es creciente y cada una de ellas incluye aspectos relevantes de las precedentes. El análisis de pasos de baile ó de coreografías que se abordan en esta Memoria forma parte del tercer item de la relación anterior.

2.3.1. Aportaciones iniciadas en los noventa

Una de las primeras revisiones del trabajo realizado se debe a Aggarwal et al [Agg94] quienes desarrollan un enfoque alternativo al de multicuerpos en Robótica, adoptando un enfoque basado en el movimiento para cuerpos articulados y elásticos dotados de movimiento no-rígido. Una distinción básica de este trabajo consiste en introducir una ***primera taxonomía*** basada en el conocimiento ó no a priori de modelos de formas. A la vista de la enorme diversidad de movimientos que pueden presentar las coreografías de uno ó varios personales esta Memoria se enmarca dentro de la aproximación que no utiliza modelos previos

Desde un punto de vista histórico, el siguiente survey (realizado por Cedras y Shah [Ced95]) se centra en cuestiones relacionadas con la extracción del movimiento humano incluyendo reconocimiento de las partes del cuerpo y una estimación de las diferentes configuraciones que se adoptan en tareas complejas. Este enfoque proporciona una distinción básica en la que podríamos

calificar como ***segunda taxonomía*** entre la ***captura de las apariencias*** y el análisis de las ***representaciones simbólicas***. Esta taxonomía reintroduce el problema del reconocimiento y la dependencia con respecto al punto de vista asociado a cada cámara.

Las distinciones entre planos sagital, frontal y coronal son típicas de los estudios biomecánicos para el movimiento humano que se realizan a finales de los noventa. Esta aproximación resulta de gran utilidad para multicuerpos y, en particular, para robots humanoides y sus aplicaciones tanto a la robótica de discapacitados como a mejorar la interacción hombre máquina. Sin embargo, esta descomposición en referencias ortogonales móviles de poca utilidad en el análisis visual del movimiento humano, pues la descomposición cartesiana (proyección sobre referencias coordenadas ortogonales eventualmente móviles) no permite "pegar" los datos procedentes de diferentes vistas ni tampoco una síntesis automática, requiriendo la asistencia de un operador humano para su interpretación desde diferentes localizaciones de las cámaras. Asimismo, tampoco permite realizar la síntesis de movimientos que se requiere para simulación ó animación en cuestiones de producción de contenidos digitales, p.e.

Las limitaciones del enfoque anterior se abordan en el survey de Aggarwal y Cai [Agg99] con tres grandes áreas para la interpretación del movimiento humano en relación con:

- ***Análisis por componentes*** del movimiento humano para cada una de las partes del cuerpo, lo cual requiere un trabajo de segmentación previa que se realiza utilizando color (bajo nivel, bajo coste) ó mediante la extracción de curvas significativas (nivel y coste medio)
- ***Seguimiento*** a partir de una ***única cámara*** para identificar gestos (secuencia de "posturas") ó de ***múltiples cámaras*** para el análisis y síntesis de movimientos en 3D.
- ***Reconocimiento de tareas*** ó actividades a partir de secuencias de video, para el que se requiere el desarrollo de Sistemas Expertos, habitualmente de tipo supervisado (usando plantillas deformables, típicamente).

La complejidad de cada una de las técnicas asociadas a los enfoques mencionados, así como la diversidad de las taxonomías a las que hacen referencia (segmentación, análisis / síntesis de movimiento, reconocimiento) hacen realmente difícil la transferencia entre resultados procedentes de estos enfoques. Por ello, a partir de finales de los noventa se producen aportaciones independientes en cada uno de esto campos con escasas conexiones entre sí, a pesar del gran interés para cuestiones de animación 3D, p.e..

2.3.2. Hacia una integración

De cara a una integración de las diferentes aproximaciones señaladas más arriba, es conveniente identificar un esquema básico para facilitar el modelado del movimiento humano que pueda facilitar la compatibilidad entre las representaciones basadas en las apariencias y las representaciones simbólicas. De acuerdo con Wange et al [Wan03], un esquema ***pipeline común*** consiste en los pasos siguientes:

- ***Segmentación del movimiento*** con la identificación de cambios bruscos de tipo cualitativo (en términos de shots) para cada una de las componentes del cuerpo. Algunas técnicas típicas son sustracción del fondo, aproximaciones estadísticas para fondo adaptativo (superposición de distribuciones normales en el caso más simple, p.e.), diferencias entre frames muestreados en la secuencia ó variantes de flujo óptico.

- ***Clasificación de objetos*** con especial atención a la visibilidad de objetos móviles (tratamiento de oclusiones) y la posible restauración de la información capturada. Las dos estrategias típicas se basan en la extracción de la forma ó de características del movimiento a partir de formas toscas móviles (blobs)
- ***Seguimiento humano*** que incluye la integración de componentes en un modelo global coherente a lo largo de cada minisecuencia de video, incluyendo técnicas como variantes de Filtros Kalman, Redes Bayesianas Dinámicas, ó algoritmos de condensación (para nubes de puntos móviles ó partículas, p.e.) para el seguimiento de regiones (bajo nivel) ó de hechos significativos (nivel medio)
- ***Reconocimiento de acciones*** asociados a gestos ó tareas desarrollados por cada uno de los agentes móviles en escena, incluyendo versiones cualitativas de sistemas dinámicos (tales como envoltura dinámica temporal ó los modelos ocultos de Markov) y su aplicación al reconocimiento de tareas propiamente dicho (mediante pegado de plantillas deformables ó bien la identificación de los estados "estables" o bien críticos)
- ***Descripción semántica*** en relación con las taxonomías básicas asociadas al tipo de movimiento, incluyendo estimación, indexación y clasificación de las acciones ó tareas desarrolladas por el sujeto. Esta aproximación requiere una mayor especificación de modelos y hechos correspondientes a las 4 anteriores, así como el desarrollo de herramientas más avanzadas correspondientes a la realimentación entre los aspectos estáticos/dinámicos o bien entre las aproximaciones basadas en apariencias/esqueletos.

Las soluciones desarrolladas desde finales de los noventa utilizan aproximaciones híbridas que combinan varios aspectos de los que acaban de ser mencionados y no ha sido posible encontrar tablas en las que se presente un análisis comparativo con respecto a un banco de pruebas aceptable y compartido. En términos generales, es necesario desarrollar una realimentación dinámica entre las apariencias (bajo nivel correspondientes a los dos primeros items) y las representaciones simbólicas (medio nivel correspondientes a los items 3 y 4) que puedan facilitar un soporte para el desarrollo de la descripción semántica (item 5). En relación con esta realimentación es conocido que:

- ***Apariencias*** pueden ir ligadas a diferentes aspectos como puntos de control $0D$ (incluyendo o no marcadores para facilitar el seguimiento), curvas $1D$ significativas (siluetas, pliegues, contornos) ó regiones coloreadas $2D$. Todas ellas deberían ser fácilmente detectables, robustas y estables a lo largo de minisecuencias de video
- ***Representaciones simbólicas*** son de tipo esqueletal y se pueden obtener mediante un "adelgazamiento" de los bordes de las regiones

En la realimentación clásica entre **S** (Structure/Shape) y **M** (movimiento) mencionada más arriba, los descriptores KLT (Kanade, Lucas, Tomasi) juegan un papel inicialmente importante para facilitar el tracking de objetos **0D** en objetos con geometría rígida (bien definida) y para una cámara móvil. En nuestro caso, la geometría es deformable y la cámara está fija. Por otro lado, los detectores asociados a formas **1D** utilizan modelos deformables (snakes para contornos, splines para superficies) con un elevado número de parámetros que no es posible ajustar on-line debido a la complejidad, variabilidad y ausencia de repetición idéntica en los movimientos llevados a cabo por los bailarines. Estas circunstancias motivan que centremos la atención sobre las regiones **2D** que sean fáciles de identificar y seguir, para asociar representaciones simbólicas correspondientes.

2.3.3. Restricciones asociadas a la captura de información

A la vista de la diversidad de técnicas y estrategias, en una memoria de estas características es imposible abordar todos los aspectos mencionados. La selección de estrategias viene condicionada por los materiales disponibles (tomas realizadas con videocámara no-profesional) y por algunos de los puntos más débiles detectados en la literatura existente (representaciones simbólicas) a las que se dedican este apartado y el siguiente.

Los ***inputs*** con los que se cuenta proceden de secuencias de video monocameral a baja resolución, tomadas bajo condiciones de iluminación muy deficientes, con desenfoques frecuentes, una muy baja definición cromática. Las primeras etapas del procesamiento están ligadas a procedimientos de filtrado de altas frecuencias (extracción de regiones) y bajas frecuencias (extracción de elementos característicos).

En las secuencias de video consideradas la detección y el seguimiento de puntos característicos es prácticamente imposible debido a la ausencia de marcadores, la baja definición de las imágenes, las (auto)oclusiones frecuentes y los rápidos movimientos de los bailarines en escena. Por ello y a la vista de los inputs disponibles, se ha dado prioridad al filtrado de altas frecuencias para la detección de regiones con características radiométricas persistentes **2D** asociadas a "bandas de color" previamente identificadas.

2.4. Técnicas para la representación del cuerpo humano y reconocimiento de la pose.

Existen principalmente dos tipos generales de formas de representar el cuerpo humano para su reconocimiento, las basadas contornos y en esqueletos; sin embargo las estrategias utilizadas para generarlas depende mucho de las características de los datos que se van a usar para generarlas. En este epígrafe sólo tendremos en cuenta las posibilidades con una sola cámara y ningún otro elemento adicional, ya que es el contexto tecnológico más próximo a nuestro caso particular (para mas información de algunos de estos otros sistemas [Web05]).

En el caso más sencillo que se puede plantear, nos encontraremos ante datos procedentes de un video grabado una cámara estática con objetos en movimiento. Ante esta perspectiva, la estrategia más común es la de realizar una operación de resta sobre dos frames consecutivos para evaluar las disparidades entre ambas[Dim05]. Esto nos da un contorno aparente generado a través del movimiento del individuo, que puede intentar ser procesado directamente para identificar hechos en la poligonal que lo forma, o bien realizarse una serie de operaciones que permitan generar un esqueleto que resuma la silueta. Se trata de una estrategia capaz de simplificar los datos a analizar de forma muy eficiente, aun cuando el entorno es geométricamente muy complejo, sin embargo pierde eficacia cuando se permite que la cámara pueda estar en movimiento, efectos de zoom-in y zoom-out, o cuando el individuo esté completamente inmóvil.

Cuando la cámara está en movimiento el problema se complica demasiado, y las estrategias que se pueden plantear son numerosas pero a costa de un aumento significativo del tiempo de procesamiento, siendo la mayoría de los casos offline.

Para imágenes sueltas, ya existen trabajos que, por ejemplo, estudian simultáneamente la estimación de la pose y el reconocimiento de acciones [Yan10], basado en el aprendizaje, con modelos esqueletales de 4 secciones (un brazo izquierdo, un brazo derecho, las dos piernas y el tronco superior), aunque en general los resultados en tiempo de procesamiento y resultados del esqueleto pueden ser mejores.

2.4.1. Representaciones esqueletales.

Un *esqueleto* **Sk(B)** asociado a una representación simbólica humana del cuerpo **B** se puede entender como una colección de segmentos conectados entre sí mediante nodos que representan las articulaciones, teniendo en cuenta las diferentes componentes *visibles* en la ejecución de una tarea. Las representaciones del cuerpo humano (utilizadas habitualmente en Robótica) son de tipo árbol con 2 segmentos correspondiente al tronco (componentes superior e inferior) al que se conectan las demás componentes (cabeza y 4 extremidades) con sus correspondientes juntas. Esta representación no es realista y requiere información completa sobre la captura, algo que no está disponible. En particular, las representaciones esqueletales consideradas en esta Memoria son móviles y la información es incompleta con fenómenos de auto-oclusión muy frecuentes. Además, a la vista de determinados pasos de baile (incluyendo cruces de brazos), no se puede pretender que el grafo sea un árbol, permitiéndose la generación de lazos o su desdoblamiento en cadenas (subárboles) que representan el estiramiento de extremidades no solapadas. Los problemas más complicados de resolver son la generación de esqueletos a partir de las apariencias (en ausencia de modelos previos), la relevancia de la representación (debido al carácter incompleto de la información) y la conectividad en las representaciones proporcionadas.

Tal y como se enuncia en el trabajo [Alo08], el esqueleto representa la estructura de un objeto (conservando la conectividad, los agujeros y, en cierto modo, la extensión del mismo) con un número pequeño de píxeles.

Las características que debería tener un esqueleto cualquiera son [Ogn92]:

- Estructura unidimensional, lo más delgada posible (1 pixel).
- Mantener la conectividad de la figura original.
- Invariante frente a transformaciones isométricas de la figura original.
- Centrado en la imagen original.
- Reversibilidad: sería también deseable que la información del esqueleto permitiese la reconstrucción de la figura original, si bien esto rara vez es posible.

Los métodos de obtención del esqueleto pueden encuadrarse dentro de cuatro categorías [Ogn92]:

- Relacionados con el modelo de adelgazamiento, también llamado grass-fire.
- Obtención del eje medio a través procedimientos analíticos.
- Erosionado topológico.
- Transformadas topológicas: obtención del eje medio a partir de la transformada de distancia

Todos los métodos presentan deficiencias y limitaciones. En general, son muy sensibles al ruido. Los más habituales los de "adelgazamiento" o bien construcción de "ejes medianos":

- ***El adelgazamiento*** o grass-fire consiste en la aplicación iterada de operadores morfológicos básicos (erosión o dilatación) ó una combinación de ambos (apertura y cierre) para minimizar el riesgo de la pérdida de conectividad del modelo.
- ***La construcción de ejes medianos*** utiliza una variante de los diagramas de Voronoi asociada al mapa de bisectrices para cada una de las componentes (cabeza, tronco, extremidades) detectadas en el borde ∂B ó silueta del cuerpo.

Ambos métodos son muy dependientes de la apariencia de la poligonal $\boldsymbol{\partial B}$ correspondiente a los bordes del objeto **B**. En particular, una falta de simetría da lugar a ramas del esqueleto que carecen de realismo y la variabilidad de las representaciones hace prácticamente imposible mantener una estructura estable; en la representación simbólica desarrollada se detectan "bifurcaciones" asociadas al solapamiento de extremidades que generan "dislocaciones en la cinemática" que no resultan naturales desde el punto de vista de la representación. Por ello, ha sido necesario desarrollar una construcción independiente para las representaciones simbólicas que se expone más abajo.

En relación con los problemas abiertos mencionados más arriba, nótese que el análisis de tipo simbólico permite conectar de una forma más directa con el punto de más débil (descripción semántica) del marco planteado más arriba. En este caso, es necesario desarrollar una "gramática de movimientos" especificada por elementos (sustantivos, verbos y adverbios) y la sintaxis asociada. Según [Zil03] la metodología Labanotation fue introducida por Hutchinson (1991) para describir los "saltos" en el ballet clásico, permitiendo una descripción de las diferentes fases (preparación, vuelo, aterrizaje) para este tipo de movimientos. En cierto modo, Labanotation se puede considerar como un precedente de la parte sintáctica asociada al enfoque presentado en esta Memoria. No obstante, en nuestro caso no se pretende desarrollar una taxonomía para los movimientos de baile individual o coreografías más generales; este objetivo es demasiado ambicioso y requiere un análisis más detallado de las diferentes variantes que presentan pasos de baile incluso para un mismo sujeto.

Como puede observarse, la elección del modelo esqueletal y su representación no es trivial, y se hablará más en profundidad de las decisiones realizadas sobre este punto en el epígrafe Generación del esqueleto de este mismo capítulo; no obstante podemos adelantar que la solución que proponemos está basada en la obtención del eje medio a través de procedimientos analíticos con ciertas peculiaridades.

2.4.1.1. Representaciones esqueletales aplicadas al reconocimiento del cuerpo humano

Para concluir este capítulo, finalmente queda reseñar algunos trabajos orientados al uso de representaciones esqueletales como:

- [Apa05], afirmando que trabajando con imágenes sueltas parte de un fondo sencillo y bajo condiciones de iluminación controladas es capaz de generar un esqueleto en un tiempo de 1.25 seg. para imágenes de tamaño 640x480 aplicando técnicas de erosión topológica.
- [Mor04] cuyo trabajo está orientado también para imágenes individuales, y que consiste en aplicar técnicas de segmentación de bajo nivel a la imagen para la generación de dicho esqueleto.

Bloque II: Metodología de desarrollo

Capítulo 3. Introducción.

3.1. Introducción a la metodología.

Este bloque pretende ilustrar el procedimiento metodológico empleado para resolver las diferentes etapas en la resolución del problema de generación de esqueleto a partir de imágenes de video de danza.

Se analizarán los puntos fuertes y débiles de cada una de las etapas que utiliza el algoritmo, así como su grado de relación o dependencia con el marco del proyecto, es decir, con los videos de danza para los que se ha decidido desarrollar este proyecto.

3.2. Estructura general.

La metodología define un procedimiento a realizar por el cual unos datos de entrada (input) son tratados y estudiados para generar una salida. En nuestro caso, el intput de este proyecto será siempre una secuencia de imágenes que corresponden a un único individuo realizando movimientos de coreografía de danza sobre un escenario. En cambio, el output corresponde al resultado de estudiar dichas imágenes con el cual se pretende reconocer cuál es la posición y postura que adopta el individuo a lo largo de la secuencia de video del input.

Para generar un output por cada frame de la secuencia de imágenes, se divide el problema en una serie de etapas, cada una de ellas con una tarea general a realizar, que a su vez puede dividirse en una o varias sub-etapas concretas. En concreto, nosotros hemos dividido el problema en dos etapas generales, procesamiento, y análisis, que están estrechamente relacionadas en un bucle que se retroalimenta con las iteraciones previas dentro del ciclo metodológico.

- El procesamiento define una serie de operaciones a nivel global sobre el frame (input) que serán presentadas con mayor detalle en el siguiente capítulo de este bloque. Se trata de una etapa de filtrado, en la que se extraerán el fondo con la mayor precisión posible, separándolo de la silueta del bailarín.
- El análisis corresponde a un tratamiento de los datos más selectivo y localizado, y debido a su envergadura, se divide en varias sub-etapas. Por un lado, utilizando los resultados del procesamiento, habrá que identificar la región de la imagen en la que se encuentra exactamente el bailarín, resumir esos datos y finalmente estudiarlos para identificar la postura en la que se encuentra el bailarín.

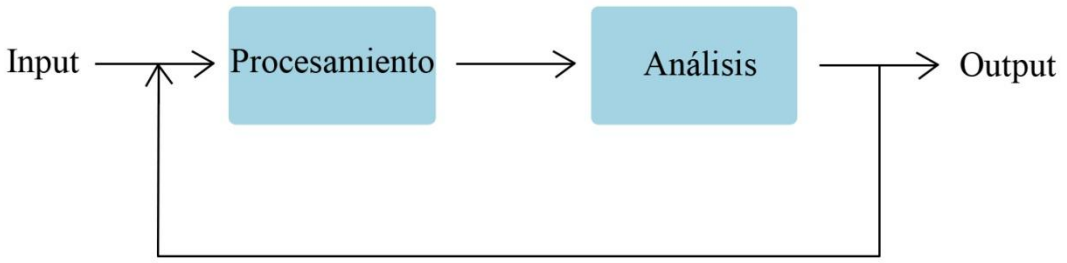


Ilustración 3-1 - Estructura conceptual del proyecto

Es importante hacer mención especial al hecho de que aunque los objetivos de este proyecto abarcan hasta el reconocimiento de la postura del cuerpo humano y la identificación de las características que definen dicha postura, dicho resultado, a su vez, pueden ser utilizados para la interpretación del significado del movimiento, que ha sido dejado como parte de trabajo futuro por limitaciones de tiempo.

Ilustración 3-2 - Estructura conceptual del sistema y grado de "inteligencia"

3.3. Conceptos generales.

Antes de empezar cualquier explicación es importante afianzar algunos conceptos, imprescindibles para la comprensión del resto del capítulo.

Entendemos por *frame* o *imagen digital* (a la que en numerosas ocasiones referenciaremos únicamente como *imagen*), un mapa de bits con forma de matriz, en el que cada elemento de la matriz es un pixel. Los *frames* pueden tener 1 o varios canales.

Una imagen de un canal puede ser, por ejemplo, una imagen en escala de grises, es decir una imagen en la que cada pixel tiene un valor de degradado de gris cuyo valor de intensidad oscila entre 0 y 255. Sin embargo, para representar una imagen a color es necesario organizar o codificar la imagen mediante lo que se denomina “espacios de color” o “modelos de color”. Algunos de estos modelos de color son el clásico RGB o el HSV.

El modelo RGB se basa en la idea de que todos los colores son una combinación de tres colores básicos o primarios, rojo (Red), verde (Green) y azul (Blue). Típicamente se suele representar este sistema con un triángulo o un círculo en el que el blanco está en el centro.

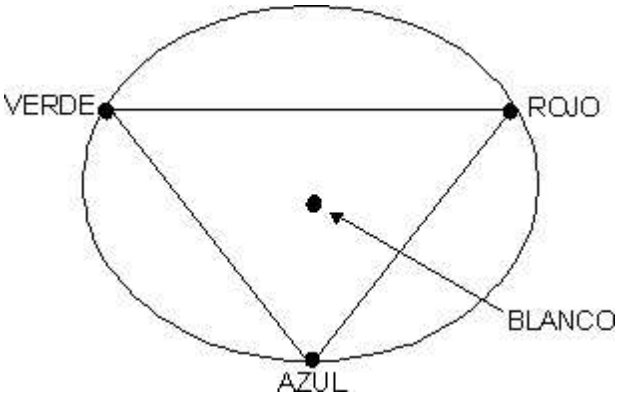


Ilustración 3-3 Representación del sistema RGB

El modelo HSV fue creado en 1978 por Alvy Ray Smith y es otro modelo de representación de imagen basado también en la representación de color en tres canales de información, matiz (Hue), saturación (Saturation) y brillo (Value).

- Matiz o tono: es el tipo de color. Se representa como el graduaje.
- Saturación: determina la pureza del color. Se representa como la distancia al eje del brillo.
- Brillo: representa la altura en el eje blanco-negro

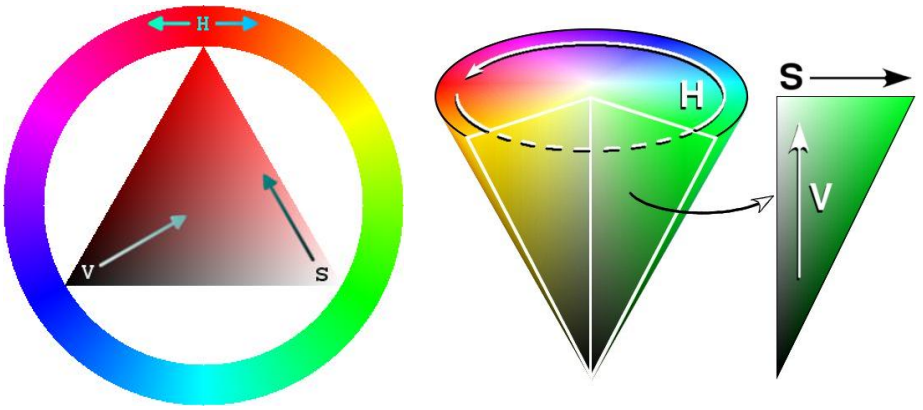


Ilustración 3-3 Representación del sistema HSV

El modelo HSV es una transformación no lineal del modelo RGB, de tal forma que se puede pasar de uno a otro aplicando las siguientes fórmulas:

$$H = cos^{-1}\left(\frac{\frac{1}{2}(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)\cdot(G-B)}}\right)$$

$$S = 1-\left(\frac{3\cdot \min(R,G,B)}{R+G+B}\right)$$

$$I = \frac{1}{3}(R+G+B)$$

Un histograma de frecuencias de una imagen es un gráfico en el que se evalúa las modas presentes en la imagen en cada valor de intensidad (0-255) de cada canal. Es una forma de resumir el contenido de la imagen, muy rápida de evaluar, sin embargo es una medida un tanto tosca.

Un video no es sino una secuencia de frames. Representan la toma de sucesivas imágenes con un periodo de muestreo homogéneo por un dispositivo, normalmente una videocámara o una webcam. Ese periodo de muestreo se le denomina FPS (Frames Per Second).

Entendemos como escena al contexto en el que se realiza la grabación de video.

3.4. Algoritmo de cálculo de extremos relativos.

Como se verá a lo largo de este bloque, para resumir gran cantidad de datos de la matriz de la imagen, se procederá a sintetizar ese volumen de datos en una o varias funciones discretas, y a su vez, los datos de esas funciones que mas información nos va a aportar serán los extremos relativos (tanto máximos como mínimos relativos) de dichas funciones.

El proceso que utilizamos para calcular los extremos relativos de la función es extremadamente sencillo; sólo hay que identificar aquellos intervalos de la función donde hay un cambio de signo de la primera derivada, y después elegimos un punto representativo de dicho intervalo.

Para la estimación de la derivada utilizamos la siguiente aproximación, con $y_{i+1} - y_i$ e suficientemente pequeño:

$$\frac{dx}{dy} \approx \frac{x_{i+1} - x_i}{y_{i+1} - y_i}$$

Donde x_{i+1} y x_i corresponden a términos de la variable independiente, y tanto y_{i+1} como y_i, corresponden a términos de la variable dependiente de la función en cuestión, y para que realmente la aproximación a la derivada que realizamos converja realmente a la derivada, la diferencia entre ambos la aplicamos para funciones discretas cuyo intervalo entre valores es de distancia 1 pixel, que es la distancia más pequeña de que disponemos.

No obstante, el problema está en que la función que nosotros disponemos es de la forma $g(x) = f(x) + \varepsilon(x)$, siendo $f(x)$ la función que nosotros queremos medir, $g(x)$ la función que nosotros conocemos y $\varepsilon(x)$ el ruido que presenta toda medición.

Con ello, este procedimiento retorna todos los extremos relativos presentes en una función que contiene ruido, y por tanto es bastante probable que varios de los extremos relativos que devolvería este procedimiento en realidad sean inútiles y confusos. Es por ellos que introducimos 3 parámetros de filtrado de extremos relativos para intentar descartar aquellos extremos relativos que tengan muchas probabilidades de haber sido causados por el ruido:

- El primer parámetro mide la relevancia del valor del extremo relativo i-ésimo con respecto a su predecesor. Si toman valores similares, entonces debe descartarse aquél que sea menos significativo con respecto a su entorno.
- El segundo parámetro compara el valor del i-ésimo máximo relativo con el i+1-ésimo máximo relativo y el i-ésimo mínimo relativo con el i+1-ésimo mínimo relativo.
- El tercer parámetro compara la proximidad del extremo relativo i-ésimo con respecto a su predecesor. Si toman valores similares, entonces debe descartarse aquél que sea menos significativo con respecto a su entorno.

Capítulo 4. Metodología del procesamiento de video.

4.1. Introducción.

La fase de procesamiento realiza una serie de operaciones a nivel global sobre toda la imagen que recibe en cada iteración.

A lo largo del desarrollo del proyecto se hicieron diferentes pruebas con algunas de las posibles metodologías candidatas, que se mostraban prometedoras. No obstante, para entender los motivos que llevan a elegir una metodología frente a otra hay que tener presente dos restricciones muy importantes, por un lado el contexto del video y por otro, la limitación de resolver la etapa en tiempo real.

El contexto del video corresponde a una grabación de una coreografía de danza [Ilustración 4-1]. En ella habrá un bailarín que irá desplazándose sobre un escenario mientras la cámara lo sigue, siendo posible que la grabación se produzcan efectos zoom-in o zoom-out. Además, frecuentemente se producen deficiencias serias en la iluminación, que varían con el tiempo en función de los diferentes focos de luz que se aplican, y parte de la indumentaria del bailarín (los pantalones) se confunde cromáticamente con el telón de fondo. La resolución del video es normal (640p x 480p) y la tasa de frames relativamente baja (25fps), lo que sin duda puede acarrear problemas en la coherencia geométrica cuando los movimientos sean muy rápidos o bruscos.

A favor, nos encontramos con que el fondo es geométrica y radiométricamente bastante simple, monocromo, caracterizado por un gran número de aristas pseudo-verticales, aunque la intensidad del brillo derivado de la irradiación lumínica de los focos afecta de forma heterogénea al telón de fondo y al suelo, eso sí, a menudo dentro de un rango.

Ilustración 4-1 - Ejemplo de imagen sin procesar (1)

Ilustración 4-2 - Ejemplo de imagen sin procesar (2)

4.2. Metodología del filtrado

El proceso metodológico empleado intenta beneficiarse de los contrastes radiométricos en la matriz de brillo de la imagen (value, en el sistema de representación HSV, y que a menudo nos referiremos a ella como la "*matriz de la imagen*") entre el fondo y el bailarín.

La secuencia de operaciones son las siguientes:

- Extraemos el canal de brillo (value) de la imagen según la representación HSV.
- Extraemos las aristas presentes en el frame, aplicando el detector de bordes Canny [véase capítulo C de Anexos].
- Realizamos una operación de dilatación sobre la imagen para unificar los minisegmentos resultantes del proceso anterior.
- Aplicamos una umbralización para ponderar positivamente las intensidades más claras en los píxeles de la imagen.
- Volvemos a hacer una dilatación para intentar cohesionar los segmentos que hayan podido separarse en el proceso anterior.

El primer paso en la secuencia de operaciones pretende simplificar los datos a estudiar. De los tres canales del modelo HSV, el que conserva mas información de los tres es siempre el de brillo, ya que tiene la propiedad de conservar toda la información geométrica de los elementos presentes en la imagen. Los otros dos canales ofrecen información complementaria que en este caso particular no compensa procesar.

Aplicando el algoritmo de Canny podemos extraer los bordes más importantes de la escena. Esto nos permite hacer una primera discriminación entre qué pixeles corresponden a zonas relevantes y

qué pixeles no. Desgraciadamente, dado que los pantalones tienen una gama cromática muy similar al telón de fondo, es muy común que esa región no pase el filtro, ya que los bordes presentan variaciones de intensidad insignificantes en la matriz de la imagen.

Ilustración 4-3 - Filtro Canny en color

Ilustración 4-4 - Filtro Canny en negativo

El proceso de dilatación permite unir pequeños segmentos resultantes de aplicar Canny, haciendo que la silueta del bailarín sea más apreciable. Adicionalmente, otros segmentos, como lineas de proyección, pueden estar presentes. En la siguiente figura se puede ver un ejemplo.

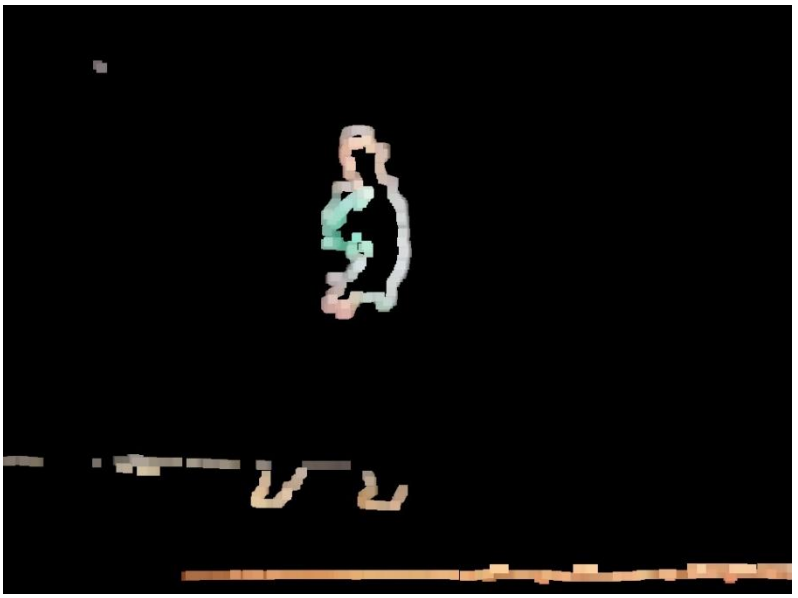


Ilustración 4-5 - Filtro Canny en color con dilatación

Ilustración 4-6 - Filtro Canny en negativo con dilatación

Una sub-etapa intermedia en el procesamiento consistía en aplicar una estrategia de rellenado sobre la matriz de la imagen resultante. No obstante, fue descartado por exceso de consumo de procesamiento y por no aportar demasiado de cara a la etapa de análisis. Se puede ver un ejemplo en la siguiente figura:

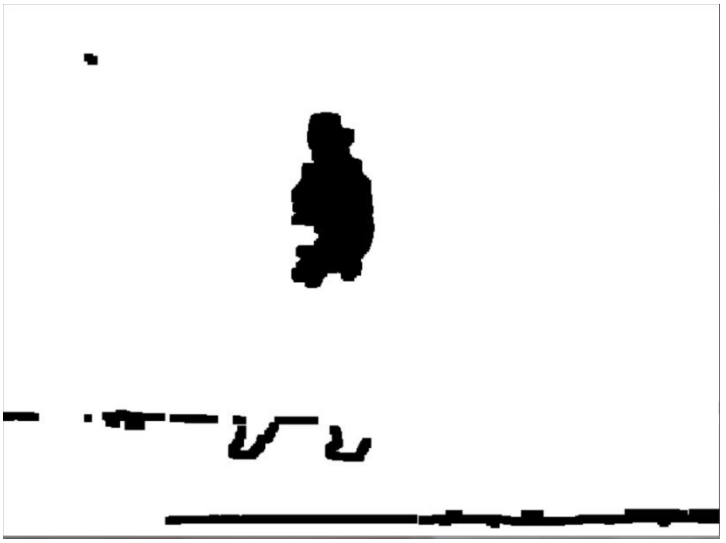


Ilustración 4-7 - Rellenado de poligonales cerradas

4.3. Puntos fuertes y puntos débiles del procesamiento

La estrategia de procesamiento realizada es el resultado de ponderar las características positivas y negativas de la imagen, ajustándose al tiempo y los recursos disponibles. En resumen, presenta las siguientes cualidades:

Positivas:

- Tiempo de procesamiento por frame razonable
- Buena tasa de extracción del fondo y de simplificación de los datos de entrada.
- Suaviza algunas inestabilidades en los contornos del individuo.

Negativas:

- Es específico para el contexto del video debido a que el resto de alternativas encontradas eran computacionalmente muy costosas o impracticables con contexto en cuestión (por ejemplo, uso de cámara estéreo). Este es uno de los motivos por el cual se ha realizado un esfuerzo especial en diseñar una solución que encapsule el procesamiento para una fácil sustitución en la estrategia de filtrado.
- Parte de la silueta del individuo no pasa el filtrado.
- Parte del fondo puede pasar el filtrado.

Capítulo 5. Metodología del análisis de video.

5.1. Introducción.

El análisis corresponde a la segunda fase metodológica tras el procesamiento. En él hay una primera sub-etapa que pretende acotar el área de búsqueda de información dentro de los datos de la matriz de la imagen filtrada, después hay una segunda sub-etapa centrada en el resumen de los datos de la imagen para un tratamiento posterior más sofisticado, seguido de una tercera sub-etapa de clasificación de los datos a priori, que concluye con la identificación y hallazgo de características relevantes para la identificación de la postura que adopta el cuerpo humano, lo que permitirá una representación de la misma.

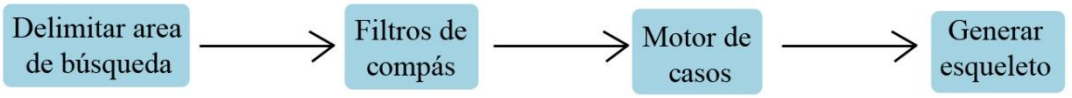


Ilustración 5-1 - Pasos en la metodología del análisis

En esta etapa es fundamental tener muy claro cuál va a ser el sistema de representación para la información. De ello dependerá tanto la metodología que se va a implementar como la calidad y claridad de los resultados que se van a obtener.

Es relativamente común el representar a la persona con la poligonal que delimita su cuerpo en la imagen para ver cómo cambian las concavidades y convexidades de la misma. Sin embargo nosotros hemos optado por ofrecer una alternativa más sugerente, que consiste en realizar una **pseudo-representación esqueletal de alto nivel semántico** del bailarín.

5.2. Localización del área que ocupa el bailarín en la imagen.

Con la imagen ya procesada en la etapa anterior, estamos en disposición de intentar localizar cual es el área que ocupa la persona. Para ello necesitamos diseñar un algoritmo que sea capaz de buscar por toda la matriz de la imagen filtrada y seleccionar el área completa en la que se encuentra la persona. Este algoritmo tiene que intentar aunar precisión y velocidad en tiempo de ejecución.

El método desarrollado es extremadamente sencillo e intuitivo:

- Partimos de una imagen filtrada en blanco y negro (figura 4-6, p.e.). Por convenio, entenderemos que la silueta de la persona y de todo aquello que ha pasado el filtro, corresponde a los pixeles de color negro.
- Asumimos como hipótesis que la imagen no ha sufrido transformaciones de rotación en torno al eje de la profundidad, o por lo menos dichas transformaciones no superan los 35º de inclinación en un sentido u otro, de tal manera que el suelo quede en las zonas más bajas dentro de la matriz de la imagen. Se trata de una hipótesis importante, ya que atenta contra una de las características deseables de las representaciones esqueletales (que el esqueleto sea invariante frente a transformaciones isométricas de la figura original). No obstante, dado que la mayoría lo normal es realizar este tipo de grabaciones orientadas de tal manera que el suelo quede abajo, por eficiencia en la búsqueda del área en la que se encuentra la persona, hemos decidido aceptar dicha hipótesis.

- Declaramos dos vectores, que denominaremos PNegrosFila y PNegrosColumna, con tantos elementos como filas y columnas haya respectivamente en la matriz de la imagen.
- Recorremos la matriz de la imagen pixel a pixel, y cada vez que encontramos un pixel negro, lo apuntamos (sumamos +1) en la posición i-esima de PNegrosFila, y en la posición j-esima de PNegrosColumna. Con ello obtenemos dos vectores que contienen el resultado del conteo de cuantos pixels negros hay en cada fila y en cada columna.
- Trabajamos con dichos vectores como si fuesen dos funciones discretas, y calculamos los extremos relativos de dichas funciones.
- Atendiendo a los diferentes extremos relativos encontradas en ambas funciones, dividimos la imagen en áreas rectangulares, que participarán en una competición como candidatas a ser elegidas el área que corresponde a la persona.
- Se descartan las zonas vacías o zonas que en iteraciones previas correspondían a la región del suelo.
- Se evalúan los candidatos en función de su densidad (cuanto ocupa), su distribución (ni excesivamente alargado a lo alto ni a lo ancho), que esté más centrado en la imagen (es decir, que aquellos bloques que estén más esquinados tendrán menos posibilidades que aquellos que estén más centrados), etc. ponderando cada criterio en función de su relevancia. Si se tiene información de iteraciones previas, el criterio de ponderar beneficiosamente a aquellos que están más centrados, es sustituido por aquellos que estén más próximos al área localizada en la iteración anterior.
- Se evalúa la posibilidad de que el área de la persona haya sido fragmentado en más de 1 bloque y en caso de que las regiones colindantes tengan también una suficiente probabilidad de ser áreas en la que se encuentra parte del individuo, se añaden al bloque ya seleccionado.

El resultado es un área rectangular en el que se encuentra la silueta del individuo. Con ello hemos conseguido:

- Evitar que en posteriores sub-fases analicemos algunas secciones de la imagen que hayan podido pasar con éxito el filtro sin pertenecer a la silueta del bailarín.
- Reducir el tiempo de ejecución en cálculos como los filtros de compás.

Ilustración 5-2 - Detección del área que ocupa el cuerpo humano

5.3. Localización del suelo.

Parte de los mismos principio que el sistema de localización de la persona, aunque los criterios de selección en la competición son obviamente diferentes. Los criterios a ponderar son:

- Solo se procesa el tercio inferior de la imagen por la hipótesis de que el suelo se encuentra en las zonas inferiores de la imagen.
- Intentamos localizar las líneas de intersección entre el plano vertical (telón) y el horizontal (tarima). Para ello ponderamos beneficiosamente regiones en los que PNegrosFila[i] tenga valores más elevados.
- Consideramos que son mejores candidatos aquellos que ocupan posiciones más altas en la matriz de la imagen, ya que pueden aparecer varias líneas de proyección. En caso de que haya información de iteraciones anteriores, se ponderará que el área nueva sea lo más parecida a la antigua.

También se tiene en cuenta la posibilidad de que el suelo no haya sido grabado con la cámara en el frame en cuestión, en cuyo caso el software trabajará adaptado a esta circunstancia.

Ilustración 5-3 - Localización del área del suelo fuera de cámara

Ilustración 5-4 - Localización del área del suelo

5.4. Filtros de compas.

Tras identificar el área en el que se encuentra la persona, se debe iniciar un proceso de simplificación de los datos filtrados. La idea consiste en construir una estructura auxiliar que

permita un acceso rápido y sencillo a los elementos mas característicos de la silueta, de tal manera que:

- Permita servir de resumen para identificar con relativa facilidad cual es el caso al que corresponde la distribución de datos.
- En la sub-etapa de generación del esqueleto podamos ahorrar tiempo de recorrido de la imagen.

Consiste en calcular las proyecciones de la silueta del cuerpo desde diferentes puntos de vista de la imagen. El proceso se realiza recorriendo la imagen en las direcciones deseadas dentro de la zona en la que se ha localizado al bailarín, parando y marcando cuando en el recorrido te encuentras con un pixel que pertenezca a la silueta del individuo. El resultado será un vector de puntos de proyección por cada proyección que se realiza. En concreto, nuestro proyecto trabaja con 5 proyecciones, dos horizontales, una cenital y dos diagonales. A continuación pueden verse algunos ejemplos de los resultados obtenidos:

Ilustración 5-5 - Filtro de compás de la proyección vertical superior

Ilustración 5-6 - Filtro de compás de la proyección horizontal. Barrido de izquierda a derecha

Ilustración 5-7 - Filtro de compás de la proyección horizontal. Barrido de derecha a izquierda

5.4.1 Consideraciones de los filtros de compás

Como puede intuirse, el algoritmo requiere trabajar sobre todo el área de la imagen en la que se ha localizado la persona, cuanto más grande sea y más definición tenga la imagen, más lento será este proceso. De hecho, es uno de los procesos que mas esfuerzo computacional requiere, debido al gran numero de accesos que hay que hacer sobre la matriz de la imagen y los recorridos a los que estamos obligados a realizar.

Es por tanto uno de los cuellos de botella de la aplicación, pero también uno de los pilares fundamentales sobre los que se sostiene la velocidad de procesamiento en operaciones posteriores, basada en una filosofía de "hacer un recorrido organizado en una fase previa para luego evitar tener que realizar numerosos recorridos mas caóticos para construir la representación esqueletal".

5.5. Motor de casos.

Originariamente se planteó la metodología de la presente representación esqueletal como un sistema que primero resumía los datos del frame y después, mediante algún tipo de algoritmo, generaba un esqueleto, siempre de la misma forma. La idea intuitiva, en resumen, consistía en construir el esqueleto en cascada, primero localizar la cabeza, debajo estaría el cuello, seguido los hombros, y conectado a estos, los brazos; debajo de los hombros estaría la columna, y más abajo, la cadera, y terminar llegando a las piernas. Sin embargo esta idea intuitiva era muy optimista, y no tenía en cuenta la posibilidad de que, por ejemplo, la cabeza estuviese oculta tras un brazo que el bailarín había levantado.

Poco a poco, se empezó a observar que era inviable desarrollar un algoritmo que buscase unas mismas características concretas en los resultados de los filtros de compás para realizar la representación esqueletal. Razón por la cual se decidió transformar la idea intuitiva en un sistema basado por casos.

La forma de proceder sería:

- Definir una colección programada de casos posibles
- Determinar las características que definen a cada caso
- Contrastarlas con los datos generados por los filtros de compas
- Determinar qué criterios se usarán para determinar el caso y ponderarlos para llegar a una única solución con la mayor tasa de acierto posible

5.5.1. Casos.

Dado que los elementos que van a definir el caso deben ser los brazos y la cabeza, dichos casos dependerán de una serie de configuraciones generales que pueden adoptar, siguiendo el estereotipo de las agujas de un reloj.

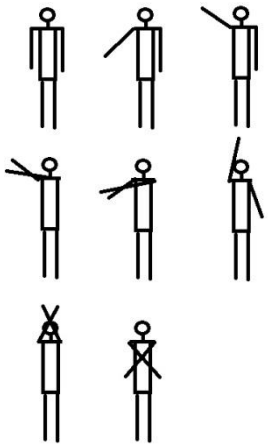


Ilustración 5-8 - Esquema de casos

Dichos casos se pueden clasificar en familias en función de sus rasgos particulares y de su complejidad.

Por un lado tenemos la denominada familia de casos 1-4, que corresponde a los casos en los que tenemos los dos brazos bajados, uno de los dos brazos levantado o los dos brazos levantados pero sin cruzamiento entre brazos o entre brazo y tronco.

Ilustración 5-9-Familia de casos 1-4

La segunda familia de casos corresponde a la existencia de un cruzamiento entre uno de los brazos con el tronco o entre los dos brazos. Sin embargo, con estos cruzamientos la cabeza no queda ocluida (o mejor dicho, no debería) por alguno de los brazos.


Ilustración 5-10-Familia de casos 5-6

Finalmente, la tercera familia de casos planteado para este trabajo en el que uno de los brazos está completamente levantado por encima de la cabeza, y el otro está más o menos levantado.

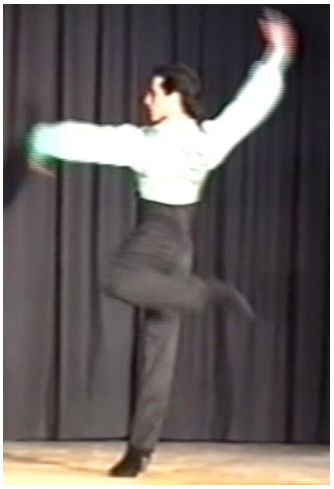



Ilustración 5-11-Familia de casos 7-8

5.5.2. Criterios.

Estos casos programados se identifican gracias al cálculo de extremos relativos de las funciones asociadas a los filtros de compás, de tal manera que se intenta crear un una correspondencia entre el numero y características de esos extremos relativos con los causantes de dichos extremos relativos, es decir, los dos brazos, la cabeza y el ruido. Dicho caso serviría de etiqueta al frame filtrado, para

que se le pueda aplicar el algoritmo específico para dicho caso, que permitirá generar el esqueleto del individuo.

5.5.3. Conclusiones acerca del motor de casos.

El motor de casos es un elemento clave dentro del sistema. Tiene un carácter inercial dada la retroalimentación entre iteraciones anteriores de frames analizados, ya que si en el frame i-ésimo ha sido clasificado como un "caso k", el frame i+1-ésimo tiene muchas probabilidades de pertenecer al mismo caso.

Ocasionalmente se producen eventos geométricos que pueden producirse de una manera más o menos gradual, que denotan que los datos están cambiando de estructura y que deben ser tratados como otro caso diferente. En ese proceso de cambio de datos se van deteriorando los resultados del algoritmo particular de generación del esqueleto. Razón por la cual sería deseable encontrar el punto de equilibrio exacto en el que la calidad del resultado generado para ese caso es inferior a la que podría realizarse con el tratamiento bajo el enfoque de otro caso. Sin embargo no disponemos de una herramienta online que nos permita contrastar los resultados obtenidos en la iteración i, con un caso ideal o esperado. Por ello, para identificar el caso al que pertenece el frame i-ésimo únicamente podemos fijarnos en los datos resultantes del cálculo de los filtros de compás, los resultados en iteraciones previas y una serie de reglas que ponderan todas estas características particulares para tomar la decisión de procesar de una forma o de otra.

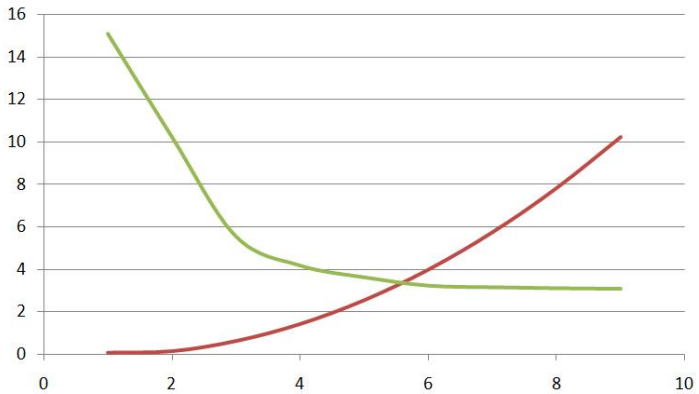


Ilustración 5-12 - Ilustración de relación entre calidad de resultados aplicando dos casos diferentes a lo largo del tiempo. No corresponde a un caso real.

Por otro lado, programar una gran variedad de casos permite que los algoritmos específicos que generan el esqueleto sean más sencillos y hacen que el resultado de éstos sea más preciso. Sin embargo, experimentalmente observamos que cuantos más casos programas, es más probable que en algún momento el motor de casos no identifique correctamente el caso, razón por la cual se intentó realizar una labor de agrupamiento de casos, haciendo que aumentase la tasa de aciertos del motor de casos.

5.6. Generación del esqueleto.

La idea intuitiva inicial para la generación del esqueleto consistía en aprovecharse del conocimiento a priori del cuerpo humano para identificar zonas características, y una vez localizadas esas zonas características, calcular el resto en función de los datos obtenidos.

En una situación ideal, esa zona característica debería corresponder a la cabeza. Una vez localizada la cabeza, los hombros deberían estar localizados muy cerca del cuello, y por tanto, de la cabeza. Conectado a los hombros, deberían estar los dos brazos, el cuello debería estar conectado con el resto de la columna vertebral, y éste con la cadera, que a su vez estaría conectado a las dos piernas.

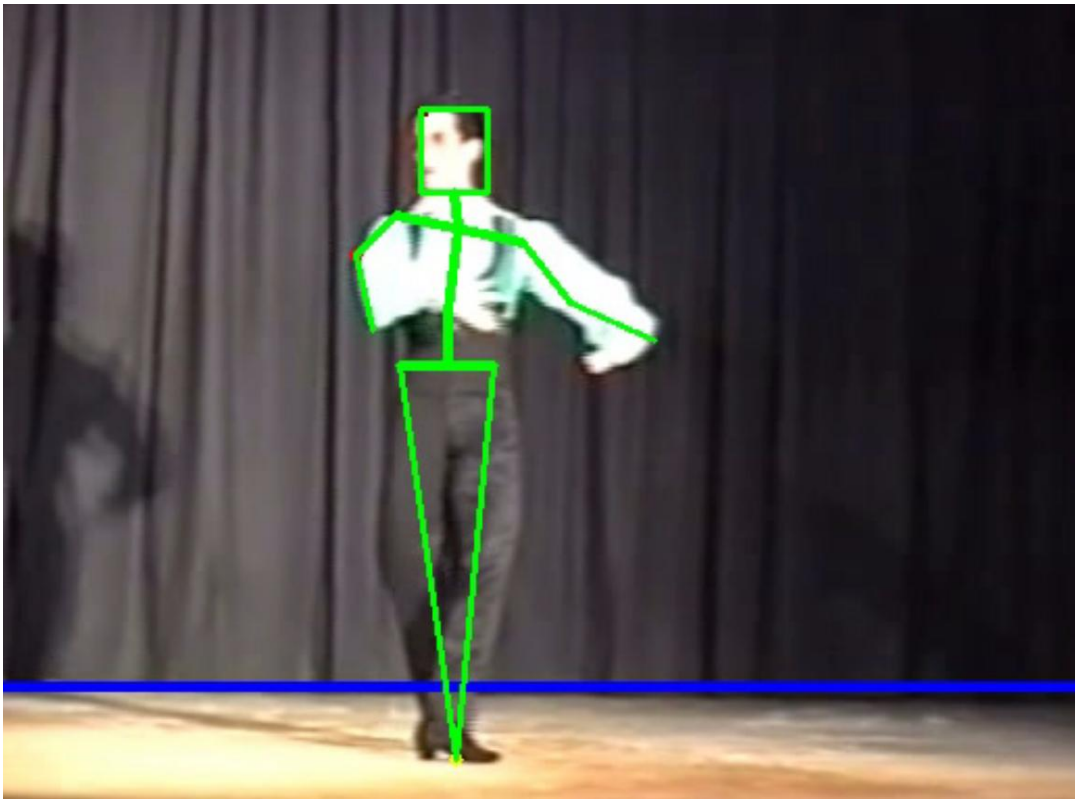


Ilustración 5-13 - Esqueleto

Sin embargo, el problema está en que la cabeza no tiene por qué ser el elemento más representativo en la silueta. Si bien es verdad que algunas veces alguno de los brazos puede quedar auto-ocluido por otra parte del cuerpo (por otro brazo o por el propio torso), también es verdad que la cabeza puede estar tapada por alguno de los brazos, o incluso no pasar el filtro debido a que el cabello del bailarín pueda coincidir con el color de fondo y dicho bailarín esté de espaldas. Por ello, se vio que era necesario dividir el problema en casos, y que las regiones más relevantes serían unas u otras en función del caso en particular, de tal manera que esas regiones permitirían, en el peor de los casos, calcular el resto de información del esqueleto de forma implícita.

En cualquier caso, el cuerpo del bailarín se calcula separando lo que hay de cintura para arriba de lo que hay de cintura hacia abajo, con el fin de que la información de los mismos se calcule de manera independiente, de tal manera que el cuerpo queda previamente dividido para su tratamiento en aproximadamente dos mitades.

Finalmente añadir que es muy importante remarcar el hecho de que el interior de la silueta que se obtiene del procesamiento del frame, no tienen por qué estar necesariamente rellenado, sino que puede contener zonas que han pasado el filtrado y zonas que no, lo que imposibilita recorrer la matriz de la imagen por el interior de la silueta.

5.6.1. Localizador de cabeza.

La cabeza corresponde a uno de los elementos clave para la generación del esqueleto junto con los brazos. No obstante, su cálculo depende mucho del caso particular en el que se necesite generar el esqueleto.

Para la familia de casos 1-4, el algoritmo procede de forma iterativa partiendo de la asignación de un extremo relativo de la proyección vertical (al que se ha identificado como un punto de la cabeza) al que llamaremos eje, partiendo también de un tamaño estándar para el área rectangular que debería ocupar la cabeza en función de las dimensiones de la silueta, y partiendo de dos pivotes elegidos en función de la configuración de extremos relativos en las funciones asociadas a los filtros de compás, e iterativamente se calculan los nuevos pivotes, aproximándose al eje, e intentando minimizar las diferencias con respecto a área rectangular precalculada en función de las dimensiones de la silueta, permitiendo que, incluso en caso de que el individuo esté de espaldas, el algoritmo pueda dar una solución aproximada donde cree que se encuentra la cabeza.

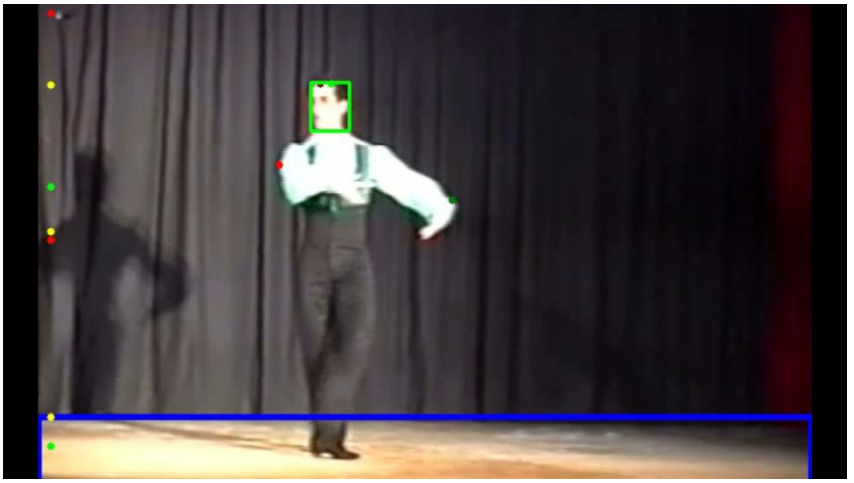


Ilustración 5-14 - Resultados del localizador de la cabeza para familia de casos 1-4

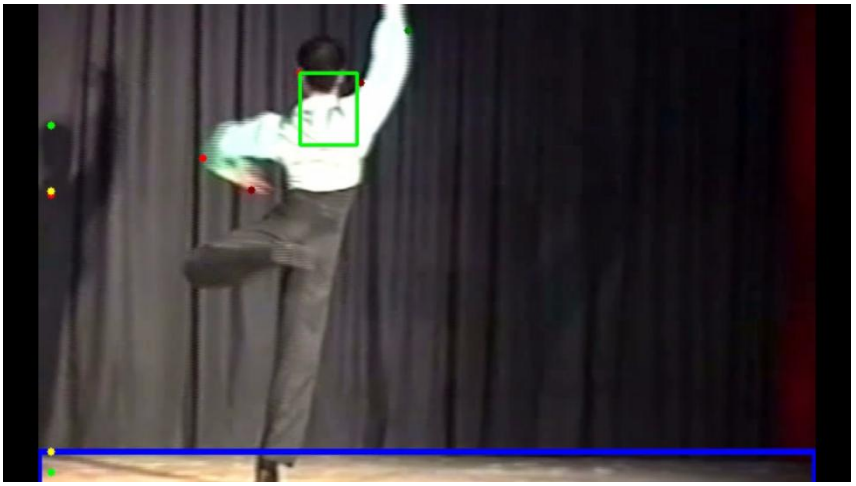


Ilustración 5-15 - Resultados del localizador de la cabeza de espaldas

Para la familia de casos 5-6 se opera ajustando un tamaño ideal del área que debería ocupar la cabeza, sobre un extremo relativo de la proyección vertical.

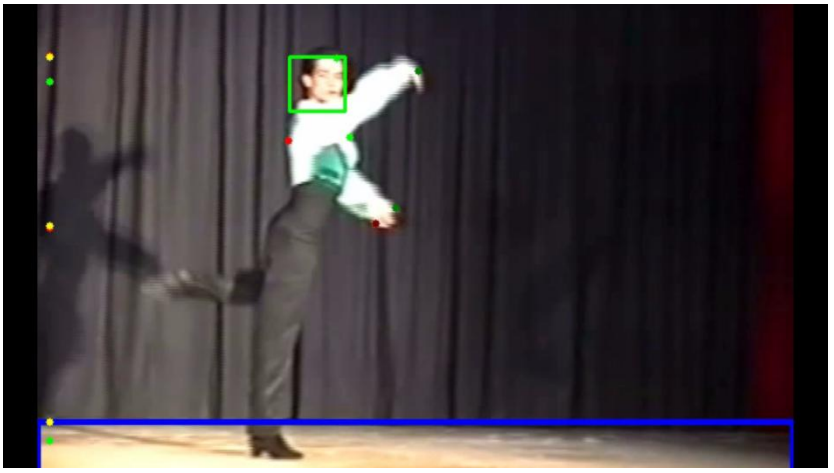


Ilustración 5-16 - Resultados del localizador de la cabeza para casos 5-6

Finalmente para la familia de casos 7-8, la cabeza se calcula directamente en función de la posición calculada de los hombros y del tamaño de la silueta que se ha generado en el filtrado.

5.6.2. Localización de brazos.

Los brazos corresponden a uno de los elementos clave para la generación del esqueleto junto con la cabeza. A menudo serán fáciles de localizar, pero también serán el principal causante de ambigüedad en la silueta.

El procedimiento general para generar su esqueleto consiste en:

- Identificar la zona en la que se encuentran las manos, en función de los extremos relativos en las funciones de proyecciones asociadas a los filtros de compás.
- Calcular la posición más probable de los hombros (opcionalmente en este paso puede ser necesario calcular la posición de la axila y la zona superior del hombro y calcular el punto medio).
- Asignar un hombro a cada mano.
- Localizar el codo.

Aunque en realidad el procedimiento específico para cada familia de algoritmos es bastante diferente, ya que los principales problemas son, determinar en qué proyección es mas "significativa" cada mano, distinguir la cabeza de una mano (ambigüedad entre la familia de casos 1-4 y la familia de casos 7-8) y sobre todo, estimar la posición de los hombros (y si están siendo la causa de la existencia de algún extremo relativo o no) y asignaros a cada mano, como se verá en el capítulo 6 dedicado a ambigüedades.

Ilustración 5-17 - Ejemplo gráfico del proceso de identificación de los brazos

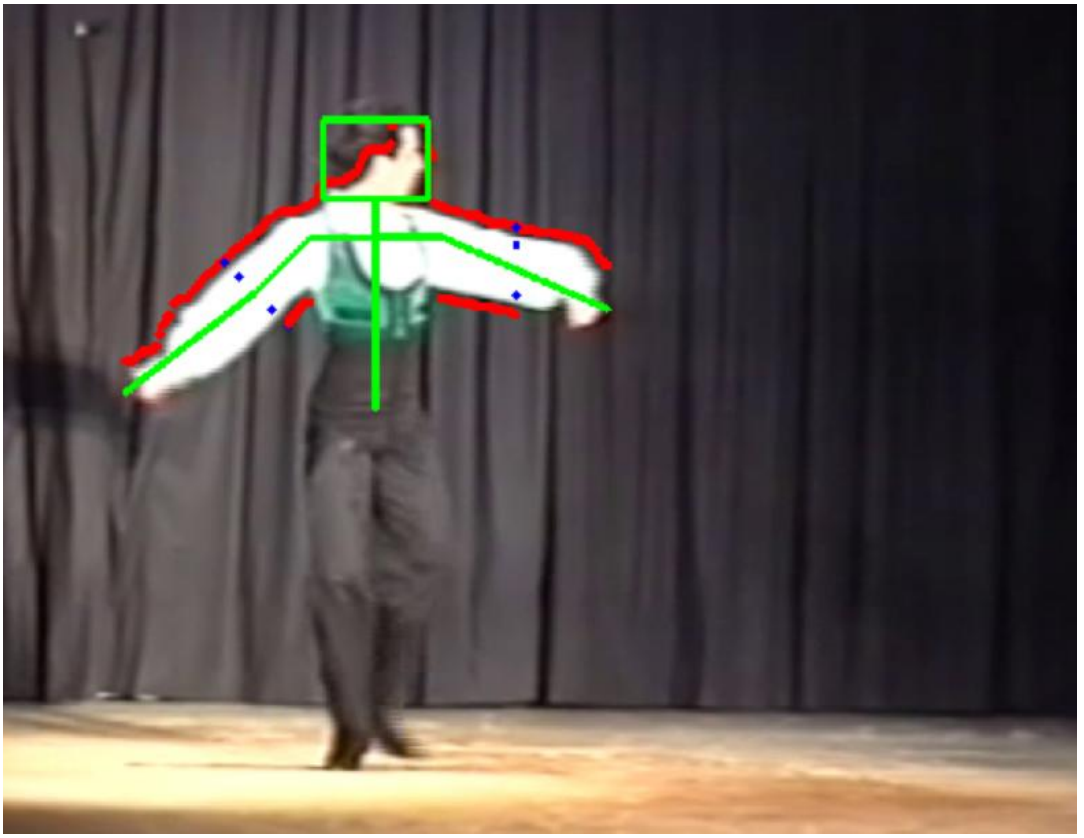


Ilustración 5-18 - Ejemplo de trazas utilizadas para la generación del esqueleto en familia de casos 1-4

5.6.3. Calculo de posición del cuello, el tronco y la cadera.

Siempre son calculados en función de los datos obtenidos con la localización de la cabeza y los brazos:

- El área que une los dos hombros define la clavícula
- El cuello se calcula uniendo el segmento de la clavícula con la cabeza
- La columna vertebral se calcula con un proceso de interpolación-extrapolación, usando como puntos de referencia, el cuello, las axilas (cuando se conoce esta información) y la estimación de dónde acaba el torso.
- Sabiendo el punto donde termina la columna vertebral se coloca una cadera de una dimensiones precalculadas en función del tamaño de la silueta, ya que debido a la radiometría del pantalón, la zona de la cadera no suele ser muy estable en la imagen filtrada.

5.6.4. Localización de piernas.

Se calculan únicamente partiendo de los datos de la posición de la cadera.

Dado que en el video el bailarín puede ir con unos pantalones cuyas características radiométricas son prácticamente idénticas a las del telón de fondo, y, por tanto, no es posible extraerlo del fondo en el procesamiento global del frame, para localizar las piernas, se han planteado varios métodos (técnicas de operadores integro-diferenciales de un proyecto de don Javier Finat y don Eduardo Cuesta).

La solución propuesta asume que no se puede obtener dato alguno de las piernas, y se centra en intentar encontrar candidatos para los pies, para luego conectarlos con la cadera. Los candidatos se generan con los siguientes criterios:

- Supone que se encuentran en el área de la imagen asociada al suelo (en caso de no haber suelo, asume que los pies no son visibles en la imagen).
- Los candidatos se les tendrá mejor en consideración cuanto mas próximos estén los candidatos al eje definido por la columna vertebral del bailarín.

Capítulo 6. Ambigüedades.

6.1. Introducción.

Entendemos por ambigüedad dentro de nuestro sistema, a un punto dentro de la aplicación de la metodología en el cual hay una decisión que tomar de entre dos o más posibilidades y no se dispone de información (o no se dispone de los medios para obtenerla) que permita discriminar de forma razonable a unos candidatos de otros para poder tomar una decisión razonable.

6.2. Ambigüedades encontradas

Hay tres tipos generales de ambigüedades que se han encontrado en este proyecto, las ambigüedades a nivel de motor de casos, a nivel de algoritmo (generación de esqueleto) y de singularidad, las cuales no son excluyentes.

- Las ambigüedades a nivel de algoritmo normalmente parten del problema de basar sus resultados únicamente en función de los contornos de la silueta y no de información que acontece en el interior de la silueta (ya que el interior de la silueta es a menudo excesivamente complejo como para plantear un recorrido en la matriz de la imagen que explote dicha información). Suelen hacer que se genere mal una pequeña parte del esqueleto, pero los resultados no suelen ser excesivamente distantes de la solución óptima.
- Las ambigüedades a nivel de motor de casos suelen nacer de la simplificación de la información. De producirse esta ambigüedad y de elegirse mal la asignación del caso, el resultado de generar el esqueleto suele distar bastante de una solución ideal.
- En cuanto a las ambigüedades de singularidad, que son un caso especial, el problema de tratar con estos casos nace del alineamiento de varias extremidades, dificultando unas veces identificar el caso, otras veces generando el esqueleto, y ocasionalmente ambas a la vez. Los resultados de generar el esqueleto cuando se materializa esta ambigüedad, al igual que con las ambigüedades a nivel de motor de casos, suelen ser bastante negativos e inestables.

Como ambigüedad a nivel de generación de esqueleto, el único caso detectado es el que se encuentra en la familia de casos 5-6, donde tenemos dos brazos estirados hacia un mismo lado. Dado que no somos capaces de seguir el recorrido completo de los brazos hasta los hombros, no podemos determinar qué mano se corresponde con qué hombro, lo que obliga a hacer una asignación al azar. No obstante, los resultados de hacer una asignación u otra afectan "mínimamente" al esqueleto resultante. Un ejemplo claro de este problema (combinado con una singularidad al estar los hombros cuasi-alineados) es el que se muestra a continuación:

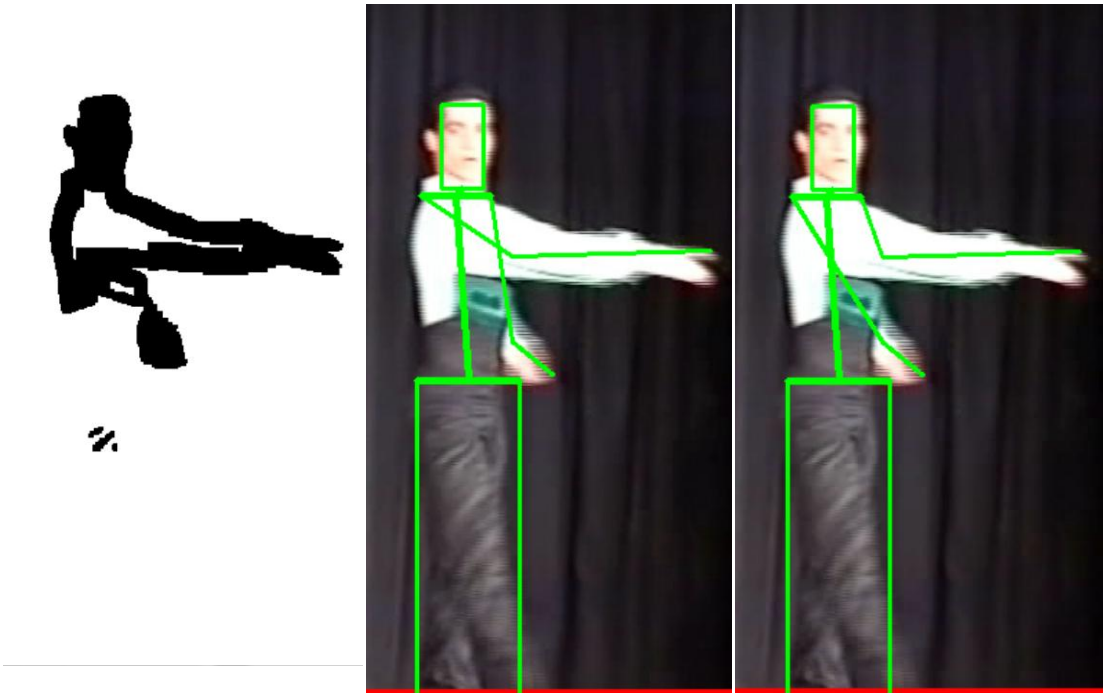


Ilustración 6-1 - Ambiguiedad a nivel de algoritmo

En lo que respecta a las ambigüedades de motor de caso, se presentan dos típicas, las ambigüedades entre las familias 1-4 con la 7-8 y las ambigüedades de la familia 7-8 consigo misma. Todas se basan en la posibilidad de que el número de extremos relativos en una o varias proyecciones sea el mismo y a raíz de esto se deban aplicar combinaciones de criterios que no aseguran completamente que vaya a acertase en la elección que se va a tomar. Este problema es atenuado al aplicar retroalimentación con iteraciones anteriores para intentar determinar la causa de los extremos relativos, pero en cualquier caso, no desaparece.

- **Caso de ambigüedad entre las familias 1-4 con la 7-8.** El fundamento de esta ambigüedad radica en las dificultades de diferenciar si el máximo relativo (cuando es uno sólo) en el filtro de compás de la proyección vertical corresponde a un brazo o a una cabeza. Para atenuar este problema, se ha utilizado la información del esqueleto generado en la iteración previa para intentar diferenciar un caso de otro; sin embargo es frecuente que se produzca una singularidad al cuasi-alinearse el brazo levantado y la cabeza, y por ello, además hay que recurrir al uso de reglas basadas en comparación de máximos-mínimos y concavidad en la región en cuestión.
- **Caso de ambigüedad entre la familia 7-8 consigo misma.** Consiste en la dificultad de diferenciar el caso 7 del caso 8 debido a que el brazo superior puede ocupar una posición libre siempre y cuando se mantenga como máximo relativo en la función asociada a la proyección vertical, y sobre todo si el otro brazo corresponde a un máximo relativo no excesivamente significativo. Normalmente se intentar dar una solución en función de cuán significativos sean los máximos relativos con respecto a los mínimos relativos de las proyecciones en cuestión, pero dicha ambigüedad se puede ver agravada en función de las singularidades que se puedan presentar.

Finalmente, respecto a las singularidades, hay tantos tipos de ambigüedades de singularidad como combinaciones de pseudo-alineamientos de ejes entre los diferentes elementos del cuerpo (hombros, brazos, cabeza) y en general tienen un comportamiento caprichoso y no hemos sido capaces de resolverlas.

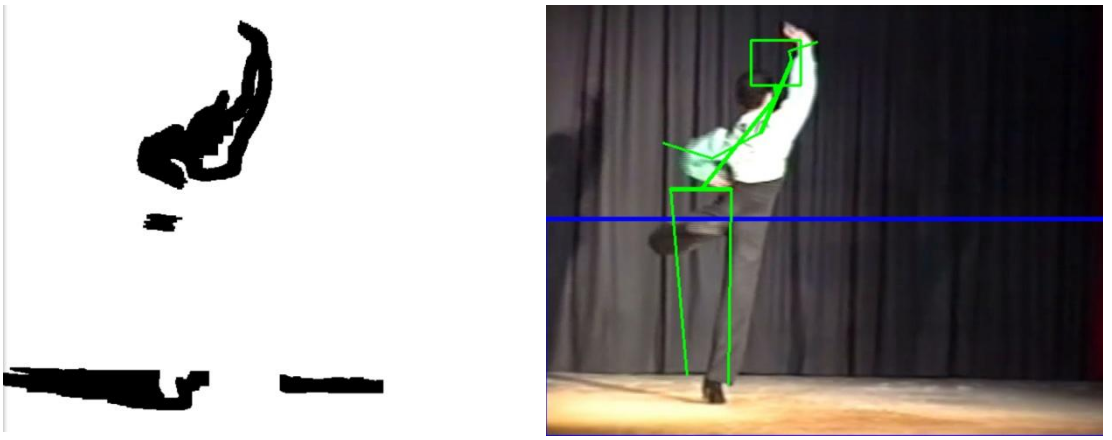


Ilustración 6-2-Singularidad mal resuelta

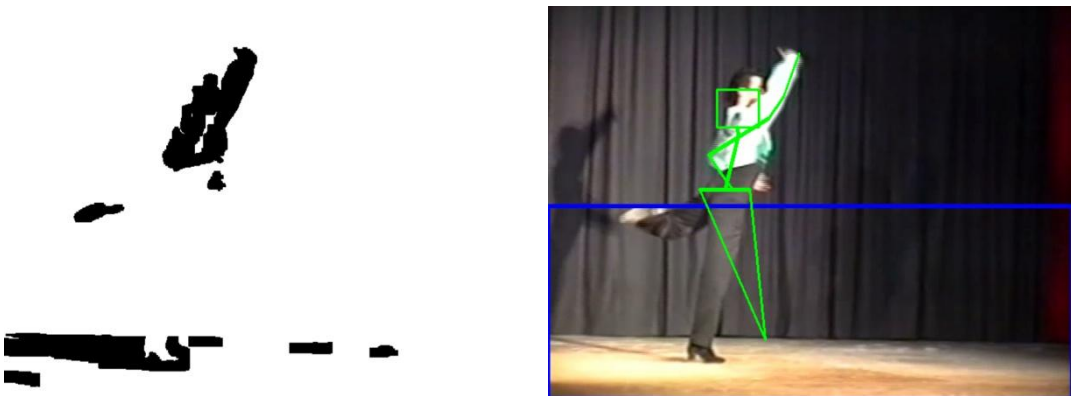


Ilustración 6-3-Singularidad bien resuelta

Bloque III: Planificación

Capítulo 7. Introducción a la planificación.

En este bloque, dedicado al plan de desarrollo software, se va a reflejar la gestión del proyecto software "AMH VIDA" correspondiente al trabajo final de grado de la carrera de Grado en Ingeniería Informática

Para su desarrollo se seguirá la metodología del Proceso Unificado en la que generalmente en la fase de Inicio no se realiza ninguna iteración por lo que dadas las características de nuestro proyecto, siguiendo con esta generalidad, no realizaremos tampoco iteraciones sucesivas en esta fase de Inicio.

7.1. Propósito

La planificación tiene como finalidad realizar un análisis exhaustivo de los recursos disponibles para el desarrollo del proyecto, es decir, evaluar su viabilidad, así como medir el tiempo y otros costes que se prevé que se requerirán para la realización de las diversas etapas y actividades del mismo, además de distribuir y gestionar otros tipos recursos que intervienen en el mismo.

Un proyecto no es viable, desde el punto de vista de los costes, si el coste especulativo es superior a los recursos de que se disponen para desarrollar el proyecto. Es muy recomendable prever un cierto margen de error entre lo que creemos que nos va a costar desarrollar el proyecto y lo que disponemos para realizarlo. El coste de un proyecto, en general, puede reflejarse en varios aspectos. En nuestro caso particular, el coste monetario en materiales se ha reducido a 0 gracias al uso de herramientas de libre distribución y de otros tipos de Software Libre, siendo el único coste económico, el que corresponde a la matriculación del TFG e impresión de los tomos de las memorias. Sin embargo el coste de tiempo requerido por el proyecto es un parámetro de mayor relevancia, digno de estudio.

El proceso de planificación no sólo sirve para prever los costes derivados del desarrollo del proyecto, sino también para comprobar que la evolución del mismo se desarrolla dentro de unos límites aceptables, lo que se traduce en este caso particular en que el tiempo de desarrollo de cada una de las etapas no se demora en exceso en comparación con las previsiones iniciales.

Es importante repartir temporalmente las diversas tareas en función de la duración estimada, de su dificultad, de la disposición de tiempo de cada de los recursos humanos del proyecto, procurando que las tareas sean alcanzables y amenas; en pocas palabras, nunca han de proponerse hitos excesivamente complejos o largos, pues éstos incrementan la frustración del trabajador que los realiza, reduciendo la productividad del mismo, así pues, los conocimientos del individuo, son un valioso recurso que será más útil para unas actividades que para otras.

Con todo ello, el resultado de la planificación debe mantener un equilibrio entre los costes económicos y temporales.

7.2. Alcance

Con el Plan de Desarrollo Software se pretende establecer la planificación general que se seguirá para el desarrollo del proyecto "AMH VIDA".

La planificación general queda definida en el anteproyecto y en este mismo bloque de este documento, quedando sujeta a los requisitos del SRS [capítulo 12].

Capítulo 8. Visión general.

8.1. Objetivo general.

Con este trabajo se pretende desarrollar un sistema capaz de analizar en tiempo real (RT) un video de danza de cámara móvil para reconocer el movimiento de un único individuo mientras baila.

8.2. Objetivos específicos.

- Identificar y aplicar una metodología que permita segmentar el fondo de la imagen (background) del bailarín, al mismo tiempo que permita seguir al bailarín por la escena. Todo ello debe realizarse en tiempo real.
- Identificar y aplicar una metodología que permita detectar características que definan el cuerpo humano en movimiento y que sean significativas para reconocer la pose del mismo.

8.3. Funcionalidades.

- Abrir un video
- Realizar una vista previa del video
- Analizar el video
- Pausar el video
- Detener el video

8.4. Suposiciones y restricciones

El proyecto se desarrolla basándonos en las siguientes suposiciones:

- El equipo de trabajo está formado por 1 personas que desempeña múltiples roles.

Las principales restricciones de la herramienta son las siguientes:

- **Tiempo:** El sistema deberá listo para ser presentado en las convocatorias académicas anunciadas por la UVA (Junio, Julio, Septiembre).
- **Documentación:** Se deberá incluir la documentación pertinente del software desarrollado, incluyendo todos los diagramas que expliquen el funcionamiento del mismo.
- **Control**: Se realizará un análisis y seguimiento de riesgos.
- **Material:** El software desarrollado deberá trabajar que ha facilitado el grupo Mobivap.
- **Requisitos:** Las funcionalidades de la aplicación deberán cubrir los requisito, que además deberán estar recogidos en el documento de requisitos.

8.5. Prerrequisitos.

- Marco conceptual para el desarrollo de la aplicación: Programación Orientada a Objetos (OOP).
- *Lenguaje:* El lenguaje a utilizar es C++ ó C\#, en los casos en que están disponibles aplicaciones más avanzadas.

- *Formatos:* La aplicación debe ser compatible con diferentes formatos de video.
- *Captura de datos:* La cámara de video es de bajo coste, tiene parámetros fijos (aunque desconocidos) y no calibrada para facilitar la extensión de resultados a diferentes infraestructuras.
- *Recursos computacionales:* La aplicación a desarrollar utilizará con preferencia la biblioteca OpenCV como marco computacional. Se requiere una familiaridad con las funcionalidades de esta biblioteca.

8.6. Entregables

De acuerdo con la filosofía de RUP, a lo largo del desarrollo del proyecto se irán generando una serie de artefactos, que irán siendo completados a lo largo de las diversas iteraciones del proyecto, marcando así los hitos del proyecto.

Los principales artefactos que se tendrán en cuenta en el proyecto son:

1) Anteproyecto

Da una visión general del proyecto. Define los principales objetivos del proyecto, planificación general y recursos del proyecto.

2) Modelo de Casos de Uso del Negocio

Es un modelo de las funciones de negocio vistas desde la perspectiva de los actores externos (Agentes de registro, solicitantes finales, otros sistemas etc.). Permite situar al sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito. Este modelo se representa con un Diagrama de Casos de Uso usando estereotipos específicos para este modelo.

3) Modelo de Objetos del Negocio

Es un modelo que describe la realización de cada caso de uso del negocio, estableciendo los actores internos, la información que en términos generales manipulan y los flujos de trabajo (workflows) asociados al caso de uso del negocio. Para la representación de este modelo se utilizan Diagramas de Colaboración (para mostrar actores externos, internos y las entidades (información) que manipulan, un Diagrama de Clases para mostrar gráficamente las entidades del sistema y sus relaciones, y Diagramas de Actividad para mostrar los flujos de trabajo.

4) Glosario

Es un documento que define los principales términos usados en el proyecto. Permite establecer una terminología consensuada.

5) Modelo de Casos de Uso

El modelo de Casos de Uso presenta las funciones del sistema y los actores que hacen uso de ellas. Se representa mediante Diagramas de Casos de Uso.

6) Especificaciones de Casos de Uso

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de

documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

7) Modelo de Análisis y Diseño

Este modelo establece la realización de los casos de uso en clases y pasando desde una representación en términos de análisis (sin incluir aspectos de implementación) hacia una de diseño (incluyendo una orientación hacia el entorno de implementación), de acuerdo al avance del proyecto.

8) Casos de Prueba

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados. Estos casos de prueba son aplicados como pruebas de regresión en cada iteración. Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones para realizar la prueba.

9) Plan de Iteración

Es un conjunto de actividades y tareas ordenadas temporalmente, con recursos asignados, dependencias entre ellas. Se realiza para cada iteración, y para todas las fases, incluido en este documento.

10) Seguimiento de Iteración

Este documento incluye le evaluación de los resultados de cada iteración, el grado en el cual se han conseguido los objetivos de la iteración, las lecciones aprendidas y los cambios a ser realizados.

11) Lista de Riesgos

Este documento incluye una lista de los riesgos conocidos y vigentes en el proyecto, ordenados en orden decreciente de importancia y con acciones específicas de contingencia o para su mitigación, incluido en el presente documento.

12) Material de Apoyo al Usuario Final

Corresponde a un conjunto de documentos y facilidades de uso del sistema, incluyendo: Guías del Usuario, Guías de Operación, Guías de Mantenimiento y Sistema de Ayuda en Línea.

13) Producto

El producto, desarrollado de manera intensiva a partir de la primera iteración de la fase de Construcción es desarrollado incremental e iterativamente, obteniéndose una nueva release al final de cada iteración.

Alguno de estos artefactos se generarán a partir de la fase de Construcción, con lo cual se han incluido aquí sólo para dar una visión global de todos los artefactos que se generarán en el proceso de desarrollo.

Capítulo 9. Organización del proyecto.

9.1. Estructura de la organización

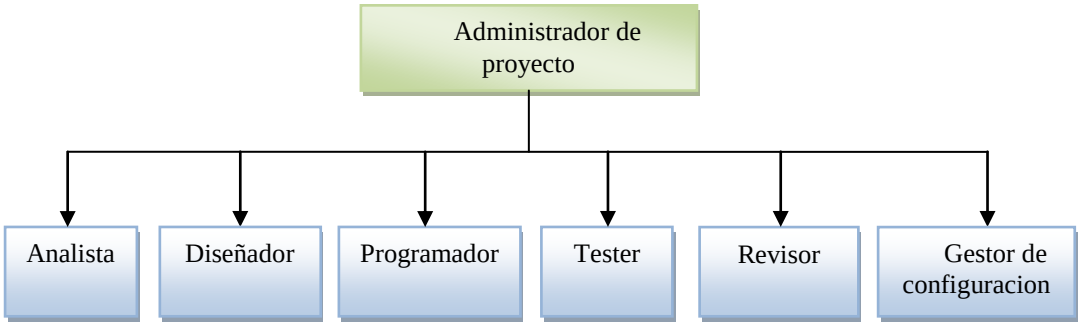
Con motivo del escaso número de trabajadores en la plantilla del proyecto, todos los roles deberán ser desempeñados por el único trabajador encargado de la realización del proyecto.

9.2. Roles y responsabilidades

Atendiendo a la naturaleza de las distintas actividades que deben ser realizadas en el proyecto, la plantilla de trabajadores deberán adoptar los diferentes roles que se enuncian a continuación:

Rol	Responsabilidad
Administrador de proyecto	El jefe de tarea asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. El jefe de proyecto también establece un conjunto de prácticas que aseguran la integridad y calidad de los artefactos del proyecto. Además, el jefe de proyecto se encargará de supervisar el establecimiento de la arquitectura del sistema. Gestión de riesgos. Planificación y control del proyecto.
Analista	Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos.
Diseñador	Diseño del sistema genérico y diseño de prototipos
Programador	Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario
Tester	Gestión de requisitos, gestión de configuración y cambios, elaboración del modelo de datos, preparación de las pruebas funcionales, elaboración de la documentación. Elaborar modelos de implementación y despliegue.
Revisor	Controla la calidad del software desarrollado y del proceso de desarrollo.
Gestor de configuraciones	Apoya a la actividad de desarrollo de software dotándola de un entorno de trabajo para construir y probar los componentes del sistema. Garantiza la constante disponibilidad de una versión estable de cada artefacto del proceso para todos los miembros del equipo. Gestiona el entorno de control de configuraciones para facilitar las actividades de revisión, cambio y seguimiento. Es también el responsable de redactar el Plan de Gestión de Configuraciones, de gestionar e informar de las Peticiones de Cambio.

Tabla 9-1 - Roles y responsabilidades



9.3. Interfaces Externas

El equipo de desarrollo interactuará con las siguientes entidades:

- Los tutores de proyecto, don Javier Finat Codes y don Valentín Cardeñoso Payo, que actuarán como clientes del producto, definiendo el ámbito del proyecto y los requistos del mismo.

9.4. Recursos hardware

- Ordenador personal

9.5. Recursos software

Durante esta fase se tendrán en cuenta los siguientes recursos software:

- OpenCV
- Qt
- Visual Studio 2008
- Microsoft Office 2008
- StarUml
- REM

9.6. Recursos humanos

- Alumno:
  - José Luis Martínez Jiménez
- Tutores:
  - Don Javier Finat Codes
  - Don Valentín Cardeñoso Payo
- Personal del Laboratorio Mobivap:
  - Don Rubén Martínez García
  - Don Francisco Delgado del Hoyo

Capítulo 10. Proceso de gestión.

10.1. Resumen

El proceso de gestión de este proyecto pretende controlar el desarrollo del proyecto, intentando minimizar el coste de desarrollo del mismo. Dicho coste se materializa, en este proyecto, en una componente económica (dinero invertido en la matriculación del TFG) y en el esfuerzo humano del grupo de trabajo encargado de desarrollar el proyecto.

Dado que la matriculación del TFG debe hacerse a priori, el coste económico a lo largo del curso en el que se está matriculado es constante. No obstante si el proyecto no alcanza los niveles adecuados de completitud de acuerdo con la funcionalidad requerida y éste no puede ser presentado dentro de una de las convocatorias en las que has sido matriculado, entonces los costes económicos ascienden debido a que se debe matricular otra vez en el TFG, manteniéndose constantes durante ese nuevo curso. Además, teniendo en cuenta que las presentaciones de TFG sólo pueden ser presentadas en convocatorias de Junio, Julio y excepcionalmente en Septiembre de 2011, no presentar en una de las fechas de ese curso, hace que el tiempo requerido para presentar en el siguiente curso ascienda a un año adicional.

Teniendo en cuenta, que la matriculación del TFG en 2011 ya había sido realizada, y que las convocatorias oficiales para el curso 2010-2011 eran Junio, Julio y Septiembre, se decidió realizar una planificación que pretende seguir un proceso de desarrollo software basado en RUP, con cuatro fases (Inicio, Elaboración, Construcción y Transición) y total de 6 iteraciones, como se muestra en en la [Tabla 10-1] de resumen. Con ella se pretendía realizar un trabajo intensivo de esfuerzo incremental, con el fin de que el proyecto estuviese acabado para la convocatoria de Septiembre de 2011, evitando así una nueva matriculación para otro curso, con los correspondientes costes adicionales que de ello se derivarían.

Etapa	Inicio	Iteraciones	Duración
Inicio	19/05/2011	0	11 días
Elaboración	03/06/2011	2	22 días
Construcción	05/07/2011	3	33 días
Transición	19/08/2011	1	9 días

Tabla 10-1 - Duración general estimada

Se estimaba que el esfuerzo temporal estaría entre las 300 y las 400 horas. Sin embargo, finalizando la fase de elaboración, se observó que la metodología que había que desarrollar para realizar el proyecto era más laboriosa de lo que previamente se había planteado, y el rendimiento de trabajo durante la fase de elaboración no era el suficiente como para completar el proyecto para Septiembre de 2011, por ello, se asumió la necesidad de ampliar el desarrollo del proyecto al curso 2011-2012.

Etapa	Inicio	Iteraciones	Duración
Inicio	19/05/2011	0	11 días
Elaboración	03/06/2011	2	44 días
Construcción	05/10/2011	3	128 días
Transición	02/05/2011	1	23 días

Tabla 10-2 - Duración general del proyecto

10.2. Gestión de requisitos

Los requisitos del sistema serán capturados en el documento SRS.pfd (especificación de requisitos software), además de ser plasmados en este mismo documento, en el [capítulo 12].

10.3. Control de calidad

Al final de cada fase se realizara una revisión formal de la situación del proyecto.

10.4. Análisis y gestión de riesgos

En este documento se realiza un listado de los riesgos encontrados y se dan una serie de pautas para gestionarlos en caso de que sucedan.

"*Risk in itself is not bad; risk is essential to progress, and failure is often a key part of learning. But we must learn to balancethe possible negative consequences of risk against the potential benefits of its associated opportunity."* [Sco92]

Criterios de clasificación:

- **Probabilidad:** este parámetro se mide en una escala de 0-100 % siendo 0% la probabilidad mínima y 100% la probabilidad máxima. Este indicador muestra el grado de certidumbre de que ocurra un determinado suceso.
- **Descripción:** Descripción detallada del riesgo identificado.
- **Impacto:**
  - C - Crítico
  - A - Alto
  - M - Medio
  - B - Bajo

- **Fase:** Etapa del desarrollo en la que la probabilidad del riesgo es máxima.
- **Plan de contingencia:** Solución aportada para la gestión del riesgo.

Listado d riesgos:

El listado de riesgos que puede verse a continuación:

<RSK-001> Demoras por falta de experiencia en tecnologías a usar					
Probabilidad	10%	Impacto	C	Fase	Construcción
Descripción	Algunas librerías utilizadas en este proyecto requieren un especial conocimiento del funcionamiento de las mismas. Es por ello que dominar su uso en un tiempo razonable es esencial para el desarrollo del proyecto y encontrar algún inconveniente en su uso (incompatibilidades, funcionamiento indebido o no esperado) puede parar el ritmo del desarrollo del proyecto				
Indicador	Los principales indicadores se manifiestan cuando el riesgo se materializa, no antes. Los citamos a continuación: • Demasiado tiempo gastado buscando información en foros y documentación oficial de formas de uso de tecnología.				
Plan de acción/contingencia	Plan acción: • Ampliar el tiempo dedicado al aprendizaje de las tecnologías, recurriendo a horas extra. El tiempo se prorrogará en 3 días, y en caso de que no sea suficiente, se irá ampliando el margen. • Si el tiempo consumido excede el margen de adjudicado, se pedirá ayuda técnica o consejo a compañeros y otros recursos humanos relacionados con el proyecto.				
Observaciones	Dado que intentar medir cuánto ha aprendido un trabajado es una tarea bastante difícil, se ha optado por no realizar ningún tipo de prueba para tomar dicha medición. El impacto es C (critico) debido a que la materialización de este riesgo provoca un parón en el desarrollo del proyecto La probabilidad de aparición del riesgo es sólo del 10% porque ya se tiene algo de experiencia con las principales tecnologías a usar.				

Tabla 10-3 - <RSK-001>

<RSK-002> Pérdida o corrupción de ficheros del proyecto					
Probabilidad	1%	Impacto	M	Fase	Cualquiera
Descripción	Algún artefacto generado durante el proyecto es eliminado.				
Indicador	Numero de ficheros a los que no se puede acceder. Umbral: >0				
Plan de acción/contingencia	Plan de reducción de impacto: • Copia digital programada en dos discos duros externos al acabar. • Copia automática en dropbox. Plan de contingencia: • Cargar copia de seguridad				
Observaciones	Este riesgo ha sido tiene una probabilidad mínima debido a los planes de prevención de los mismos, aunque siempre es importante tenerlo en cuenta.				

Tabla 10-4 - <RSK-002>

<RSK-003> Inhabilitación del equipo personal de desarrollo					
Probabilidad	10%	Impacto	M	Fase	Cualquiera
Descripción	El equipo personal de desarrollo queda inoperativo.				
Indicador	Numero de servicios críticos del ordenador a los que el usuario no tiene acceso (p.e. el propio sistema operativo no arranca, perfil corrupto). Umbral: >0				
Plan de acción/contingencia	Plan de prevención: • Realizar una revisión y mantenimiento del equipo principal cada 2 meses (Vaio Care). Plan de contingencia: • Usar otro ordenador personal (ordenador de reserva). Requiere instalación del software apropiado (coste temporal estimado por puesta a punto: 1-2 día). • Modificar la planificación para asignar tareas de documentación que no requieran de instalación de software específico y usar los ordenadores de la facultad. • Llamar al servicio técnico en caso de grave daño del equipo (coste temporal estimado (si no se dispone de ordenador de reserva): 2 semanas)				
Observaciones					

Tabla 10-5 - <RSK-003>

<RSK-004> Enfermedad/Accidente/Indisposiciones					
Probabilidad	20%	Impacto	M	Fase	Cualquiera
Descripción	El trabajador cae enfermo, padece de estrés o simplemente acontece algún tipo de evento que le impide continuar de manera temporal con su labor en este proyecto (enfermedad o accidente grave de un familiar).				
Indicador	Percepción de cansancio/fatiga/malestar...				
Plan de acción/contingencia	Plan de prevención: • Descansar cada 3 horas de trabajo • Dieta sana y equilibrada No hay plan de contingencia debido a que sólo se dispone de un miembro de trabajo.				
Observaciones	Los indicadores de este riesgo seguramente no sean suficientes para tomar medidas de prevención a tiempo, y su monitorización puede ser demasiado costosa como para si quiera plantear una métrica razonable. Se recomienda seguir constantemente el plan de prevención.				

Tabla 10-6 - <RSK-004>

<RSK-005> Retraso en concluir alguna fase o iteración del proyecto.					
Probabilidad	60%	Impacto	A	Fase	Cualquiera
Descripción	La cantidad de esfuerzo requerido para acabar a tiempo el proyecto puede ser superior al número de horas/hombre disponibles.				
Indicador	Días de retraso con respecto a lo planificado Umbral de aviso: 1 semana Umbral crítico: 2 semanas				
Plan de acción/contingencia	Ajustarse en la medida de lo posible al plan de cada fase, con el fin de evitar que los trabajadores se desmoralicen al ver que las tareas se aglutinan. Mantener una actitud positiva. En caso de que se supere el umbral de aviso, se redistribuirán la organización tareas tras identificar los motivos del retraso. En caso de que se produzca un retraso crítico, se deberá incrementar el número de horas extra de trabajo, hasta que se suprima el retraso.				
Observaciones					

Tabla 10-7 - <RSK-005>

<RSK-006> Error en la captura de requisitos.					
Probabilidad	5%	Impacto	M	Fase	Inicio Elaboración Construcción
Descripción	Es posible que no se entiendan correctamente los requisitos del proyecto				
Indicador	Nº de críticas del cliente de las entregas que se le han mostrado				
Plan de acción/contingencia	Plan de prevención: • Realizar reuniones con el cliente con la mayor frecuencia posible • Escuchar atentamente las sugerencias del cliente				
Observaciones					

Tabla 10-8 - <RSK-006>

<RSK-007> Errores/Problemas en software de terceros.					
Probabilidad	5%	Impacto	M	Fase	Cualquiera
Descripción	Aparecen errores en la implementación del sistema debido al software empleado para el desarrollo del mismo (servidor web, sistema gestor de bases de datos, librerías utilizadas, etc.). Problemas de comunicación/compatibilidad con software construido por otros.				
Indicador	Origen de un error en la implementación del sistema.				
Plan de acción/contingencia	Plan de prevención: • Se estudiarán los elementos que van a componer el sistema para detectar posibles incompatibilidades o errores. En caso necesario se valorará la posibilidad de cambiar el software utilizado por una versión distinta o un software equivalente. • Utilizar versiones finales.				
Observaciones	Se ha considerado que todas las fases pueden verse afectadas por este riesgo porque desde la etapa de inicio ha planteado trabajar con tecnologías, en concreto, en la fase de inicio se probará Symfony, XAMPP, etc. Además este riesgo también incluye defectos en herramientas CASE.				

Tabla 10-9 - <RSK-007>

<RSK-008>Deficiencias en el diseño.					
Probabilidad	10%	Impacto	A	Fase	Elaboración Construcción
Descripción	El diseño no es implementable, no cumple con los requisitos, etc				
Indicador	Grado de disconformidad del cliente.				
Plan de acción/contingencia	Plan de prevención: • Realizar reuniones con el cliente lo antes posible para validar el diseño. Plan de contingencia: • En caso de detectarse defectos, repararlos inmediatamente, en vez de demorar su reparación.				
Observaciones	Indicador subjetivo, fruto de las reuniones con el cliente.				

Tabla 10-10- <RSK-008>

10.5. Planificación y seguimiento del proyecto

El número de iteraciones planeado para cada fase depende, básicamente de la complejidad del sistema propuesto. Dadas las características del presente proyecto se establece el siguiente plan de iteraciones:

- **Fase de inicio:** Define el ámbito del sistema. En nuestro caso particular, no tendremos ninguna iteración "real" en esta fase del proyecto.
- **Fase de elaboración:** dos iteraciones, la primera para esbozar la arquitectura y la segunda para completar la línea base de la arquitectura.
- **Fase de construcción:** tres iteraciones, para asegurar que los incrementos resultantes funcionan satisfactoriamente.
- **Fase de transición:** una única iteración.

Fase	Iteración	Descripción
Inicio	Preliminar	Definir el modelo de negocio, los productos de requerimientos y el plan de desarrollo de software.
Elaboración	E1 - Desarrollo del prototipo de la arquitectura	Análisis para todos los casos de uso del Release 1.0
	E2 - Desarrollo del prototipo de la arquitectura	Análisis y Diseño de todos los casos de uso restantes del Release 1.0 y 2 .0
Construcción	C1 – Desarrollo del Release 0.5 Pre-Alfa	Implementar y probar todos los casos de uso para liberar la versión Pre-Alfa
	C1 – Desarrollo del Release 1.0 Alfa	Implementar y probar todos los casos de uso para liberar la versión Alfa
	C3 – Desarrollo del Release 2.0 Beta	Diseño, implementación y prueba de los casos de uso del R2. Incorporar nuevos requerimientos y defectos de R1. Desarrollar el Release 2.0 del sistema.
Transición	Final	Pruebas finales, distribución e instalación del Release 2.0.

Tabla 10-11 - Distribución general de actividades en cada fase

Duración esperada del proyecto:

Tiempo esperado	Inicio	Elaboración	Construcción	Transición
Porcentaje	14%	30%	44%	12%
Días	11 días	22 días	33 días	9 días

Tabla 10-12 - Proporción del coste temporal (en días) por fase

El esfuerzo esperado asociado a cada día, en función de la etapa del desarrollo y del plan previsto, siguiendo la tabla que se muestra a continuación.

Fase	Número de iteraciones	Promedio horas/hombre por día	Total horas/fase
Inicio	1	3 horas/hombre	33
Elaboración	2	4 horas/hombre	88
Construcción	2	8 horas/hombre	264
Transición	1	3 horas/hombre	27

Tabla 10-13 - Esfuerzo en horas/hombre por fase

En total se estima que el proyecto debería requerir aproximadamente un total de **416 horas** de desarrollo.

A continuación se muestra un resumen de la planificación y del seguimiento del proyecto, que sintetiza el contenido de los correspondientes documentos de Planificación y Seguimiento que acompañan a esta memoria en los que se detalla con más profundidad la evolución del proyecto, horas invertidas al día en cada tarea, etc.

10.5.1. Plan fase de inicio

	Nombre de tarea	Duración	Comienzo	Fin
1	Inicio	0 días	jue 19/05/11	jue 19/05/11
2	**Fase Inicio**	**11 días?**	**jue 19/05/11**	**jue 02/06/11**
3	Realizar planificación general	1 día?	jue 19/05/11	jue 19/05/11
4	Visión del proyecto	2 días	vie 20/05/11	lun 23/05/11
5	Estudio de posibles metodologías	6 días	lun 23/05/11	lun 30/05/11
6	Análisis de riesgos	1 día?	mar 24/05/11	mar 24/05/11
7	Inicio realización SRS	3 días	mié 25/05/11	vie 27/05/11
8	Detección casos de uso del sistema	2 días	lun 30/05/11	mar 31/05/11
9	Realizar anteproyecto	3 días	mar 31/05/11	jue 02/06/11
10	Planificación fase elaboración + seguimiento fase inicio	1 día?	jue 02/06/11	jue 02/06/11
11	Documento casos de uso versión inicial	0 días	vie 27/05/11	vie 27/05/11
12	Hito principal: Anteproyecto	0 días	jue 02/06/11	jue 02/06/11

Ilustración 10-1

Ilustración 10-2 - Plan fase Inicio

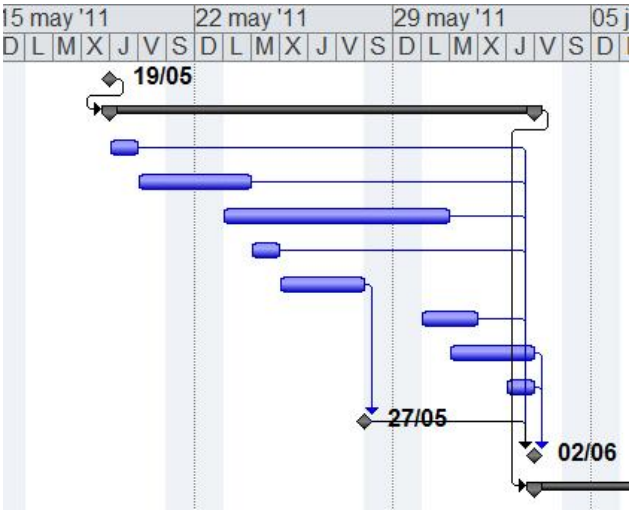


Ilustración 10-3 - Diagrama de Gantt en fase de Inicio

10.5.2. Fase Inicio: Seguimiento fase 1

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Inicio	0 días	jue 19/05/11	jue 19/05/11	
Fase Inicio	**11 días?**	**jue 19/05/11**	**jue 02/06/11**	**1**
Realizar planificación general	1 día?	jue 19/05/11	jue 19/05/11	
Visión del proyecto	1 día	jue 19/05/11	jue 19/05/11	
Estudio de posibles metodologías	6 días	vie 20/05/11	vie 27/05/11	
Análisis de riesgos	1 día?	mar 24/05/11	mar 24/05/11	
Inicio realización SRS	2 días	mié 25/05/11	jue 26/05/11	
Detección de casos de uso del sistema	1 día	lun 30/05/11	lun 30/05/11	
Realizar anteproyecto	3 días	mar 31/05/11	jue 02/06/11	
Planificación fase elaboración + seguimiento fase inicio	1 día?	jue 02/06/11	jue 02/06/11	
Documento casos de uso versión inicial	0 días	jue 26/05/11	jue 26/05/11	7
Hito principal: Anteproyecto	0 días	jue 02/06/11	jue 02/06/11	9;3;4;5;8;6;10

Ilustración 10-4 - Seguimiento en fase de Inicio

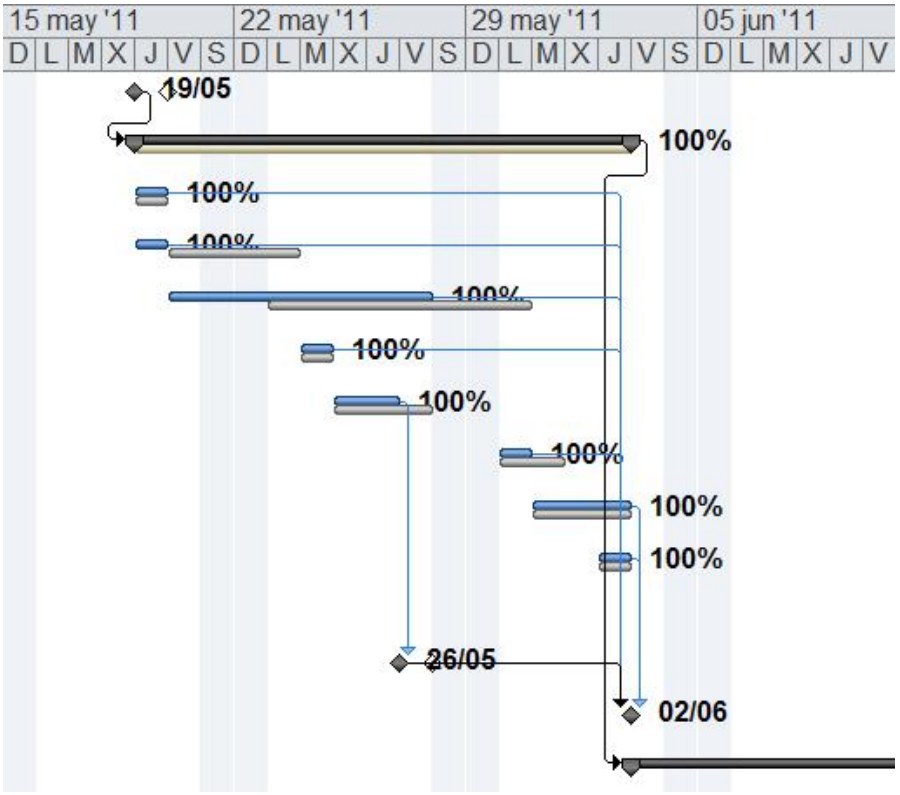


Ilustración 10-5 - Diagrama de Gantt en seguimiento de fase de Inicio

10.5.3. Fase Elaboración: Planificación iteración 1

Nombre de tarea	Duración	Comienzo	Fin
Fase Elaboración	**22 días?**	**vie 03/06/11**	**lun 04/07/11**
Iteración 1	**8 días?**	**vie 03/06/11**	**mar 14/06/11**
Identificar arquitectura del sistema	1 día?	vie 03/06/11	vie 03/06/11
Refinar casos de uso	1 día	lun 06/06/11	lun 06/06/11
Refinar SRS	2 días	lun 06/06/11	mar 07/06/11
Estudio metodologías (pruebas de viabilidad+alternativas) I	3 días	lun 06/06/11	mié 08/06/11
Diagrama de clases de análisis	2 días	mié 08/06/11	jue 09/06/11
Diagramas de secuencia de análisis	3 días	mié 08/06/11	vie 10/06/11
Diagramas diseño	2 días	vie 10/06/11	lun 13/06/11
Planificación Elaboración fase 2 + seguimiento fase elaboración iteración 1+ gestión de riesgos	1 día	mar 14/06/11	mar 14/06/11
Hito principal: Documento SRS refinado, 1º versión documento análisis	0 días	mar 07/06/11	mar 07/06/11
Iteración 2	**14 días?**	**mié 15/06/11**	**lun 04/07/11**

Ilustración 10-6 - Plan de iteración 1 de fase de Elaboración

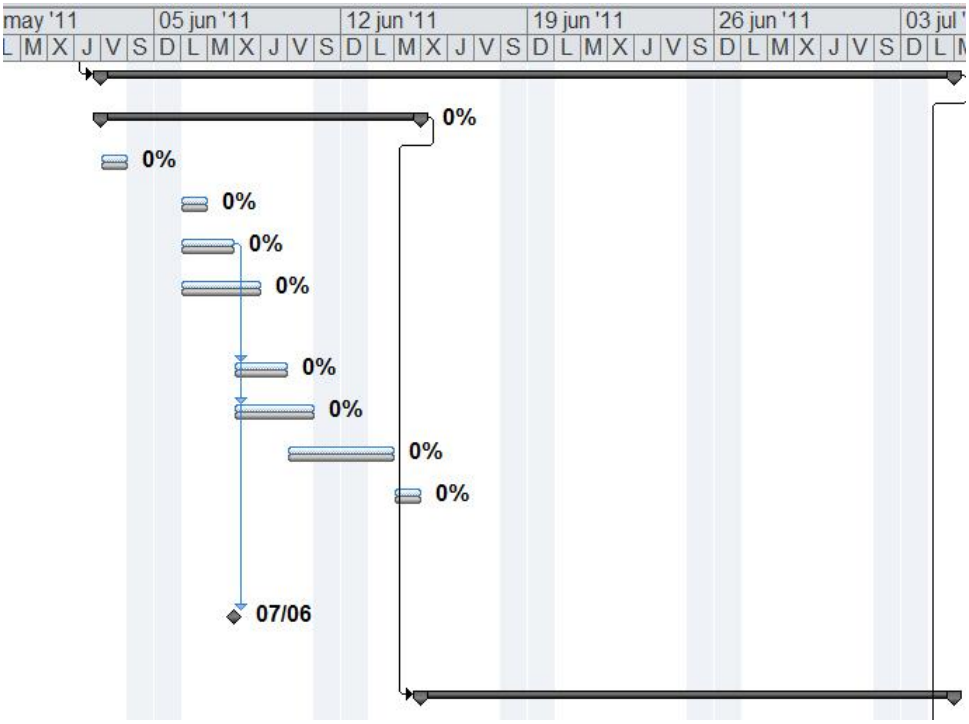


Ilustración 10-7 - Diagrama de Gantt en iteración 1 de fase de Elaboración

10.5.4. Fase Elaboración: Seguimiento iteración 1

Inicio	0 días	jue 19/05/11	jue 19/05/11	
Fase Inicio	**11 días?**	**jue 19/05/11**	**jue 02/06/11**	**1**
Fase Elaboración	**22 días?**	**vie 03/06/11**	**lun 04/07/11**	**2**
Iteración 1	**13 días**	**vie 03/06/11**	**mar 21/06/11**	
Identificar arquitectura del sistema	1 día	vie 03/06/11	vie 03/06/11	
Refinar casos de uso	1 día	vie 03/06/11	vie 03/06/11	
Refinar SRS	1 día	lun 06/06/11	lun 06/06/11	
Estudio metodologías (pruebas de viabilidad+alternativas) I	5 días	lun 13/06/11	vie 17/06/11	
Diagrama de clases de análisis	2 días	mar 07/06/11	mié 08/06/11	17
Diagramas de secuencia de análisis	1 día	mar 07/06/11	mar 07/06/11	17
Diagramas diseño	2 días	jue 09/06/11	vie 10/06/11	
Planificación Elaboración fase 2 + seguimiento fase elaboración iteración 1+ gestión de riesgos	1 día	mar 21/06/11	mar 21/06/11	21;20;19;18
Hito principal: Documento SRS refinado, 1º versión documento análisis	0 días	lun 06/06/11	lun 06/06/11	17
Iteración 2	**9 días?**	**mié 22/06/11**	**lun 04/07/11**	**14**

Ilustración 10-8 - Seguimiento de iteración 2 de fase de Elaboración

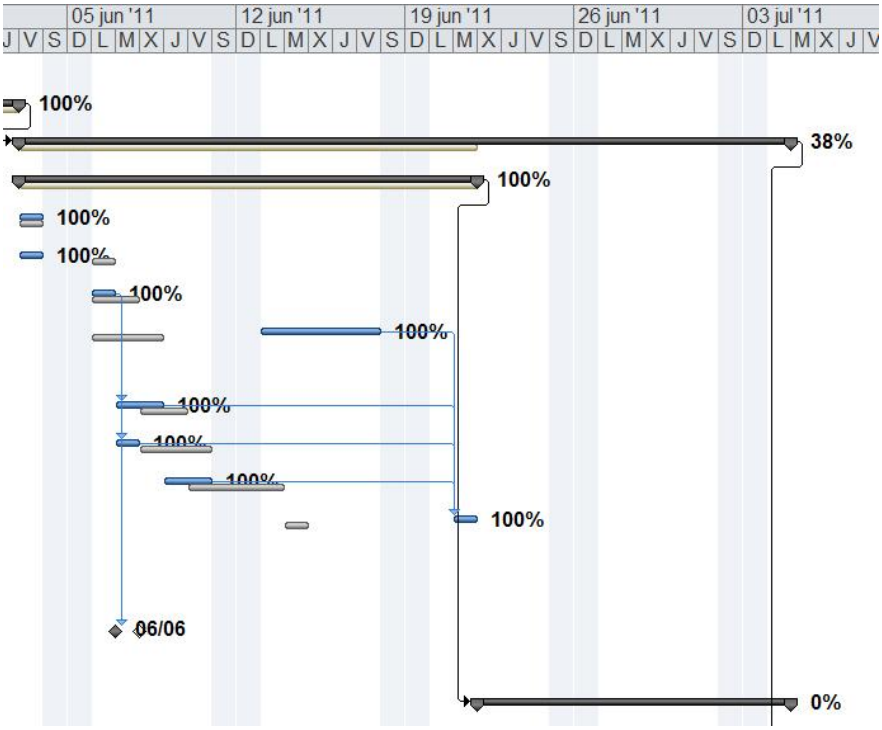


Ilustración 10-9 - Diagrama de Gantt de seguimiento de iteración 1 de fase de Elaboración

10.5.5. Fase Elaboración: Planificación iteración 2

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Iteración 1	**13 días**	**vie 03/06/11**	**mar 21/06/11**	
Iteración 2	**8 días**	**mié 22/06/11**	**vie 01/07/11**	**14**
Refinar diagrama de clases de análisis	1 día	mié 22/06/11	mié 22/06/11	
Refinar diagrama de secuencia de análisis	1 día	mié 22/06/11	mié 22/06/11	
Diagrama de clases de diseño	2 días	jue 23/06/11	vie 24/06/11	
Diagramas de secuencia de diseño	2 días	jue 23/06/11	vie 24/06/11	
Planificación fase construcción + seguimiento	1 día	vie 01/07/11	vie 01/07/11	
Probar metodología (pruebas de viabilidad+a	5 días	lun 27/06/11	vie 01/07/11	
Hito principal: Documento de análisis, arquitectura, 1º versión diseño, codigo version preliminar	0 días	vie 01/07/11	vie 01/07/11	25;26;28;27;30

Ilustración 10-10 - Plan de iteración 2 en fase de Elaboración

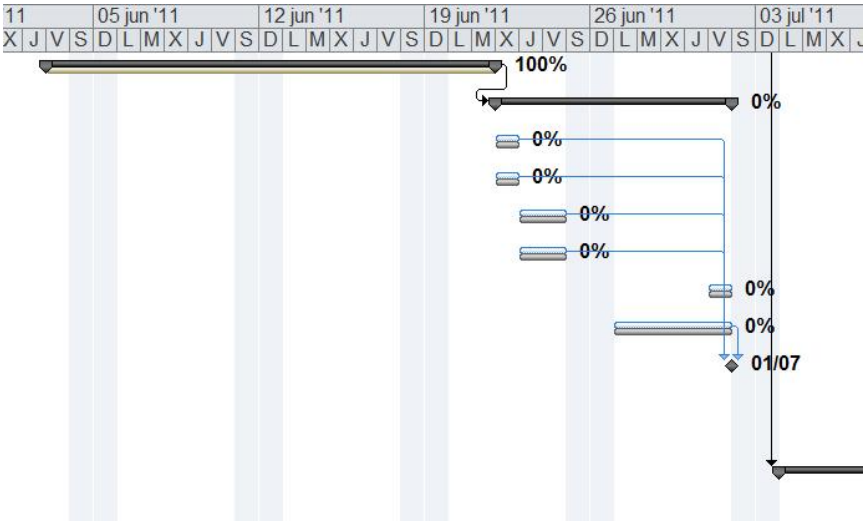


Ilustración 10-11 - Diagrama de Gantt en iteración 2 de fase de Elaboración

10.5.6. Fase Elaboración: Seguimiento iteración 2

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Iteración 1	**13 días**	**vie 03/06/11**	**mar 21/06/11**	
Iteración 2	**31 días**	**mié 22/06/11**	**mié 28/09/11**	**14**
Refinar diagrama de clases de análisis	1 día	mié 22/06/11	mié 22/06/11	
Refinar diagrama de secuencia de análisis	1 día	mié 22/06/11	mié 22/06/11	
Diagrama de clases de diseño	1 día	jue 23/06/11	jue 23/06/11	
Diagramas de secuencia de diseño	1 día	jue 23/06/11	jue 23/06/11	
Implementación (+pruebas+alternativas) II	5 días	lun 27/06/11	vie 01/07/11	
Replanificación de urgencia fase elaboración + seguimiento fase elaboración iteración 2 + riesgos	1 día	lun 04/07/11	lun 04/07/11	29
Diseño sistema adaptado a casos	2 días	mar 05/07/11	mié 06/07/11	30
Implementación (+pruebas+alternativas) III	7 días	jue 07/07/11	vie 15/07/11	31
Implementación (+pruebas+alternativas) IV	12 días	lun 12/09/11	mar 27/09/11	32
Planificación fase construcción + seguimiento fase elaboración iteración 2	1 día	mié 28/09/11	mié 28/09/11	31;33
Hito principal: Documento de análisis, arquitectura, 1º versión diseño, codigo version preliminar	0 días	mié 28/09/11	mié 28/09/11	25;26;28;27;34

Ilustración 10-12 - Seguimiento de iteración 2 en fase de Elaboración

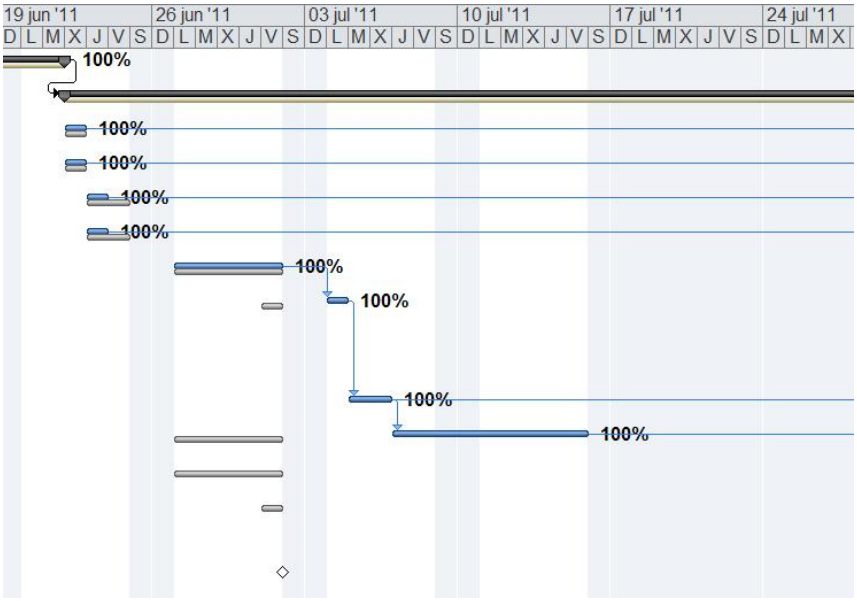


Ilustración 10-13 - Diagrama de Gantt en seguimiento de iteración 2 de fase de Elaboración

10.5.7. Fase Construcción: Planificación iteración 1

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Fase Construcción	**36 días**	**jue 29/09/11**	**jue 17/11/11**	**13**
Iteración 1	**18 días**	**jue 29/09/11**	**lun 24/10/11**	
Refinar diagrama de clases de diseño	2 días	jue 29/09/11	vie 30/09/11	
Refinar diagramas de secuencias de diseño	2 días	jue 29/09/11	vie 30/09/11	
Implementación	**14 días**	**lun 03/10/11**	**jue 20/10/11**	
Mejorar calculo de extremos relativos	3 días	lun 03/10/11	mié 05/10/11	
Mejorar filtros de compás y resolver sus problemas derivados	4 días	jue 06/10/11	mar 11/10/11	41
Detectar brazos familia caso 5	3 días	mié 12/10/11	vie 14/10/11	42
Detectar cabeza caso 5	3 días	lun 17/10/11	mié 19/10/11	43
Generar datos implicitos de caso 5 (cuello, tronco, etc)	1 día	jue 20/10/11	jue 20/10/11	44
Pruebas implementación	11 días	vie 07/10/11	vie 21/10/11	
Planificación fase construcción iteración 2+ seguimiento fase construcción iteración 1+gestión	1 día	lun 24/10/11	lun 24/10/11	38;39;40;46

Ilustración 10-14 - Plan de iteración 1 en fase de Construcción

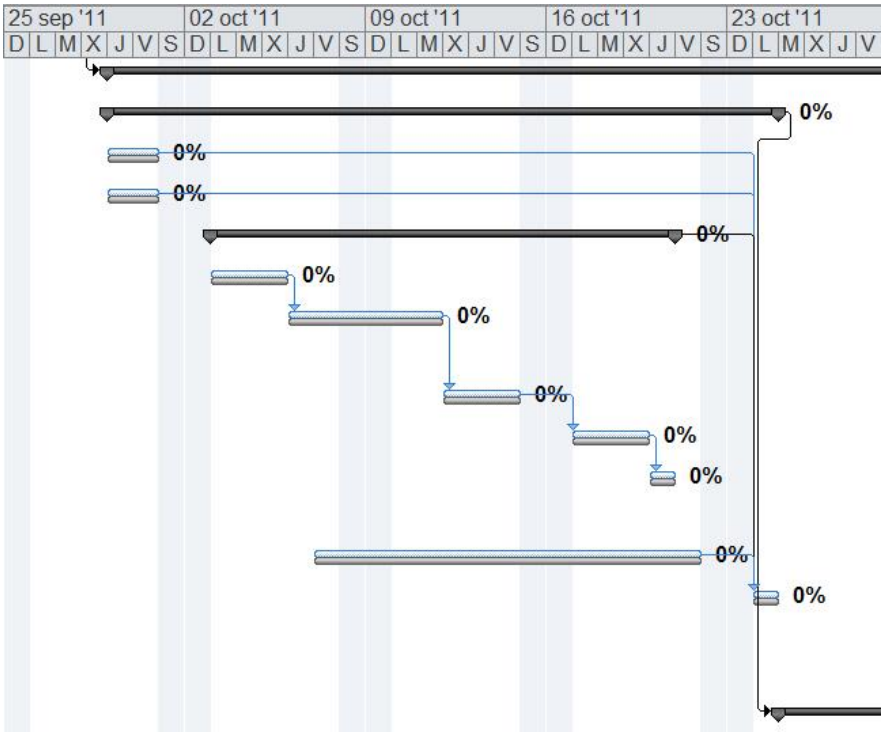


Ilustración 10-15 - Diagrama de Gantt de iteración 2 en fase de Construcción

10.5.8. Fase Construcción: Seguimiento iteración 1

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Fase Construcción	**36 días**	**mié 05/10/11**	**lun 28/11/11**	**13**
Iteración 1	**18 días**	**mié 05/10/11**	**mié 02/11/11**	
Refinar diagrama de clases de diseño	1 día	mié 05/10/11	mié 05/10/11	
Refinar diagramas de secuencias de diseño	1 día	mié 05/10/11	mié 05/10/11	
Implementación	**16 días**	**jue 06/10/11**	**vie 28/10/11**	
Mejorar calculo de extremos relativos	4 días	jue 06/10/11	mar 11/10/11	
Mejorar filtros de compás y resolver sus problemas derivados	2 días	jue 13/10/11	vie 14/10/11	41
Detectar brazos familia caso 5	6 días	lun 17/10/11	lun 24/10/11	42
Detectar cabeza caso 5	2 días	mar 25/10/11	mié 26/10/11	43
Generar datos implicitos de caso 5 (cuello, tronco, etc)	2 días	jue 27/10/11	vie 28/10/11	44
Pruebas implementación	12 días	jue 13/10/11	vie 28/10/11	
Planificación fase construcción iteración 2+ seguimiento fase construcción iteración 1+gestión	1 día	mié 02/11/11	mié 02/11/11	38;39;40;46

Ilustración 10-16 - Seguimiento de iteración 1 en fase de Construcción

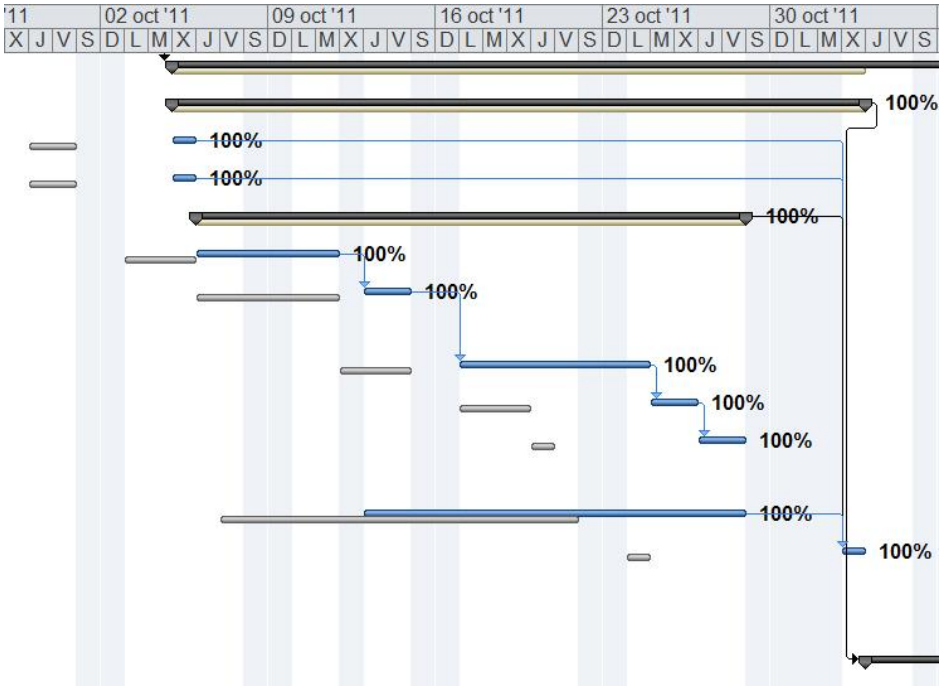


Ilustración 10-17 - Diagrama de Gantt de seguimiento de iteración 1 en fase de Construcción

10.5.9. Fase Construcción: Planificación iteración 2

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Iteración 1	**18 días**	**mié 05/10/11**	**mié 02/11/11**	
Iteración 2	**14 días**	**lun 14/11/11**	**jue 01/12/11**	**37**
Grafo de transiciones	3 días	lun 14/11/11	mié 16/11/11	
Implementación	**13 días**	**lun 14/11/11**	**mié 30/11/11**	
Revisión de implementación realizada (mejoras de ER con filtros de compás)	3 días	lun 14/11/11	mié 16/11/11	
Deteccion brazos caso 7	5 días	jue 17/11/11	mié 23/11/11	51
Deteccion cabeza caso 7	4 días	jue 24/11/11	mar 29/11/11	52
Calcular datos implicitos (cuello, tronco, etc)	1 día	mié 30/11/11	mié 30/11/11	53
Pruebas implementación	13 días	lun 14/11/11	mié 30/11/11	
Planificación fase construcción iteración 2 + seguimiento fase construcción iteración 2 + gestión	1 día	jue 01/12/11	jue 01/12/11	50;55
Hito principal: versión proyecto	0 días	mié 30/11/11	mié 30/11/11	50;55

Ilustración 10-18 - Plan de iteración 2 en fase de Construcción

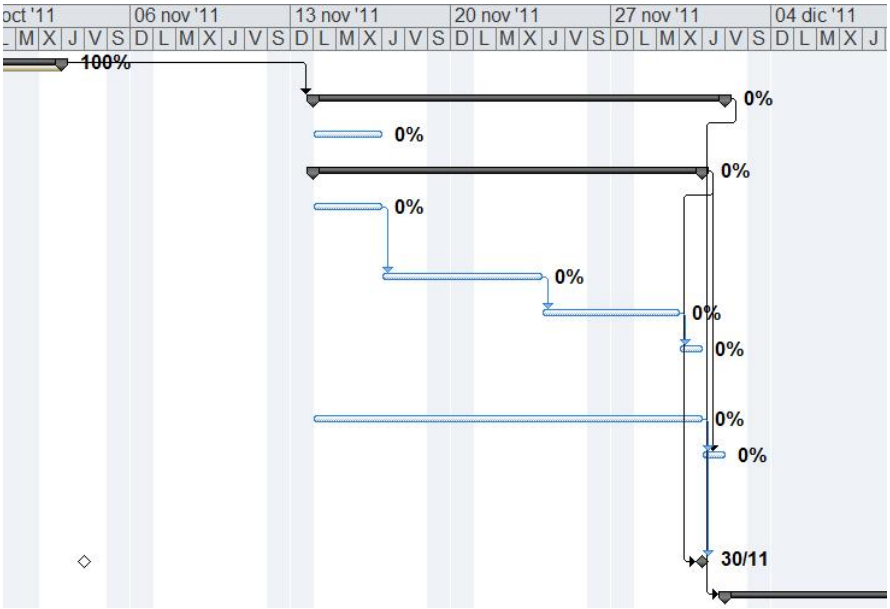


Ilustración 10-19 - Diagrama de Gantt de iteración 2 en fase de Construcción

10.5.10. Fase Construcción: Seguimiento iteración 2

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Iteración 1	**18 días**	**mié 05/10/11**	**mié 02/11/11**	
Iteración 2	**19,5 días**	**lun 14/11/11**	**vie 16/12/11**	**37**
Grafo de transiciones	2 días	lun 14/11/11	mar 15/11/11	
Implementación	**8,5 días**	**mar 15/11/11**	**vie 25/11/11**	
Revisión de implementación realizada (mejoras de ER con filtros de compás)	1 día	jue 17/11/11	jue 17/11/11	
Deteccion brazos caso 7	6 días	mar 15/11/11	mié 23/11/11	
Deteccion cabeza caso 7	1 día	jue 24/11/11	jue 24/11/11	52
Calcular datos implicitos (cuello, tronco, etc)	1 día	vie 25/11/11	vie 25/11/11	53
Pruebas implementación	9 días	mié 16/11/11	lun 28/11/11	
Replanificar iteración actual	1 día	mar 29/11/11	mar 29/11/11	
Implementación	**6 días**	**mié 30/11/11**	**jue 15/12/11**	**56**
Mejora calidad resultado caso 7	1 día	vie 02/12/11	vie 02/12/11	
Resolver problema de fiabliidad del detector del cuerpo	6 días	mié 30/11/11	jue 15/12/11	
Planificación fase construcción iteración 2 + seguimiento fase construcción iteración 2 + gestión	1 día	jue 15/12/11	vie 16/12/11	50;57
Hito principal: versión proyecto	0 días	jue 15/12/11	jue 15/12/11	50;55;57

Ilustración 10-20 - Seguimiento de iteración 2 en fase de Construcción

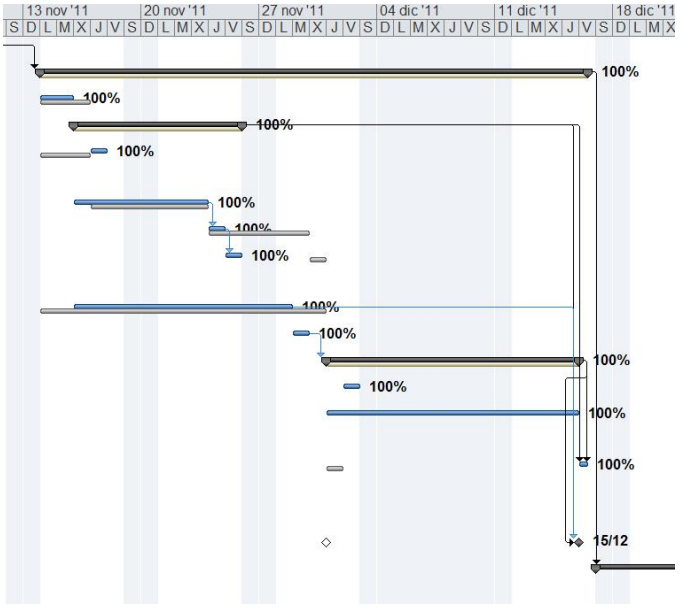


Ilustración 10-21 - Diagrama de Gantt de seguimiento de iteración 2 en fase de Construcción

10.5.11. Fase Construcción: Planificación iteración 3

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Iteración 2	**19,5 días**	**lun 14/11/11**	**vie 16/12/11**	**37**
Iteración 3	**50,5 días**	**vie 16/12/11**	**lun 12/03/12**	**48**
implementación	**36,5 días**	**vie 16/12/11**	**mar 21/02/12**	
Adaptación proyecto para video	3 días	vie 16/12/11	mié 21/12/11	
Sistema de retroalimentacion entre iteraciones del algoritmo	10 días	vie 16/12/11	lun 16/01/12	
Motor de casos: transiciones de casos	7 días	mar 17/01/12	mié 25/01/12	
Resolución de posibles conflictos derivados de nuevas pruebas globale	10 días	vie 27/01/12	jue 09/02/12	
Algoritmo detectar suelo	3 días	lun 13/02/12	mié 15/02/12	
Algoritmo generar esqueleto piernas-cadera	4 días	jue 16/02/12	mar 21/02/12	68
Pruebas implementación	37 días	vie 16/12/11	mié 22/02/12	
Guardar datos en XML	2 días	vie 16/12/11	mar 20/12/11	
Interfaz	**7 días**	**lun 27/02/12**	**mar 06/03/12**	
Diseño interfaz	3 días	lun 27/02/12	mié 29/02/12	
Implementación interfaz	4 días	jue 01/03/12	mar 06/03/12	73
Redactar documentación memoria final I	10 días	lun 16/01/12	vie 27/01/12	
Redactar documentación memoria final II	10 días	lun 06/02/12	vie 17/02/12	
Redactar documentación memoria final III	10 días	lun 27/02/12	vie 09/03/12	
Planificación fase Transición + seguimiento fase construcción iteración 2 + gestión	1 día	lun 12/03/12	lun 12/03/12	77;72;63;70
Hito principal: Versión beta + primera versión de la memoria	0 días	mar 21/02/12	mar 21/02/12	63

Ilustración 10-22 - Plan de iteración 3 en fase de Construcción

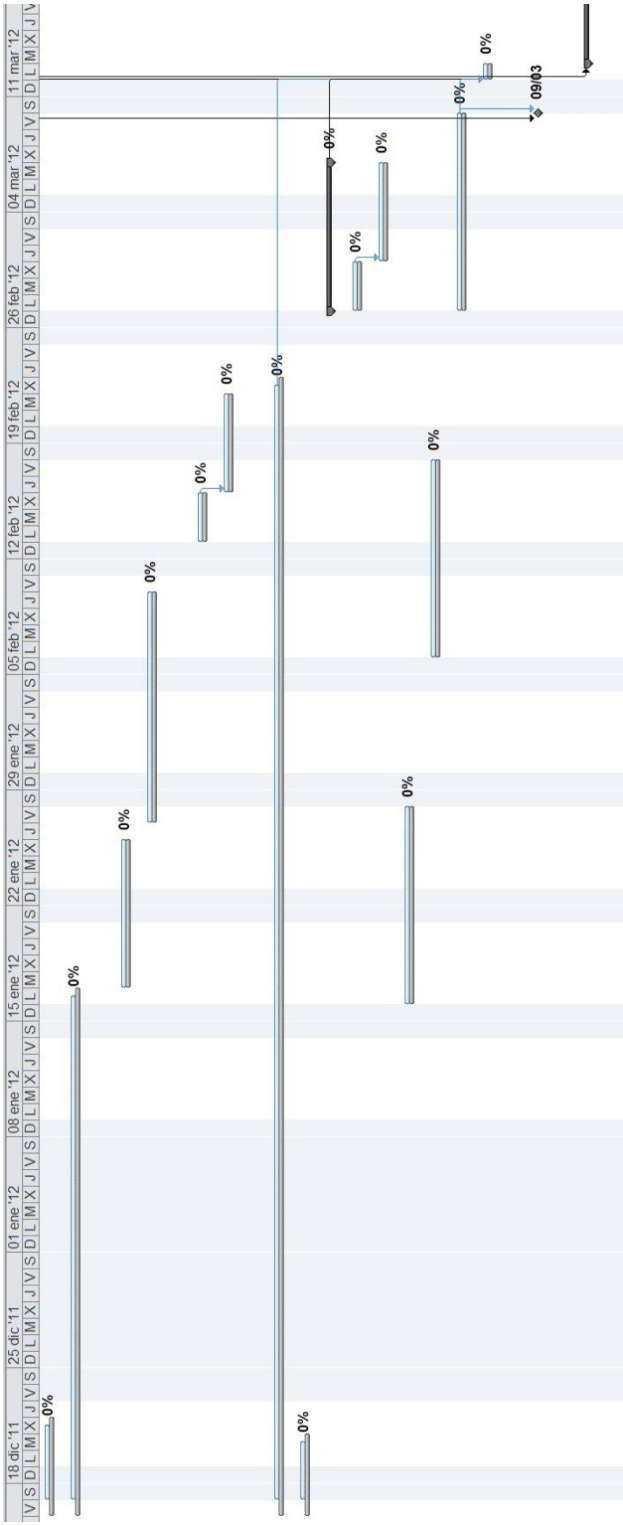


Ilustración 10-23 - Diagrama de Gantt de iteración 3 en fase de Construcción

10.5.12. Fase Construcción: Seguimiento iteración 3

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Iteración 3	**83 días**	**vie 16/12/11**	**mié 02/05/12**	**48**
implementación	**74,5 días**	**mié 21/12/11**	**vie 20/04/12**	
Adaptación proyecto para video	2 días	mié 21/12/11	jue 22/12/11	
Sistema de retroalimentacion entre iteraciones del algoritmo	10 días	mar 24/01/12	mar 07/02/12	
Motor de casos: transiciones de casos	30,5 días	jue 09/02/12	vie 23/03/12	
Resolución de posibles conflictos derivados de nuevas pruebas globale	48,5 días	vie 10/02/12	vie 20/04/12	
Algoritmo detectar suelo	1 día	vie 03/02/12	vie 03/02/12	
Algoritmo generar esqueleto piernas-cadera	1 día	jue 16/02/12	jue 16/02/12	68
Pruebas implementación	37 días	vie 16/12/11	mié 22/02/12	
Guardar datos en XML	1 día	jue 29/03/12	jue 29/03/12	
Interfaz	**4 días**	**mar 24/04/12**	**vie 27/04/12**	
Diseño interfaz	2 días	mar 24/04/12	mié 25/04/12	
Implementación interfaz	2 días	jue 26/04/12	vie 27/04/12	73
Redactar documentación memoria final I	3 días	vie 24/02/12	mar 28/02/12	
Redactar documentación memoria final II	2 días	vie 30/03/12	lun 02/04/12	
Redactar documentación memoria final III	6,5 días	jue 19/04/12	lun 30/04/12	
Planificación fase Transición + seguimiento fase construcción iteración 3 + gestión	1 día	lun 30/04/12	mié 02/05/12	77;72;63;70
Hito principal: Versión beta + primera versión de la memoria	0 días	lun 30/04/12	lun 30/04/12	63;77;72

Ilustración 10-24 - Seguimiento de iteración 3 en fase de Construcción

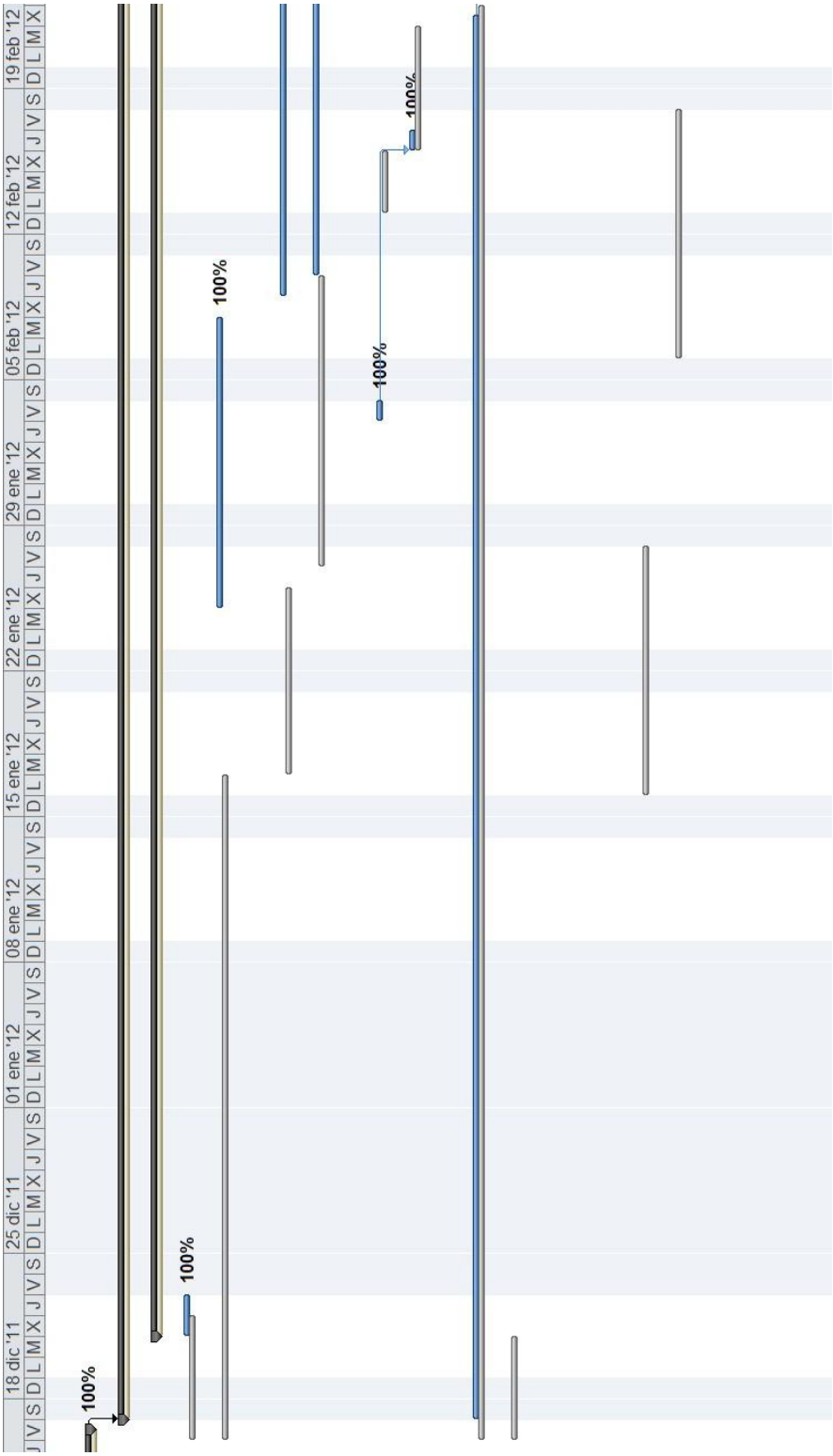


Ilustración 10-25 - Diagrama de Gantt de seguimiento de iteración 3 en fase de Construcción (1)

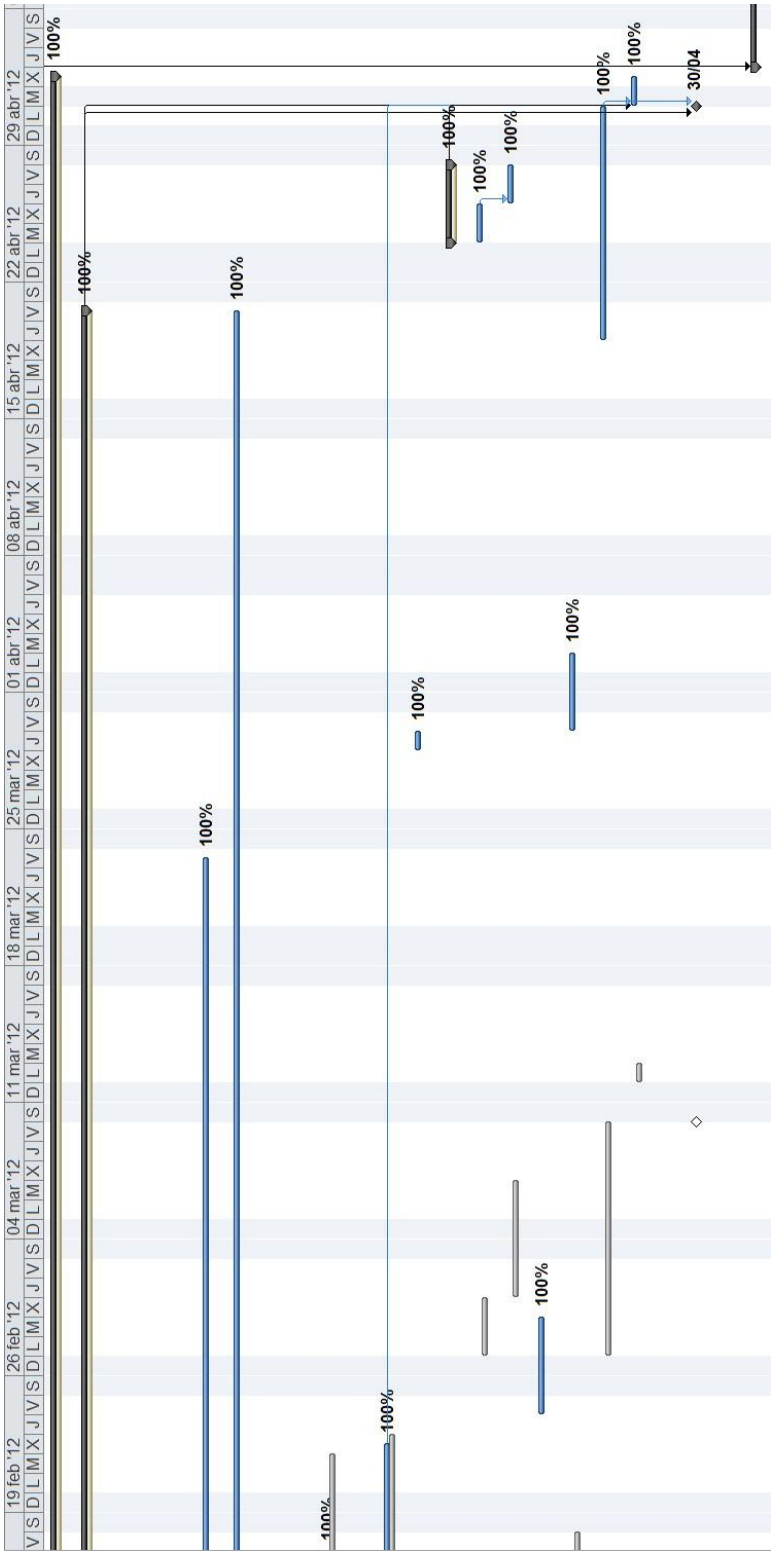


Ilustración 10-26 - Diagrama de Gantt de seguimiento de iteración 3 en fase de Construcción (2)

10.5.13. Fase Transición: Planificación iteración 1

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Fase Elaboración	**44 días**	**vie 03/06/11**	**mié 28/09/11**	**2**
Iteración 1	**13 días**	**vie 03/06/11**	**mar 21/06/11**	
Iteración 2	**31 días**	**mié 22/06/11**	**mié 28/09/11**	**14**
Fase Construcción	**27,5 días**	**mié 05/10/11**	**mié 02/05/12**	**13**
Iteración 1	**18 días**	**mié 05/10/11**	**mié 02/11/11**	
Iteración 2	**19,5 días**	**lun 14/11/11**	**vie 16/12/11**	**37**
Iteración 3	**83 días**	**vie 16/12/11**	**mié 02/05/12**	**48**
Fase Transición	**20,5 días**	**mié 02/05/12**	**jue 31/05/12**	**36**
Iteración 1	**20,5 días**	**mié 02/05/12**	**jue 31/05/12**	
Redactar memoria I	2 días	mié 02/05/12	vie 04/05/12	
Pruebas y resolución de defectos encontrados en la versión beta	5 días	lun 07/05/12	vie 11/05/12	
Redactar memoria II	9 días	mar 15/05/12	vie 25/05/12	82
Pruebas finales	3 días	lun 28/05/12	mié 30/05/12	84
Documento de pruebas	0 días	mié 30/05/12	mié 30/05/12	83;85
Redactar memoria II	3 días	lun 28/05/12	mié 30/05/12	84
Seguimiento fase transición	1 día	jue 31/05/12	jue 31/05/12	87
Versión final	0 días	jue 31/05/12	jue 31/05/12	88;86
Final	0 días	vie 08/06/12	vie 08/06/12	81

Ilustración 10-27 - Plan de fase de Transición

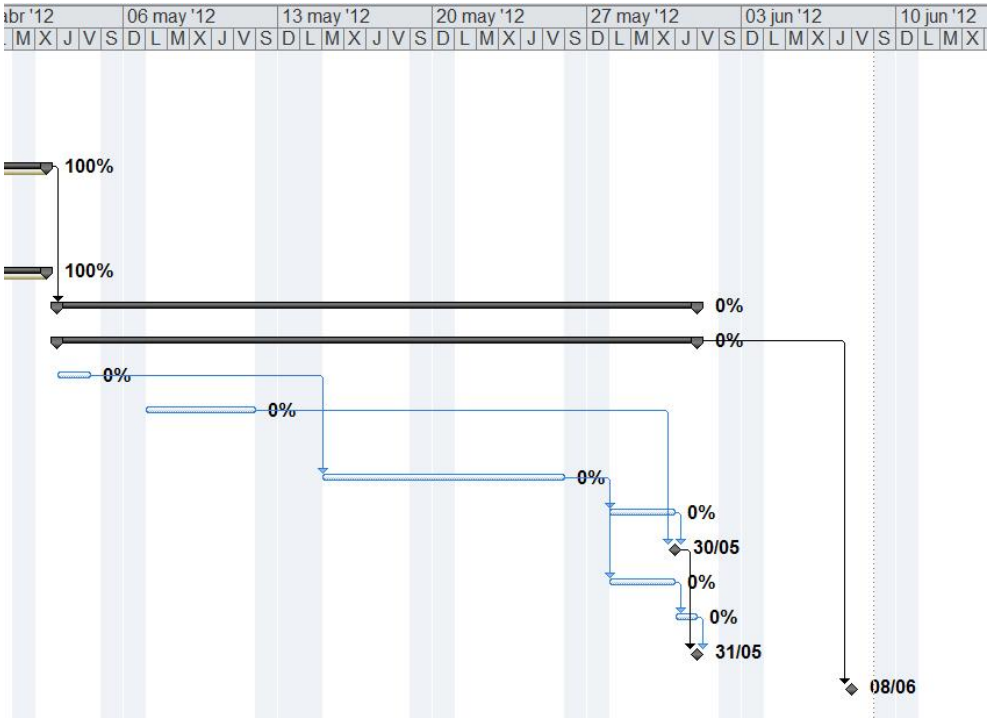


Ilustración 10-28 - Diagrama de Gantt de fase de Transición

10.5.14. Fase Transición: Seguimiento iteración 1

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Fase Transición	**38 días**	**mié 02/05/12**	**mar 26/06/12**	**36**
Iteración 1	**38 días**	**mié 02/05/12**	**mar 26/06/12**	
Redactar memoria I	2 días	mié 02/05/12	vie 04/05/12	
Pruebas y resolución de defectos encontrados en la versión beta	5 días	lun 07/05/12	vie 11/05/12	
Redactar memoria II	7 días	mar 15/05/12	mié 23/05/12	82
Pruebas finales	4 días	jue 24/05/12	mar 29/05/12	84
Documento de pruebas	0 días	mar 29/05/12	mar 29/05/12	83;85
Redactar documentación restante	6 días	vie 25/05/12	vie 01/06/12	84
Seguimiento fase transición	1 día	lun 04/06/12	lun 04/06/12	87
Revisión memoria con tutores. Seguimiento fase transición	1 día	lun 25/06/12	lun 25/06/12	88
Versión final	0 días	lun 25/06/12	mar 26/06/12	89;86
Final	0 días	mar 26/06/12	mar 26/06/12	81;90

Ilustración 10-29 - Seguimiento de iteración 1 en fase de Transición

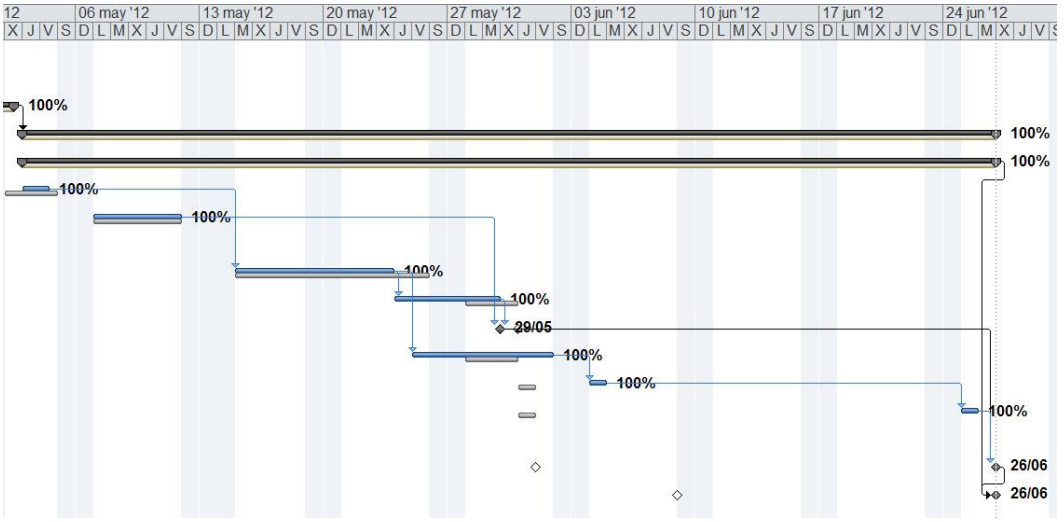


Ilustración 10-30 - Diagrama de Gannt de seguimiento en iteración 1 de fase de Transición

10.6. Resultados del seguimiento del proyecto

Debido a motivos personales, a la complejidad del producto que había que desarrollar y del tiempo tan ajustado del que se disponía, no se pudo completar el trabajo en el tiempo propuesto y fue necesario ampliar el tiempo de desarrollo en un año.

Los resultados generales del desarrollo son los siguientes:

Tiempo dedicado	Inicio	Elaboración	Construcción	Transición
Porcentaje	5%	22%	63%	10%
Días	11 días	44 días	128 días	24 días

Tabla 10-14 - Proporción de tiempo consumido por fase

Y el resumen del esfuerzo invertido:

Fase	Número de iteraciones	Promedio horas/hombre por día	Total horas/fase
Inicio	0	3.3 horas/hombre	37
Elaboración	2	2.7 horas/hombre	31 + 91 = 122
Construcción	3	2.5 horas/hombre	40+45+227=312
Transición	1	3.6 horas/hombre	85

Tabla 10-15 - Relación de esfuerzo consumido por fase

El total de horas netas dedicadas al proyecto es: **556 horas**. Los datos completos de horas dedicadas al día y resultados de la evolución de cada iteración se pueden ver en los documentos de seguimiento adjuntos a la memoria.

Bloque IV: Análisis

Capítulo 11. Introducción al análisis.

En este bloque se va a proceder a exponer todo lo relacionado con el análisis. Se hablará de los stakeholders del proyecto, de los requisitos del sistema, de los casos de uso y se mostrará un primer acercamiento a la aplicación con el modelo de dominio.

11.1. Stakeholders.

Los desarrolladores que han intervenido en este proyecto son:

- José Luis Martínez Jiménez

Los principales clientes relacionados con este proyecto son:

- Don Francisco Javier Finat Codes, investigador de la UVA en el grupo MoBiVAP y cotutor de este proyecto.
- Don Valentín Cardeñoso Payo, cotutor de este proyecto.
- Don Rubén Martínez García, investigador de la UVA del grupo MoBiVAP.

11.2. Seguridad.

Dada la naturaleza stand alone de la aplicación (para más información de los motivos véase el epígrafe entorno del capítulo de introducción) y que no almacena contenido que pueda estar sujeto a la Ley de Protección de Datos, no se ha visto necesario implementar medida de seguridad alguna.

11.3. Glosario de términos.

- **Entregable:** artefacto que debe ser entregado al cliente.
- **Artefacto:** Es una pieza de información utilizada o producida mediante un proceso de desarrollo de software. Puede ser un modelo, una descripción o un software. No siempre es un entregable.
- **Etapa del proceso:** Conjunto de actividades orientadas a la consecución de una serie de objetivos, que se denominan objetivos de la etapa.
- **Hito:** Evento de relevancia durante el proceso. A menudo surgen de la consecución de algún objetivo.
- **Planificación:** Se compone del plan general de ejecución del proyecto, así como del desarrollo real de dicho plan, incluyendo información detallada sobre el tiempo que requerirá cada actividad y los recursos que se destinarán (o destinaron) a la misma.
- **Planificación inicial:** Es la estimación a priori de los tiempos y recursos que requerirá cada una de las actividades a desempeñar en el proceso. Se trata de la planificación teórica del proceso, y los tiempos y recursos que establece no tienen por qué coincidir con los que inicialmente se planteó para este desarrollo de software.
- **Stakeholder:** Toda persona o entidad

11.4. Acrónimos utilizados.

- OpenCV (Computer Vision): Plataforma software de Visión Computacional Open Source bajo C++
- OOP (Object Oriented Program): Programación Orientada a Objetos)
- RT (Real Time): Tiempo Real

Capítulo 12. Funcionalidad Requerida para el sistema.

12.1. Análisis de requisitos.

En este epígrafe se estudian los diferentes requisitos que el sistema desarrollado debe cumplir explicando primero los requisitos funcionales, seguidos de los requisitos no funcionales.

Lista de requisitos funcionales:

FRQ-0001	Vista previa video
Versión	1.0 (26/05/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *permitir al usuario realizar una vista previa de un video seleccionado. Esta vista previa consiste en la visualización del video en bruto, es decir, sin ningún tipo de tratamiento especial.*
Importancia	quedaría bien
Urgencia	puede esperar
Estado	completado
Estabilidad	estable
Comentarios	Ninguno

Tabla 12-1 - FRQ-0001

FRQ-0002	Carga de videos
Versión	1.0 (25/05/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *permitir al usuario cargar un video en el sistema, especificando una ruta a dicho archivo. Esos videos serán de tipo .avi*
Importancia	vital
Urgencia	urgente
Estado	completado
Estabilidad	estable
Comentarios	Ninguno

Tabla 12-2 - FRQ-0002

FRQ-0003	Segmentación de video
Versión	1.0 (25/05/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *ser capaz de segmentar un video de danza, extrayendo lo mejor posible al bailarín del fondo.*
Importancia	vital
Urgencia	urgente
Estado	completado
Estabilidad	estable
Comentarios	De momento se ha previsto que esto solo funcione para un sólo bailarin. Dado que puede darse el caso de que la ropa del individuo (o parte del cuerpo del mismo) sea radiométricamente similar al fondo sobre la que se encuentra, es posible que la extracción de dicho cuerpo no sea perfecta, pero en este caso, se considerará que se ha tenido existo en el cumplimiento de este requisito si lo extraído permite reconocer la postura del individuo.

Tabla 12-3 - FRQ-0003

FRQ-0004	Seguimiento bailarín
Versión	1.0 (25/05/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *ser capaz de localizar y seguir la posición del bailarín dentro de la escena durante el análisis del video*
Importancia	vital
Urgencia	urgente
Estado	completado
Estabilidad	estable
Comentarios	Ninguno

Tabla 12-4 - FRQ-0004

FRQ-0005	Identificar características del bailarín
Versión	1.0 (26/05/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *identificar características del bailarín en la imagen del video, de tal manera que éstas sean capaces de definir la postura del bailarín, para su posterior reconocimiento*
Importancia	vital
Urgencia	urgente
Estado	completado
Estabilidad	estable
Comentarios	Se plantea la posibilidad de que estas características sean el esqueleto del cuerpo del bailarín, aunque podría ser también la poligonal de la silueta del cuerpo del bailarín, siempre y cuando ésta esté suficientemente bien definida

Tabla 12-5 - FRQ-0005

FRQ-0006	Configuración parámetros
Versión	1.0 (26/05/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *permitir modificar la configuración de los diferentes parámetros de filtrado, segmentación, umbrales, etc*
Importancia	quedaría bien
Urgencia	puede esperar
Estado	completado
Estabilidad	estable
Comentarios	Ninguno

Tabla 12-6 - FRQ-0006

Lista de requisitos no funcionales:

NFR-0001	Tiempo de respuesta.
Versión	1.0 (25/05/2007)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *implementar una metodología que permita realizar todas las operaciones de segmentación y reconocimiento en tiempo real.*
Importancia	quedaría bien
Urgencia	puede esperar
Estado	pendiente de validación
Estabilidad	Estable
Comentarios	Ninguno

Tabla 12-7 - NRF-0001

NFR-0002	Recuperación de errores
Versión	1.0 (06/06/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá , *en caso de que se produzca algún tipo de error, ser capaz de recuperarse de esos errores en tiempo real, en la medida de lo posible.*
Importancia	Importante
Urgencia	Vital
Estado	PD
Estabilidad	PD
Comentarios	Ninguno

Tabla 12-8 - NRF-0002

NFR-0003	Interfaz.
Versión	1.0 (06/06/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *disponer de una interfaz sencilla para el usuario*
Importancia	quedaría bien
Urgencia	puede esperar
Estado	Completado
Estabilidad	Estable
Comentarios	Ninguno

Tabla 12-9 - NRF-0003

NFR-0004	Limitación de tecnología.
Versión	1.0 (26/05/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá *usará las siguientes herramientas: Visual C++, Qt, OpenCV*
Importancia	Vital
Urgencia	PD
Estado	Completado
Estabilidad	Estable
Comentarios	Ninguno

Tabla 12-10 - NRF-0004

Lista de requisitos de información:

IRQ-0001	Video a procesar
Versión	1.0 (25/05/2011)
Autores	José Luis Martínez Jiménez
Dependencias	Ninguno
Descripción	El sistema deberá trabajar con la información correspondiente a*l video adjunto a este documento, o en su defecto, partes de dicho video*.
Datos específicos	Ninguno
Comentarios	Ninguno

Tabla 12-11 - IRQ-0001

12.2. Roles.

El estudio automático de movimiento humano

12.3. Casos de uso.

A continuación se muestra el diagrama de casos de uso del sistema:

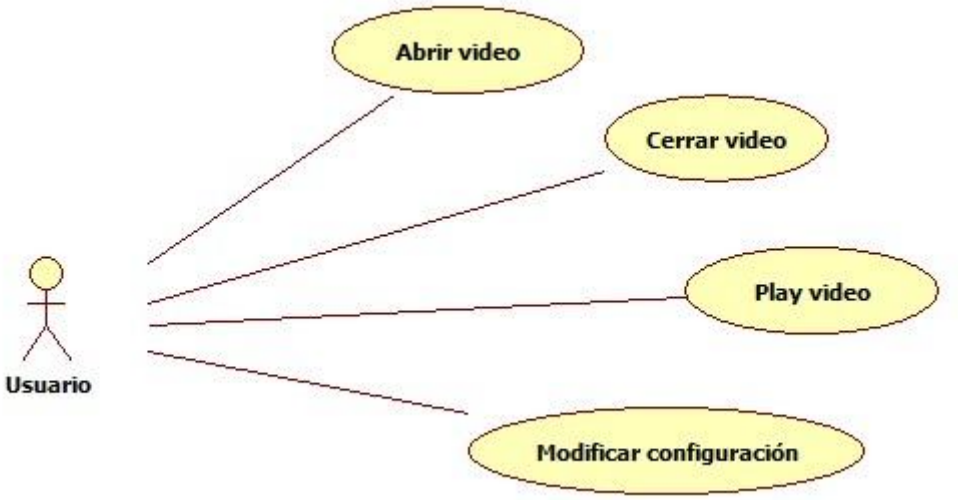


Ilustración 12-1 - Diagrama de casos de uso

Donde Modificar configuración tiene un escenario posible por cada posible parámetro a modificar en la configuración y dondel el caso de uso Play Video tiene a su vez dos escenarios posibles:

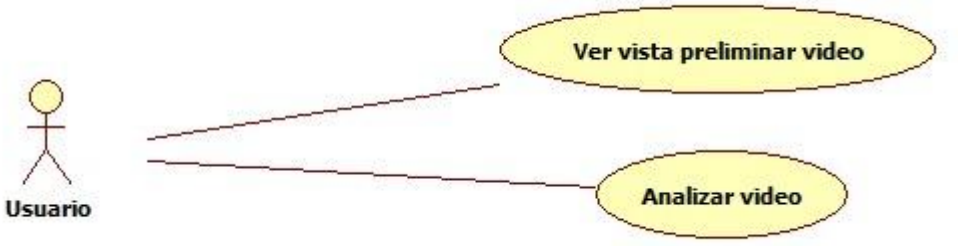


Ilustración 12-2 - Escenarios de "Play video"

A continuación detallamos los diversos casos de uso del sistema, en la que se detalla nombre descripción del caso de uso, pasos, precondiciones, postcondiciones y excepciones.

UC-0001	Abrir video	
Versión	1.0 (30/05/2011)	
Autores	• José Luis Martínez Jiménez	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el usuario desea abrir un video*	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-0001) *solicita abrir un nuevo video*
	2	El sistema *solicita al actor Usuario seleccionar la ruta en la que se encuetra el video*
	3	El actor Usuario (ACT-0001) *selecciona la ruta en la que se encuentra el video*
	4	El sistema *abre el video, procesando la operación*
Postcondición	Se ha abierto el video seleccionado por el usuario (guardar información del video para su posterior tratamiento, incluyendo el feedback correspondiente para el usuario)	
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Importancia	vital	
Urgencia	inmediatamente	
Estado	completado	
Estabilidad	alta	
Comentarios	Ninguno	

Tabla 12-12 - UC-0001

UC-0002	Ver vista preliminar video	
Versión	1.0 (30/05/2011)	
Autores	• José Luis Martínez Jiménez	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *un usuario desee ver el contenido de un video abierto*	
Precondición	Hay un video abierto como resultado del caso de uso [UC-0001] Abrir video	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-0001) *solicita realizar una vista preliminar del video*
	2	El sistema *carga el video y se lo muestra al usuario (reproducción del video)*
	3	El sistema *termina con la reproducción del video tras haberse llegado al final de dicho video*
Postcondición	Se muestra al usuario el video abierto	
Excepciones	**Paso**	**Acción**
	2	Si *el usuario cancela la visualización del video*, el sistema *cierra la reproducción del video*, a continuación este caso de uso *queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Importancia	importante	
Urgencia	hay presión	
Estado	completado	
Estabilidad	alta	
Comentarios	Ninguno	

Tabla 12-13 - UC-0002

UC-0003	Analizar video	
Versión	1.0 (30/05/2011)	
Autores	• José Luis Martínez Jiménez	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *un usuario desee analizar un video de coreografías de baile*	
Precondición	Hay un video abierto como resultado del caso de uso [UC-0001] Abrir video	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-0001) *solicita analizar el video abierto*
	2	El sistema *carga el video y se lo muestra al usuario (reproducción del video) junto con los resultados del análisis (filtros/detectores)*
	3	El sistema *termina con el análisis del video tras haberse llegado al final de dicho video*
Postcondición	Se ha realizado un análisis del video abierto	
Excepciones	**Paso**	**Acción**
	2	Si *el usuario cancela el análisis del video*, el sistema *cierra la reproducción del video*, a continuación este caso de uso *queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Importancia	vital	
Urgencia	hay presión	
Estado	completado	
Estabilidad	media	
Comentarios	Ninguno	

Tabla 12-14 - UC-0003

UC-0004	Modificar configuración	
Versión	1.0 (03/06/2011)	
Autores	• José Luis Martínez Jiménez	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *un usuario del sistema dese modificar algún parámetro generales de configuración de los filtros/detectores de que dispone el sistema*	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-0001) *solicita modificar la configuración del los filtros/detectores disponibles en el sistema*
	2	El sistema *muestra la configuración actual y permitiendo al usuario modificarla*
	3	El actor Usuario (ACT-0001) *modifica la configuración*
	4	El sistema *guarda la nueva configuración*
Postcondición	Se ha guardado la nueva configuración de los parámetros de configuración del sistema	
Excepciones	**Paso**	**Acción**
	3	Si *el actor Usuario cancela la operación*, el sistema *no hace nada*, a continuación este caso de uso *queda sin efecto*
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Importancia	importante	
Urgencia	hay presión	
Estado	completado	
Estabilidad	baja	
Comentarios	Especificación general para cualquier escenario particular de este caso de uso	

Tabla 12-15 - UC-0004

UC-0005	Cerrar video	
Versión	1.0 (03/06/2011)	
Autores	• José Luis Martínez Jiménez	
Dependencias	Ninguno	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *un usuario desee cerrar un video abierto (eliminar la referencia al video cargado actualmente)*	
Precondición	Hay un video abierto como resultado del caso de uso [UC-0001] Abrir video	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-0001) *solicita cerrar un video abierto como resultado del caso de uso [UC-0001] Abrir video*
	2	El sistema *cierra el video, liberando cualquier información temporal almacenada sobre el mismo*
	3	El actor Usuario (ACT-0001) *indica la ruta a la imagen y si lo desea completa también la descripción*
	4	El sistema *reconoce la posición y la almacena*
Postcondición	Se ha eliminado la referencia al video cargado actualmente, liberando la información temporal de dicho video dentro del sistema	
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	en construcción	
Estabilidad	alta	
Comentarios	Ninguno	

Tabla 12-16 - UC-0005

Capítulo 13. Modelo de dominio.

En la Figura 8.1 se puede observar el diagrama de clases de análisis completo, planteado a comienzo del desarrollo de este proyecto. En él se plasma el esfuerzo por detallar completamente las posibles clases de nuestro sistema.

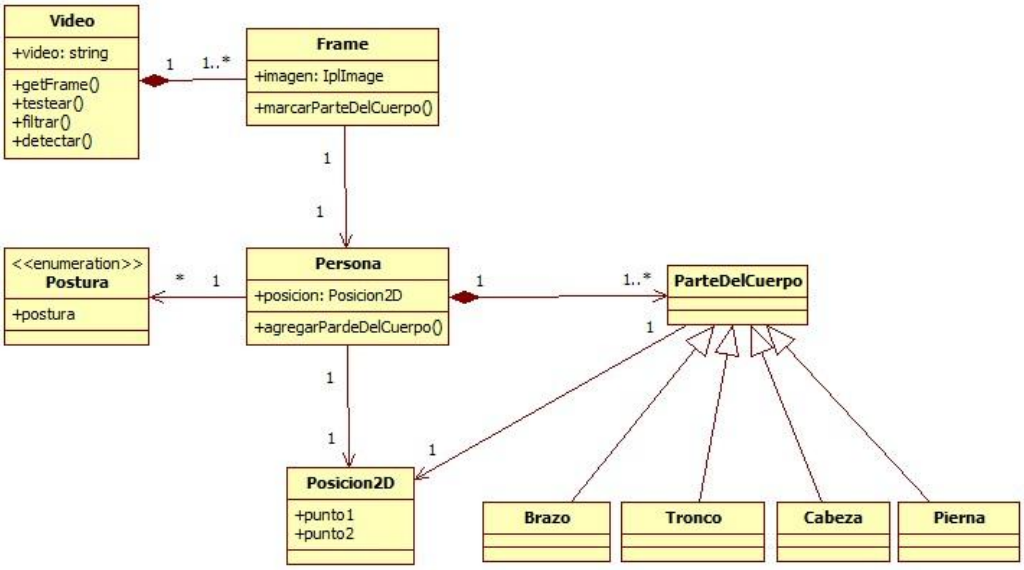


Ilustración 13-1 - Diagrama de clases de análisis

En dicha figura podemos observar una clase "Video" que es la que tiene la capacidad de recuperar la colección de frames del mismo. Ese video puede ser "tratado de alguna manera" para poder detectar los individuos que en él aparecen, de tal manera que en cada frame aparecerá una persona (modelada en la clase homónima) colocada en una determinada posición de la imagen (Posicion2D) y en una determinada postura, que corresponderá a alguno de los casos que se mencionaban en el capítulo 5 de Metodología. A su vez, el cuerpo puede ser dividido en partes de interés, como son los brazos, la cabeza, el tronco o las piernas, que también estarán localizadas en una determinada posición.

Capítulo 14. Diagramas de secuencia de análisis.

14.1. Diagrama de secuencia asociado al caso de uso "Abrir video"

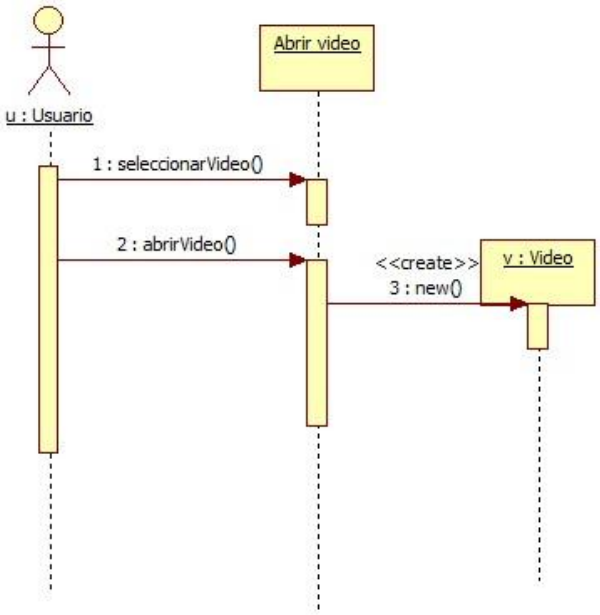


Ilustración 14-1 - Diagrama de secuencia de análisis "Abrir video"

14.2. Diagrama de secuencia asociado al caso de uso "Cerrar video"

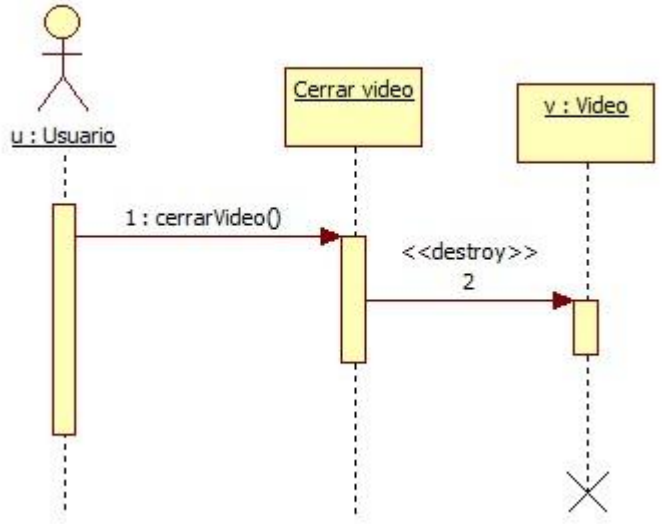


Ilustración 14-2 - Diagrama de secuencia de análisis "Cerrar video"

14.3. Diagrama de secuencia asociado al caso de uso "Play video" - Escenario "Ver vista preliminar video"

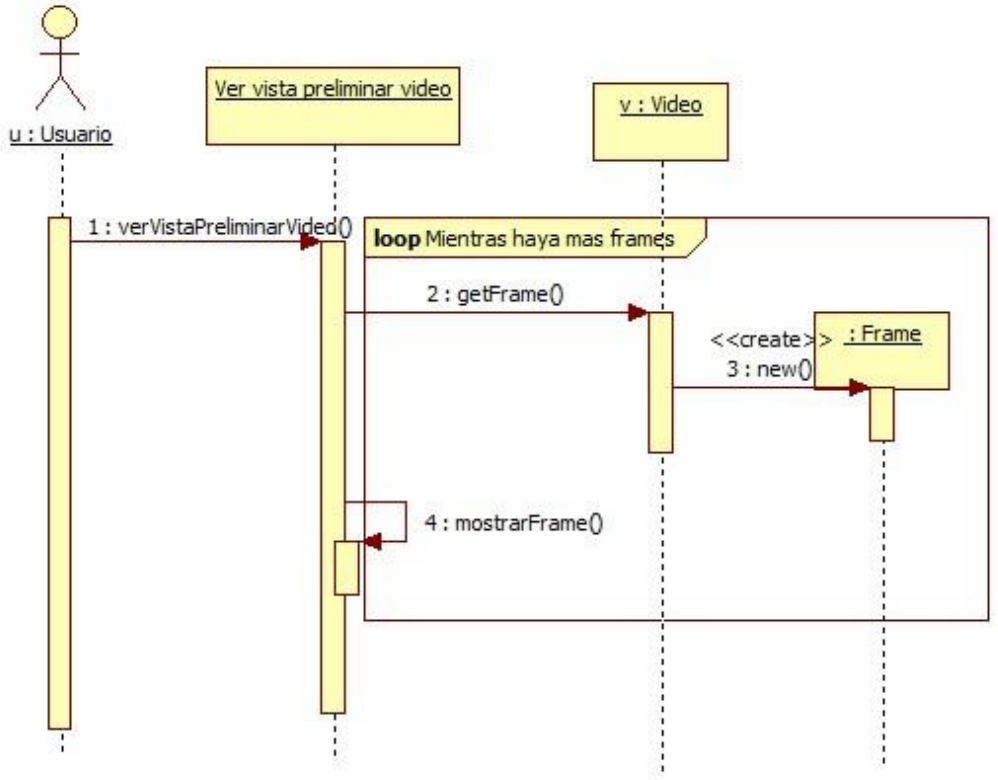


Ilustración 14-3 - Escenario "Ver vista preliminar video"

14.4. Diagrama de secuencia asociado al caso de uso "Play video" - Escenario "Analizar video"

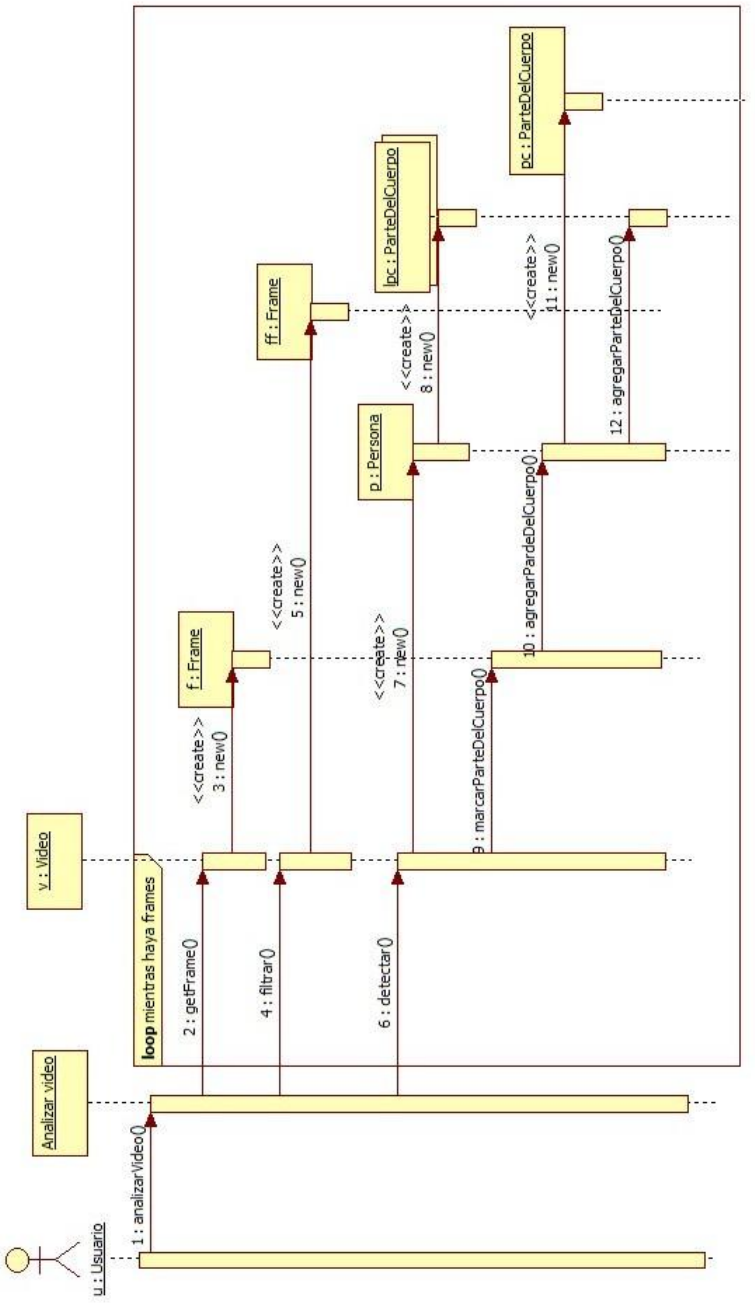


Ilustración 14-4 - Escenario "Analizar video"

14.5. Diagrama de secuencia asociado al caso de uso "Modificar configuración"

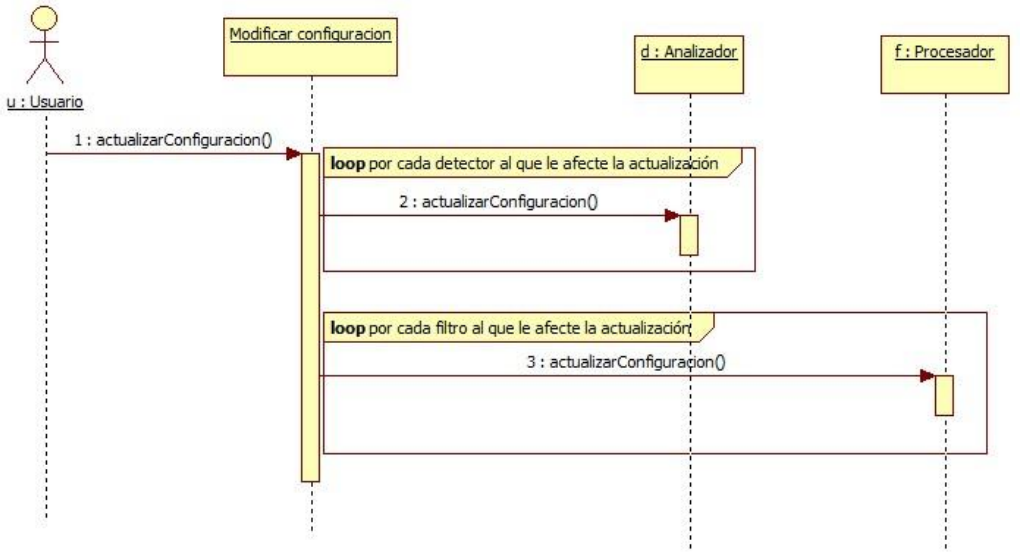


Ilustración 14-5 - Diagrama de secuencia de análisis "Modificar configuración"

Bloque V: Diseño

Capítulo 15. Introducción al diseño.

En este capítulo se va a proceder a exponer todo lo relacionado con el diseño del sistema de este proyecto. En él se explicarán los diferentes diagramas de secuencia y de clases, al mismo tiempo que se exponen los principales patrones de diseño empleados.

Capítulo 16. Arquitectura del sistema.

En todo proyecto informático es importante que los componentes construidos tengan una serie de características, tales como reusabilidad, alta cohesión, bajo acoplamiento, etc., que definen su la calidad como producto final. Por ello es importante que los pequeños cambios que se puedan producir dentro de una clase, no afecte al resto del sistema (o en su defecto, afecte lo mínimo posible). Con este propósito, se ha optado por una arquitectura de dos capas.

La primera capa corresponde a la capa de presentación, formada por las clases que implementan la interfaz gráfica mediante el uso de la librería Qt. Su funcionalidad es la de interactuar con el usuario para lanzar los casos de uso.

En la segunda capa agrupamos la lógica. Es la capa sobre la que se centra la mayor parte del esfuerzo empleado en nuestro trabajo, y por tanto la de mayor interés. Su tarea es realizar operaciones y transformaciones sobre la información de entrada (en el caso particular de este proyecto, nos referimos a frames de una secuencia de video) para generar un resultado útil (que en el caso que nos ocupa es el extracción de la silueta de un bailarín y la generación de un esqueleto que la resuma perteneciente a la secuencia de frames de entrada). Para el apoyo de esta capa, se utiliza la librería de visión computacional de nombre OpenCV, sobre la que se habló en el capítulo 21 de esta memoria.

Finalmente, teniendo en cuenta que el propósito de este software es el de mostrar la aplicación de una metodología desarrollada, así como la naturaleza y extensión de éste proyecto, se ha optado por no incluir funcionalidad de almacenamiento de resultados.

Capítulo 17. Interfaz del sistema.

17.1. Introducción

En este capítulo veremos el diseño asociado a la capa de vista del proyecto. En ella se podrá apreciar que no hay diferencias entre los distintos escenarios del caso de uso "Play video", ya que donde realmente se reflejan diferencias es en la capa de Lógica, encargada de devolver el frame procesado en función del escenario concreto.

17.2. Bosquejo

Se plantea un bosquejo que esboza el concepto de lo que se quiere plasmar en la interfaz.

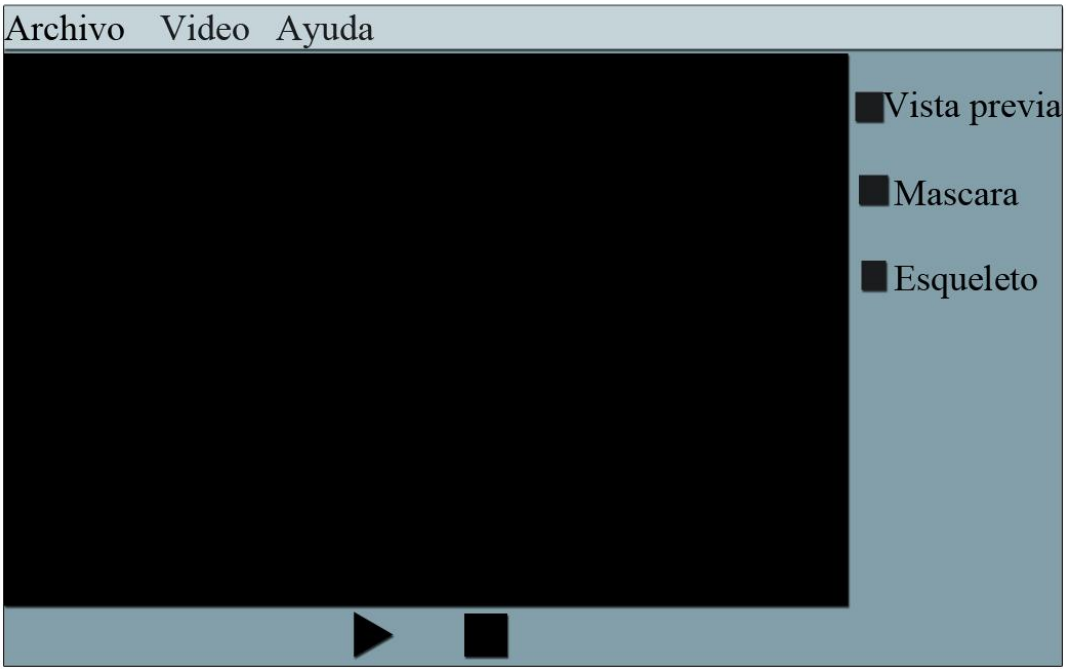


Ilustración 17-1 - Bosquejo de la ventana principal

El diseño propuesto adopta la metáfora y el aspecto de un reproductor de archivos de video clásico, con botones similares a los que en ellos se utilizan con el fin de minimizar el tiempo de adaptación y aprendizaje del usuario a la interfaz.

El video, como es costumbre, se abriría, como es costumbre, accediendo a la barra de menú, "Inicio-->Abrir". Las principales opciones de configuración se mostrarían en el lado derecho de la interfaz, con el fin de reducir el número de pasos que debe realizar el usuario, para una navegación más cómoda.

17.3. Diagrama de clases para la capa de Presentación

La fuente del sistema de comunicación entre clases en la capa de vista (así como el código que ejemplificaba la resolución al problema propuesto en el hilo de debate de la fuente, y que en gran parte ha sido utilizado para la Capa de Presentación de este software) está extraído de [Web06], siendo dicho contenido "Open Source". A continuación se muestra un diagrama de clases de la capa de Vista del sistema y su relación con el controlador de datos de la capa de Lógica:

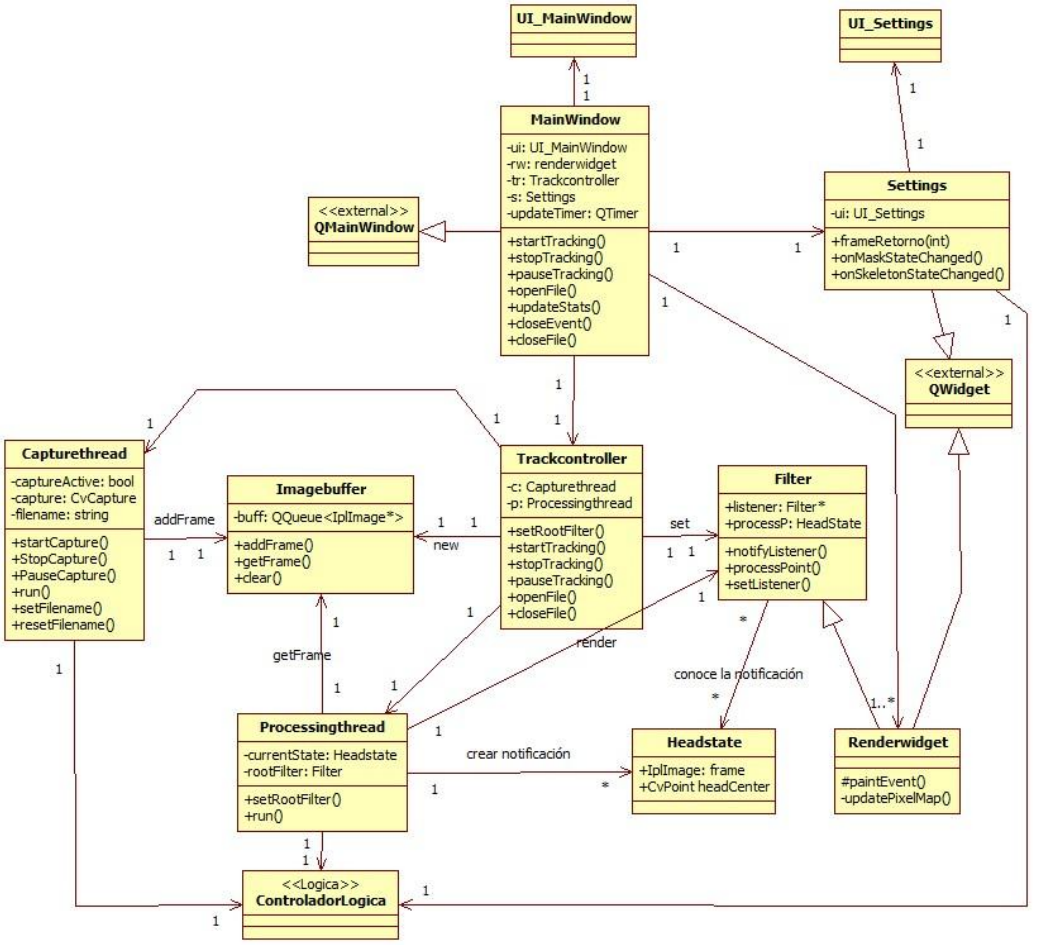


Ilustración 17-2 - Diagrama de clases de diseño de la capa de presentación

Las clases MainWindow y Setting corresponden a la gestión de eventos en las ventanas homónimas y tienen relación con dos clases "UI" que son generadas automáticamente por QT como código de carácter general para la gestión de dichas ventanas. El resto de clases de la capa de Vista se utilizan para la gestión de presentación de frames:

- Trackcontroller es el que coordina todo el proco, creando el buffer de frames. Éste crea dos hilos, Capturethread y Processingthread, además de establecer un filter, que es el encargado de presentar la imagen.
- Capturethread recupera frames de un archivo de video a través de la interfaz de OpenCV y los va publicando en un buffer (Imagebuffer).

- Processingthead se encarga de coordinar el procesado de dicha imagen, es decir, de solicitar a la capa de Lógica (a través de ControladorLogica) que procese dicho frame y le retorne otro ya procesado. El frame resultante es notificado al Filter dentro de un HeadState.
- Finalmente Renderwidget (Filter), encargado de dibujar por pantalla la imagen notificada, la transforma de IplImage (imagen del formato OpenCV) a QImage (imagen del formato de QT), reescala y la pinta.

17.4. Diagramas de secuencia para la capa de Presentación

17.4.1. Diagrama de secuencia de diseño asociado al caso de uso "Abrir video"

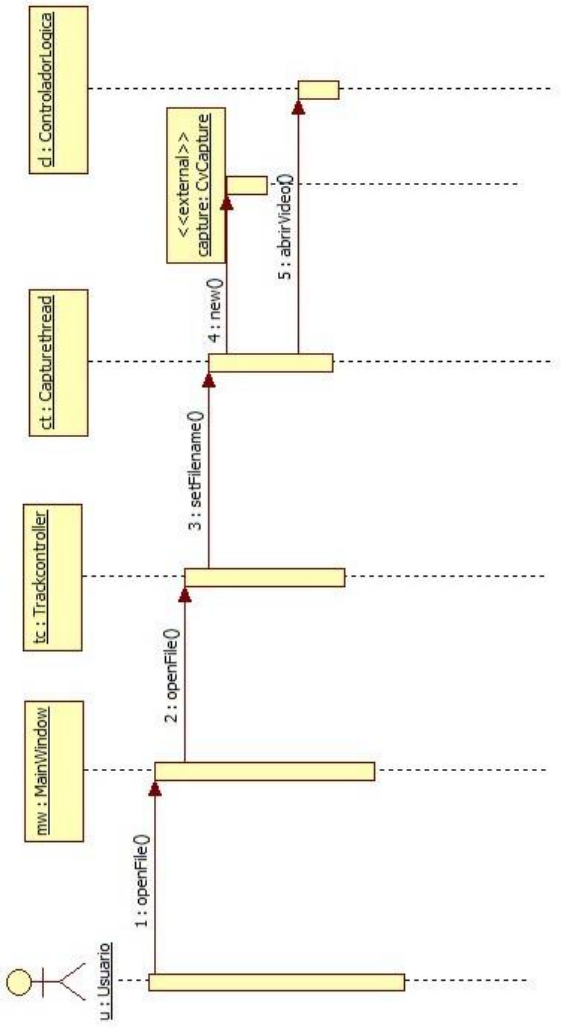


Ilustración 17-3 - Diagrama de secuencia de diseño de la capa de presentación "Abrir video"

17.4.2. Diagrama de secuencia común para los escenarios del caso de uso "Play video"

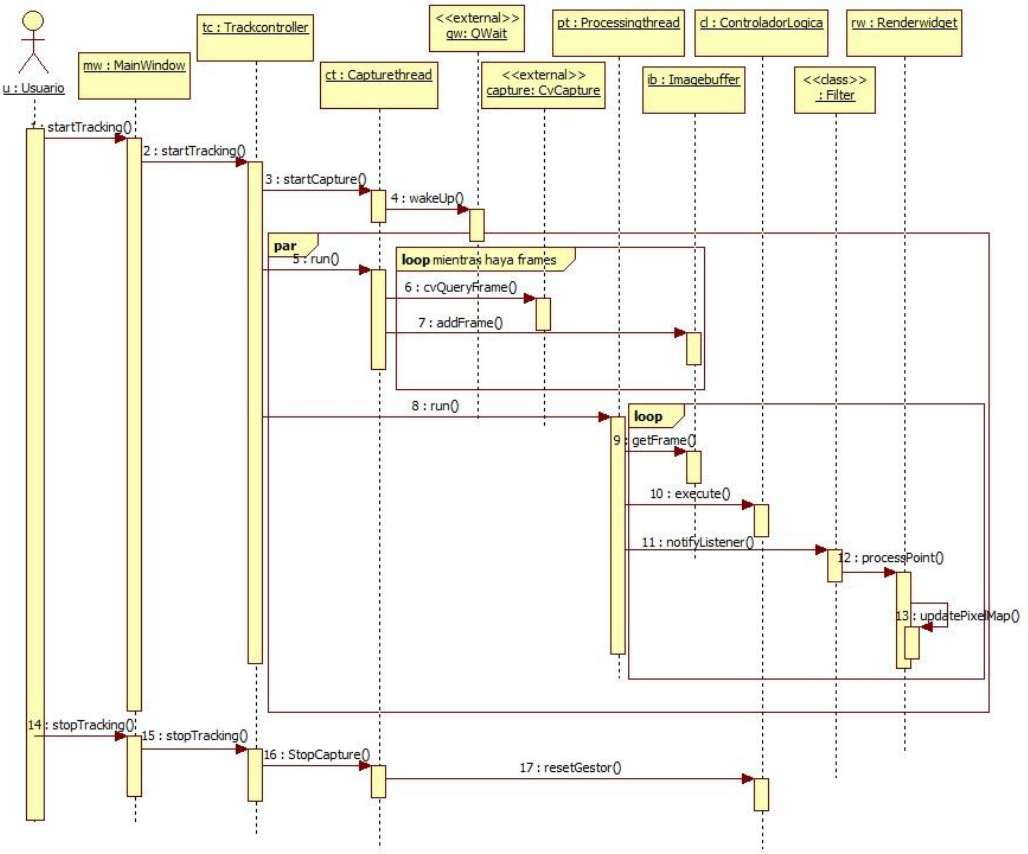


Ilustración 17-4 - Diagrama de secuencia de diseño de la capa de presentación "Play video"

17.4.3. Diagrama de secuencia asociado al caso de uso "Cerrar video"

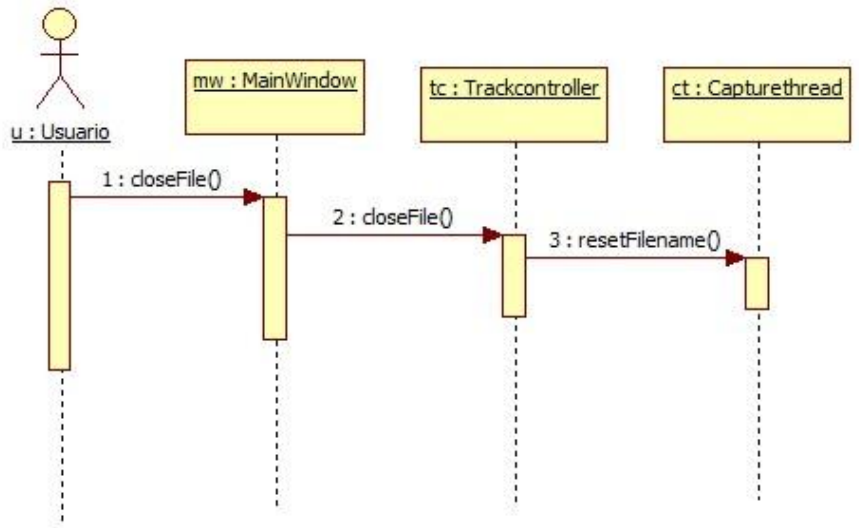


Ilustración 17-5 - Diagrama de secuencia de diseño de la capa de presentación "Cerrar video"

17.4.4. Diagrama de secuencia asociado al caso de uso "Modificar configuración" - Escenario "Mostrar mascara"

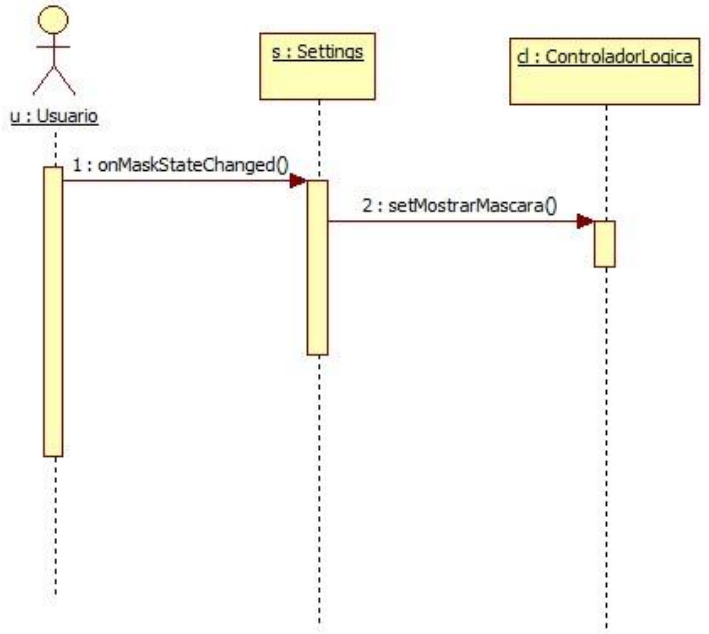


Ilustración 17-6 - Diagrama de secuencia de diseño de la capa de presentación "Modificar configuración" (1)

17.4.5. Diagrama de secuencia asociado al caso de uso "Modificar configuración" - Escenario "Mostrar esqueleto"

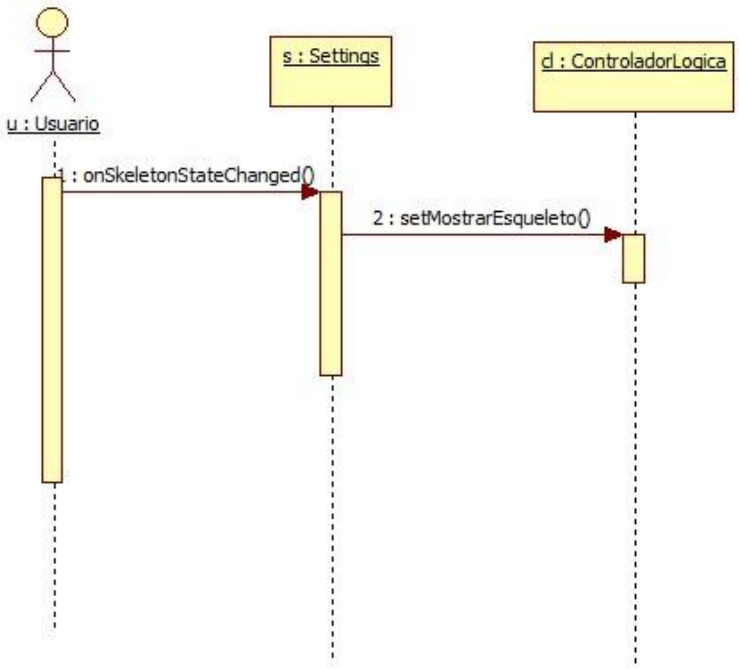


Ilustración 17-7 - Diagrama de secuencia de diseño de la capa de presentación "Modificar configuración" (2)

Capítulo 18. Diagrama de clases de diseño.

A continuación se presenta el diagrama de clase de diseño correspondiente a la capa de Lógica. Debido a la gran cantidad de atributos y métodos presentes en las clases, hemos visto oportuno mostrar primero un diagrama simplificado, en el que sólo vemos las clases y sus relaciones entre ellas, y detallar a parte el contenido de las clases.

18.1. Diagrama de clases

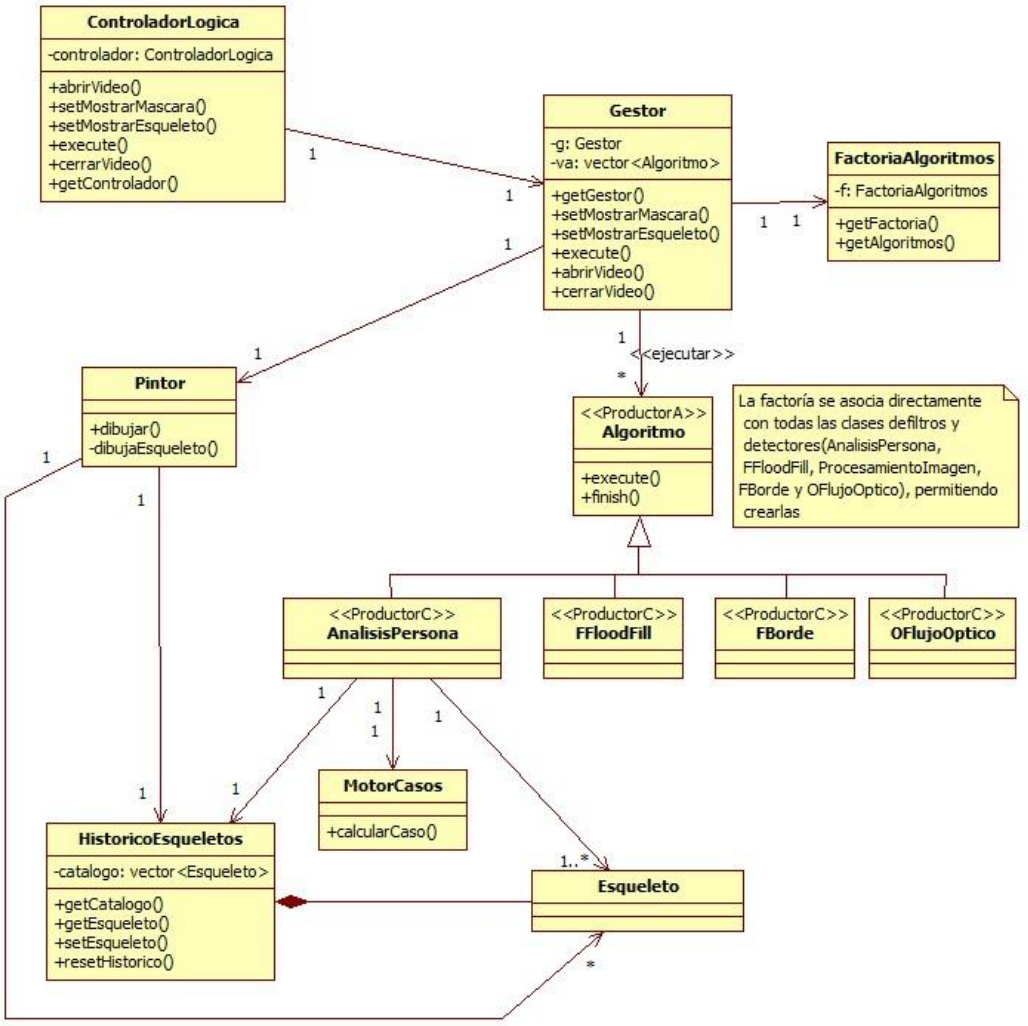


Ilustración 18-1 - Diagrama de clases de diseño de la capa de Lógica

En el diagrama de clases podemos observar la presencia de algunos patrones de diseño.

Por un lado implementamos el patrón controlador para un acceso uniforme en la capa de Lógica. El controlador de lógica, representado por la clase ControladorLogica, a su vez implementa un patrón singleton que garantiza una única instancia de dicha clase.

La clase que mas transformaciones ha sufrido con respecto a la etapa de análisis ha sido "Video". Se ha separado su semántica de su funcionalidad, quedando representada por la clase CVCapture (externa a este proyecto y perteneciente a la biblioteca OpenCV) y la capacidad de localizar a la persona e identificar sus características han quedado distribuidos en los diferentes algoritmos de inteligencia artificial que implementan el procesamiento y análisis que se realiza sobre los frames, con el fin de diseñar un software mas reutilizable, capaz de permitir probar al desarrollador diferentes estrategias de filtrado y reconocimiento de información, con la posibilidad futura de extender o modificar las prestaciones del análisis y del procesamiento de video sin tener que verse afectado el resto del código.

Para ello se ha implementado un patrón fachada, que podemos ver con más claridad en la siguiente figura:

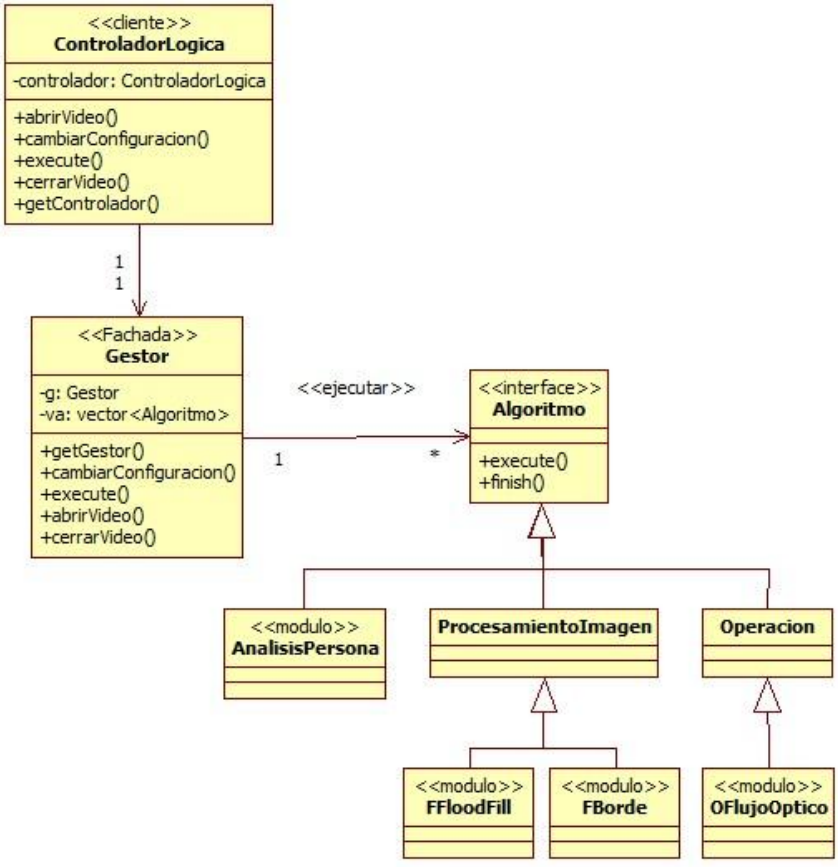


Ilustración 18-2 - Patrón fachada

Donde la clase ControladorLogica es el cliente, Gestor corresponde a la fachada, y FFloodFill, FBorde, OFlujoOptico y AnalisisPersona son las clases que típicamente se denominan módulos dentro del patrón fachada.

Para abstraer a la fachada de las diversas clases de algoritmos que se pueden ejecutar, se ha construido una interfaz que permita su acceso homogéneo en las múltiples llamadas que se le haga dentro del bucle principal de recepción de frame / tratamiento de frame, que queda mejor ilustrado en los diagramas de secuencia correspondientes.

Esta abstracción se complementa aplicando un patrón Factoría, que puede verse con más claridad en la siguiente figura:

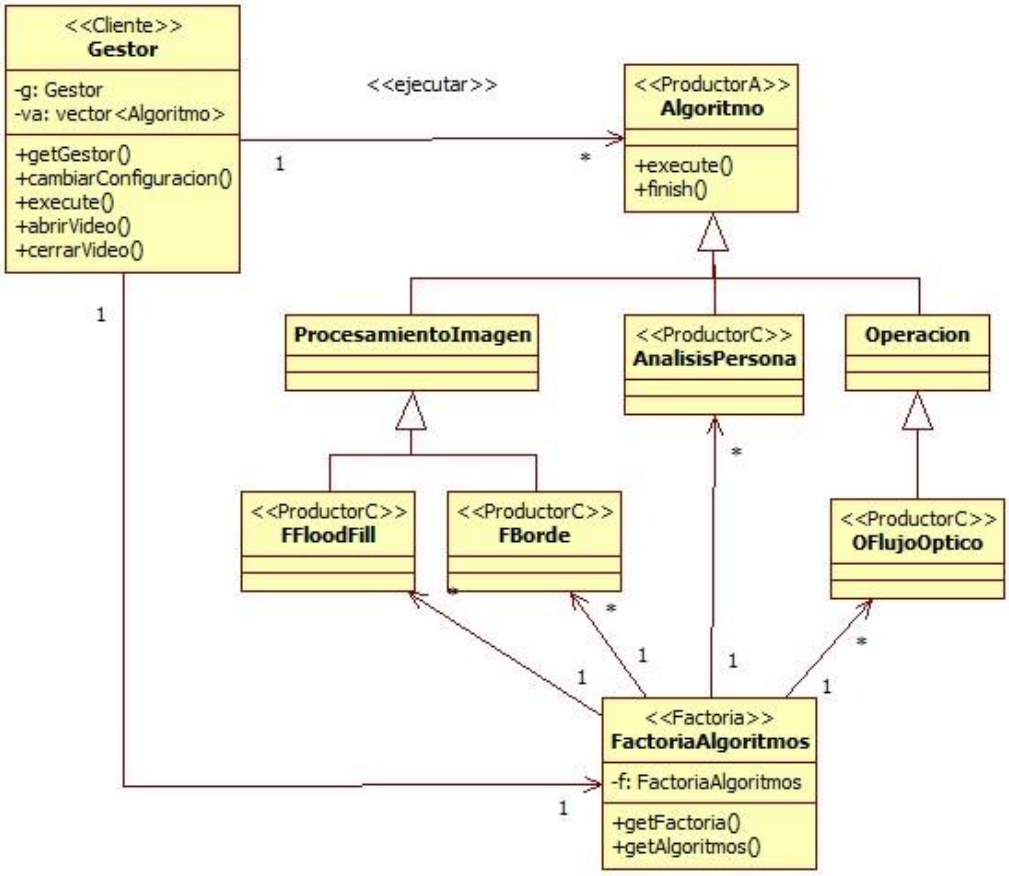


Ilustración 18-3 - Patrón factoría

Donde en esta ocasión, Gestor corresponde a la clase Cliente en el contexto del patrón factoría, FactoriaAlgoritmos actúa como la clase Factoría, mientras Algoritmo hace de ProductoAbstracto en el ejemplo típico del patrón factoría, y FFloodFill, FBorde, OFlujoOptico y AnalisisPersona son ProductoConcreto.

Con ello cumplimos el objetivo de crear una familia de algoritmos, abstrayendo completamente a Gestor de toda implementación de algoritmo.

18.2. Especificación de atributos y métodos de las clases del sistema

En este epígrafe nos disponemos a describir por orden alfabético las diversas clases de lógica que, por dimensión, habían sido resumidas en la [Ilustración 18-1] del diagrama del clases.

Algoritmo corresponde a una interface común a todos los algoritmos que se encargan de analizar el contenido del video. Como parámetros recibe la imagen del frame a analizar y un número que corresponde al orden que tiene ese frame con respecto al resto de frames que conforman el video. La salida que ofrece es un objeto de tipo Imagen, es decir, un IplImage. Estos algoritmos están también pensados para poder ser utilizados en bucle, es decir, para recibir un frame tras otro, con el fin de evaluar dichos elementos a lo largo del tiempo, con lo cual, también se incluye una función finish cuya función puede variar dependiendo del tipo de algoritmo que implemente la interfaz.

Ilustración 18-4 - Clase Algoritmo

La clase AnalisisPersona implementa la funcionalidad del proceso de análisis dentro del sistema. Su funcionalidad es la de generar el esqueleto del bailarín de los frames procesados que se le van enviando en el método implementado execute de la interfaz Algoritmo. Internamente utiliza dos estructuras especiales para transmitir datos, por un lado ExtremosRelativos, que contiene la lista de maximos y mínimos relativos asociados a una determinada función de valores discretos, y la estructura Persona, que contiene la información temporal del esqueleto mas sus proyecciones.

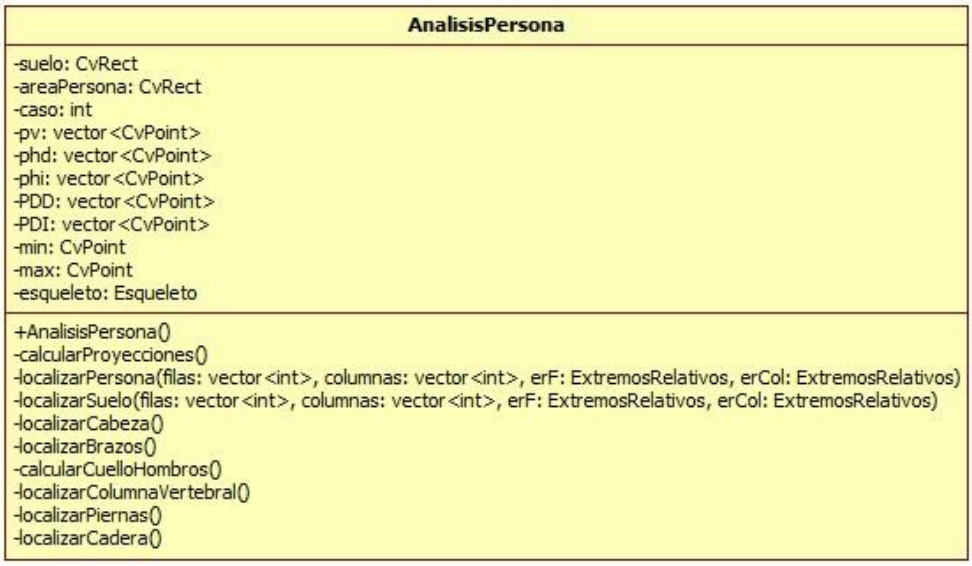


Ilustración 18-5 - Clase AnalisisPersona

La clase ControladorLogica sirve como interfaz para la capa de presentación que utiliza este sistema. Implementa un patrón Singleton y se comunica únicamente con Gestor.

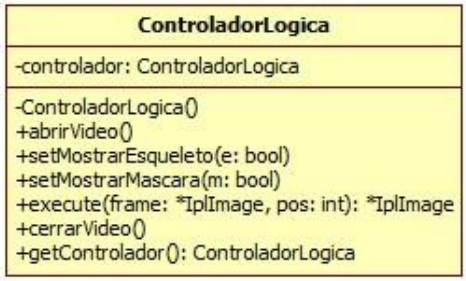


Ilustración 18-6 - Clase ControladorLogica

La clase Esqueleto almacena la información del esqueleto que genera AnalisisPersona.

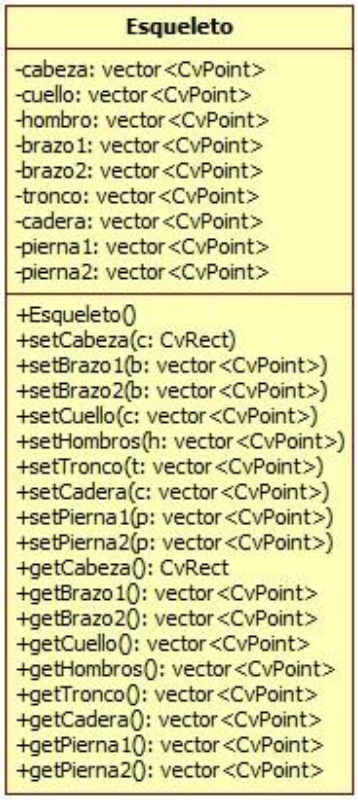


Ilustración 18-7 - Clase Esqueleto

La clase FactoriaAlgoritmos es la encargada de crear los objetos que implementan la interfaz Algoritmo. Implementa un patrón "Factoría".

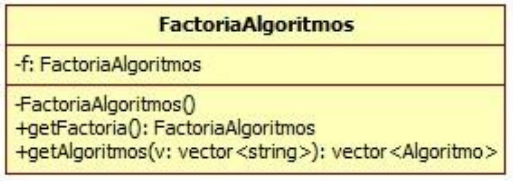


Ilustración 18-8 - Clase FactoriaAlgoritmos

La clase Gestor es la encargada de tramitar el proceso de estudio del frame que recibe como parámetro en el método execute, coordinando el procesamiento y análisis de la imagen en cuestión. Implementa el patrón factoría y fachada.

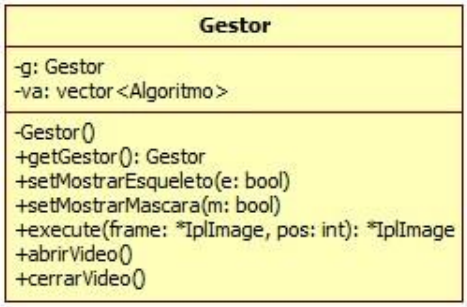


Ilustración 18-9 - Clase Gestor

La clase HistoricoEsqueletos abstrae de Esqueleto el sistema de gestión, almacenamiento y recuperación de Esqueleto's.

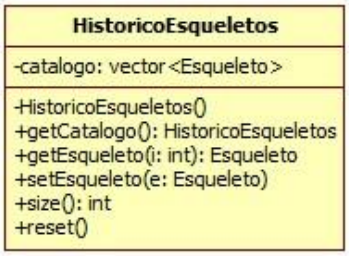


Ilustración 18-10 - Clase HistoricoEsqueletos

La clase MotorCasos es la encargada de identificar que caso se adapta mejor a los datos pasados como parámetro cuando se invoca el método calcularCaso.

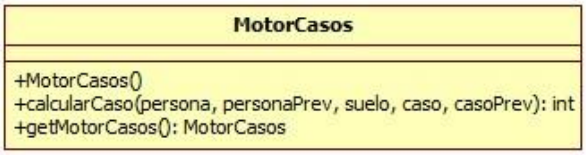


Ilustración 18-11 - Clase MotorCasos

La clase Pintor es la encargada de determinar que frame se va a mostrar posteriormente en la capa de presentación. Es la encargada de pintar el esqueleto en un objeto de tipo IplImage. El retorno puede ser una imagen a color sin pintar o con el esqueleto pintada, o una máscara de la imagen con o sin pintar el esqueleto.

Ilustración 18-12 - Clase Pintor

Capítulo 19. Interacción entre las clases del sistema.

19.1. Diagrama de secuencia asociado al caso de uso "Abrir video".

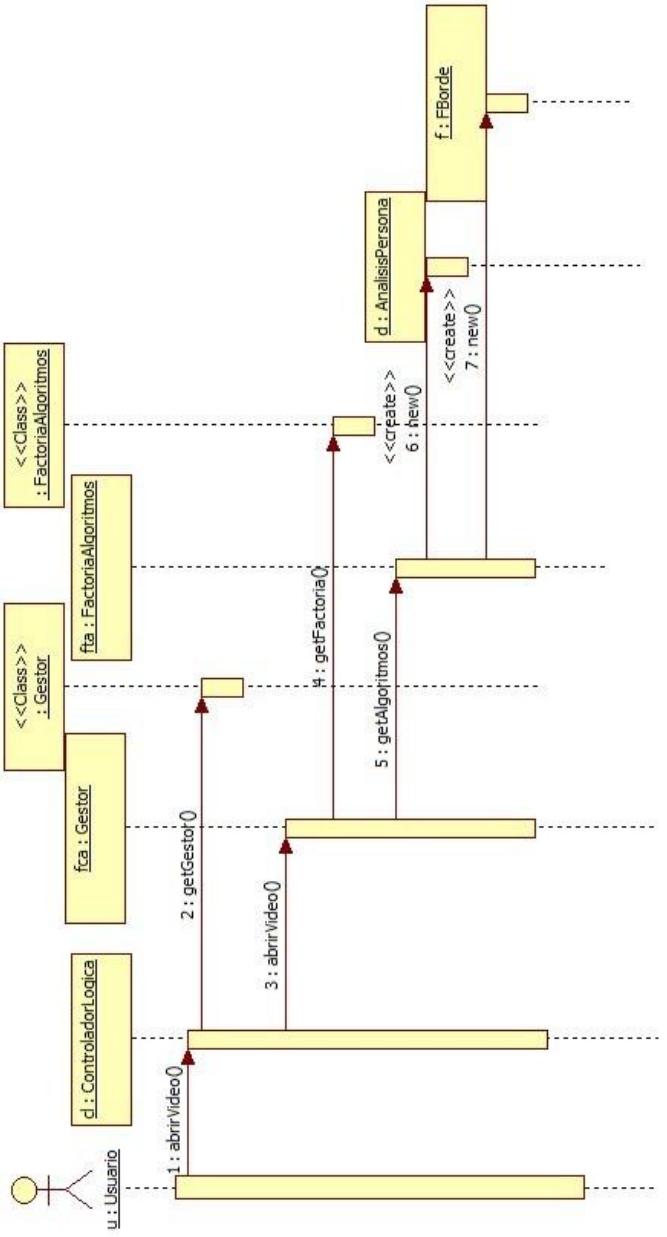


Ilustración 19-1 - Diagrama de secuencia "Abrir video"

19.2. Diagrama de secuencia asociado al caso de uso "Play video"

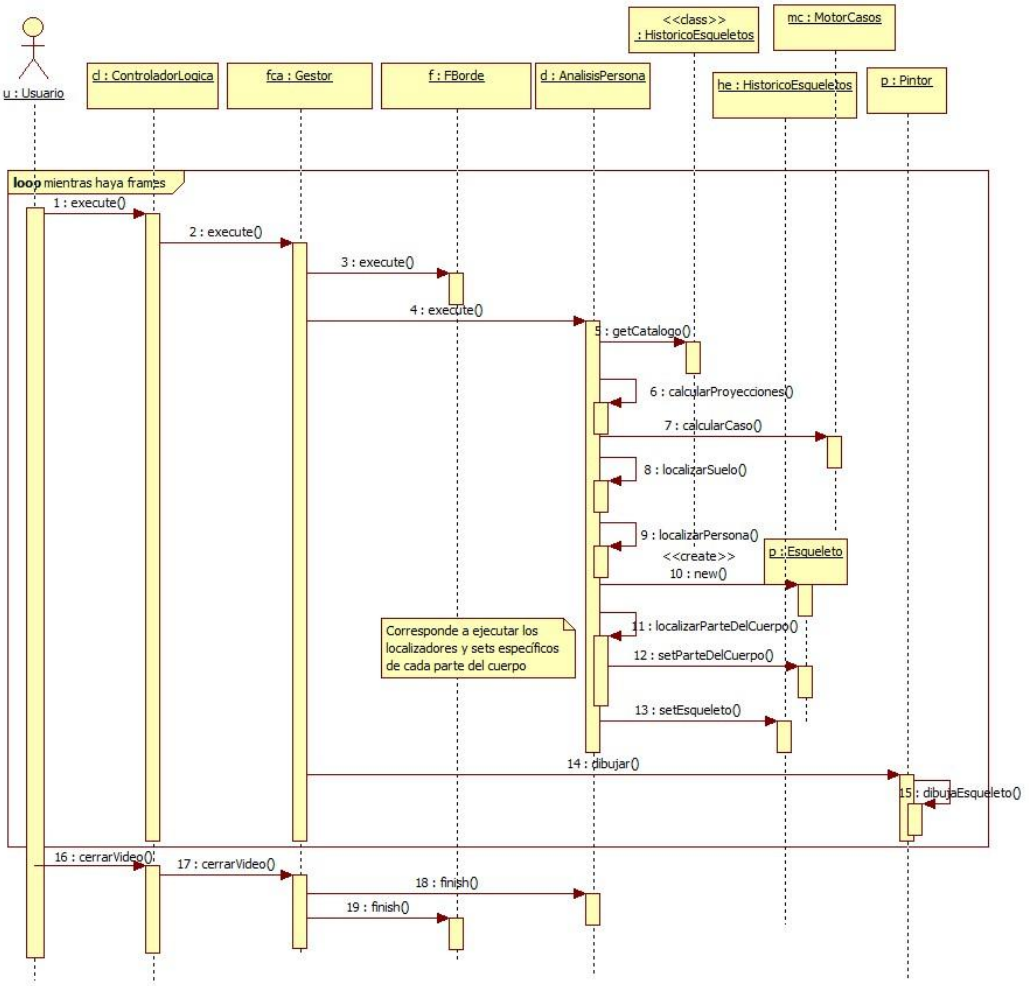


Ilustración 19-2 - Diagrama de secuencia "Play video"

19.3. Diagrama de secuencia asociado al caso de uso "Modificar configuración" - Escenario "Mostrar máscara"

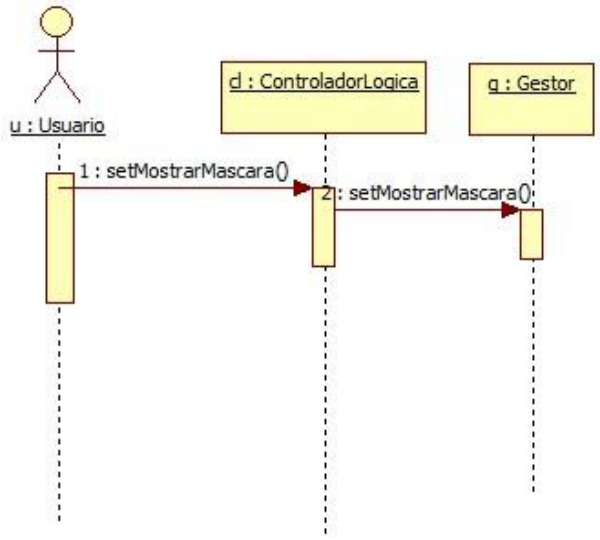


Ilustración 19-3 - Diagrama de secuencia "Mostrar máscara"

19.4. Diagrama de secuencia asociado al caso de uso "Modificar configuración" - Escenario "Mostrar esqueleto"

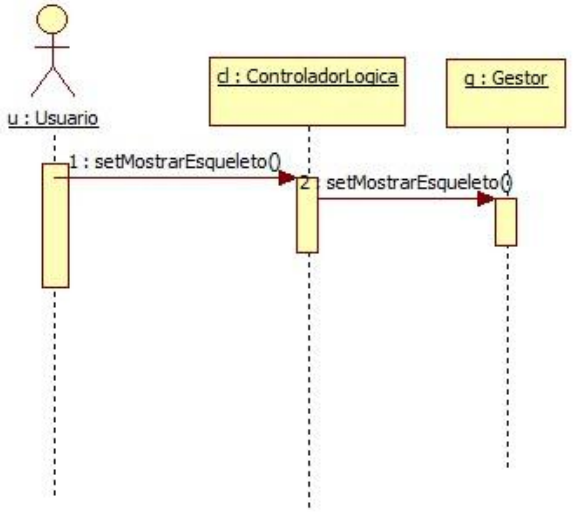


Ilustración 19-4 - Diagrama de secuencia "Mostrar esqueleto"

Bloque VI: Desarrollo

Capítulo 20. Resultados.

20.1. Relación de resultados del análisis de imágenes.

Los resultados obtenidos en el procesamiento del video "Prueba1.avi" de 101 frames con una iluminación aceptable son:

Prueba	Resultado
Tasa de aciertos en localizar el área suelo	97/101=0,96
Tasa de aciertos en localizar el área que ocupa la persona	101/101=1.0
Tasa de aciertos del motor de casos	98/101=0,97
Tasa de aciertos generando el esqueleto	93/101=0,92

Tabla 20-1 - Resultados de generación del esqueleto

20.2. Consumo de tiempo del sistema

A continuación se muestra una tabla con resultados de los datos de prueba de procesamiento y análisis de frames de tamaño 640x480 de un video de prueba. En ellos se muestra el tiempo consumido (en milisegundos) en realizar cada una de las principales tareas para un ordenador medio para después poder sacar conclusiones.

Filtro Global (ms)	Filtro compás(5) (ms)	Calcular caso (ms)	Generar Esqueleto (ms)	Tiempo Total (ms)
84,26	173,33	3,52	6,92	268,03

Tabla 20-2 - Promedio de tiempos de procesamiento (1)

En la siguiente tabla mostramos adicionalmente el tiempo promedio (en milisegundos) que se consume utilizando los 5 filtros de compás, el tiempo consumido si no se procesan los dos filtros de compás asociados a las proyecciones diagonales, el ancho y alto (medido en pixels) del área sobre el que se va ha realizado el cálculo de los filtros de compás, el valor del área propiamente dicho y cociente entre el tiempo que tardaron los dos filtros de compás para ver cuánto ha mejorado:

Filtro de compás(5) (ms)	Filtro de compás(3) (ms)	X (pixel)	Y (pixel)	Área (x·y) (pixel^2)	FC5/FC3
173,33	141,81	201	161,73	32602,50	1,22

Tabla 20-3 - Promedio de tiempos de procesamiento (2)

Como conclusiones, en el siguiente gráfico circular, puede observarse que la mayor parte del tiempo se invierte en calcular los 5 filtros de compás (64%), seguido de un 31% en el filtrado

global, y una insignificante proporción a la generación del esqueleto en función de los datos procesados. Esto es un buen dato, teniendo en cuenta que el cálculo de filtros de compás altamente paralelizable.

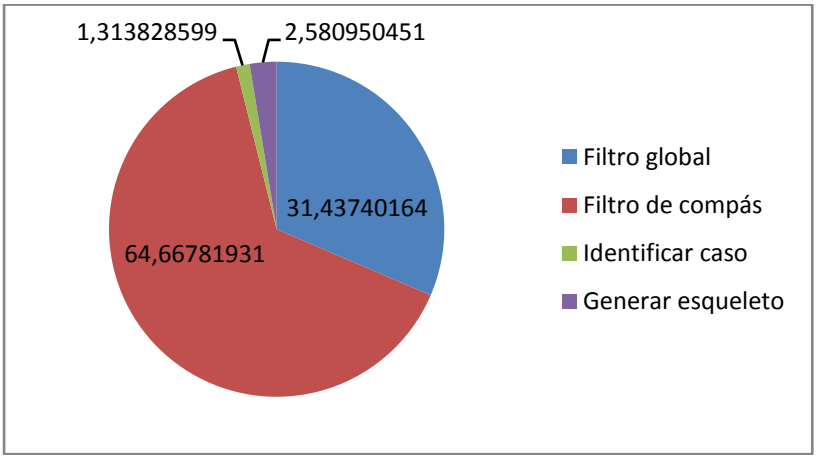


Ilustración 20-1 Porcentaje del consumo de tiempo de procesamiento

Por otro lado, queda claro que cuanto mayor sea el área del tronco humano (en pixels^2), mayor es el tiempo que se tarda en calcular las proyecciones, lo que indica que obviamente también puede reducirse el tiempo de procesamiento de forma proporcional, reduciendo definición de la imagen.

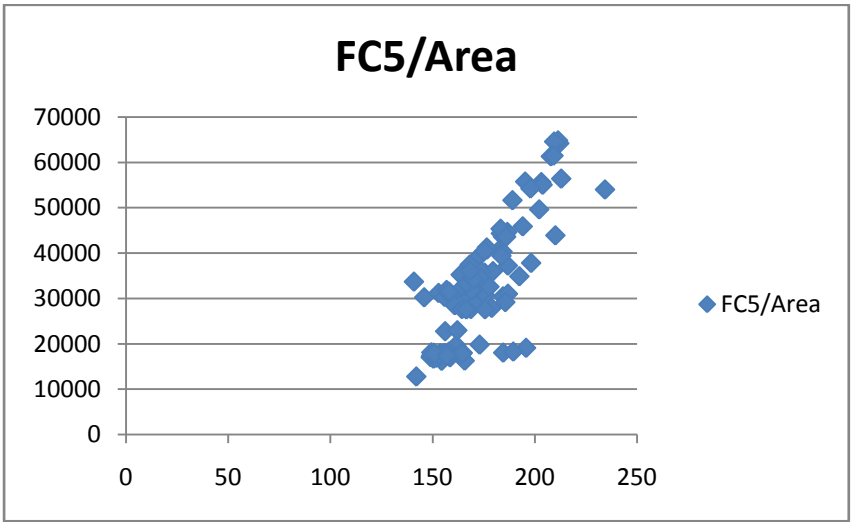


Ilustración 20-2 - Correlación entre tiempo consumido por el cálculo de filtros de compás y el área sobre el que se calcula

En cuanto a la ganancia de tiempo al quitar las dos proyecciones diagonales (que son las únicas opcionales ya que su papel es únicamente el de dar mayor estabilidad a los resultados), se observa que el aunque hay bastantes zonas en la que la reducción de tiempo es significativa, en general, parece ser que no son las proyecciones que más se tarda en calcular.

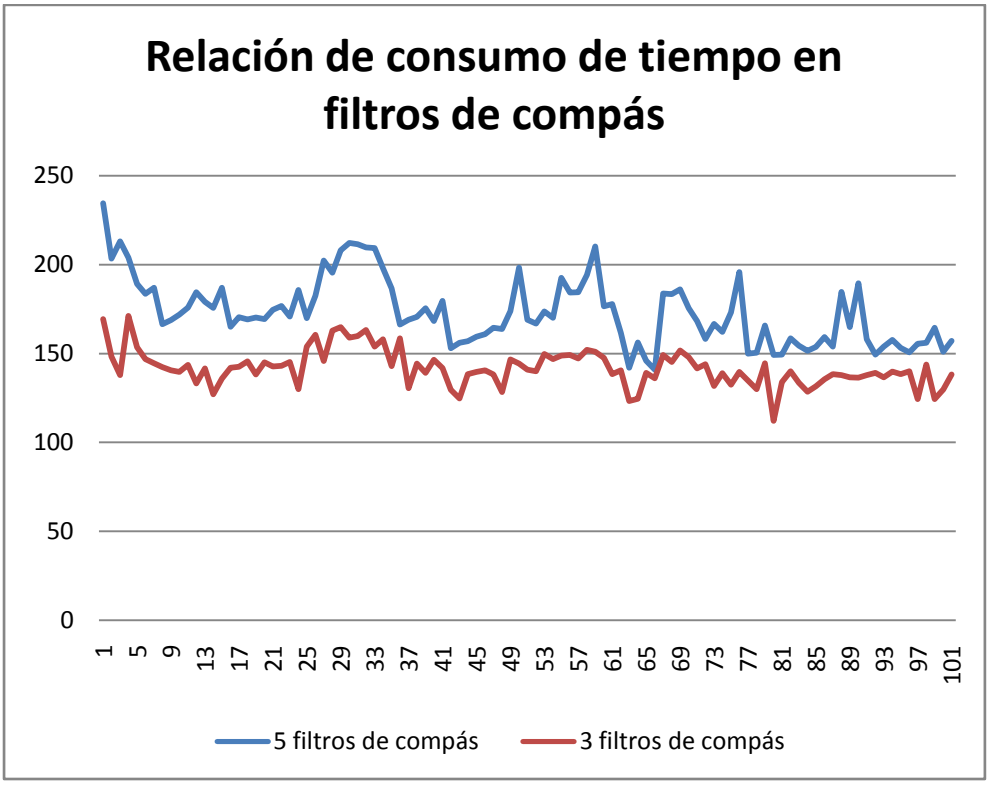


Ilustración 20-3 - Relación de tiempo consumido con 5 3 filtros de compás

Capítulo 21. Contexto tecnológico.

21.1. Entorno de trabajo

En este proyecto nos hemos decantado por el IDE Visual Studio 2008. Los motivos por los que hemos escogido éste y no cualquier otro son varios.

En primer lugar por ser un entorno el cual ya nos era familiar. Es un IDE realmente fácil para empezar a trabajar con él y muy intuitivo. También hay que indicar que es uno de los más extendidos en el mundo laboral. Además es muy fácilmente asociable con las bibliotecas que hemos usados, tales como OpenCV y Qt. Sin olvidad el hecho de que por ser alumnos de Ingeniería Técnica de Informática de Gestión podemos descargarlo de forma gratuita.

21.1.1. Visual Studio.

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

A partir de la versión 2005 Microsoft ofrece gratuitamente las *Express Edition*. Estas son varias ediciones básicas separadas por lenguajes de programación o plataforma enfocadas para novatos y entusiastas. Estas ediciones son iguales al entorno de desarrollo comercial pero sin características avanzadas.

Enunciaremos ahora las distintas versiones que han ido apareciendo:

Visual Studio 6.0: se lanzó en 1998 y fue la última versión en ejecutarse en la plataforma Win9x Los números de versión de todas las partes constituyentes pasaron a 6.0, incluyendo Visual J++ y Visual InterDev que se encontraban en las versiones 1.1 y 1.0 respectivamente. Esta versión fue la base para el sistema de desarrollo de Microsoft para los siguientes 4 años, en los que Microsoft migró su estrategia de desarrollo al .NET Framework.

Visual Studio 6.0 fue la última versión en que Visual Basic se incluía de la forma en que se conocía hasta entonces; versiones posteriores incorporarían una versión muy diferente del lenguaje con muchas mejoras, fruto de la plataforma .NET. También supuso la última versión en incluir Visual J++, que proporcionaba extensiones de la plataforma Java, lo que lo hacía incompatible con la versión de Sun Microsystems. Esto acarreó problemas legales a Microsoft, y se llegó a un acuerdo en el que Microsoft dejaba de comercializar herramientas de programación que utilizaran la máquina virtual de Java.

Aunque el objetivo a largo plazo de Microsoft era unificar todas las herramientas en un único entorno, esta versión en realidad añadía un entorno más a Visual Studio 5.0: Visual J++ y Visual

Interdev se separaban del entorno de Visual C++, al tiempo que Visual FoxPro y Visual Basic seguían manteniendo su entorno específico.

Visual Studio .NET (2002): en esta versión se produjo un cambio sustancial, puesto que supuso la introducción de la plataforma .NET de Microsoft. .NET es una plataforma de ejecución intermedia multilenguaje, de forma que los programas desarrollados en .NET no se compilan en lenguaje máquina, sino en un lenguaje intermedio (CIL - Common Intermediate Language) denominado Microsoft Intermediate Language (MSIL). En una aplicación MSIL, el código no se convierte a lenguaje máquina hasta que ésta se ejecuta, de manera que el código puede ser independiente de plataforma (al menos de las soportadas actualmente por .NET). Las plataformas han de tener una implementación de Infraestructura de Lenguaje Común (CLI) para poder ejecutar programas MSIL. Actualmente se pueden ejecutar programas MSIL en Linux y Mac OS X usando implementaciones de .NET que no son de Microsoft, tales como Mono y DotGNU.

Visual Studio .NET 2002 supuso también la introducción del lenguaje C#, un lenguaje nuevo diseñado específicamente para la plataforma .NET, basado en C++ y Java. Se presentó también el lenguaje J# -sucesor de J++- el cual, en lugar de ejecutarse en una máquina virtual de Java, se ejecuta únicamente en el framework .NET. El lenguaje Visual Basic fue remodelado completamente y evolucionó para adaptarse a las nuevas características de la plataforma .NET, haciéndolo mucho más versátil y dotándolo con muchas características de las que carecía. Algo similar se llevó a cabo con C++, añadiendo extensiones al lenguaje llamadas Managed Extensions for C++ con el fin de que los programadores pudieran crear programas en .NET. Por otra parte, Visual FoxPro pasa a comercializarse por separado.

Todos los lenguajes se unifican en un único entorno. La interfaz se mejora notablemente en esta versión, siendo más limpia y personalizable.

Visual Studio .NET puede usarse para crear programas basados en Windows (usando Windows Forms en vez de COM), aplicaciones y sitios web (ASP.NET y servicios web), y dispositivos móviles (usando el .NET Compact Framework).

Esta versión requiere un sistema operativo basado en NT. La versión interna de Visual Studio .NET es la 7.0

Visual Studio .NET 2003: esta supone una actualización *menor* de Visual Studio .NET. Se actualiza el .NET Framework a la versión 1.1. También se añade soporte con el fin de escribir aplicaciones para determinados dispositivos móviles, ya sea con ASP.NET o con el .NET Compact Framework. Además el compilador de Visual C++ se mejora para cumplir con más estándares, el Visual C++ Toolkit 2003.

Visual Studio 2003 se lanza en 4 ediciones: Academic, Professional, Enterprise Developer, y Enterprise Architect. La edición Enterprise Architect incluía una implentación de la tecnología de modelado Microsoft Visio, que se centraba en la creación de representaciones visuales de la arquitectura de la aplicación basadas en UML. También se introdujo "Enterprise Templates", para ayudar a grandes equipos de trabajo a estandarizar estilos de programación e impulsar políticas de uso de componentes y asignación de propiedades.

Microsoft lanzó el *Service Pack 1* para Visual Studio 2003 el 13 de Septiembre de 2006.

La versión interna de Visual Studio .NET 2003 es la 7.1 aunque el formato del archivo es 8.0.

Visual Studio 2005: esta versión se empezó a comercializar a través de Internet a partir del 4 de Octubre de 2005 y llegó a los comercios a finales del mes de Octubre en inglés. En castellano no salió hasta el 4 de Febrero de 2006. Microsoft eliminó *.NET*, pero eso no indica que se alejara de la plataforma .NET, de la cual se incluyó la versión 2.0.

La actualización más importante que recibieron los lenguajes de programación fue la inclusión de *tipos genéricos*, similares en muchos aspectos a las plantillas de C#. Con esto se consigue encontrar muchos más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible. C++ tiene una actualización similar con la adición de C++/CLI como sustituto de C# manejado.

Se incluye un diseñador de implantación, que permite que el diseño de la aplicación sea validado antes de su implantación. También se incluye un entorno para publicación web y pruebas de carga para comprobar el rendimiento de los programas bajo varias condiciones de carga.
Visual Studio 2005 también añade soporte de 64-bit. Aunque el entorno de desarrollo sigue siendo una aplicación de 32 bits Visual C++ 2005 soporta compilación para x86-64 (AMD64 e Intel 64) e IA-64 (Itanium). El SDK incluye compiladores de 64 bits así como versiones de 64 bits de las librerías.

Visual Studio 2005 tiene varias ediciones radicalmente distintas entre sí: Express, Standard, Professional, Tools for Office, y 5 ediciones Visual Studio Team System. Éstas últimas se proporcionaban conjuntamente con suscripciones a MSDN cubriendo los 4 principales roles de la programación: Architects, Software Developers, Testers, y Database Professionals. La funcionalidad combinada de las 4 ediciones Team System se ofrecía como la edición Team Suite.

Tools for the Microsoft Office System está diseñada para extender la funcionalidad a Microsoft Office.

Las ediciones Express se han diseñado para principiantes, aficionados y pequeños negocios, todas disponibles gratuitamente a través de la página de Microsoft se incluye una edición independiente para cada lenguaje: Visual Basic, Visual C++, Visual C#, Visual J# para programación .NET en Windows, y Visual Web Developer para la creación de sitios web ASP.NET. Las ediciones express carecen de algunas herramientas avanzadas de programación así como de opciones de extensibilidad.

Se lanzó el service Pack 1 para Visual Studio 2005 el 14 de Diciembre de 2006.
La versión interna de Visual Studio 2005 es la 8.0, mientras que el formato del archivo es la 9.0.

Visual Studio 2008: fue publicado (RTM) el 17 de Noviembre de 2007 en inglés, mientras que la versión en castellano no fue publicada hasta el 2 de Febrero de 2008.

El nuevo framework (.Net 3.5) está diseñado para aprovechar las ventajas que ofrece el nuevo sistema operativo "Windows Vista" a través de sus subsistemas "Windows Communication Foundation" (WCF) y "Windows Presentation Foundation" (WPF).El primero tiene como objetivo la construcción de aplicaciones orientadas a servicios mientras que el último apunta a la creación de interfaces de usuario más dinámicas que las conocidas hasta el momento.

A las mejoras de desempeño, escalabilidad y seguridad con respecto a la versión anterior, se agregan entre otras, las siguientes novedades.

La mejora en las capacidades de Pruebas Unitarias permiten ejecutarlas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos. Se incluye además un nuevo soporte para diagnosticar y optimizar el sistema a través de las herramientas de pruebas de Visual Studio. Con ellas se podrán ejecutar perfiles durante las pruebas para que ejecuten cargas, prueben procedimientos contra un sistema y registren su comportamiento; y utilizar herramientas integradas para depurar y optimizar.

Con Visual Studio Tools for Office (VSTO) integrado con Visual Studio 2008 es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario (UI) de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project. Una completa compatibilidad para implementación con ClickOnce garantiza el entorno ideal para una fácil instalación y mantenimiento de las soluciones Office.

Visual Studio 2008 permite incorporar características del nuevo Windows Presentation Foundation sin dificultad tanto en los formularios de Windows existentes como en los nuevos. Ahora es posible actualizar el estilo visual de las aplicaciones al de Windows Vista debido a las mejoras en Microsoft Foundation Class Library (MFC) y Visual C++. Visual Studio 2008 permite mejorar la interoperabilidad entre código nativo y código manejado por .NET. Esta integración más profunda simplificará el trabajo de diseño y codificación.

LINQ (Language Integrated Query) es un nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos, a través de extensiones para C++ y Visual Basic así como para Microsoft .NET Framework. Permite filtrar, enumerar, y crear proyecciones de muchos tipos y colecciones de datos utilizando todos la misma sintaxis, prescindiendo del uso de lenguajes especializados como SQL o XPath.

Visual Studio 2008 ahora permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework: 2.0. (Incluido con Visual Studio 2005), 3.0 (incluido en Windows Vista) y 3.5 (incluido con Visual Studio 2008).

.NET 3.5 incluye biblioteca ASP.NET AJAX para desarrollar aplicaciones web más eficientes, interactivas y altamente personalizadas que funcionen para todos los navegadores más populares y utilicen las últimas tecnologías y herramientas Web, incluyendo Silverlight y Popfly.

Actualmente ya se encuentra disponible la versión 2010.

21.2. Lenguaje de programación

En este proyecto y sobre todo por razones de compatibilidad con librerías hemos escogido el lenguaje de programación C++.

21.2.1. C++

C++ es un lenguaje de programación creado a mediados de los años ´80 por el danés Bjarne Stroustrup. La intención del creador fue la de proveer al lenguaje C de herramientas para la manipulación de objetos. Por este motivo y desde el punto de vista de los lenguajes orientados a objetos, se puede considerar C++ como un lenguaje híbrido.

Posteriormente se añadieron utilidades para la programación genérica, lo cual se sumó a los dos paradigmas que ya estaban admitidos (programación estructurada y programación orientada a objetos). Por este motivo, se suele decir que C++ es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

El nombre C++ fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "*C con clases*". En C++, la expresión "C++" significa "*incremento de C*" y se refiere a que C++ es una extensión de C.

21.3. OpenCV – Conceptos generales

En este capítulo vamos a hablar de los aspectos generales de la librería más importante usada en este proyecto, OpenCV, más concretamente de la versión 2.0, aunque ya ha sido lanzada la versión 2.1.

21.3.1. OpenCV.

OpenCV es una biblioteca libre de visión computacional originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es multiplataforma, Existiendo versiones para Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

Como objetivos, el proyecto pretende proveer un "Tool-Kit" o Marco de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multi-nucleo. OpenCV puede además utilizar el sistema las Primitivas de Rendimiento Integradas de Intel. Que es un conjunto de rutinas de bajo nivel específicas para procesadores Intel (IPP).

21.4. Qt

En este apartado vamos a hablar de Qt, una biblioteca la cual usamos para el desarrollo de la GUI (Interfaz Gráfica de Usuario). Actualmente se encuentra disponible la versión 4.6.3, sin embargo nosotros hemos utilizado la versión 4.6.2, ya que era la más reciente en el momento en que empezamos a desarrollar con la misma.

21.4.1. Qt.

Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores. Qt es utilizada principalmente en KDE, Google Earth, Skype, Qt Extended, Adobe Photoshop Album, VirtualBox y Opie. Es producido por la división de software Qt de Nokia, que entró en

vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008.

Qt es utilizada en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de *bindings*.

Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

Distribuida bajo los términos de GNU Lesser General Public License (y otras), Qt es software libre y de código abierto.

Inicialmente Qt apareció como biblioteca desarrollada por Trolltech (en aquel momento «Quasar Technologies») en 1992 siguiendo un desarrollo basado en el código abierto, pero no completamente libre. Originalmente permitía desarrollo de software cerrado mediante la compra de una licencia comercial, o el desarrollo de software libre usando la licencia Free Qt. Esta última no era una licencia real de software libre dado que no permitía redistribuir versiones modificadas de Qt.

Se usó activamente en el desarrollo del escritorio KDE (entre 1996 y 1998), con un notable éxito y rápida expansión, camino de convertirse en uno de los escritorios más populares de GNU/Linux.

Este hecho causaba preocupación desde el proyecto GNU, ya que veían como una amenaza para el software libre que uno de los escritorios libres más usados se apoyase en software propietario. Para contrarrestar esta situación se plantearon dos ambiciosas iniciativas: por un lado el equipo de GNU en 1997 inició el desarrollo del entorno de escritorio GNOME con GTK+ para GNU/Linux. Por otro lado se intentó hacer una biblioteca compatible con Qt pero totalmente libre, llamada Harmony.

En 1998 desarrolladores de KDE se reunieron con Trolltech para establecer la KDE Free Qt Foundation, que establecía que si Trolltech dejaba de desarrollar la versión gratuita y semi-libre de Qt la propia Fundación podría liberar la última versión publicada de la biblioteca Qt bajo una licencia tipo BSD.

Con la versión 2.0 se cambió a la licencia Q Public License, considerada de código abierto. Este cambio pretendía acallar las críticas a Qt y KDE que alegaban que no era software libre. Sin embargo, QPL no era compatible con la licencia GPL que usaba KDE, por lo que hubo voces que afirmaban que se estaba violando la licencia GPL al mezclar software QPL (la biblioteca Qt) con software GPL (KDE).

El 4 de septiembre de 2000, Trolltech comenzó a ofrecer la biblioteca Qt en su versión 2.1 bajo la licencia GPL en su versión para Linux. La versión para Mac OS X no se publicó bajo GPL hasta junio de 2003, mientras que la versión para Windows fue publicada bajo la licencia GPL en junio de 2005.

El 18 de enero de 2008, Trolltech anunció que también ofrecería Qt bajo la licencia GPL v3.

El 14 de enero de 2009, Nokia anunció que Qt v4.5 se licenciaría adicionalmente bajo la licencia LGPL 2.1, con el lema «Qt Everywhere».

Qt cuenta actualmente con un sistema de triple licencia: GPL v2/v3 para el desarrollo de software de código abierto y software libre, la licencia de pago QPL para el desarrollo de aplicaciones comerciales, y a partir de la versión 4.5 una licencia gratuita pensada para aplicaciones comerciales, LGPL.

Qt se encuentra disponible para sistemas tipo Unix con el servidor gráfico X Window System (Linux, BSDs, Unix), para Apple Mac OS X, para sistemas Microsoft Windows, para Linux embebido (en inglés *Embedded Linux*, para sistemas embebidos como PDA, Smartphone, etc.) y para dispositivos que utilizan Windows CE.

Qt Software anunció el 20 de octubre de 2008 una versión de Qt para la plataforma S60.
Adicionalmente también está disponible QSA (*Qt Scripts for Applications*), que, basándose en ECMAScript/JavaScript, permite introducir y crear scripts en las aplicaciones creadas con Qt.

Hay tres ediciones de Qt disponibles en cada una de estas plataformas, llamadas:
GUI Framework – edición con nivel reducido de GUI, orientado a redes y bases de datos.
Full Framework – edición completa comercial.
Open Source – edición completa Open Source.
Qt dispone de una serie de bindings para diversos lenguajes de programación, entre los que cabe destacar: PythonQt (Python), QT Jambi (Java), QtAda (Ada), FreePascal Qt4 (Pascal), PHP-QT (PHP), lqt (Lua).

21.5. Otros

En el capítulo presente hablaremos del resto de librerías y herramientas que, en un menor grado, también hemos requerido, como por ejemplo XML o los códecs de video.

21.5.1. Códecs de video.

Códec es la abreviatura de *codificador-decodificador*. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos (*stream*) o una señal. Los códecs pueden codificar el flujo o la señal (a menudo para la transmisión, el almacenaje o el cifrado) y recuperarlo o descifrarlo del mismo modo para la reproducción o la manipulación en un formato más apropiado para estas operaciones. Los códecs son usados a menudo en videoconferencias y emisiones de medios de comunicación.

La mayor parte de códecs provoca pérdidas de información para conseguir un tamaño lo más pequeño posible del archivo destino. Hay también codecs sin pérdidas (lossless), pero en la mayor parte de aplicaciones prácticas, para un aumento casi imperceptible de la calidad no merece la pena un aumento considerable del tamaño de los datos. La excepción es si los datos sufrirán otros tratamientos en el futuro. En este caso, una codificación repetida con pérdidas a la larga dañaría demasiado la calidad.

Muchos archivos multimedia contienen tanto datos de audio como de vídeo, y a menudo alguna referencia que permite la sincronización del audio y el vídeo. Cada uno de estos tres flujos de datos puede ser manejado con programas, procesos, o hardware diferentes; pero para que estos streams sean útiles para almacenarlos o transmitirlos, deben ser encapsulados juntos. Esta función es realizada por un formato de archivo de vídeo (contenedor), como .mpg, .avi, .mov, .mp4, .rm, .ogg, .mkv o .tta. Algunos de estos formatos están limitados a contener streams que se reducen a un pequeño juego de códecs, mientras que otros son usados para objetivos más generales.

Un endec es un concepto similar (pero no idéntico) para el hardware.

Un códec de video es un tipo de códec que permite comprimir y descomprimir video digital. Normalmente los algoritmos de compresión empleados conllevan una pérdida de información.

El problema que se pretende acometer con los códec es que la información de video es bastante ingente en relación a lo que un ordenador normal es capaz de manejar. Es así como un par de segundos de video en una resolución apenas aceptable puede ocupar un lugar respetable en un medio de almacenamiento típico (disco duro, Cd, Dvd) y su manejo (copia, edición, visualización) puede llevar fácilmente a sobrepasar las posibilidades de dicho ordenador o llevarlo a su límite.

Es así como se ha preferido construir y ocupar estos algoritmos de compresión y descompresión en tiempo real: Los códec. Su finalidad es obtener un almacenamiento sustancialmente menor de la información de vídeo. Esta se comprime en el momento de guardar la información hacia un archivo y se descomprime, en tiempo real, durante la visualización. Se pretende, por otro lado, que el proceso sea transparente para el usuario, es decir, que no intervenga o lo haga lo menos posible.

Existe un complicado equilibrio entre la calidad de video, la cantidad de datos necesarios para representarlo (también conocida como tasa de bits), la complejidad de los algoritmos de codificación y decodificación, la robustez frente a las pérdidas de datos y errores, la facilidad de edición, la posibilidad de acceder directamente a los frames, y otros factores.

Bloque VII: Conclusiones

Capítulo 22. Conclusiones.

El área de las aplicaciones basadas en contenido multimedia, y en concreto las aplicaciones de visión computacional son cada vez más importantes y comunes para la sociedad. No obstante, el desarrollo de las mismas plantea problemas muy interesantes para su estudio.

La idea inicial de este proyecto era construir una aplicación capaz de reconocer pasos de baile de un bailarín en movimiento en un video de una sola cámara móvil. . No obstante, esta tarea era muy ambiciosa para la extensión del trabajo que se quería presentar, por lo que desde la concepción de este proyecto esta idea fue reducida a encontrar y describir una metodología que permitiese "identificar las características que definían los movimientos del bailarín" para su tratamiento informático y la creación de una "demo" que mostrase las capacidades técnicas que ofrecía la metodología. Se realizó una búsqueda de posibles ideas que pudiesen resolver el problema a tratar, y de las pocas candidatas, se optó por definir un sistema basado por casos capaz de generar una pseudo-representación esqueletal de la silueta del bailarín por su intuitiva interpretación, rápido post-procesamiento y por ser un área no tan explotada como otras como el estudio de poligonales.

El proyecto seguía siendo ambicioso, tanto por la exigencia del mismo como por motivos personales, se tuvo que prorrogar el tiempo de desarrollo, permitiendo desarrollar las ideas e implementarlas más en detalle la metodología que se desarrollaría a lo largo del proyecto.

Como resultado, se consiguió desarrollar un sistema de filtrado de imágenes que permitía extraer al bailarín del fondo a partir del detector de bordes Canny y una umbralización del frame. Sin embargo adolecía de dos inconvenientes que no pudieron ser resueltos, por un lado, que el filtrado es específico para ese video, y que no es capaz de extraer completamente al bailarín, ya que los pantalones tenían una características radiométricas iguales a las del fondo. Dado que no podíamos mejorar el filtrado (por el significativo coste computacional que requeriría), tuvo que mejorarse la metodología en la fase de analizado del resultado de filtrar la imagen.

Para las pruebas se eligieron cuatro fragmentos de video de dimensiones 640x480 pixels en los que sólo apareciese un único bailarín, que ocupase un área de tamaño suficientemente significativo en la imagen y en condiciones de iluminación no excesivamente desfavorables (no en penumbra). Los resultados fueron sorprendentes. En la mayor parte de los frames, el sistema no solamente identificaba el área que ocupaba el suelo, el bailarín y la postura que éste adoptaba, sino que también se podía seguir el movimiento de las diferentes partes del cuerpo, e incluso aún cuando se producían auto-oclusiones de unas partes del cuerpo con otras.

En cuanto al rendimiento del proceso, no ha sido optimizado paralelizando parte del código, pero los resultados permiten procesar al menos a una tasa de unos 3-4 frames por segundo en un ordenador medio, teniendo en cuenta que gran parte del código que consume ese tiempo es altamente paralelizable, lo que nos hace ser bastante optimistas de cara al futuro de este trabajo.

Capítulo 23. Trabajo futuro.

Cumplidos los objetivos propuestos, queda claro que el proyecto realizado abre nuevas líneas de trabajo de gran interés.

A. En primer lugar, tras los datos de tiempo obtenidos, se ha observado que se puede mejorar mucho el rendimiento del programa paralelizando ciertos procesos (filtros de compás, procesamiento global, etc) dentro del algoritmo, permitiendo mejorar el ratio de frames por segundo.
B. A lo largo de varios capítulos se ha comentado que el presente proyecto permitía generar un esqueleto del bailarín para seguir sus movimientos; sin embargo, la interpretación de secuencias de posturas y aprendizaje de pasos de baile fue descartado de este proyecto por ser un objetivo demasiado ambicioso para el tiempo de desarrollo que se le adjudicó., quedando como trabajo futuro.
C. Algunas escenas del video, del que parte este estudio, no se han podido procesar debido a la enorme dificultad que presentaban, debido a problemas de iluminación, excesivo efecto de zoom-out, captura parcial del cuerpo del bailarín, etc. Sería de especial interés, crear un sistema automático que pudiese descartar aquellos frames que no van a poder ser procesados.
D. Queda pendiente estudiar cómo mejorar el filtrado global para extrapolar los resultados a otros contextos más complejos (entornos urbanos, p.e.).
E. El trabajo actual contempla únicamente el caso en el que sólo hay un bailarín en escena, pero sería deseable que se pudiesen generar el esqueleto de más de un individuo por escena, aunque esto plantea nuevas dificultades de enorme complejidad.
F. También sería interesante ver si se pueden relajas algunas de las suposiciones iniciales (el individuo está de pie, p.e.) con las que trabaja el proceso de análisis de imagen implementado.
G. El actual proyecto se ha centrado en probar el sistema para imágenes de tamaño 640x480 y ocasionalmente imágenes de tamaño 1080x720. Sería extremadamente interesante comprobar cómo se comporta el algoritmo para diferentes tamaño, con el fin de evaluar la dependencia que exista con el rendimiento.

Bloque VIII: Bibliografía

Capítulo 24. Referencias.

24.1. Bibliografía.

[Agg94] J.K. Aggarwal, Q. Cai, W. Liao, B. Sabata: "Articulated and elastic non-rigid motion: a review", Proceedings of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects, 2–14, 1994

[Agg99] J.K. Aggarwal and Q. Cai: "'Human motion analysis: a review", Comput. Vision Image Understanding 73 (3), 428–440, 1999.

[Alo08] Juan Manuel Alonso Weber, Araceli Sanchis de Miguel (2008), Un Nuevo Modelo Autoorganizado Aplicado a la Resolución de Problemas de Geometría Computacional.

[Apa05] Tatiana Aparicio Vergara, Cristóbal Sebastián González Dixon (2005), Sistema de Reconocimiento de Posturas del Cuerpo Humano.

[Bau00] A. Baumberg (2000), Reliable feature matching acrosswidely separated views.

[Bea97] P. Beardsley, A. Zisserman, D. W. Murray (1997), Sequential updating of projective and affine structure from motion.

[Bla93] A. Blake, R. Curwen, A. Zisserman (1993), A framework for spatiotemporal control in the tracking of visual contours.

[Blu67] H. Blum (1967), A transformation for extracting new descriptors of shape.

[Bra93] K. J. Bradshaw, P. F. McLauchlan, I. D. Reid, D. W. Murray (1993), Saccade and pursuit on an active head/eye platform.

[Bra08] G. Bradski, A. Kaehler (2008), *Learning OpenCV*, O'Reilly.

[Bur89] P. J. Burt, J. R. Bergen (1989), Object tracking with a moving camera.

[Can86] Canny, J., (1986), *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence.

[Ced95] C. Cedras, M. Shah: "Motion-based recognition: a survey", Image Vision Comput. 13 (2), 129–155., 1995

[Dim05] M. Dimitrijevic, V. Lepetit and P. Fua (2005), Human Body Pose Recognition Using Spatio-Temporal Templates.

[Har88] C. Harris and M. Stephens (1988), A combined corner and edge detector, Fourth Alvey Vision Conference.

[Har92] C. Harris (1992), Tracking with rigid models.

[Ke04] Y. Ke, R. Sukthankar (2004), PCA-SIFT: A more distinctive representation for local image descriptors.

[Laz03] S. Lazebnik, C. Schmid, J. Ponce (2003), Sparse texture representation using affine-invariant neighborhoods.

[Kol94] D. Koller, J. Weber, J. Malik (1994), Robust multiple car tracking with occlusion reasoning.

[Let93] J. M. Letang, V. Rebuffel, P. Bouthemy (1993), Motion detection robust to perturbations: a statistical regularization and temporal integration framework.

[Low99] D.G. Lowe (1999), Object recognition from local scale-invariant features.

[Low04] D. G. Lowe (2004), Distinctive image features from scale-invariant keypoints.

[Luc81] B. D. Lucas, T. Kanade (1981), *An iterative image registration technique with an application to stereo vision. Proceedings of Imaging understanding workshop*.

[Mcl92] P. F. McLauchlan, I. D. Reid, D. W. Murray (1992), Coarse motion for saccade control.

[Mcl95] P. F. McLauchlan, D. W. Murray (1995), A unifying framework for structure and motion recovery from image sequences.

[Mcl97] P. McLauchlan, J. Malik (1997), Vision for longitudinal control.

[Mik01] K. Mikolajczyk, C. Schmid (2001), Indexing based on scale invariant interest points.

[Mik05] K. Mikolajczyk, C. Schmid (2005), A performance evaluation of local descriptors.
[Mor04] Greg Mori, Xiaofeng Ren, Alexei A. Efros, Jitendra Malik (2004), Recovering Human Body Configurations: Combining Segmentation and Recognition.
[Ogn92] R. Ogniewicz, M. Ilg (June 1992), Voronoi Skeletons: Theory and Applications.
[Pah93] K. Pahlavan, T. Uhling, J. O. Eklund (1993), Dynamic fixation.
[Ro01] Y. M. Ro, M. Kim, H. K. Kang, B. S. Manjunath, J. Kim (2001), MPEG7 homogeneous texture descriptor.
[Sch00] C. Schmid, R. Mohr, C. Bauckhage (2000), Evaluation of interest point detectors.
[Sco92] Van Scoy, Roger L. Software Development Risk (September 1992), Opportunity, Not Problem. Software Engineering Institute, CMU/SEI-92-TR-30, ADA 258743.
[Sha95] L. S. Shapiro, A. Zisserman, J. M. Brady (1995), 3D motion recovery via affine epipolar geometry.
[Shi94] J. Shi, C. Tomasi (1994), Good Features to Track.
[Smi95] S. M. Smith (1995), ASSET 2: Real-Time Motion segmentation and Shape Tracking.
[Tuy00] T. Tuytelaars, L. V. Gool (2000), Wide baseline stereo matching based on local, affinely invariant regions.
[Wan03] L.Wang, W.Hu, and T.Tan: Recent developments in human motion analysis, Pattern Recognition 36, 585 – 601, 2003
[Yan10] Weilong Yang, Yang Wang, and Greg Mori (2010), Recognizing Human Actions from Still Images with Latent Poses.
[Zab94] R. Zabih, J. Woodfill (1994), Non-parametric local transforms for computing visual correspondence.
[Zha92] Z. Zhang, O. Fuageras (1992), Dynamic Scene Analysis.
[Zil03] S. Zillner, M. Gelautz, M. Kallinger: "THE RIGHT MOVE" – A CONCEPT FOR A VIDEO-BASED CHOREOGRAPHY TOOL, ISPRS 2003

De las cuales queremos destacar como lectura especialmente recomendada:

[Alo08] Juan Manuel Alonso Weber, Araceli Sanchis de Miguel (2008), Un Nuevo Modelo Autoorganizado Aplicado a la Resolución de Problemas de Geometría Computacional.
[Agg94] J.K. Aggarwal, Q. Cai, W. Liao, B. Sabata: "Articulated and elastic non-rigid motion: a review", Proceedings of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects, 2–14, 1994
[Mcl97] P. McLauchlan, J. Malik (1997), Vision for longitudinal control.
[Mor04] Greg Mori, Xiaofeng Ren, Alexei A. Efros, Jitendra Malik (2004), Recovering Human Body Configurations: Combining Segmentation and Recognition.
[Sch00] C. Schmid, R. Mohr, C. Bauckhage (2000), Evaluation of interest point detectors.
[Yan10] Weilong Yang, Yang Wang, and Greg Mori (2010), Recognizing Human Actions from Still Images with Latent Poses.

24.2. Webgrafía.

[Web01] Seminario Grupo de Topología Computacional y Matemática Aplicada. Última visita: 18-07-2012.
http://alojamientos.us.es/gtocoma/pid/tema6.pdf
[Web02] Documentación oficial de QT. Última visita: 18-07-2012.
http://doc.qt.nokia.com/4.6/classes.html
[Web03] Información sobre la instalación de QT. Última visita: 18-07-2012.
http://doc.trolltech.com/4.6/install-win.html
[Web04] Documentación oficial de OpenCV. Última visita: 18-07-2012.

http://opencv.willowgarage.com/documentation/

[Web05] Viaclinica. Captura de movimiento aplicado a biomecánica. Última visita: 18-07-2012. http://viaclinica.com/article.php?pmc_id=1513229

[Web06] Foro de QT donde se explica cómo recomiendan implementar la interacción entre OpenCV y QT. Última visita: 18-07-2012. http://www.qtcentre.org/threads/11655-OpenCV-integration

Bloque IX: Anexos

A. Instalación.

A.1: Introducción.

En este anexo describiremos cómo instalar las diferentes herramientas usadas para el desarrollo de esta aplicación en una máquina Windows, comenzando por los codecs de video necesarios para la correcta visualización de los mismos, pasando por Visual Studio y terminando con las diferentes librerías necesarias para el correcto funcionamiento de la aplicaión.

A.2: Instalación de códecs de video.

Lo primero es localizar en el CD el fichero *"Combined-Community-Codec-Pack-2009-09-09.exe"*, situado en el directorio Combined Community Codec Pack.

Antes de empezar a instalar, este pack avisará si encuentra en el sistema algún elemento que conflictivo y solicita deshabilitarlo. Decimos que deshabilite todo lo que considere necesario.

Luego únicamente dejamos las opciones por defecto asegurándonos al final de la instalación de que establezca las opciones por defecto.

Una vez instalado este pack ya seremos capaces de poder visualizar todo tipo de ficheros de video en nuestro ordenador.

A.3: Instalación de Visual Studio 2008.

En la realización de este proyecto nos hemos servido del IDE Visual Studio 2008, disponible de forma gratuita para alumnos de Ingeniería Técnica de Informática de Gestión.

La instalación se realiza de forma automática, únicamente hay que tener en cuenta el instalar Visual C++, ya que es el lenguaje de programación empleado en este trabajo.

En caso de no disponer de Visual Studio 2008 de forma gratuita, en el CD se adjunta la versión Express de Visual C++ 2008, la cual ofrece Microsoft sin coste alguno.

A.4: Instalación de OpenCV 2.2.

Lo primero es localizar en el CD el fichero *"cmake-2.8.2-win32-x86.exe"*, situado en la carpeta CMake 2.8.2, e instalarlo utilizando las opciones por defecto. Esto instalará CMake 2.8 en nuestro equipo.

Una vez instalado CMake, localizamos en el CD el fichero *"OpenCV-2.2.0a-win32.exe"*, situado en el directorio OpenCV 2.2, y lo ejecutamos para comenzar la instalación de OpenCV. Durante la instalación nos dirá si queremos que introduzca el directorio $opencv/bin al PATH. Le diremos que sí (podemos comprobar después de la instalación que el directorio está correctamente incluido al PATH, yendo a Inicio → botón derecho sobre Mi PC → Propiedades → Opciones avanzadas → Variables de entorno → Variables de sistema → PATH.

Una vez instalado, vamos a línea de comandos (Inicio → Ejecutar → cmd) y nos dirigimos al directorio dónde esté instalado OpenCV. En ese directorio tecleamos "cmake-gui ." (el punto es importante).

Se nos abrirá una ventanita. Pulsamos sobre Configure y elegimos el compilador para el que queramos que genere el proyecto. En nuestro caso, para el VS2008, aunque puede ser 2005, 2003, etc. Volvemos a pulsar sobre configure hasta que el botón de Generate esté activo, entonces lo pulsamos.

En ese momento tendremos generado un archivo en el directorio de OpenCV que se llama "OpenCV.sln". Le abrimos. En el Visual studio vamos a Generar → Limpiar solución, y después Generar → Volver a generar todo. Una vez hecho esto, vamos al directorio dónde esté instalado OpenCV (llamaremos $opencv de ahora en adelante).

Entramos en $opencv/bin/debug y copiamos todos los archivos. Los pegamos en $opencv/bin.

Entramos en $opencv/lib/debug y copiamos todo su contenido a $opencv/lib. Ya tenemos listas las librerías del modo Debug.

Abrimos de nuevo el proyecto de solución que generamos anteriormente, y cambiamos al modo Release (seleccionamos de la lista de desplegables donde pone Debug, Release). Ahora vamos a compilar las librerías para el modo Release. Una vez compilados tenemos que hacer los mismos pasos de copiado que anteriormente.

Entramos en $opencv/bin/release y copiamos todos los archivos a $opencv/bin. Lo mismo hacemos en el otro directorio.

Entramos en $opencv/lib/release y copiamos todos los archivos a $opencv/lib. Con ello tendremos generadas las librerías para el modo Release.

Ahora añadiremos la configuración de directorios para el Visual Studio. Abrimos el VS, y vamos a Herramientas → Opciones. Seleccionamos Proyectos y Soluciones → Directorios de VC++. Ahí vamos añadiendo en las diferentes secciones:

1. Archivos de inclusión:
    1. C:\OpenCV2.2\include\opencv
    2. C:\OpenCV2.2\3rdparty\include\ffmpeg_
    3. C:\OpenCV2.2\3rdparty\include\flann
    4. C:\OpenCV2.2\3rdparty\include\jasper
    5. C:\OpenCV2.2\3rdparty\include\OpenEXR
2. Archivos de biblioteca:
    1. C:\OpenCV2.2\lib
    2. C:\OpenCV2.2\3rdparty\lib
    3. C:\OpenCV2.2\3rdparty\libjasper
    4. C:\OpenCV2.2\3rdparty\libjpeg
    5. C:\OpenCV2.2\3rdparty\libpng
    6. C:\OpenCV2.2\3rdparty\libtiff
    3. Archivos de Código fuente:
    1. C:\OpenCV2.2\src\cv
        2. C:\OpenCV2.2\src\cvaux
        3. C:\OpenCV2.2\src\cvaux\vs

4. C:\OpenCV2.2\src\cxcore
5. C:\OpenCV2.2\src\highgui
6. C:\OpenCV2.2\src\ml

Una vez terminada la configuración de directorios ya tendremos OpenCV 2.2 correctamente instalado en nuestro equipo.

A.5: Instalación de Qt 4.6.2.

Localizamos en el CD el fichero *"qt-win-opensource-4.6.2-vs2008.exe"*, situado en la carpeta Qt 4.6.2, y lo ejecutamos para iniciar la instalación. Dejemos las opciones por defecto. En caso de querer cambiar el directorio de instalación, hay que tener en cuenta que en la ruta NO puede haber espacios.

Después hay que añadir al la variable de entorno PATH la dirección: "C:\Qt\4.6.2\bin\", ya explicamos cómo acceder a la variable en el anexo anterior.

Abrimos la consola de comandos de Visual Studio, Inicio → Programas → Microsoft Visual Studio 2008 → Visual Studio Tools → Símbolo de sistema de Visual Studio 2008.

Vamos al directorio de instalación de Qt, es decir, tecleamos "cd \D C:\Qt\4.6.2"

Tecleamos "configure -platform win32-msvc". Esta operación tardará alrededor de 45 minutos, dependiendo del ordenador en el que se ejecute.

Ahora ejecutamos el comando "nmake". Esta operación puede tardar varias horas.
Una vez terminada la operación anterior vamos a configurar los directorios de Visual Studio.

Añadimos los siguientes:

1. Archivos de inclusión:
    1. C:\Qt\4.6.2\include
    2. C:\Qt\4.6.2\include\QtOpenGL
    3. C:\Qt\4.6.2\include\qtgui
    4. C:\Qt\4.6.2\include\QtCore
    5. C:\Qt\4.6.2\include\qt
    6. C:\Qt\4.6.2\include\QtXml
    7. C:\Qt\4.6.2\include\phonon
2. Archivos de biblioteca:
    1. C:\Qt\4.6.2\lib

Una vez finalizada la configuración, ya tenemos Qt 4.6.2 instalada en nuestro equipo.

Ahora instalaremos el AddIn de Qt para Visual Studio 2008, para ello localizamos en el CD el fichero *"qt-vs-addin-1.1.5.exe"*, ubicado en el directorio VS Qt Addin 1.1.5, y lo ejecutamos para comenzar la instalación, dejando las opciones por defecto.

Una vez terminada la instalación abrimos Visual Studio. Abrimos el menú Qt → Qt Options y añadimos el directorio de instalación donde hemos instalado Qt.

Cuando hayamos terminado ya tenemos todo instalado correctamente.

A.6: Instalación de la aplicación.

Antes de nada hay que localizar el fichero *"AMH-VIDA.exe"*, situado en la carpeta Instalador del CD, y disponer de 15,5 Mb de espacio libre en el disco. Tenga en cuenta que este instalador sólo sirve para entornos Windows.

Una vez localizado, damos doble click para comenzar con la instalación, la cual se realiza de forma automática.

Ilustración A-1 - Segunda ventana del instalador

En la primera pantalla pulsamos en Siguiente. En la segunda pantalla podemos especificar la carpeta donde se instalará la aplicación.

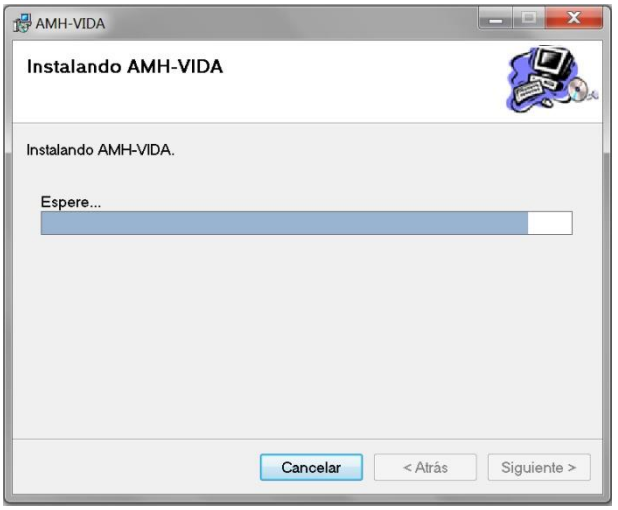


Ilustración A-2 - Cuarta ventana del instalador

En la tercera pantalla pulsamos en Siguiente para que comience con la instalación. Nos mostrará una cuarta ventana, reflejada en la figura A.2, la cual muestra el proceso de instalación. Una vez finalizado nos mostrará una quinta ventana confirmando que la instalación se ha realizado correctamente, como se refleja en la siguiente imagen:

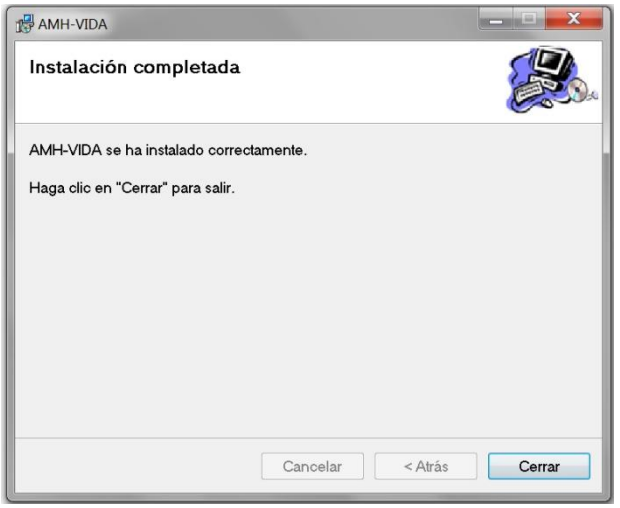


Ilustración A-3 - Quinta ventana del instalador

A.6: Desinstalación de la aplicación.

El software de instalación proporcionado se desinstala como cualquier otro programa en Windows, accediendo a "Panel de control" - "Programas y características", y seleccionando AMH-VIDA y pulsando en la opción Desinstalar que aparecerá en pantalla.

B. Manual de usuario.

B.1. Introducción

En este capítulo de los anexos mostraremos el manual de usuario que describirá las posibles formas de interacción del usuario con las prestaciones del sistema.

B.2. Manual de usuario

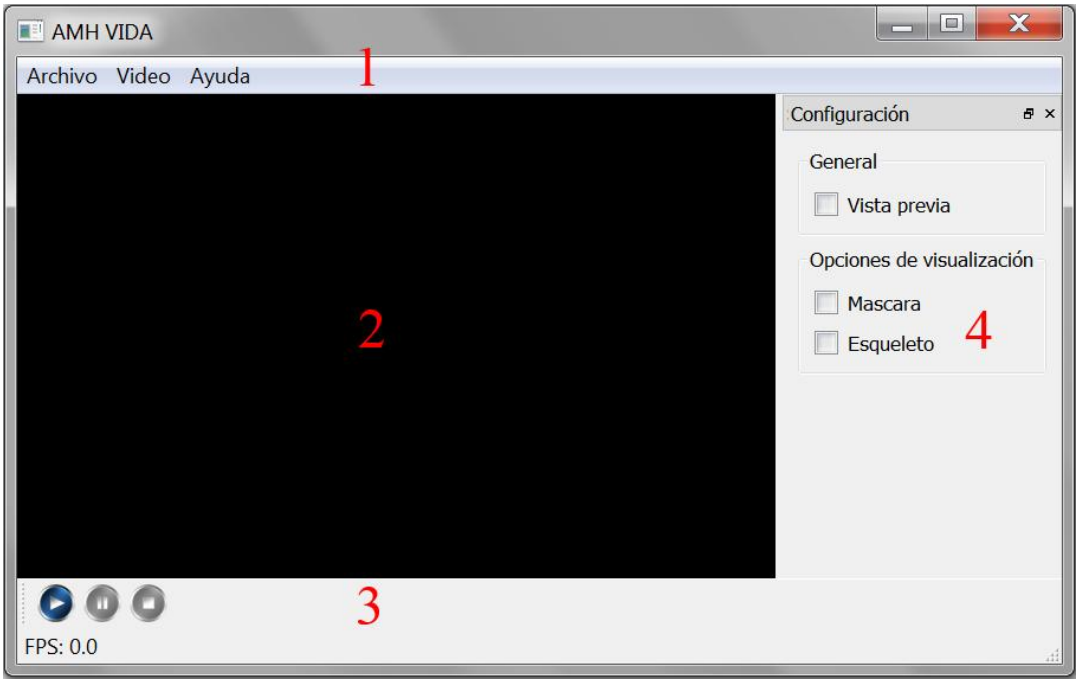


Ilustración B-1 - Pantalla principal de la aplicación

En la [Ilustración B-1] se muestra la pantalla principal de la aplicación. Las funcionalidades de la aplicación se han distribuido en cuatro áreas diferentes:

1) La barra de menú. Contiene las principales funciones del sistema, permitiendo abrir videos, cerrarlos, controlar el video, etc.
2) El área de reproducción. Es el lugar en el que se reproduce el video que ha sido seleccionado para su reproducción y los resultados del tratamiento de las imágenes del mismo.
3) Barra de acciones. Permite ejecutar acciones básicas sobre el video que ha sido abierto, tales como reproducir el video, pausarlo, detenerlo y reanudarlo.
4) Panel de configuración. Incrustado en la página principal, el panel de configuración permite un acceso instantáneo a las principales opciones de configuración, que afectan a los resultados mostrados en el área de reproducción durante la reproducción.

B.2.1. Realización de la tarea de Análisis de un video

A la hora de realizar la tarea de analizar un video, deberemos realizar las siguientes subtareas:

- Abrir un video
- Reproducir un video.
- Cerrar un video

B.2.1.1. Realización de la subtarea de Abrir un video.

Accedemos a la barra de menú y en el menú "Archivo" observaremos las siguientes opciones:

- "Abrir" permite seleccionar el video que se quiere reproducir. Si ya hay un video abierto, cierra el video y abre el nuevo video que se seleccione.
- "Cerrar video" vuelve al estado original de la aplicación tras haber abierto un video.
- "Salir" permite cerrar la aplicación.

Ilustración B-2 - Menú "Archivo"

Seleccionamos Abrir y se nos mostrará una ventana emergente donde deberemos elegir un video.

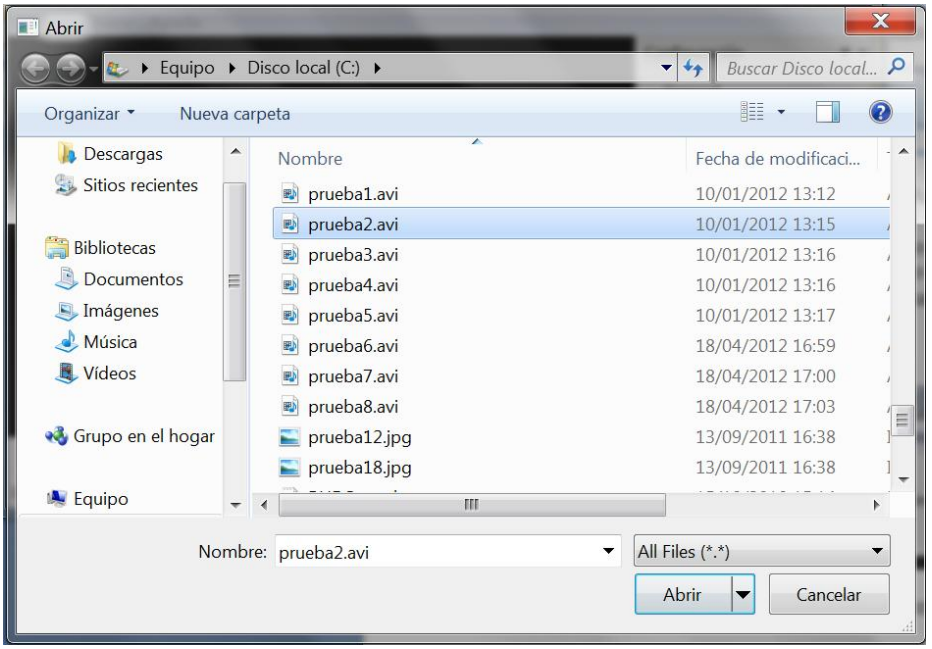


Ilustración B-3 - Ventana de selección de video

B.2.1.2. Realización de la subtarea de Reproducir un video.

La subtarea de "Reproducir un video" tiene como objetivo mostrar información en forma de imágenes en el área de reproducción. El tipo de imágenes dependerá de las opciones de configuración seleccionadas.

Previo inicio de la reproducción del video, se puede seleccionar la opción "Vista previa" del menú de configuración. Con ello, cuando se inicie la reproducción del video seleccionado no se utilizará ningún tipo de procesamiento ni análisis sobre las imágenes, mostrándose el video tal cual, y se deshabilitarán las opciones de configuración mientras el video esté en reproducción en el modo "Vista previa".

Existen dos formas de iniciar la reproducción del video abierto (con idénticos resultados), pulsando el botón con el símbolo de "Play" o bien accediendo al menú "Video" de la barra de menú y pulsando la opción "Start".

Opcionalmente antes y durante la reproducción del video (siempre y cuando no se esté en modo "Vista previa") se pueden marcar las casillas de "Máscara" y "Esqueleto" para mostrar la máscara resultante del filtrado global de la imagen y el esqueleto generado con el análisis de la imagen respectivamente.

La subtarea termina cuando el video llega a su fin, cuando el usuario pulsa el botón con el símbolo "Stop" o cuando el usuario accede a la opción "Stop" del menú "Video" de la barra de menú.

C. Detector de bordes Canny.

C.1. Introducción

El operador Canny [Can86] es un detector de aristas diseñado por John F. Canny en 1986. Este algoritmo es considerado como uno de los mejores métodos para detectar contornos. Para la detección de bordes, se basa en el cálculo de la primera derivada, ya que será constante dicho valor en aquellas regiones cuya intensidad no cambie demasiado en un entorno; en cambio los bordes presentaran cambios bruscos de intensidad y éste hecho se verá reflejado en la derivada primera.

El objetivo de Canny era construir un detector de aristas óptimo, lo que implicaba que el algoritmo debería ser capaz de:

- Detectar aquellos bordes que sean más importantes, presentes en la imagen.
- Minimizar la distancia entre la arista donde se detecta con respecto a la posición real de la arista.
- Ser capaz de identificar aquellos bordes que se detectan varias veces y unificarlas en una única

El algoritmo de Canny se puede subdividir en tres etapas principales:

- Cálculo del vector gradiente en cada pixel.
- Supresión no máxima: consiste en la reducción del grosor de los bordes a un pixel
- Uso de una función de histéresis a partir de dos umbrales de entrada, para eliminar posibles falsos bordes.

C.2. Procedimiento

Obtención del gradiente.

Para el cálculo del vector gradiente se realiza el siguiente procedimiento

- El primer paso que realiza el detector Canny consiste en aplicar un filtro Gaussiano sobre la imagen, con la intención de suavizar la imagen. El suavizado se realiza calculando el promedio de las intensidades de los pixeles vecinos a otro con una máscara de convolución de media 0.
- Se calcula el gradiente en cada putno como un vector bidimensional de la siguiente forma:

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\delta}{\delta x} f(x,y) \\ \frac{\delta}{\delta y} f(x,y) \end{bmatrix}$$

El cual es un vector perpendicular al borde cuyo modulo y orientación son respectivamente:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

$$\emptyset(x,y) = tan^{-1}\frac{G_y}{G_x}$$

El resultado es una imagen con la magnitud de gradiente y otra con la dirección.

Ilustración C-1 - Imagen natural y transformación por filtro Gaussiano. La imagen original de Lena (izquierda) pasa por un proceso de suavizado, transformándose en la imagen que se puede ver a la derecha

Supresión no maximal

Dada la estimación del gradiente de la imagen, se consideran las direcciones 0º, 45º, 90º y 135º para evaluar en cada pixel, si la magnitud toma un valor máximo en la dirección del gradiente. Después, por cada pixel, se le asigna valor 0 si la magnitud de su gradiente es menor que la de al menos uno de sus vecinos. En caso contrario se le asigna el valor de la magnitud de su gradiente. El resultado es una imagen con bordes más finos.

Ilustración C-2 - Imagen tras finalizar el proceso de *supresión maximal* de Canny

Rastreo de aristas a través de la umbralización de la histéresis.

Para la eliminación del ruido que posiblemente pueda contener la imagen obtenida en el paso anterior, se suele aplicar una función de histéresis. En esta etapa se requieren adicionalmente dos parámetros, un umbral inferior y otro superior. El procedimiento a seguir por cada punto es el siguiente:

- Localizar el siguiente punto de borde no evaluado, con valor de intensidad superior al umbral mínimo pasado como parámetro de entrada.
- Recorrer todos los puntos de máximos locales con intensidad menor que el umbral máximo recibido como entrada.
- Marcar todos los puntos recorridos y almacenar la lista de puntos que se han seguido en cadena.

Ilustración C-1 - Imagen tras pasar por el filtro de histéresis del algoritmo Canny

D. Contenido del CD.

En este anexo vamos a mostrar los contenidos almacenados en el CD que acompaña a esta memoria. Primeramente mostraremos una imagen con el árbol de directorios del mismo, y luego detallaremos el contenido de cada una de las carpetas.

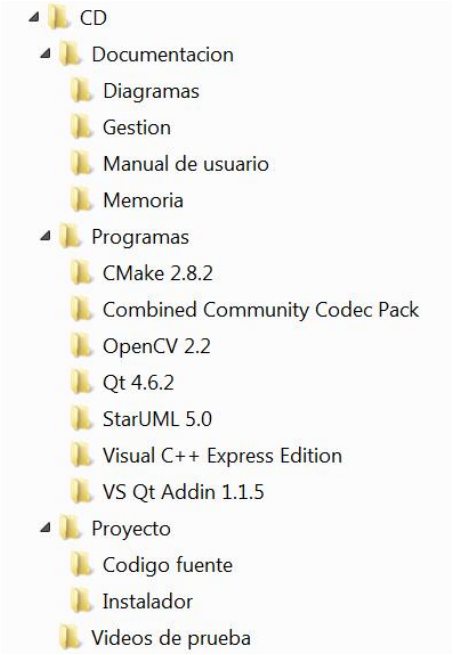


Ilustración D-1 - Árbol de directorios del CD

Diagramas: En esta carpeta encontraremos los ficheros de los diagramas de clase y secuencia (.uml).

Gestion: Contiene los documentos (.pdf) de planificación del proyecto, seguimiento, resumen de riesgos y resultados.

Manual de usuario: Contiene una copia (.pdf) del manual de usuario.

Memoria: dentro podemos encontrar el fichero "Memoria.pdf", versión digital de este mismo documento.

CMake 2.8.2: dentro encontraremos el fichero "cmake-2.8.2-win32-x86.exe", el cual es el instalador de la aplicación CMake, versión 2.8.2.

Combined Community Codec Pack: en su interior hayaremos el fichero "Combined-Community-Codec-Pack-2009-09-09.exe", que es el instalador del pack de codecs del mismo nombre.

OpenCV 2.2: en su interior podemos ver el fichero "OpenCV-2.0.0a-win32.exe", instalador de las librerías de OpenCV en su versión 2.0.

Qt 4.6.2: dentro del directorio podemos hallar el fichero "qt-win-opensource-4.6.2-vs2008.exe", que es el instalador de las librerías Qt en su versión 4.6.2, preparadas para ser compiladas con Visual Studio 2008.

StarUML 5.0: dentro de la carpeta podemos encontrar el fichero "staruml-5.0-with-cm.exe", instalador del programa StarUML, usado para la realización de los diagramas UML. Permite abrir los ficheros .uml de la carpeta Diagramas.

Visual C++ Express Edition: en ella podemos localizar el fichero "vcsetup.exe", que nos permite instalar El entorno de desarrollo Visual C++ 2008 en su edición Express, la cual es completamente gratuita.

VS Qt Addin 1.1.5: en su interior localizamos el fichero "qt-vs-addin-1.1.5.exe", que nos permite instalar el "addin" de Qt para el entorno de desarrollo Visual Studio.

Código Fuente: dentro podemos encontrar el código fuente de la aplicación, así como el fichero .sln de Visual Studio.

Instalador: en ella se pueden localizar los ficheros "AMH-VIDA.msi" y "AMH-VIDA.exe", los cuales sirven para la instalación de la aplicación realizada en este trabajo fin de grado.

Videos de prueba: en ella podemos ver los archivos de video de prueba utilizados en este proyecto.