Universidad de las Ciencias Informáticas

Facultad 6

Componente para elaborar resúmenes de video escalables automáticamente

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Maykel López Meneses

Tutor: Ing. Abel Díaz Berenguer

La Habana,

Junio de 2013

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Maykel López Meneses

_____

Firma del Autor

Ing. Abel Díaz Berenguer

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Abel Díaz Berenguer

Formación Académica: Ingeniero en Ciencias Informáticas.

Centro Laboral: Universidad de las Ciencias Informáticas (UCI).

Correo Electrónico: aberenguer@uci.cu

Agradecimientos

A toda mi familia por el apoyo incondicional, la confianza y la atención que siempre me han dado. Especialmente a mis abuelos Delfina y Ever, y mis tías Ada y Ana y Roberta que en estos cinco años siempre han tenido comida, cama y conversación, lista para cuando pasara por la Lisa.

A mis hermanos de crianza Richard y Yaisel, mis ejemplos a seguir en la vida, principales responsables de la fuerza de voluntad y los deseos de mejorar que me han empujado hasta el final de mi carrera.

Al piquete de la Redbull, comelones del doble, bullangueros profesionales, fanáticos al deporte, enchuchadores y pendencieros; por hacer tan amena la estancia en el edificio 95, sin duda alguna, lo que más voy a extrañar de la UCI.

Al tribunal, al oponente y a mi tutor, por ser tan pacientes conmigo, y guiar esta investigación por el camino correcto.

A todos mis amistades en general, por la compañía, pero muy especialmente a Adrián, Karel y Aramis, los tres pilares que me han sostenido en pie, a lo largo de este recorrido tan sinuoso e instructivo, que representa la Universidad.

Dedicatoria

A mi Hermano, mi Mamá y mi Papá, por inculcarme la lealtad, la amabilidad y la razón, valores que tantas puertas me han abierto en la vida.

A mis amistades más cercanas, por la compañía, los buenos consejos y por obrar a mi favor, en las buenas y en las malas.

A las piedras con las que tropecé en el camino, por ayudarme a entender el mundo como es, y no como nos gustaría que fuera

RESUMEN

Los algoritmos de creación de resúmenes de video surgen por la necesidad de obtener representaciones visuales compactas de los contenidos audiovisuales disponibles en la red. Estas representaciones pueden obtenerse como un índice conformado por imágenes (*storyboard*) o como una secuencia de video mucho más pequeña que la original (*skims*). En ambos casos la longitud del resumen tiene gran importancia, debido a que según esta, varía la cantidad de información que se puede aportar al usuario y el tiempo que le tomará visualizar el resultado. Para efectuar esta tarea existen gran variedad de técnicas que persiguen obtener menores tiempos de cómputo o mejores resultados. Este trabajo propone la creación de un algoritmo que obtenga, como resultado de un solo análisis, múltiples secuencias del video original (*skims*, de longitudes diferentes). Con el objetivo de construir un resumen escalable, se utiliza un algoritmo de ranking iterativo que hace uso de distintos criterios para dar una puntuación a las tomas y agregarlas a una lista, en la que según el orden de aparición será su relevancia; logrando de esta forma una representación visual escalable del material audiovisual. También se introduce la utilización de más de un descriptor visual para realizar los primeros análisis con los descriptores más rápidos y optimizar el resultado con los más precisos, con la intención de obtener un resumen de mayor calidad sin sacrificar la velocidad del algoritmo. El resultado demostró poseer gran precisión y flexibilidad, gracias a que se pueden adicionar o quitar otros descriptores visuales.

PALABRAS CLAVE

Descriptores, escalable, *ranking*, *skim*, *storyboard*.

ABSTRACT

The video summary generation algorithms arise by the necessity of gaining compact visual representations of video sequences in the net. These representations can be obtained as an index of images (storyboard) or a smaller video sequence than the original (skims). In both cases, the length of the summary has a great importance, because, in dependence of it, will be the amount of viewed information and the time of visualization varies. To make video summarization possible there exists a great variety of technics that seek obtaining better computing times or sharper accuracy. This work proposes the creation of an algorithm that gets, as a result of a single analysis, multiple skims (of different length) of a video. Aiming to construct a scalable summary, an iterative ranking algorithm is used, that, with the aid of different criteria gives a grade to the shots, and adds them to a list, in which the order of appearance will mean the relevance of them; making possible a scalable visual representation of the original video. There is also introduced the use of multiple visual descriptors, to do the heavier job with those who are faster, and optimize the result with those more accurate, with the intention of generating a better summary without losing computing speed. The resultant solution proved to have a great accuracy and flexibility, so that different descriptors can be added or removed.

KEYWORDS

Descriptor, ranking, scalable, skim, storyboard.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNIDICE DE TABLAS

INTRODUCCIÓN

El surgimiento de la radio y la televisión marcan el inicio de lo que hoy se conoce como una sociedad mediática. La televisión ha sido desde la década de los 70's, el medio de comunicación masivo con mayor impacto en nuestra sociedad. Este impacto lejos de decrecer se incrementa y transforma con la aparición de nuevas tecnologías como **la televisión digital** y **la Internet** (1).

La introducción de la televisión digital trae consigo numerosas ventajas tanto a la hora de entender como de utilizar la televisión; el uso de la representación numérica de la señal hace posible la utilización de compresores, filtros digitales, detección y corrección de errores, canales de doble vía, control de conexión local, entre otras. Características que garantizan un mejor servicio así como un mejor aprovechamiento del ancho de banda (2) (3).

Desde finales de la década de 1990 se puede apreciar la creciente relación que establecen los sistemas de televisión digital en sus diferentes soportes con la denominada red de redes, **Internet**. Se trata de una relación de ida y vuelta -la televisión en Internet e Internet en la televisión- que se desarrolla, día tras día. En el marco de esta relación los televidentes se convierten cada vez con más frecuencia en usuarios-consumidores que pagan distintos servicios interactivos; entre éstos, actualmente, los más importantes son el **video bajo demanda** y el **video en vivo** (4).

En los últimos años se ha evidenciado un notable aumento de la producción y distribución de contenidos de video. El evidente atractivo y la aceptación que poseen estos contenidos para los usuarios, el abaratamiento y crecimiento de los medios de almacenamiento, el incremento de las capacidades de transmisión de video en las redes de ordenadores y el desarrollo de estándares para la compresión y descripción video, constituyen elementos que influyen en este auge de producción y distribución audiovisual. Esta situación condiciona una elevada demanda de aplicaciones informáticas para la gestión, procesamiento y almacenamiento de estos contenidos, facilitando el acceso eficiente de los usuarios a la información almacenada.

En Cuba, existen varias instituciones que trabajan en pos de desarrollar aplicaciones que aseguren un mejor aprovechamiento de las tecnologías de información y comunicaciones. Entre esta instituciones se encuentra la Universidad de Ciencias Informáticas (UCI), en la cual se encuentran proyectos de investigación, innovación y desarrollo en el seno del Departamento de Señales Digitales del centro

GEYSED de la facultad 6, encargados de la gestión, el procesamiento y transmisión de contenidos audiovisuales.

En los sistemas antes referidos, los investigadores han obtenido aplicaciones informáticas, como la plataforma de transmisión de noticias (Primicia), el sistema de catalogación y publicación de videos (VideoWeb), el Sistema de Transmisión de Canales Virtuales (STCV), y el sistema de gestión y transmisión de contenidos audiovisuales (SIAV). En estas aplicaciones efectuar tareas como la clasificación y aprobación de las medias a publicar o transmitir se hace muy difícil, debido a que por lo general implica un proceso de pre-visualización del video, que requiere una gran cantidad de tiempo y uso del ancho de banda disponible.

A partir de la situación anterior se determina el siguiente **problema a resolver** ¿Cómo reducir el tiempo y el ancho de banda requerido en los procesos de clasificación y aprobación de contenidos de video? Se define como **objeto de estudio** las técnicas para obtener representaciones compactas de una secuencia de video y como **objetivo general** desarrollar un componente que permita generar resúmenes de video, obteniendo como **campo de acción** las técnicas para construir representaciones compactas escalables de contenidos de video.

Con el fin de dar solución a la situación problemática se define como **idea a defender:** si se desarrolla un componente para generar automáticamente múltiples resúmenes de una secuencia de video se reducirá el tiempo y el ancho de banda requerido en los procesos de clasificación y aprobación de contenidos audiovisuales.

Para desarrollar el componente propuesto se establecen las siguientes **tareas de investigación:**

1. Determinación de los descriptores Visuales a utilizar para confeccionar el resumen de video.
2. Selección de los algoritmos y técnicas factibles para realizar resúmenes automáticos de video.
3. Selección de las herramientas y tecnologías de desarrollo a utilizar en la construcción del software.
4. Desarrollo de los artefactos y documentación requerida según la metodología de desarrollo seleccionada.
5. Construcción del componente.
6. Validación del funcionamiento del componente.

La investigación se efectúa haciendo uso de los siguientes métodos científicos:

Métodos teóricos

Analítico-Sintético: Este método se utiliza con el objetivo de analizar y procesar los fundamentos científicos y las teorías relacionadas con el objeto de estudio de la investigación. Lo que permitirá extraer los aspectos significativos que sustentarán la propuesta de algoritmo a seguir para la implementación del componente.

Inductivo-Deductivo: Este método posibilita el desarrollo de un conocimiento más profundo de los algoritmos y técnicas para realizar resúmenes automáticos de video, apoyándose en conclusiones obtenidas a partir del análisis del objeto de estudio y campo de acción.

Modelación: Este método se utilizará en la creación de varios modelos necesarios en el proceso de desarrollo del componente, como son el modelo de dominio, de diseño, y de implementación.

Finalmente, con el objetivo de dar cumplimiento a las tareas de investigación se estructura el documento de la siguiente forma:

Capítulo 1: “Fundamentación teórica”. En este capítulo se explican los conceptos asociados al dominio del problema y se aborda el estado del arte del tema tratado que incluye la descripción del objeto de estudio y el análisis de soluciones existentes.

Capítulo 2: “Herramientas y Tecnologías”. En este capítulo se presenta todo lo relacionado con la metodología de desarrollo a utilizar. También se seleccionan la biblioteca y el lenguaje de programación para implementar el componente, que son respaldados por un entorno de desarrollo y un marco de trabajo también descritos.

Capítulo 3: “Análisis y Diseño”. En este capítulo se desarrolla el análisis y diseño del componente, siguiendo las pautas de la metodología seleccionada. Después se define la arquitectura del componente y se presenta la descripción de la propuesta de algoritmo. Por último se presenta el modelo de implementación de la solución.

Capítulo 4: “Validación de la propuesta”. En este capítulo se realiza la validación del componente, efectuando pruebas como son las pruebas de formatos y de precisión en la detección de tomas.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se abordarán los conceptos asociados al objeto de estudio y campo de acción, con el propósito lograr que el lector se familiarice con los temas relacionados a la investigación. Además se hace una descripción detallada del dominio del problema y se efectúa un análisis de las soluciones existentes. Por lo que se persigue como objetivo principal, aumentar el conocimiento sobre los descriptores visuales, las técnicas y algoritmos a utilizar durante la elaboración de resúmenes automáticos de video, y por tanto, conformar los fundamentos teóricos de la presente investigación.

1.1 Conceptos asociados al dominio del problema

En el trabajo se hace uso de un grupo de términos cuyos significados, pueden ser poco conocidos o mal enfocados, disminuyendo la claridad de las ideas expuestas en el mismo. Por lo cual, se definen a continuación, los conceptos asociados a la investigación que se desarrolla, comenzando por los que son parte del título del trabajo.

Componente

Un componente es una unidad de composición de una aplicación de software determinada, que posee una interfaz de comunicación y satisface un conjunto de requisitos, con el objetivo de que sea desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente. (5)

Resumen

Un resumen no es más que una explicación corta en la que se presenta lo principal de un asunto o materia (6). Un ejemplo de resumen sería la redacción de un texto nuevo a partir de otro texto, exponiendo las ideas principales o más importantes del texto original de manera abreviada. El resumen no es sólo una simple reducción informativa de un original, sino un texto nuevo que intenta adaptarse a las características de un nuevo contexto comunicativo. Al hacer un resumen, es preciso plantearse primero con que finalidad se realiza, quién será su destinatario, qué espera el destinatario del resumen. (7)

Enfocando este concepto al objetivo de este trabajo, se concluye que un resumen de video busca proveer al usuario con una representación visual compacta que contenga la mayor cantidad de información posible. (8) (9) (10)

Automático

Se aplica al mecanismo que funciona por sí solo o que realiza, total o parcialmente, su proceso sin ayuda de una persona (6). Para este componente representa la generación de resúmenes según las preferencias establecidas sin la supervisión de una persona.

Escalable

Este término ha sido utilizado en muchos campos de investigación con significados diferentes, pero la idea que prevalece es que algo es escalable cuando es capaz de adaptarse adecuadamente a diferentes condiciones. (9)

El concepto de escalabilidad puede ser utilizado como una nueva propiedad de los resúmenes de video en sí. La escala es relacionada con la longitud del resumen (duración o cantidad de imágenes) (8) y es precisamente el significado que se utiliza en la investigación, dando a entender a través del título del trabajo, la capacidad del componente de generar no solo uno, sino múltiples resúmenes de distintas duraciones a partir de un único análisis.

Video

Un video es una grabación realizada con un sistema que permite la captura y reproducción de imágenes y sonidos (11). También se asocia a la tecnología de procesamiento, almacenamiento, transmisión y reconstrucción por medios electrónicos digitales o analógicos de una secuencia de imágenes (fotogramas) que representan escenas en movimiento (12).

En el marco de esta investigación se define como una secuencia de **tomas** que transmite cierta información.

Toma

Una toma es un fragmento de una película de fotografía o cine que se impresiona o se graba de la misma (6). Es la unidad básica de un resumen video y su identificación es imprescindible para entender su estructura (13).

Por tanto, es una secuencia de imágenes y sonidos que tienen gran similitud y cercanía temporal, que representa la unidad básica utilizada para la generación de resúmenes de video. Para procesar una toma, ya sea para clasificarla o extraer información de la misma, se hace necesario el uso de los **descriptores visuales**.

Descriptor visual

Un descriptor no es más que una palabra clave que define el contenido de un documento (14). Los descriptores son utilizados con el fin de describir el contenido de los distintos tipos de información multimedia para efectuar la gestión posterior de acuerdo a la finalidad del sistema. Los descriptores visuales son utilizados para resaltar un grupo de píxeles que es relevante de alguna manera desde el punto de vista humano. Estos describen las principales características de las imágenes como forma, color, textura o movimiento (15). Por tanto, son datos que permiten obtener características distintivas de un **fotograma**.

Fotograma

Cada una de las imágenes que se suceden en una película cinematográfica consideradas de forma aislada (16). Por tanto, se define como las imágenes individuales que al concatenarse, conforman una toma, estas pueden ser comparadas entre sí obteniendo un valor numérico que representa que tan diferentes son, este valor es conocido como **distancia**.

Distancia

Espacio o periodo de tiempo que medio entre dos cosas o sucesos (17). Enmarcado en el objeto de estudio este concepto hace referencia a las diferencias que se establecen entre dos fotogramas a través del uso de descriptores visuales (18). Por tanto, no es más que un valor numérico que representa que tan diferentes son dos fotogramas analizados.

1.2 Descripción actual del dominio del problema

Lo mencionado en el epígrafe anterior facilita un mejor entendimiento del proceso de generación de resúmenes de video, en su ámbito general. En este epígrafe se profundiza sobre la problemática que existe en el proyecto STCV, y la plataforma SIAV.

Dentro de los subsistemas con los que cuenta STCV se encuentra el subsistema de programación de canales virtuales y el de transmisión de audiovisuales. Por otra parte SIAV cuenta con un subsistema de publicación de materiales audiovisuales en la web. Tanto en uno como en otro, el proceso de clasificación y aprobación de los materiales a publicar se hace difícil en el sentido que los editores, o el responsable de esta actividad debe visualizar de forma parcial o total los materiales a transmitir o publicar en la web. Este proceso resulta altamente ineficiente debido que necesita hacer uso de gran cantidad de tiempo y ancho de banda. Una variante para dar solución a este problema es buscar la sinopsis del video pero entonces el proceso depende de la existencia y veracidad de la misma.

En el caso de los usuarios que quieren ver determinada programación o material audiovisual, necesitan también de una sinopsis que los ayude a obtener una idea del mismo, sin necesidad de visualizarlo en su totalidad y de esta manera motivarse a ver o no dicho material. Es importante señalar que una sinopsis es un resumen de texto, por lo que hay muchas características de un video que no puede reflejar como son la fotografía, efectos especiales, ambientación, entre otros, y la creación de la misma, depende de que una persona visualice el video en su totalidad, haciendo mas lento el proceso de publicación de un video muy reciente. Por lo cual se necesita de una forma de generar resúmenes que aporten la mayor cantidad de información posible al cliente.

1.3 Descripción del objeto de estudio

Las técnicas para obtener representaciones compactas de una secuencia de video, son un tema en el cual muchos especialistas han profundizado en las últimas dos décadas, con el objetivo de satisfacer necesidades muy diversas, que van desde la detección de incidentes en grabaciones de cámaras de seguridad hasta la creación de resúmenes de video de materiales deportivos o noticiarios. Como resultado de los estudios realizados sobre el tema, han surgido modelos de estructuración que tienen, como objetivo principal, lograr una arquitectura más modular y sencilla en estas soluciones. De esta manera, autores con gran experiencia en el tema como son Víctor Valdez y José M. Martínez, separan el proceso de elaboración de resúmenes distintas etapas, las cuales serán explicadas a continuación. (19)

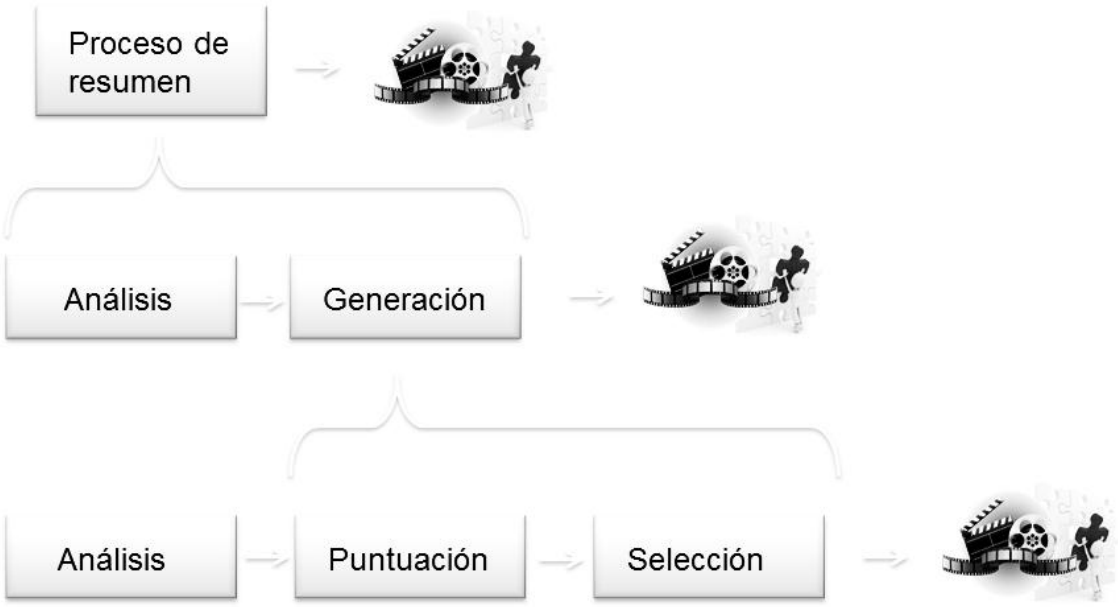

Figura 1. Proceso de elaboración de resúmenes.

En la figura 1 se observa como el proceso de elaboración de resúmenes está compuesto, según los autores mencionados, por dos etapas principalmente. La primera es el análisis, en la cual se efectúa la detección de las tomas y sus características. La segunda etapa, generación, que está a su vez compuesta por 2 etapas, identificadas como Puntuación, etapa donde se le asigna cierto nivel de relevancia a las tomas atendiendo a las características extraídas de las mismas, y Selección, etapa donde se seleccionan las tomas que pasarán a formar parte del resumen de video. (19) Para lograr una mejor comprensión del objeto de estudio se explicará a continuación cada una de estas etapas.

1.3.1 Análisis

Durante el análisis del video original, se realiza la detección de tomas y la extracción de características de las mismas. La detección de tomas o segmentación del video, permite obtener una lista de las tomas que conforman el video original, que se utilizará en las etapas restantes del proceso de generación de resúmenes. Tanto para efectuar la segmentación del video como para extraer características de las tomas, es necesario hacer uso de descriptores visuales. Los descriptores visuales, se pueden dividir en dos grupos principalmente: descriptores globales y descriptores locales (20). A continuación se abordan algunas de las técnicas que contienen estos grupos.

1.3.1.1 Descriptores globales

Realizan un análisis de la imagen completa para extraer un vector de características. Tras realizar un análisis con un descriptor global no se obtienen características de objetos o zonas específicas del video sino un conjunto de datos que describen la imagen completa permitiendo de esta manera diferenciarla de otras. (20)

Diferencias entre píxeles

La forma más fácil de detectar si dos fotogramas son significativamente diferentes, consiste en contar el número de píxeles que varían por encima de cierto umbral. Este total es comparado contra un segundo umbral para determinar si un límite de toma ha sido encontrado. Este método, a pesar de ser de fácil implementación y tener bajos costo computacional, es muy sensible al movimiento de cámara. (21)

Diferencias estadísticas

Las diferencias estadísticas se basan en la idea del descriptor anteriormente mencionado, pero dividiendo las imágenes en regiones y comparando las medidas estadísticas de los píxeles en esas regiones. Por ejemplo, computar una medida basada en el sentido y la desviación estándar de los niveles de grises en regiones de imágenes. Este método es razonablemente tolerante al ruido, pero es lento debido a su complejidad de fórmulas estadísticas. También genera muchos falsos positivos lo cual produce que se obtengan las tomas fragmentadas. (22)

Histogramas

Los histogramas son el más común de los métodos utilizados para la detección de tomas. El método más simple de histogramas procesa la escala de grises o en colores de dos imágenes. Si la diferencia entre estos histogramas sobrepasa un umbral definido, se detecta una toma. Entre sus principales ventajas está que es de simple implementación. El histograma de color de una imagen es relativamente invariable con traslaciones o rotaciones sobre el eje de visión, y varia solo suavemente con el ángulo de visión (23). Pero también tiene una alta sensibilidad a los cambios de iluminación e imposibilidad de representar una distribución espacial de los colores (10).

Se han investigado distintas formas de detectar tomas basándose en histogramas. A continuación se mencionan algunas:

- Ueda, Miyatake, y Yoshizawa usan la variación de los histogramas de colores para encontrar límites de tomas (24).
- Nagasaka y Tanaka comparan varias estadísticas simples basadas en escalas de grises e histogramas de color (25).
- Swanberg, Shu, y Jain usaron diferencias de histogramas de escala de grises en una región o sección del fotograma, valorándolos por cuanto la región había cambiado en una secuencia de video (26).

Detección de bordes

Como ejemplo de detección de bordes está la solución propuesta por Zabih, Miller y Mai. En su propuesta alinean fotogramas consecutivos para reducir los efectos de los movimientos de cámara, compararon los números y la posición de los bordes en las imágenes utilizadas. El porcentaje de bordes que entran y salen entre dos fotogramas se procesa, los límites se detectan entonces cuando encuentran grandes cambios en los porcentajes. Este método es mucho más preciso que los histogramas en situaciones que ocurren cambios de iluminación o transiciones de desaparición, aunque como desventaja con respecto al anterior tiene un mayor costo computacional. (27) (28)

1.3.1.2 Descriptores locales

Extraen puntos o regiones características de las imágenes y conforman con estos un vector que las describe (20). Estos descriptores son muy utilizados en la detección de rostros y objetos en un fotograma. Por tanto son muchos más precisos pero tienen un costo computacional más alto que los globales.

Scale Invariant Feature Transform (SIFT)

El descriptor SIFT fue desarrollado para detectar puntos característicos estables en una imagen. Estos puntos se muestran invariables frente a diferentes transformaciones como traslación, rotación, escala, iluminación y transformaciones afines. Originalmente fue desarrollado para el reconocimiento de objetos de manera general y realiza la correspondencia entre puntos basada en los vectores de características de cada punto que componen el descriptor de la imagen. (18)

Existen diversas aplicaciones de este descriptor a la detección de tomas, entre las que se encuentra la de O. Chum, J. Philbin, M. Isard y A. Zisserman que proponen una solución en la que se dividen las

imágenes a comparar en regiones, y cada región es representada por un descriptor SIFT, aprovechando que el descriptor SIFT ha demostrado ser invariable a pequeñas distorsiones geométricas y cambios de iluminación. (29)

Speeded Up Robust Features (SURF)

El descriptor SURF, fue desarrollado como un detector de puntos de interés robusto. Este guarda cierta similitud con la filosofía del descriptor SIFT, aunque presenta notables diferencias con respecto al anterior. Los autores afirman que este detector y descriptor presenta principalmente tres mejoras resumidas en los siguientes conceptos: (18)

- Velocidad de cálculo considerablemente superior sin ocasionar perdida del rendimiento.
- Mayor robustez ante posibles transformaciones de la imagen.
- Genera menor cantidad de información a partir del análisis de una imagen.

1.3.2 Generación

Durante la etapa de generación se efectúa la Puntuación y la Selección de las tomas que conformarán el resumen de video. La Puntuación es efectuada atendiendo a distintos criterios extraídos a partir del análisis de las tomas, entre los cuales los principales son: (8) (30)

- Distancia: Ofrece una idea de que tan diferente o semejante es una toma al resto.
- Variabilidad: el cual es un valor numérico que indica que tanto varía una toma dentro de sí.
- Duración: como su nombre indica, la extensión o longitud de tiempo que tiene una toma.

En la actualidad existen varios métodos para realizar la selección de las tomas en dependencia de los autores y las problemáticas que resuelvan, pero en la mayoría de los casos, se evidencia la existencia de algoritmos de agrupamiento (o de creación de clústeres), que son aplicados a las tomas detectadas con el objetivo de eliminar la redundancia, y por tanto, transmitir la mayor cantidad de información posible. Este agrupamiento se realiza en base a la distancia entre tomas, la cual es determinada mediante el uso de uno o más descriptores visuales. A continuación, se describen algunos de los tipos de algoritmos de agrupamiento o creación de clústeres que podrían utilizarse en el proceso de generación de resúmenes.

1.3.2.1 Agrupamiento jerárquico

Este algoritmo agrupa la información como una secuencia de particiones embebidas, que son organizadas en una estructura jerárquica según una matriz de proximidad basada en las didstancias. El resultado del mismo es usualmente reflejado como un árbol binario o dendograma. El nodo raíz representa todo el conjunto de datos, y cada hoja es vista como un punto de datos o información. Los nodos intermedios, por tanto, describen que tan próximos son unos objetos de otros, y la altura del dendograma usualmente expresa la distancia entre cada par de puntos de datos o clústeres. El resultado final puede ser obtenido al cortar el dendograma en diferentes niveles. Esta representación provee la descripción y visualización de las estructuras de datos (clústeres) potenciales, especialmente cuando realmente existe una relación jerárquica en la información procesada. (31)

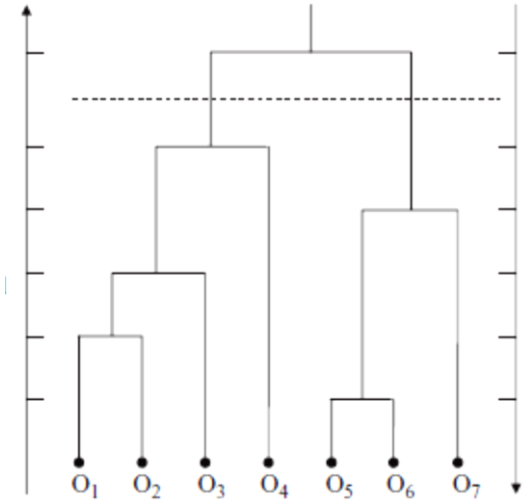

Figura 2. Ejemplo de dendograma.

La figura 2 da una idea de cómo quedaría conformado el agrupamiento jerárquico de las tomas (O_n) en base a la distancia entre estas. En los niveles inferiores se aprecia la asociación entre tomas que tienen un alto parecido como son O_5 y O_6, que posteriormente se asocian a la más cercana O_7, y así sucesivamente.

1.3.2.2 Agrupamiento K-Means

El algoritmo K-Means es uno de los algoritmos de agrupamiento más populares en el mundo. El mismo busca un particionado óptimo de la información mediante un proceso de optimización iterativo, que distribuye los datos según su parecido en distintos grupos. Este algoritmo también calcula el centroide

o punto central de los clústeres generados con su uso. Este centroide identificaría el objeto más representativo del grupo y por tanto el de mayor importancia a la hora de seleccionarlos. (31)

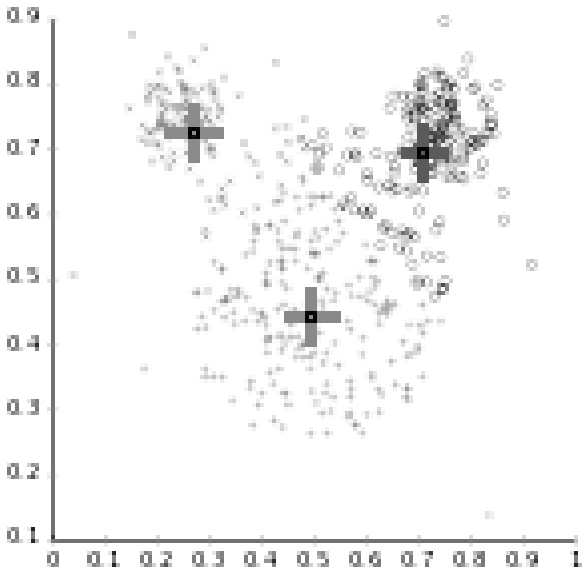

Figura 3. Ejemplo de agrupamiento K-Means.

La figura 3 representa las clústeres después de agruparse utilizando mediante K-Means. En el mismo, se pueden identificar tres grupos (rojo, verde y azul) con sus respectivos centroides, los cuales están representados por una cruz del color del grupo con el centro de color negro.

1.3.2.3 Agrupamiento basado en grafos

La teoría de grafos puede ser utilizada para los agrupamientos que no son jerárquicos. Para agrupar la información, el grafo debe ser construido con un árbol de expansión mínima basado en una matriz de distancia entre los nodos que lo componen. El objetivo es eliminar los arcos inconsistentes y construir de esta forma los clústeres. Los arcos con pesos mayores que el promedio de los pesos vecinos son tratados como inconsistentes, y la eliminación de estos convierten al grafo en conjuntos de nodos conectados, que se interpretan como clústeres. (31)

La figura 4 muestra el resultado obtenido al eliminar los arcos inconsistentes (líneas discontinuas) dentro del árbol de expansión mínima. En el mismo se pueden apreciar los tres clústeres que conforman las tomas.

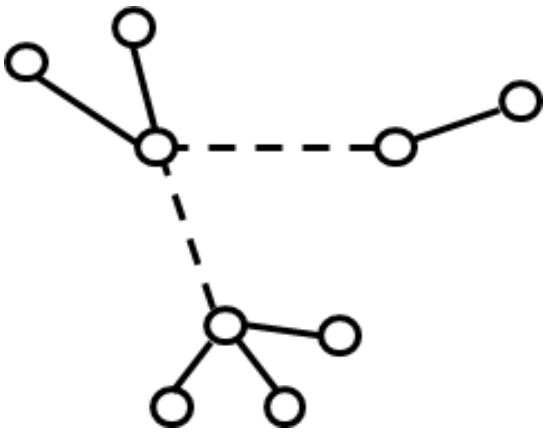

Figura 4. Ejemplo de agrupamiento mediante grafos.

1.3.3 Construcción

Después de haber recorrido los epígrafes anteriores, el lector, debe tener una idea de cómo se decide que parte del video original pasará a ser del resumen de video resultante, pero todavía se hace necesario conocer, cómo se visualizaría el resultado final. En la actualidad existen principalmente 2 formas de representar los resúmenes de video generados, como imágenes estáticas (*Storyboard*) y como secuencias de imágenes (*Skim*). A continuación se explicaran estos conceptos:

1.3.3.1 Storyboard

Un *Storyboard* es un resumen creado seleccionando algunos fotogramas separados e independientes para representar el contenido en unas pocas imágenes. En un resumen de este tipo, no hay reproducción, por lo cual es estático y permite una exploración no lineal del contenido, sacrificando el flujo temporal del video. Su creación necesita de muy pocos recursos, porque son imágenes estáticas y vez generado, su visualización es muy rápida y de fácil navegación. (32) (8)

1.3.3.2 Skim

La traducción del término Skim al español significa: hacer una lectura rápida y superficial, escrutinio o consideración, un vistazo (33). En el marco de la investigación se interpreta como un grupo de segmentos de video conectados mediante alguna transición (10). Es una de las aplicaciones más interesantes de la generación de resúmenes de video, consiste en seleccionar segmentos de video que muestren la mayor cantidad de información posible del video original. En dependencia de su

aplicación la técnica de selección puede variar obteniendo resultados desde *trailers*[1] hasta resúmenes que solo incluyen la aparición de una persona en un video (34). Es dinámico, y a diferencia del *Storyboard*, preserva el flujo temporal del video al recorrer lineal y continuamente cierta porción de contenido de video, dependiendo de una longitud determinada (32). Por tanto, es una secuencia de tomas, elegidas teniendo en cuenta un grupo de criterios. Es importante señalar que el audio de los contenidos de video será despreciado producto a que se hace muy difícil coordinar los cambios de tomas entre audio y video.

1.4 Análisis de soluciones existentes

En los últimos años han ocurrido grandes avances en la generación de resúmenes de video. Las publicaciones científicas sobre este tema hacen propuestas de múltiples técnicas para efectuar este proceso con mayor calidad o rapidez. A continuación se hace el análisis de dos soluciones, que abarcan el proceso completo (9) (8) (30).

1.4.1 Framework para la generación de resúmenes escalables de videos

El objetivo principal de esta solución es generar múltiples resúmenes de video a partir de un único análisis. En la misma se plantea que un buen resumen debe de aportar la mayor cantidad de información posible y garantizar una presentación agradable para el usuario. Para lograr lo primero, buscan seleccionar la mayor cantidad de información representativa posible al tiempo que eliminan la redundancia. La segunda propiedad busca que el resumen además sea cómodo y agradable de ver para el usuario, buscando entre otras cosas, evitar escenas de transiciones que podrían confundir o desorientar al cliente. (8)

La longitud de un resumen también juega un papel importante según los autores. Debe ser bastante reducida pero el resumen debe preservar toda la información que pueda. Porque esta no solo afecta el aspecto semántico del resumen sino también la comodidad visual.

Para efectuar la detección de tomas utilizan una característica del formato h.264 y MPEG-4. La misma les proporciona con bastante precisión el fotograma inicial de cada cambio de toma. Sin embargo depende de que la media este en este formato específicamente.

[1] Resumen o avance en imágenes de una película (14).

La escalabilidad de los resúmenes es definida como un grupo de resúmenes embebidos. De manera que si tenemos varios resúmenes $SS = \langle S^1, \dots, S^q, \dots, S^Q \rangle$ estos se corresponderán a la siguiente fórmula $S^1 \subset S^2 \dots \subset S^q \dots \subset S^Q \subset V$. Siendo V el video en su totalidad y S^Q el resumen de mayor extensión. Conformando así, un resumen de una longitud máxima, del cual se puede obtener varios resúmenes con longitudes menores y perdiendo la menor cantidad de información posible.

Para realizar la selección de las tomas se utiliza un proceso de selección iterativo. Este procedimiento consiste en ir seleccionando la toma que tenga más distancia con las seleccionadas anteriormente. Para dar una puntuación o nivel de relevancia las tomas seleccionadas se ven dos aspectos, distancia y duración, también a medida que se va seleccionando tomas va disminuyendo su relevancia.

Resultados:

Después de realizar pruebas utilizando el TRECVid[2] 2005 obtuvieron los siguientes resultados. Para la detección de tomas se obtuvo un rendimiento bastante bueno. También se demostró que el procesamiento no sería suficiente para casos de tomas de mayor longitud. También se hizo pruebas de resúmenes utilizando canales de noticias y se obtuvieron resultados bastante buenos, aunque se reconoce que el algoritmo puede ser mejorado (8).

1.4.2 Generación de resúmenes de video en línea basado en árboles binarios

Esta solución propone la utilización de un algoritmo basado en la generación dinámica de un árbol de tomas que emulan las distintas posibilidades para la inclusión o exclusión de las tomas entrantes. El mejor camino en este árbol binario es seleccionado iterativamente y provee de la información requerida para decidir entre seleccionar o descartar cada fragmento de video entrante. (30)

Para dar una puntuación a la tomas detectadas se emplean varios criterios entre estos están:

- Duración: Tomando un 2% como el tamaño máximo que puede tener una secuencia en dependencia de la extensión será su prioridad en el resumen.
- Continuidad: Se considera que un resumen sería perceptualmente más agradable si se logra que las secuencias escogidas tengan la mayor continuidad posible.
- Distancia: A mayor distancia con respecto a las otras secuencias mayor cantidad de

[2]TREC Video Retrieval Evaluations concurso o evento que se basa en distintas líneas de investigación del procesamiento de video, para esto tienen varias Base de datos de pruebas

información.

- Variabilidad o redundancia: Este indicador nos da una idea de que tan diferentes son los fotogramas dentro de una misma toma.

Esta solución provee un mecanismo genérico en el cual las características del procesamiento pueden ser controladas, en dependencia del retardo de entrada del video y las características del hardware de la computadora. Es posible configurar el algoritmo para que sea más rápido y funcione con un menor consumo de memoria (considerando árboles más pequeños con menos ramas para su evaluación) o para una generación de resúmenes mejores (árboles más profundos con mayor cantidad de ramas).

Resultados:

Dos pruebas utilizando el TRECVid 2008 con una computadora bastante modesta para la tecnología del momento demostraron que el algoritmo mejoraba los tiempos de la mayoría de los sistemas. Pero el resultado final depende de la configuración que se defina entre más moderno sea el hardware mejores son los resultados (30).

1.5 Conclusiones parciales

Al concluir el capítulo se puede afirmar que se cumplió con los objetivos propuestos. Se investigaron soluciones completas y técnicas específicas para efectuar resúmenes de video. Como resultado de esta investigación se puede concluir que:

- Las técnicas de detección de tomas tienen deficiencias que pueden ser complementadas si se utilizan combinaciones de ellas en lugar de una sola.
- Los descriptores locales representan el mecanismo más preciso para establecer distancias entre fotogramas, por lo que se debe tener en consideración para desarrollar la solución. De estos el más adecuado para utilizar es el **SURF** por sus características de mayor robustez, mejores tiempos de cómputo y por generar menos datos.
- El algoritmo más adecuado para efectuar el agrupamiento es el basado en grafos porque responde a las necesidades de la etapa de selección, además de ser muy rápido y de simple implementación.
- Los *Skims* o resúmenes de secuencias de imágenes son más adecuados para generar el resultado final debido a que aportan más información que los *Storyboards* y son más llamativos.

CAPÍTULO 2. HERRAMIENTAS Y TECNOLOGÍAS

Introducción

Una vez conocidos los fundamentos teóricos relacionados con la investigación, este capítulo tiene como objetivo seleccionar las herramientas y tecnologías adecuadas para efectuar tanto el análisis y diseño del componente, como la construcción del mismo. Se elige la metodología utilizada para efectuar el proceso de desarrollo del componente. Al tiempo, que se establecen la biblioteca de procesamiento gráfico y el lenguaje de programación, que son auxiliados por un marco de trabajo y un entorno de desarrollo integrado.

2.1 Metodología de desarrollo

Una metodología es un conjunto de procedimientos, técnicas y herramientas que ayudan a los desarrolladores a realizar un nuevo software. La misma, indica como hay que obtener los distintos productos parciales y finales, en dependencia de su utilización se logrará llegar a un producto final con la calidad que requiere. (35)

2.1.1 Rational Unified Process (RUP)

RUP es un proceso de desarrollo que define claramente quien, cómo, cuándo y qué debe hacerse; este aporta herramientas como los Casos de Uso, que definen los requisitos además de permitir la ejecución iterativa del proyecto y del control de riesgos. (35) Entre sus principales características se encuentra que es:

- Guiado por los Casos de Uso: los Casos de Uso son una técnica de captura de requisitos que representan funcionalidades del sistema, las cuales definen lo que el usuario desea obtener y permiten guiar todo el ciclo de vida de la aplicación o proyecto, para crear un resultado que satisfaga las necesidades esperadas.
- Centrado en la Arquitectura: la arquitectura es la organización o estructura de las partes más relevantes de un sistema, porque brinda una perspectiva clara y una visión común del mismo entre todos sus implicados. En esta se describen los procesos del negocio que son más importantes, teniendo en cuenta los elementos de calidad, rendimiento, reutilización y flexibilidad.

- Iterativo e Incremental: El proceso reconoce que es práctico dividir grandes proyectos en proyectos más pequeños o mini-proyectos. Cada mini-proyecto comprende una iteración que resulta en un incremento. Una iteración puede abarcar la totalidad de los flujos del proceso. Las iteraciones son planificadas en base a los Casos de Uso. (36)

Fases de un ciclo de RUP

Durante cada ciclo o iteración del proceso de desarrollo de una aplicación se ejecutan cuatro fases cuyos nombres varían en dependencia de la documentación que se revise pero siempre cumplen el mismo propósito. A continuación una breve descripción de cada fase:

- **Inicio**: Se concibe la idea del producto, y se revisa y confirma el entendimiento sobre los objetivos centrales del negocio. También se establece la viabilidad del producto y el alcance del mismo.
- **Elaboración**: Se establece la estructura base para la arquitectura del sistema, proporciona el diseño del mismo y se especifican en detalle los casos de uso.<
- **Construcción**: Completa el desarrollo del sistema basado en la estructura ya especificada en la fase anterior. También se refina el diseño para llevarlo a código fuente.
- **Transición:** Se garantiza el cumplimiento de los requisitos, buscando satisfacer a las partes interesadas. Se prepara el ambiente y se completan, identifican y corrigen defectos del sistema. La fase termina con un cierre dedicado al aprendizaje de lecciones, que quedan para futuros ciclos. (37)

Flujos principales

El Proceso Unificado identifica los flujos de trabajo de mayor importancia que se generan durante el proceso de desarrollo de software. Los mismos incluyen el modelado de negocio, requerimientos, análisis, diseño, implementación y prueba. Los flujos no son consecutivos y se efectúan durante las cuatro fases de desarrollo. (37)

Justificación de la elección de la metodología

La elección de RUP como metodología de software está fundamentada primeramente porque es la metodología sobre la que más conocimientos ha adquirido el autor durante sus estudios y por tanto tiene un mayor dominio de esta. Además la misma:

- Genera modelos que facilitan mejores prácticas de análisis y diseño del software.
- Tiene un enfoque orientado a objetos.
- Es repetible, adaptable y de evolución continúa.
- Es la metodología utilizada para desarrollar los subsistemas y componentes del proyecto al que pertenece el autor.

2.1.2 Lenguaje de modelado. Unified Modeling Language (UML)

El UML 2.0 es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. El mismo busca visualizar, especificar, construir y documentar un sistema de software a través de representaciones visuales. UML ofrece un estándar para describir un sistema a través de modelos, incluyendo aspectos conceptuales como procesos de negocio y funciones del sistema. Este lenguaje cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas. (37)

2.2 Herramienta CASE

CASE, *Computer Aided Software Engineering*, traducido como Ingeniería de Software Asistida por Computadora, consiste en un conjunto de guías para el desarrollo de aplicaciones informáticas, desde la planificación (el análisis y el diseño), hasta la generación del código de las aplicaciones. Las herramientas CASE tienen como objetivo principal proveer un lenguaje para describir el sistema, que permita su fácil entendimiento a la hora de generar determinadas aplicaciones. (38)

2.2.1 Visual Paradigm

Visual Paradigm Enterprise Edition v5.0 ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a distintas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos.

Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del

software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería del Software. (39)

2.3 Bibliotecas

En la actualidad existe un gran desarrollo en el procesamiento de imágenes mediante el uso de computadoras. Para facilitar y acelerar este desarrollo diversas entidades han creado colecciones de código que brindan funcionalidades que en un área de trabajo determinado, a estas colecciones se les conoce como bibliotecas. A continuación, se muestra un análisis sobre algunas de estas bibliotecas (VXL, LTI y OpenCV). (40) (41) (42)

2.3.1 Vision-something-Libraries (VXL)

Es una colección de bibliotecas de C++ diseñadas para la investigación e implementación del procesamiento de imágenes. Fue creado con el objetivo de lograr un sistema ligero, rápido y consistente. Está escrito en C++ y diseñado para que sea multiplataforma.

Las principales bibliotecas que contiene VXL son: (40)

- vnl (numerics): Contenedores numéricos y algoritmos.
- vil (imaging): Cargar, salvar y manipular imágenes en muchos formatos comunes.
- vgl (geometry): Puntos geométricos, curvas y otros objetos elementales en 1, 2 o 3 dimensiones.
- vsl (streaming I/O), vbl (basictemplates), vul (utilities): Funcionalidades adicionales independientes de plataforma.

Requisitos

Sistema operativo Windows NT con Microsoft Visual Studio 7 o basado en Unix (Linux/Solaris/Irix).

Licencia

El permiso para usar, copiar, modificar o distribuir este software y su documentación para cualquier propósito es garantizado mientras se cumpla con tres condiciones:

- Que el aviso de derecho de copia y el permiso aparezca en todas las copias del software la documentación relacionada.

- Que el nombre TarjetJRConsortium, no sea usado en ningún aviso o relación de publicidad para el software sin especificación.
- Cualquier modificación sea claramente marcada y resumida en un log historial de cambios.

Análisis

Biblioteca de licencia libre, pero que hace uso de aplicaciones de licencia privada. Además no cuenta con algoritmos de agrupamiento ni de extracción de características de las imágenes.

2.3.2 Lehrstuhlfuer Technische Informatik (LTI)

Biblioteca orientada a objetos, con algoritmos y estructuras de datos frecuentemente usadas en el procesamiento de imágenes. Ha sido desarrollada usando GCC[3] en Linux, y Visual C++ en Windows NT. No ha sido probada en otras plataformas.

Muchas clases encapsulan funcionalidades de Windows/Linux con el objetivo de simplificar el trabajo con especificidades de hardware o del sistema. El resto (más de 500 clases) maneja principalmente alguno de estos campos: (41)

- Algebra Lineal.
- Clasificación y Agrupamiento.
- Procesamiento de imágenes.
- Herramientas de Dibujo y Visualización.

Requisitos

Para Windows NT se necesita al menos MS[4] Visual C++ 6.0 con Service Pack 5.0 o MS Visual C++.NET 2003 (La versión .NET 2002 no es soportada).

Licencia

La LTI-Lib fue creada como software libre. El permiso para su uso, copia, modificación y distribución de este software y su documentación bajo los términos de la GNU LGPL esta admitido.

Análisis

[3] GNU Compiler Collection. Conjunto de compiladores creados por el proyecto GNU.
[4] MicroSoft

Esta biblioteca es de licencia LGPL pero necesita hacer uso de herramientas de licencia privada (Visual Studio) por lo que no cumple con el objetivo de lograr un soporte multiplataforma y que sea completamente libre.

2.3.3 Open Source Computer Vision (OpenCV)

El OpenCV 2.3.1 es una biblioteca de funciones de C++ programada para el procesamiento de imágenes en tiempo real. Sus aplicaciones abarcan desde el arte interactivo, la inspección de minas, el marcado de mapas en la web hasta la robótica avanzada. (42)

Esta biblioteca agrupa las funcionalidades en distintos módulos. A continuación, algunos de los módulos que la componen:

- core: Define las estructuras de datos básicas.
- imgproc: Módulo de procesamiento de imágenes que incluye el filtrado lineal y no lineal, transformaciones geométricas de la imagen, conversiones de color, histogramas, etc.
- video: Módulo de análisis de video que incluye estimación de movimiento, extracción de trasfondo y algoritmos de rastreo de objetos.
- features2d: Módulo que detecta características, contiene descriptores y descriptor matchers.

Requisitos

Es multiplataforma por lo que soporta sistemas operativos como Windows, Linux, Android y Mac OS.

Licencia

OpenCV es liberado bajo la licencia BSD para ambos usos tanto académico como comercial.

Análisis

Esta biblioteca es la más adecuada para desarrollar el componente propuesto porque es de licencia completamente libre, puede utilizarse en Windows o Linux (multiplataforma) y cuenta con las funcionalidades que se necesitan para implementar la solución propuesta.

2.4 Lenguajes de programación

Según la definición teórica, como lenguaje se entiende a un sistema de comunicación que posee una determinada estructura, contenido y uso. La programación es, en el vocabulario propio de la informática, el procedimiento de escritura del código fuente de un software. De esta manera, puede decirse que la programación le indica al programa informático qué acción tiene que llevar a cabo y cuál es el modo de concretarla (43). El empleo de un lenguaje de programación adecuado a las necesidades de desarrollo es imprescindible para poder efectuar la implementación del componente. La biblioteca seleccionada (OpenCV) está disponible para los lenguajes de programación Python, C y C++.

2.4.1 Python

Lenguaje de programación creado por Guido van Rossuma principios de los años 90. Se trata de un lenguaje interpretado o de script, con una asignación de tipos dinámica, multiplataforma y orientado a objetos. (44) Este lenguaje cuenta con una sintaxis simple, clara y sencilla. El gestor de memoria, la gran cantidad de bibliotecas disponibles y la potencia del lenguaje, hacen que desarrollar una aplicación en Python sea sencillo y muy rápido.

Python sin embargo, no es adecuado para la programación de bajo nivel o para aplicaciones en las que el rendimiento sea crítico. Además la biblioteca OpenCV no tiene todas sus funciones disponibles para el lenguaje Python lo cual limita las posibilidades para desarrollar el componente. (45)

2.4.2 C

El lenguaje de programación C fue diseñado e implementado por primera vez por Dennis Ritchie, en el año 1972, en los Laboratorios Bell como una evolución del lenguaje B, que estaba basado en BCPL. (46) C es un lenguaje altamente difundido y utilizado en todo el mundo, debido principalmente a su flexibilidad y eficiencia. El mismo es imperativo o procedimental. Esto significa que indica secuencias de acciones con el fin de llegar a un objetivo. Es un lenguaje estructurado debido a que los programas escritos con él se pueden organizar en módulos.

Entre sus desventajas se encuentran que no es orientado a objetos, la carencia de programación para multi-hilo de forma nativa y la no existencia de un recolector de basura. Este fue el lenguaje escogido

para desarrollar la primera versión de la biblioteca OpenCV (1.0); pero en la versión 2, existen funcionalidades en módulos que no tienen implementación para este lenguaje de programación. (47)

2.4.3 C++

Este es un lenguaje de programación creado por Bjarne Stroustrup en los laboratorios de At&T[5] en 1983. El C++ es un derivado del lenguaje C. Creado con el objetivo de agregar características que ya existían en otros lenguajes como Smalltalk[6]. (47) Algunas de estas características fueron:

- Definición de módulos (llamados clases)
- Programación orientada al objeto.
- Tipificación más estricta.
- Tratamiento de excepciones.
- Plantillas (equivalentes a módulos genéricos).

C++ es el lenguaje de programación principal de OpenCV 2.x, todas las funcionalidades que tiene están implementadas para el mismo. Debido a las características que posee este lenguaje, se selecciona como el lenguaje de programación utilizado para desarrollar la implementación del componente. (47)

2.5 Marco de trabajo

Un marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que permite resolver nuevos problemas. En el desarrollo de software constituye una estructura conceptual y tecnológica de soporte definida, normalmente cuenta con artefactos o módulos de software concretos, empleados fundamentalmente para desarrollar código fuente. (48)

2.5.1 Qt 4.8

Qt es un *framework*[7] utilizado en el desarrollo de aplicaciones informáticas creado por la compañía Trolltech, el mismo es muy utilizado para la creación de interfaces de usuario, pero también provee

[5] Compañía estadounidense de telecomunicaciones. Provee servicios de voz, video, datos, e Internet a negocios, clientes y agencias del gobierno.
[6] Lenguaje de Programación que permite realizar tareas de computación mediante la interacción con un entorno de objetos virtuales.
[7] Marco de trabajo

varias clases para facilitar otras tareas de programación como el manejo de sockets, soporte para programación multihilo, comunicación con bases de datos, manejo de cadenas de caracteres, entre otras. La elección de este marco de trabajo se debe a que es de licencia libre y se integra perfectamente con el lenguaje de programación y la biblioteca de procesamiento grafico utilizados. (49)

2.6 IDE

Un IDE o entorno de desarrollo integrado, es un editor de código que cuenta con características que facilitan la implementación con uno o más lenguajes de programación. La selección de un IDE adecuado es imprescindible para lograr mayor comodidad y eficiencia a la hora de desarrollar una solución. A continuación se exponen las características del IDE seleccionado. (50)

2.6.1 Qt-Creator

El Qt-Creator versión 2.4.4. es un IDE creado por Trolltech, multiplataforma, diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil. (51) Entre las características principales que posee el Qt-Creator, se encuentran que:

- Posee un avanzado editor de código C++.
- Posee también una GUI integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto integrada.
- Depurador visual.
- Resaltado y auto-completado de código.
- Soporte para refactorización de código.

2.7 Conclusiones parciales

Después de haber realizado un estudio y análisis de las herramientas y tecnologías de desarrollo a utilizar en la construcción del componente para elaborar resúmenes escalables de video. Se determina utilizar **Visual Paradigm** y **UML,** como Herramienta CASE y lenguaje de modelado respectivamente por constituir prácticamente el lenguaje estandarizado para la modelación de sistemas de cómputo. Al mismo tiempo se elige el **OpenCV 2.3.1** como biblioteca de procesamiento gráfico porque posee diversas funcionalidades implementadas que son necesarias para el procesamiento del video que se realiza en el componente, es multiplataforma y se distribuye bajo licencia de libre para su explotación

en cualquier ámbito. Por otra parte se opta por utilizar el lenguaje **C++** por ser el lenguaje nativo de OpenCV. Se elige al **Qt-Creator** como IDE y framework, debido a que cumple con las especificaciones necesarias para implementar el componente en C++, se integran de forma fácil con la biblioteca OpenCV, permiten obtener un resultado para entornos multiplataforma y el autor de la investigación posee habilidades de desarrollo con estas tecnologías.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

Introducción

Una vez conocidas las herramientas y tecnologías a utilizar en el proceso de creación del componente es necesario efectuar el análisis y diseño del mismo. Con el objetivo de lograr un software de calidad y siguiendo las pautas de la metodología seleccionada, se hace necesaria la determinación de los requisitos y la estructura de la solución. Resultando en varios modelos y prácticas, entre los que se encuentran el modelo de diseño, de implementación, y los patrones arquitectónicos y de diseño utilizados. En el capítulo también se presenta una descripción general del componente a implementar, para lograr un mayor entendimiento del funcionamiento del mismo.

3.1 Modelo de dominio

Se plantea la conceptualización del entorno mediante un modelo de dominio para comprender el contexto del componente. Esto se debe a que durante la investigación no se definen procesos de negocio que estén involucrados con el dominio del problema. Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Utilizando UML, un modelo del dominio se puede representar como un conjunto de diagramas de clases. Estas clases pueden representar: (52)

- Objetos del dominio o clases conceptuales.
- Asociaciones entre las clases conceptuales.
- Atributos de las clases conceptuales.

3.1.1 Descripción de los conceptos fundamentales

Cliente: Sistema o usuario que necesita obtener un video.

Servidor de archivos audiovisuales: Estación con gran capacidad de almacenamiento que provee contenidos audiovisuales.

Video: Conjunto de imágenes y sonidos que pueden ser agrupados en tomas, que tienen un nombre que los identifica, una determinada duración y una frecuencia de fotogramas por segundo (Fps).

Toma: Conjunto de imágenes o fotogramas consecutivos que representan una escena y cierta cantidad de información visual. Esta información es directamente proporcional a la duración de la toma e inversamente proporcional a su redundancia.

Fotograma: Imagen por separado que, en dependencia de su ancho, alto y profundidad de color, será la calidad visual que aporte, lo cual es directamente proporcional al espacio que ocupará este fotograma en memoria.

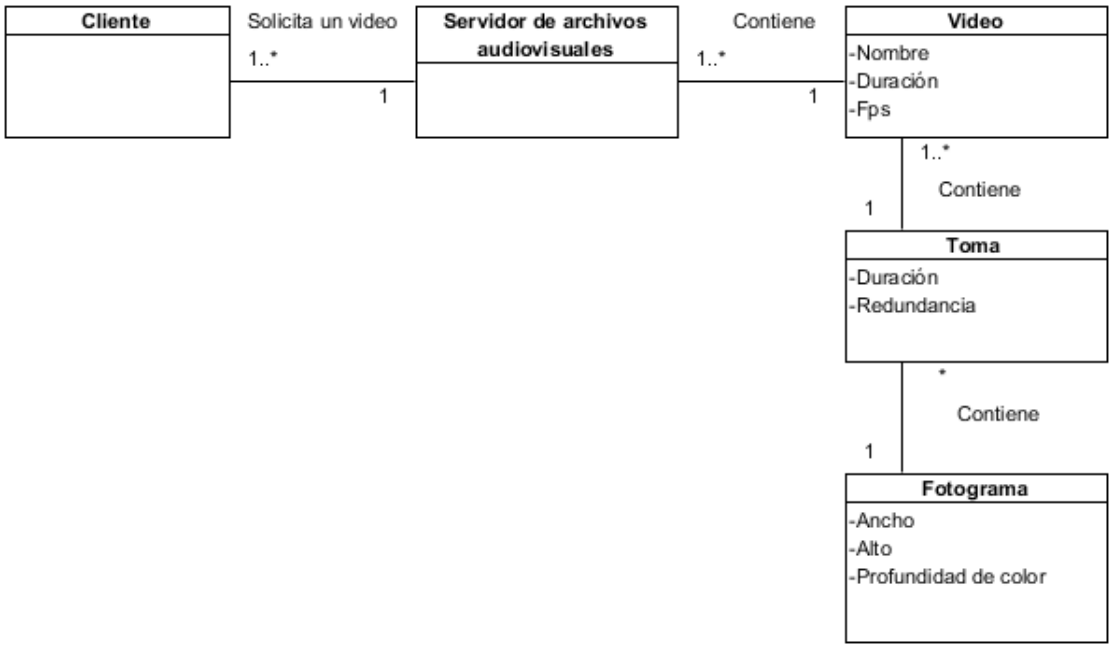

Figura 5. Modelo de dominio.

3.1.2 Descripción del modelo de dominio

El modelo reproduce una situación que ocurre en un sistema de gestión de contenido multimedia en la que el **cliente** solicita un determinado video con el objetivo de obtener información del mismo. Este **video** es buscado y ofrecido por un **servidor de contenidos audiovisuales**. Es importante señalar que los contenidos audiovisuales están compuestos por **tomas** que aportan más o menos información al cliente, y que a su vez están compuestas por **fotogramas** consecutivos.

3.2 Requisitos

Para efectuar el desarrollo de software en la actualidad se hace necesario el análisis detallado de los requisitos funcionales. Esta tarea consiste en la generación de especificaciones correctas que

describan con claridad, sin ambigüedades, en forma consistente y compacta, las necesidades de los usuarios o clientes. (53)

3.2.1 Requisitos funcionales

Con el objetivo de desarrollar la solución se identificaron los siguientes requisitos funcionales:

- **RF 1** Detectar las tomas: El componente debe permitir efectuar la segmentación del video o separación del mismo en las tomas que lo componen.
- **RF 2** Analizar las características de las tomas: El componente debe permitir obtener la duración, variabilidad y distancia de las tomas.
- **RF 3** Calificar las tomas: El componente debe permitir obtener una puntuación para cada toma en dependencia de sus características.
- **RF 4** Agrupar las tomas: El componente debe permitir agrupar las tomas según el criterio de distancia entre las mismas.
- **RF 5** Generar el resumen escalable: El componente debe permitir generar el resumen escalable utilizando la puntuación dada a las tomas en el **RF 3** y la agrupación lograda en el **RF 4**.
- **RF 6** Seleccionar las tomas: El componente debe permitir seleccionar las tomas que conformaran el resumen resultante de acuerdo a la longitud o duración solicitada.
- **RF 7** Ordenar tomas seleccionadas: El componente debe permitir organizar la aparición de las tomas en el resumen de video siguiendo el flujo temporal que tengan en el video original.
- **RF 8** Construir el resumen de video: El componente debe permitir efectuar la concatenación de las tomas para obtener el resumen de video final.

3.2.2 Requisitos no funcionales

Eficiencia

- **RNF 1** Tiempo de respuesta: El tiempo de respuesta que deberá tener durante la utilización de las diferentes detecciones:
    - Detección mediante Histograma: 0.001s aproximadamente.
    - Detección mediante Líneas: 0.02s aproximadamente.
    - Detección mediante SURF: 0.1s aproximadamente.

- **RNF 2** Capacidad: El componente debe procesar un único video a la vez, por lo que solo puede atender a un cliente al mismo tiempo.
- **RNF 3** Utilización de recursos: El componente deberá hacer el menor uso de memoria posible, dando mayor utilización al disco duro, debido a dificultades de la biblioteca a la hora de manejar gran cantidad de estructuras **Mat**.

Soporte

- **RNF 4** Software:
    - Sistema operativo: Windows versiones XP, 7 y 8. Ubuntu Linux versiones 11.10, 12.04 y 12.10.
    - Biblioteca: OpenCV versión 2.3.1
- **RNF 5** Hardware:
    - Procesador Intel Core i5 o superior.
    - RAM: 1Gb o superior.
    - Disco Duro: 80Gb o superior.

Restricciones de diseño:

- **RNF 6** Arquitectura: El producto debe diseñarse bajo la arquitectura N-Capas.
- **RNF 7** Estándares: Se deben emplear los estándares establecidos por el proyecto para la codificación del componente, estos se pueden encontrar en el Anexo 2 del documento.

3.3 Definición de los Casos de Uso

Los Casos de Uso se usan con el objetivo de representar como debe de trabajar un sistema, por lo que son descripciones de las funcionalidades del mismo. Independiente de la implementación, describen bajo la forma de acciones y relaciones, el comportamiento de un sistema desde la perspectiva del cliente.

3.3.1 Actores del componente

Los actores son las entidades externas al sistema que tienen una relación con este y le exigen una funcionalidad. Entre los ejemplo de actores podemos tener operadores humanos, sistemas externos y entidades abstractas como el tiempo. (54)

Actor	Descripción
Cliente	Representa al sistema o el usuario que accede a la funcionalidad elaborar resúmenes de video que provee el componente.

Tabla 1. Descripción de actores del componente.

3.3.2 Diagrama de Caso de Uso

El diagrama de Caso de Uso se utiliza para entender y describir los requisitos funcionales de la aplicación. (52) A continuación se muestra el diagrama de Caso de Uso perteneciente a este trabajo.

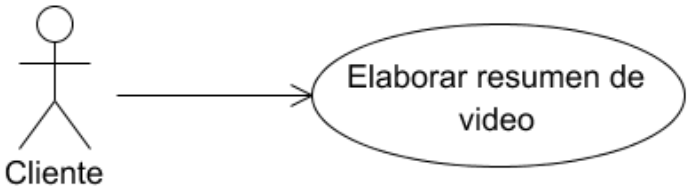

Figura 6. Diagrama de Caso de Uso.

3.3.3 Descripción del Caso de Uso

Un Caso de Uso es una técnica utilizada para la captura de requisitos de una nueva aplicación o una actualización de software. El mismo proporciona uno o más escenarios que indican cómo debe actuar el sistema con el usuario, para lograr un objetivo específico. (52) A continuación se describe el Caso de Uso Elaborar resumen de video.

Caso de Uso	Elaborar resumen de video
Actores	Cliente
Resumen	El Caso de Uso inicia cuando el actor solicita elaborar un resumen de video. Esta petición es recibida por el componente a través de la interfaz y es efectuada. Se finaliza cuando el componente devuelve el resumen de video.
Precondiciones	El componente debe tener acceso a un servidor de contenidos audiovisuales que contenga el video del cual se solicitará la elaboración del resumen.
Referencias	RF1, RF2, RF3, RF4, RF5, RF6, RF7 y RF8

Complejidad	Alta	
Prioridad	Crítico	
Flujo Normal de Eventos		
Actor		**Sistema**
1.	El Caso de Uso inicia cuando el cliente utiliza la funcionalidad Elaborar resumen de video, en la que especifica el video y la duración del resumen.	
2.		Comprueba que el video no halla sido analizado con anterioridad.
3.		Detecta las tomas que conforman al video original, efectuando la detección basada en histogramas, en líneas y en Surf.
4.		Determina la longitud y variabilidad de las tomas extraídas y la distancia de cada toma con respecto a las demás.
5.		Genera el resumen escalable utilizando las características utilizadas anteriormente.
6.		Guarda en un archivo de tipo YAML que tendrá el nombre del video el resumen escalable correspondiente a este video.
7.		Construye el resumen resultante a partir del resumen escalable con la duración solicitada.
8.		Copia el resumen a la carpeta establecida en el código para guardarlo y se finaliza el Caso de Uso.
Flujo Alterno “El video ha sido analizado con anterioridad”		
Actor		**Sistema**
3.		Carga desde un archivo el resumen escalable correspondiente al video. Y se pasa al paso 7 del Flujo Normal de

		Eventos.
Relaciones	CU Incluidos	No tiene.
	CU Extendidos	No tiene.

Tabla 2. Caso de Uso elaborar resumen de video.

3.4 Descripción general de la propuesta de algoritmo

Una vez identificados los requisitos funcionales a implementar en el componente, se hace necesario explicar o describir como se piensa dar solución a los mismos, a través de la propuesta de algoritmo a utilizar. Para lograr una mejor comprensión de este tema y haciendo uso de los conocimientos adquiridos durante la investigación se divide el proceso de generación de resúmenes en 3 etapas: Análisis, Generación y Construcción del resumen de video, las cuales serán explicadas en los próximos epígrafes.

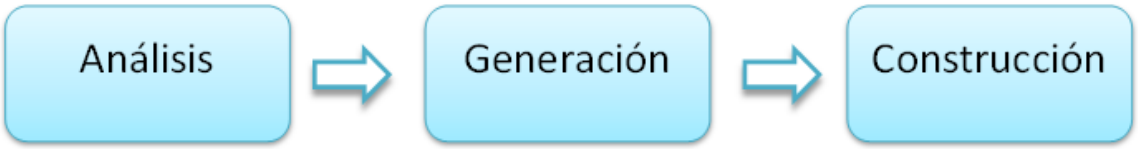

Figura 7. Descripción del proceso general.

En la figura 5 se puede observar el orden establecido para la ejecución del algoritmo. Primero se efectúa el análisis de las tomas. Como resultado de esta etapa se obtiene una lista de tomas que contiene un fotograma de cada toma, así como la variabilidad dentro de la misma y su longitud. Durante la generación, estas tomas se agrupan mediante un proceso de agrupamiento. Utilizando las características obtenidas en la etapa anterior y la distancia entre las mismas, se agrupan y se seleccionan las más adecuadas, obteniendo como resultado otra lista de tomas ordenadas según su relevancia. La etapa de construcción puede ser efectuada tantas veces como sea necesario y se encarga de ir seleccionando según su nivel de relevancia, cada una de las tomas seleccionadas en la etapa de generación hasta que el resultado cuente con la longitud deseada, luego se ordenan y se crea el video resultante.

3.4.1 Análisis

En esta etapa, primeramente se efectúa la detección de tomas, utilizando tres descriptores visuales para lograr un mejor resultado. La figura 8 muestra el orden en que se ejecutan cada uno de los descriptores. A continuación se explica el proceso mostrado.

Figura 8. Descripción del análisis del video.

Decodificación

El algoritmo comienza cuando se recibe el video del cual se quiere obtener el resumen. Una vez recibido el video se pasa a decodificar y extraer los fotogramas del mismo. Para logralo se utilizan funcionalidades de la biblioteca OpenCV. Primeramente es necesario crear una estructura de tipo **VideoCapture** encargada de decodificar el video y un objeto de tipo **Mat** que contendrá el fotograma extraído. Luego se utiliza la función **read** de la estructura que contiene el video para guardar el próximo fotograma en el objeto. Este segmento de código debe de estar encapsulado dentro de un ciclo de manera que se efectué el proceso para todos los fotogramas del video.

Detección mediante histogramas

Una vez capturados los fotogramas se pasa a la primera de las técnicas utilizadas para segmentar el video, la detección de tomas mediante histogramas. Esta técnica se divide en los siguientes pasos:

1. Calcular el histograma de cada fotograma.
2. Comparar cada fotograma con el contiguo y guardar los resultados.
3. Establecer un umbral entre las comparaciones.

4. Detectar todas las comparaciones que sobrepasen este umbral como un cambio de toma.

Para calcular el histograma de color, en este caso de tres colores, es necesario primeramente separar la imagen en sus tres canales (rojo, verde y azul) utilizando el método **split**, y luego efectuar el cálculo de histograma para cada uno de estos mediante la función **calcHist**. La comparación se efectúa utilizando la funcionalidad **compareHist** que retorna un valor numérico. El resultado de la comparación de todos los fotogramas es umbralizado y de esta forma se detectan los cambios de tomas.

Detección mediante líneas

En este punto el video ya ha sido segmentado utilizando histogramas pero este es un algoritmo muy sensible a los cambios de iluminación. Por esta razón se efectúa la detección mediante bordes a los fotogramas que representan el cambio de una toma a otra. Esta técnica de detección se logra a través de la ejecución de los siguientes pasos:

1. Convertir los fotogramas a escala de grises.
2. Realizar un desenfoque gaussiano.
3. Aplicar una función de detección de bordes.
4. Expandir las líneas resultantes.
5. Comparar los resultados.
6. Establecer un umbral con las comparaciones.
7. Detectar los falsos positivos de la etapa anterior.

El primer y segundo paso se logran utilizando las funciones **cvtColor** y **GaussianBlur** de OpenCV. Los mismos preparan la imagen para efectuarle la detección de bordes **Canny**. El resultado es luego expandido mediante el método **dilate** para disminuir la sensibilidad al movimiento y se efectúa una operación (**bitwise_and**) que genera una imagen con las coincidencias entre los fotogramas comparados.

Detección mediante SURF

Después de haber efectuado el análisis con los otros descriptores pueden existir cortes dentro de una toma conocidos como falsos positivos. Por tanto se utiliza el descriptor SURF para aquellos límites entre tomas que como resultado del análisis de histogramas pertenezcan a un margen dudoso. La ejecución de esta técnica se logra mediante los siguientes pasos:

1. Extraer las características.
2. Obtener los descriptores.
3. Buscar coincidencias entre fotogramas.
4. Umbralizar el resultado.
5. Detectar falsos positivos de la etapa anterior.

Para detectar las características de una imagen, es necesario crear un vector de **keypoints** en el que se almacenarán los resultados de la detección utilizando un objeto de tipo **SurfFeatureDetector**. Luego se convierte el vector a un objeto de tipo **Mat** utilizando **SurfDescriptorExtractor** y el resultado es comparado con el de otro fotograma mediante la estructura **BruteForceMatcher**. Como resultado de este proceso se obtiene un valor numérico que representa las coincidencias entre los fotogramas. Este valor numérico es luego umbralizado, para posteriormente utilizarse en la etapa de detección de falsos positivos.

3.4.2 Generación

Al terminar la etapa anterior se obtiene como resultado una lista que contiene los siguientes aspectos de cada toma:

- Posición inicial.
- Longitud.
- Redundancia interna.
- Fotograma intermedio.

El objetivo de esta etapa es obtener las tomas más representativas del video, buscando eliminar la redundancia y ofreciendo la mayor cantidad de información posible. El siguiente diagrama da una idea proceso efectuado.

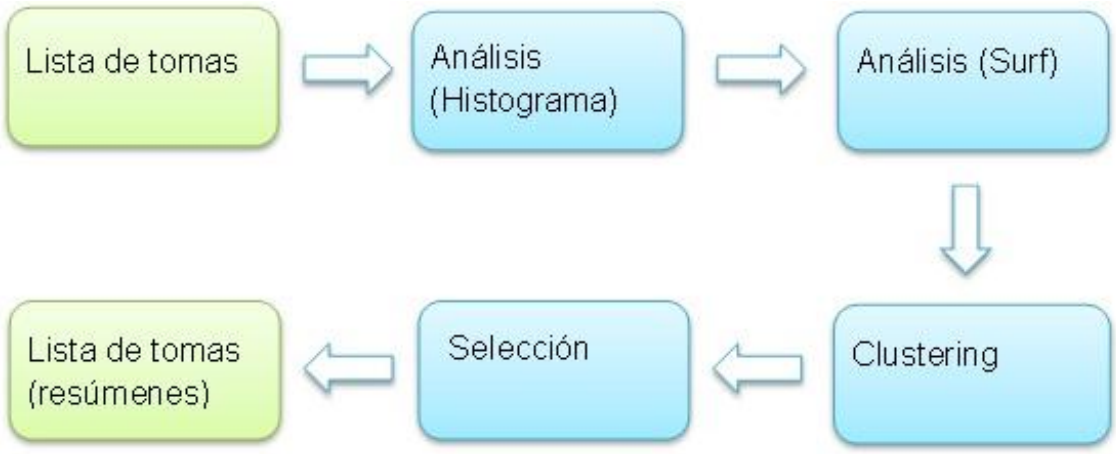

Figura 9. Descripción de la generación del resumen.

Como resultado del análisis de histograma y SURF se obtiene una matriz de proximidad (matriz de distancias). Esta matriz es utilizada en el proceso de agrupamiento (*clustering*) mediante grafos. Una vez obtenidos los grupos o clústeres se extrae la toma con mayor puntuación con respecto a las características de duración y variabilidad del grupo, esta toma sería la más representativa del grupo. La figura 10 muestra el resultado esperado.

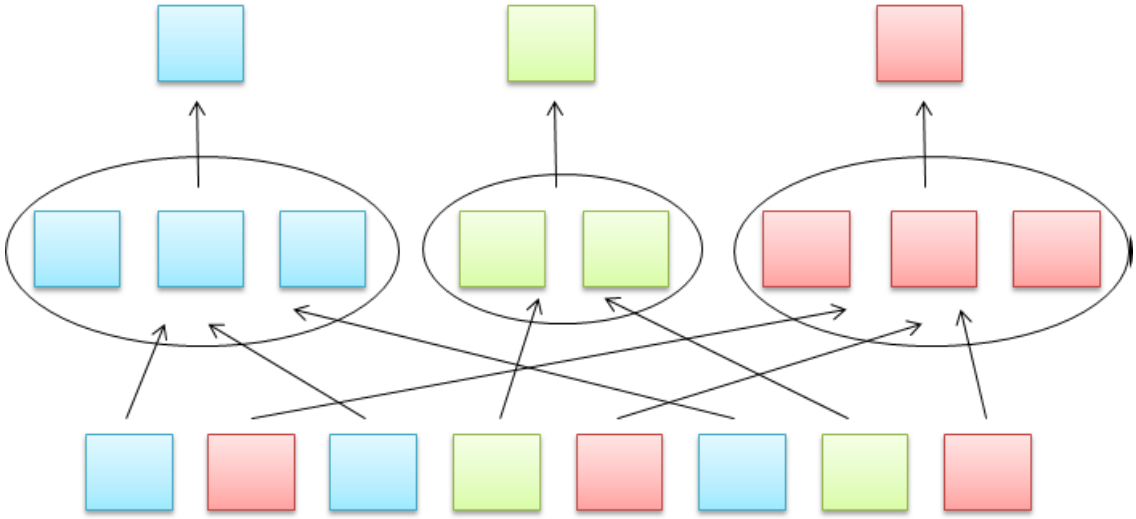

Figura 10. Descripción del agrupamiento.

Luego utilizando los demás criterios de selección se selecciona la toma con mayor puntuación y se van seleccionando de forma iterativa las tomas más diferentes (con mayor distancia) a las seleccionadas

hasta el momento. De esta manera se confecciona la lista resultante que tiene los posibles resúmenes embebidos. El orden de aparición de las tomas en la lista resultante representa la prioridad de las mismas para pertenecer a un resumen solicitado.

3.4.3 Construcción

Una vez obtenida la lista resultante con las tomas más relevantes ordenadas según su prioridad y entrada la longitud que se desea que tenga el resumen de video resultante. Se seleccionan, comenzando por el principio de la lista, las tomas que contiene la misma, hasta que se satisfaga la duración requerida. Habiendo seleccionado las tomas que formarán parte del resumen se ordenan siguiendo el flujo temporal del video original y se concatenan mediante el uso de una transición de desaparición simple. Esta etapa puede ser utilizada para obtener resúmenes de cualquier longitud deseada, por tanto, se puede repetir todas las veces que sea necesario.

3.5 Estilo arquitectónico

Los estilos arquitectónicos son modelos que describen a los patrones y las interacciones entre ellos. Estos ayudan a lograr un tratamiento estructural que se aplican más bien en la teoría, la investigación académica y la arquitectura en el nivel de abstracción más elevado. (55) El estilo seleccionado para efectuar el diseño de la aplicación es el de **llamada y retorno**, lo cual garantiza la capacidad de modificación y la escalabilidad de las aplicaciones del componente.

3.5.1 Patrón de arquitectura

Los patrones de arquitectura son la herramienta básica de un arquitecto a la hora de dar forma a la arquitectura de una aplicación. Un patrón de arquitectura se puede entender como un conjunto de principios que definen a alto nivel un aspecto de la aplicación. También se entienden como indicaciones abstractas de cómo dividir en partes el sistema y de cómo estas partes deben interactuar. Dentro de los patrones de arquitecturas que se pueden utilizar en el estilo seleccionado se encuentra la arquitectura N-Capas. (56)

Arquitectura N-Capas

El patrón de arquitectura en capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles

indican el tipo y la forma de interacción con otras capas y las responsabilidades de la funcionalidad que implementan. (57)

Figura 11. Representación de arquitectura dos capas.

Características

Sus principales características son que los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces bien conocidas y que cada nivel agrega las responsabilidades y abstracciones del nivel inferior. (56)

Beneficios

La utilización de esta arquitectura ofrece varias ventajas. A continuación se hace referencia a las que ventajas que se manifiestan en el componente:

- Aislamiento: se pueden realizar actualizaciones en el interior de las capas sin que esto afecte al resto de la aplicación.
- Rendimiento: distribuyendo las capas en distintos niveles físico se puede mejorar la escalabilidad, la tolerancia a fallos y el rendimiento.
- Capacidad de prueba: cada capa contiene una interfaz bien definida sobre la que realizar pruebas y la habilidad de cambiar entre diferentes implementaciones de una capa. (56)

Con el objetivo lograr una correcta utilización de este patrón arquitectónico se dividen las funcionalidades en dos capas:

- Interfaz: Compuesta por los métodos que garantizan la comunicación con otros componentes.
- Lógica de negocio: En esta capa se ejecutan las funcionalidades principales del componente, la misma incluye todo el proceso de generación de resúmenes.

3.6 Patrones de diseño

Un patrón de diseño nombra, abstrae e identifica los aspectos principales de una estructura común de diseño que da solución a problemas que ocurren una y otra vez en el desarrollo de software. El patrón de diseño identifica la participación de clases e instancias, sus roles y colaboración, y la distribución de responsabilidades. El mismo describe cuando se aplica, donde puede aplicarse teniendo en cuenta otras restricciones de diseño y las consecuencias de su uso. (58)

3.6.1 Patrones GRASP

Experto: Permite asignar las responsabilidades al experto en la información. La clase que cuenta con la información necesaria para cumplir determinada responsabilidad. Esta distribución puede observarse en gran parte de las clases del componente. (59) Por ejemplo, al efectuar la detección de tomas se divide el proceso en tres clases distintas (**CDeteccionHistograma**, **CDeteccionLineas** y **CDeteccionSurf**) que corresponden con cada una de las etapas de esta fase. Estas clases a su vez dependen de **CHistograma**, **CLineas** y **CSurf** que serían las clases expertas en la extracción de características de un fotograma y por tanto las responsables de efectuar esta tarea.

Controlador: Define quién deberá encargarse de atender un evento del sistema. Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar eventos del sistema. (59) Este patrón se evidencia en las clases cómo **CDeteccionHistograma**, **CDeteccionLineas** y **CDeteccionSurf** que se encargan de controlar los procesos de detección mediante histograma, detección mediante líneas y detección utilizando SURF, o sea, que actúan como intermediarios entre las clases **CHistograma**, **CLineas** y **CSurf** y la clase **CAnalisis**.

Creador: Asigna la responsabilidad de crear una clase a otra que agregue, contenga, registre o utilice la misma. (59) Este patrón se manifiesta en la clase **CAnalisis** que es la encargada de construir la lista de tomas (**CToma**) que será procesada en las próximas etapas.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases y recurre a ellas. Lograr un acoplamiento bajo significa que una clase no depende de muchas. (59) Por ejemplo, en la última fase del proceso de generación del resumen se hace uso de la clase **CConstruccion** que genera el resultado final de la solución. A pesar de su gran importancia el funcionamiento de esta clase solo depende de incluir la clase **CToma**, y ser inicializada por la clase **CComponenteAVS**.

Alta cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Lograr una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas para que no realicen un trabajo enorme. (59) Este patrón se utiliza en la totalidad de la estructuración de la implementación y se puede comprobar al ver la relación que existe entre **CComponenteAVS**, **CAnalisis**, **CDeteccionHistograma** y **CHistograma**. La clase **CComponenteAVS** engloba el proceso de generación completo pero le delega a **CControlDeteccion** la responsabilidad de realizar la detección de tomas. **CAnalisis** a su vez ejecuta a **CDeteccionHistograma** para efectuar la primera etapa de la detección y esta utiliza las funcionalidades que proporciona **CHistograma** para desarrollar el proceso.

3.6.2 Patrón GOF

Estrategia: Se utiliza cuando existen diversos algoritmos o políticas que están relacionados. Su objetivo es definir cada algoritmo o estrategia en una clase independiente, con una interfaz común. (60) Este patrón de diseño se refleja en clases como **CDistanciaHistograma** y **CDistanciaSurf** debido a que ambas reciben una lista de tomas y una matriz de distancias y trabajan sobre estas. A pesar de tener entradas y salidas equivalentes, procesan los datos de forma diferente.

3.7 Diagrama de clases de diseño

Los diagramas de clases de diseño (DCD) tienen como objetivo principal mostrar las especificaciones de las clases en una aplicación. En los mismos se representan las clases con sus asociaciones y atributos, los métodos que las componen y sus dependencias. (52)

El diagrama a continuación, muestra la distribución de las clases en el componente, donde se representan teniendo en cuenta la arquitectura en dos capas, sin mostrar sus atributos y métodos (el DCD completo puede encontrarse en el Anexo 1). En la capa de lógica de negocio se observa como la clase controladora principal **CComponenteAVS** está compuesta por las clases **CAnalisis**, **CGeneracion, CConstruccion** y **CPersistencia**, de estas clases las tres primeras se encargan de realizar las etapas establecidas en el algoritmo y la última garantiza la persistencia de los resúmenes escalables generados.

La clase **CAnalisis** está compuesta por **CDeteccionHistograma**, **CDeteccionLineas** y **CDeteccionSurf** que son generalizaciones de **CDeteccion**, clase esta que funciona como interfaz común para estandarizar las operaciones de detección mediante los distintos descriptores. Algo similar

a lo anteriormente visto sucede en la clase **CGeneracion**, esta vez en la relación que existe entre **CDistanciaHistograma** y **CDistanciaSurf** con **CDistancia**. Tanto en los casos mencionados como en **CDeteccionHistograma** y **CDeteccionSurf** se observa como agregan a las clases **CHistograma** y **CSurf** respectivamente, que son las que cuentan con las funcionalidades para determinar que la diferencia que hay entre dos fotogramas determinados. También es importante señalar la agregación de la clase **CToma** en **CDeteccion**, **CDistancia**, **CClustering** y **CGeneracion**, esta inclusión imprescindible porque la toma representa la unidad básica de procesamiento para efectuar resúmenes de video.

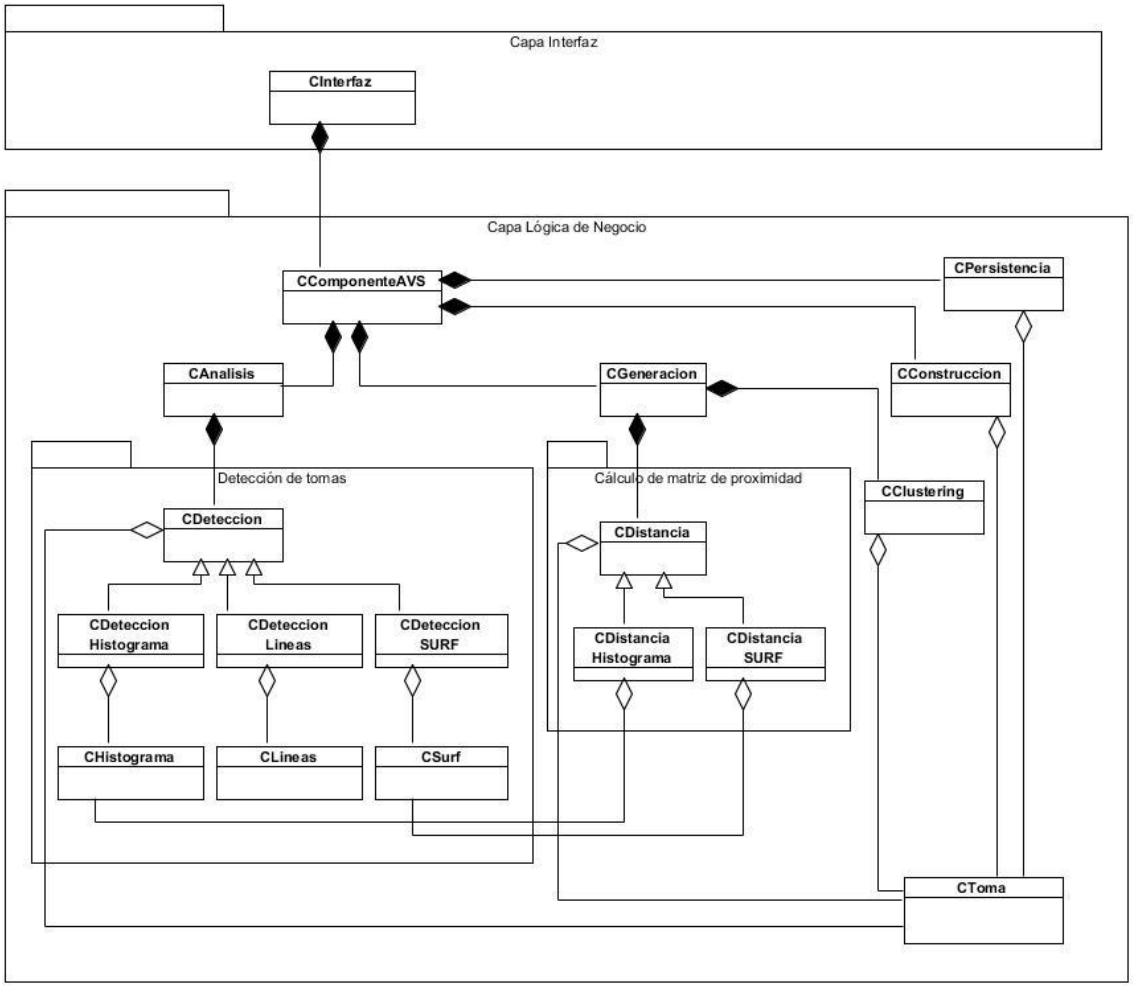

Figura 12. Clases de diseño.

3.8 Modelo de implementación

El modelo de implementación no es más que la representación de la composición física de la implementación del sistema, comprendido por los subsistemas y componentes del mismo. Con el objetivo de describir la relación que existe entre estos componentes y subsistemas. Este artefacto tiene una gran importancia porque permite a los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de sus componentes y sus relaciones. (54)

3.8.1 Diagrama de componentes

Un diagrama de componentes muestra las dependencias lógicas entre los componentes de una solución, sean estos de código fuente, componentes del código binario o componentes ejecutables. Su creación asegura un mejor entendimiento y estructuración de la implementación, y pueden ser representados mediante el uso de cinco estereotipos: datos, códigos fuente, clases, ejecutables y bibliotecas. El diagrama a continuación muestra las relaciones entre el ejecutable, las bibliotecas y los datos usados solamente, ya que el código fuente ha sido estructurado según las clases, y las relaciones entre estas ya fueron especificadas en el DCD. (54)

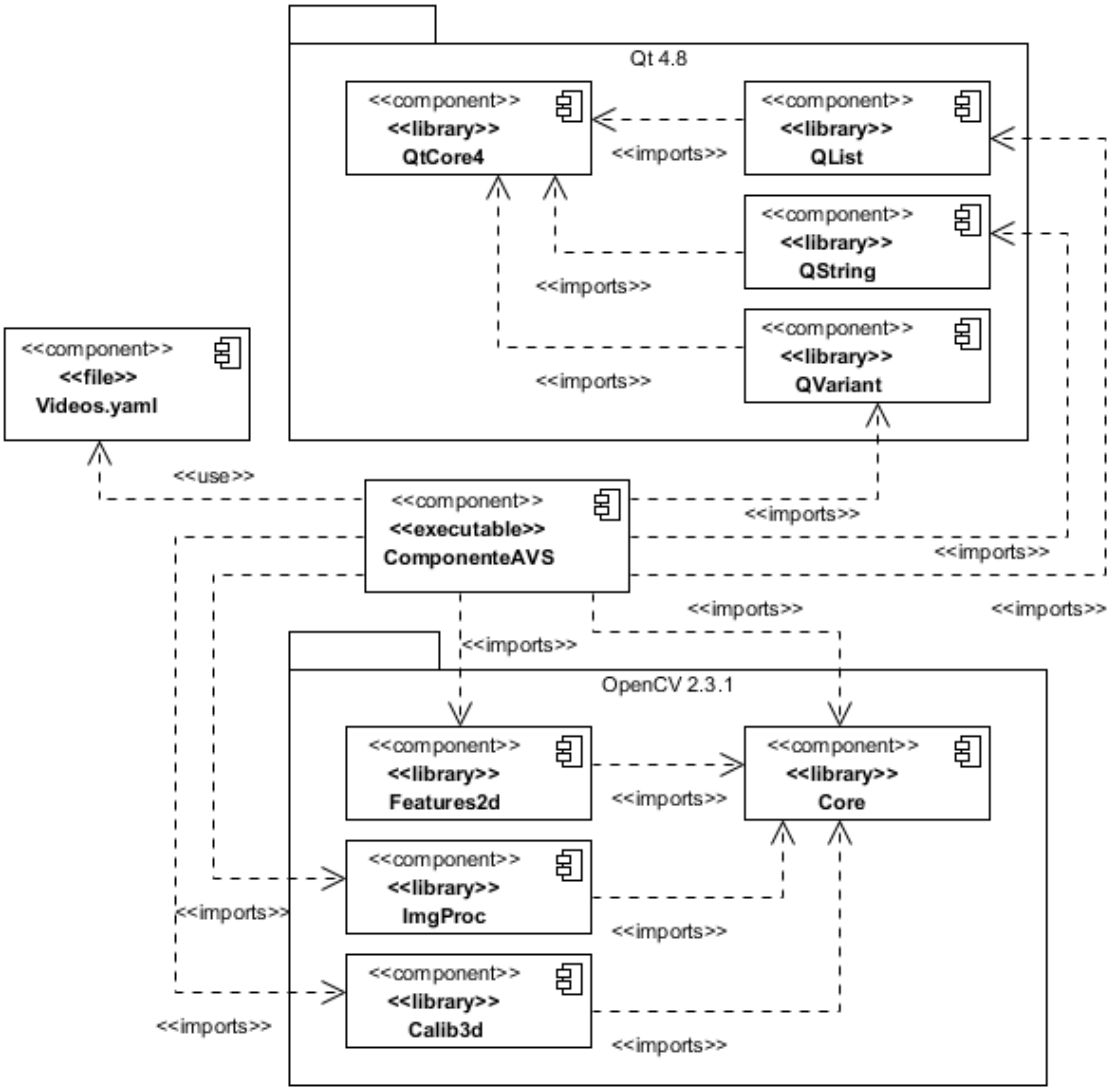

Figura 13. Diagrama de componentes.

3.8.2 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema o subsistema y el reparto de los mismos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Uno nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. (61)

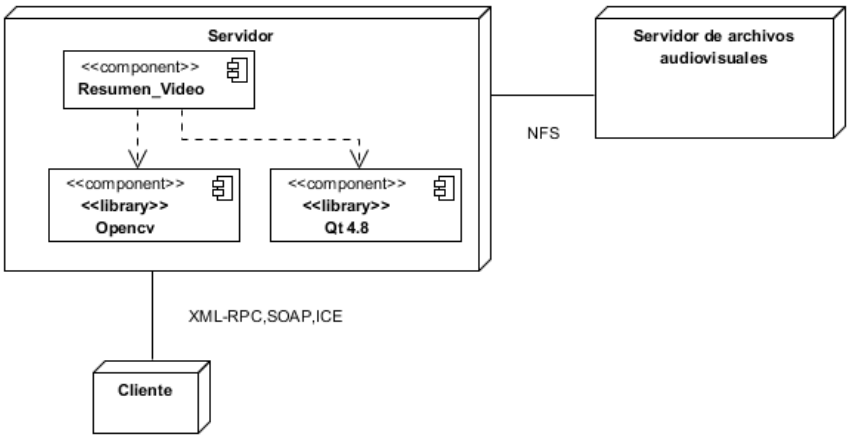

Figura 14. Modelo de despliegue

El nodo servidor debe ser una estación con las características especificadas en los requisitos no funcionales. El mismo debe conocer la ubicación de un servidor de archivos audiovisuales, para obtener el contenido de video a procesar, que especifique el nodo cliente. La comunicación con el nodo cliente es efectuada a través de los protocolos XML-RPC[8], SOAP[9] o ICE[10].

3.9 Conclusiones parciales

En este capítulo se presentaron los principios de análisis y diseño, comenzando por el modelo de dominio y la especificación de requisitos. También se presentó la propuesta de algoritmo, teniendo en cuenta las tendencias y los fundamentos teóricos vistos en el capítulo 1. El algoritmo propuesto consta de tres etapas principales (detección clasificación y generación). Durante la generación es necesario utilizar tres técnicas, basada en histogramas, en líneas y en Surf, debido a que cada una por sí sola no brinda los resultados óptimos y la mezcla de estas tres favorece una detección con mayor precisión. La selección se realiza utilizando una técnica de agrupamiento basado en grafos con el objetivo de eliminar la redundancias existentes y de esta forma aportar la mayor información posible. En la etapa de construcción se tuvo en cuenta las especificaciones requeridas con el objetivo de lograr los resúmenes de tipo *skim* mediante la inclusión de transiciones simples.

[8] Forma fácil y rápida de hacer llamadas a procedimientos. Convierte la llamada en un XML y lo envía usando HTTP, la respuesta es también
[9] Protocolo de comunicación que permite la comunicación mediante el intercambio de archivos XML con un formato definido.
[10] Plataforma orientada a objetos para la comunicación entre aplicaciones que provee de herramientas, APIs y Librerías para apoyar la construcción de aplicaciones de cliente-servidor orientada a objetos.

Después se determinó la utilización del patrón de arquitectura dos capas que asegura un correcto aislamiento y un mejor rendimiento en la solución final. Por otra parte se utilizaron los patrones de diseño (alta cohesión, bajo acoplamiento, controlador, experto, creador y estrategia) para no cometer errores ante la solución de problemas ya conocidos y solucionados anteriormente, estandarizar el diseño propuesto y obtener una arquitectura basada en buenas prácticas de diseño. Finalmente se definió la estructura y el entorno del resultado final mediante el desarrollo del modelo de implementación.

CAPÍTULO 4. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

La verificación y validación son actividades de aseguramiento de la calidad del software que buscan certificar que una solución es desarrollada de acuerdo con el proceso de desarrollo establecido y que satisface las necesidades del cliente. Una vez implementada la propuesta se hace necesaria la utilización de varias pruebas para encontrar posibles errores, conocer las limitaciones del software, entre otras. Lo que se persigue en el presente capítulo es lograr la validación del software desarrollado.

4.1 Pruebas del Sistema

Las pruebas del sistema se enfocan en requisitos funcionales tomados directamente de casos de uso y funciones de negocio. El objetivo de las mismas es verificar el ingreso procesamiento y recuperación apropiado de los datos. Este tipo de pruebas se basan en técnicas de caja negra (62), por lo que se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. (63)

Teniendo en cuenta que la solución desarrollada es un componente y no un sistema, con un solo caso de uso, se efectúan una prueba de formato, y una prueba al algoritmo de detección de tomas que se encuentra en el **RF 1**. La elección de estos dos entornos de prueba se debe a que estos son los procesos que podrían generar errores dentro de la solución

4.1.1 Prueba de soporte de formatos

Con el objetivo de comprobar el comportamiento de la solución implementada ante videos de características distintas, se conforma un caso de prueba que tiene como objetivo probar cuatro videos totalmente distintos durante el proceso completo y de esta forma identificar, si existe algún error en alguno de los requisitos. Entre las características que los distinguen se encuentran el contenedor,

codificador, bitrate[11], fotogramas por segundo (FPS) y el tamaño de los mismos. A continuación se muestra el resultado de las pruebas de soporte de formato para la ejecución del componente. (64)

Nombre del video	Stay
Contenedor	AVI
Codificador	MPEG4-Xvid
Bitrate(KB/s)	128
FPs	29.97
Video Size:	800 x 600
Valoración: El componente realizó el resumen satisfactoriamente.	

Tabla 3. Prueba de formato 1.

Nombre del video	10 Genres of metal in 3 minutes
Contenedor	MP4
Codificador	AVC(h264)
Bitrate(KB/s)	2400
FPs	30
Video Size:	320 x 240
Valoración: El componente presentó problemas en la etapa de detección SURF.	

Tabla 4. Prueba de formato 2.

Nombre del video	Girlfriend
Contenedor	WMV
Codificador	WMV2
Bitrate(KB/s)	2400
FPs	30

[11] Frecuencia de bits por segundo.

720 x 480	720 x 480
Valoración: El componente presentó problemas en la etapa de detección SURF.	

Tabla 5. Prueba de formato 3.

Nombre del video	StarCraft II.2 Campaign Cinematics 07
Contenedor	MPG
Codificador	MPEG1
Bitrate(KB/s)	1150
FPs	25
Video Size:	352 x 288
Valoración: El componente realizó el resumen satisfactoriamente.	

Tabla 6. Prueba de formato 4.

Durante las pruebas de formato se identificó un error al efectuar la detección SURF que está asociado a determinadas combinaciones de contenedores y codificadores. Después de una minuciosa investigación del problema se detectó que se debe a errores de la biblioteca de procesamiento gráfico. Por tanto, se hace necesario acotar el formato de los resúmenes entrados a los contenedores AVI y MPG, con codificadores MPEG1 y MPEG4.

4.1.2 Prueba al algoritmo de detección de tomas

La detección de tomas representa una sección crítica dentro del componente implementado debido a que como resultado de esta tarea se obtienen las unidades básicas para efectuar el procesamiento de resumen. Las pruebas desarrolladas a continuación buscan identificar las detecciones incorrectas durante este proceso. La forma más utilizada para medir el buen funcionamiento de los algoritmos de detección de cambios de tomas consiste en calcular su *recall*[12] *(R)* y su precisión[13] (P). (64)

[12] Indicador de los cortes entre tomas que no detecta el algoritmo.
[13] Indicador de los cortes entre tomas incorrectos o falsos que detecta el algoritmo.

$$R = \frac{correctos}{correctos + falsos\ negativos}$$

$$P = \frac{correctos}{correctos + falsos\ positivos}$$

Nombre del video	Número de Fotogramas	Cortes	Cortes Detectados	Fp	Fn	Recall	Precisión
CNN	30 715	54	51	4	3	94%	93%
Dias contados	27 643	13	12	1	0	100%	92%
Friends	90 341	21	21	2	0	100%	91%
La vita e bella	114 997	127	125	6	2	98%	96%

Tabla 7. Prueba de detección de tomas.

4.1.3 Prueba de elaboración de resúmenes

Con el objetivo de validar que el componente cumple con el objetivo general planteado, se diseña un caso de pruebas con distintos videos y se solicita resúmenes de distintas longitudes de los mismos. La duraci

Nombre del video	Duración en minutos	Resumen de 2 minutos	Resumen de 3 minutos	Resumen de 5 minutos
CNN	13	Efectuado correctamente	Efectuado correctamente	Efectuado correctamente
Don't say a word	21	Efectuado correctamente	Efectuado correctamente	Efectuado correctamente
Friends	17	Efectuado correctamente	Efectuado correctamente	Efectuado correctamente
La vita e bella	20	Efectuado correctamente	Efectuado correctamente	Efectuado correctamente

4.2 Conclusiones Parciales

Después de realizar las pruebas se puede comprobar que el componente realiza satisfactoriamente la detección de tomas con un grupo de formatos pero presenta problemas para decodificar otros que también son soportados por el OpenCV 2.3.1, por lo que se hace necesario acotar los formatos de los videos entrados. Por otra parte se comprobó la alta precisión obtenida durante la etapa de detección llegando a un porcentaje de un 93%, que demuestra que la selección de descriptores de imágenes fue la correcta y se ordenaron satisfactoriamente.

CONCLUSIONES

Una vez terminado el desarrollo de la solución es posible arribar a las siguientes conclusiones:

- La determinación de las técnicas de detección de tomas (utilizando descriptores de Histogramas, Líneas y Surf), demostró ser acertada debido a la alta precisión obtenida durante la fase de pruebas, lo cual aporta mayor calidad al proceso de elaboración de resúmenes en su totalidad.
- El diseño de las etapas de análisis y generación aportan mayor flexibilidad a la solución, ofreciendo la posibilidad de implementar la detección de toma o la construcción de la matriz de proximidad, utilizando más, menos, o distintos descriptores visuales.
- Las herramientas y tecnologías de desarrollo seleccionadas garantizaron un correcto desarrollo de la construcción del componente.
- El componente resuelve eficientemente la situación problemática planteada, ya que se comprobó la generación múltiples resúmenes de videos a partir de un único análisis durante la fase de pruebas, por lo que se concretó la idea a defender planteada al inicio del trabajo.

RECOMENDACIONES

Como resultado del desarrollo de la investigación y conclusiones de esta investigación, se expresan a continuación las siguientes recomendaciones.

- Acotar el dominio de los videos a resumir a distintos géneros de películas o documentales, programas deportivos o de noticias, entre otros, dando la posibilidad de especializar los descriptores, los criterios de evaluación y el algoritmo de agrupamiento.
- Implementar el trabajo con hilos al componente para asegurar un mejor rendimiento.
- Implementar la generación de resúmenes de imágenes estáticas en el componente.
- Implementar la comunicación a través de los protocolos de comunicación XML-RPC, ICE y SOAP.

BIBLIOGRAFÍA Y REFERENCIAS BIBLIOGRÁFICAS

1. ECURED. [En línea] [Citado el: 8 de 1 de 2013.] http://www.ecured.cu/index.php/Televisi%C3%B3n.

2. **Serra, Carmen Caffarel.** *Algunas reflexiones en torno a la Televisión Digital Terrestre.* s.l. : Revista ICONO Revista científica de Comunicación y Tecnologías emergentes, 2007.

3. **Rodríguez, I González.** *Televisión Digital Terrestre, el apagón analógico.* 2009.

4. **Badillo, Ramón Zallo y Ángel.** Mercado y políticas de cultura y comunicación en el mercado global. s.l. : Unión Latina de Economía Política de la Información, la Comunicación y la Cultura, 2010.

5. **Szyperski, Clemens.** *Component Software: Beyond Object Oriented Programming.* 2002.

6. *Diccionario Manual de la Lengua Española Vox.* s.l. : Larousse Editorial, S.L., 2007.

7. Ecured. [En línea] [Citado el: 16 de 12 de 2012.] http://www.ecured.cu/index.php/Resumen.

8. **Luis Herranz, José M. Martínez.** *A Framework for Scalable Summarization of Video.* 2010.

9. **Arribas, Luis Herranz.** A SCALABLE APPROACH TO VIDEO SUMMARIZATION AND ADAPTATION. 2010.

10. **Víctor Valdés, José M. Martínez.** On-line Video Skimming Based on Histogram Similarity. 2007.

11. *KDictionaries Ltd.* 2009.

12. **Thakre, Kalpana S.** *VIDEO MATCH ANALYSIS: A Comprehensive Content based Video Retrieval System Shimna Balakrishnan.* 2010.

13. **Harish, S. Manjunath D. S. Guru M. G. Suraj B. S.** A Non Parametric Shot Boundary Detection: An Eigen Gap based Approach. 2010.

14. *Diccionario de la lengua española .* s.l. : Espasa-Calpe, 2005.

15. **López, Antonio Redondo.** *Extracción de Características de Imagen para Navegación de Robots Móviles.* 2011.

16. Word Reference. [En línea] 10 de 5 de 2013. http://www.wordreference.com/definicion/Fotograma.

17. *Word Reference.* [En línea] [Citado el: 10 de 5 de 2013.] http://www.wordreference.com/definicion/distancia.

18. **García, Óscar Boullosa.** Estudio comparativo de descriptores visuales para la detección de escenas. 2011.

19. **Martínez, Víctor Valdés y José M.** *On Video Abstraction Systems' Architectures and Modelling.* Madrird : s.n., 2008.

20. *Descriptores de video, sus aplicaciones en materiales audiovisuales.* **Yoandri Quintana Rondón1, Yanio Hernández Heredia y Abel Díaz Berenguer.** Ciudad de la Habana : 1Universidad de las Ciencias Informáticas, 2010.

21. **Zhang, H.J., Kankanhalli, A., and Smoliar, S.W.** "Automatic Partitioning of Full-motion Video", Multimedia Systems. 1993.

22. **Kasturi, R. and Jain R.** "Dynamic Vision", in Computer Vision: Principles. s.l. : IEEE Computer Society Press, 1991.

23. **Shapiro, Linda G. and Stockman, George C.** "Computer Vision" Prentice Hall. 2003.

24. **Ueda, H., Miyatake, T., and Yoshizawa, S.** "IMPACT: An Interactive Natural-motion-picture Dedicated Multimedia Authoring System". 1991.

25. **Nagasaka, A. and Tanaka, Y.** "Automatic Video Indexing and Full-Video Search for Object Appearances", in Visual Database Systems II. 1992.

26. **Swanberg, D., Shu C.F., and Jain, R.** "Knowledge Guided Parsing and Retrieval in Video Databases", in Storage and Retrieval for Image and Video Databases. 1993.

27. **Zabih, R., Miller, J., and Mai, K.** "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks", Proc. ACM Multimedia 95. 1993.

28. **Lienhart, Rainer.** *Comparison of Automatic Shot Boundary Detection Algorithms.* 1997.

29. **Ondˇrej Chum, James Philbin, Michael Isard, Andrew Zisserman.** *Scalable Near Identical Image and Shot Detection.* 2006.

30. **Víctor Valdés, José M. Martínez.** Binary Tree Based On-line Video Summarization. 2008.

31. **Wunsch, Rui Xu y Donald C.** *Clustering.* s.l. : IEEE, 2009.

32. **Chong-Wah Ngo, Yu-Fei Ma, Hong-Jiang Zhang.** Automatic Video Summarization by Graph Modeling. 2003.

33. *The American Heritage, Dictionary of the English Language, Fourth Edition.* s.l. : Houghton Mifflin Company, 2009.

34. **Arribas, Luis Herranz.** A scalable approach to video summarization and adaptation. 2010.

35. **Méndez, Alejandra Virrueta.** *METODOLOGÍAS DE DESARROLLO DE SOFTWARE.* Apatzingan Michoacán : s.n., 2010. ITSA.

36. **Kruchten, P. s.l.** *The Rational Unified Process: An Introduction.* s.l. : Addison Wesley, 2000.

37. _ECURED. [En línea] [Citado el: 10 de 5 de 2013.] http://www.ecured.cu/index.php/UML.

38. **Giraldo, Luis y Zapata, Yuliana. s.l.** *Herramientas de desarrollo de ingeniería de software para Linux.* s.l. : Monitoria de Ingesoft, 2005.

39. ECURED. [En línea] [Citado el: 2013 de 5 de 24.] http://www.ecured.cu/index.php/Visual_Paradigm.

40. *The VXL Logo Homepage.* [En línea] [Citado el: 24 de 1 de 2013.] http://vxl.sourceforge.net/.

41. *LTI Lib.* [En línea] [Citado el: 24 de 1 de 2013.] http://ltilib.sourceforge.net/doc/homepage/index.shtml.

42. *Open Source Computer Vision.* [En línea] [Citado el: 24 de 1 de 2013.] http://opencv.org/.

43. Lenguaje de progrmación. *Definicion de.* [En línea] [Citado el: 17 de 1 de 2013.] http://definicion.de/lenguaje-de-programacion/.

44. *Python Programming Language – Official Website.* [En línea] [Citado el: 15 de 2 de 2013.] www.python.org/ .

45. **Duque, Raúl González.** *Python para todos.* s.l. : Creative Commons, 2006.

46. **Brian W. Kernighan, Dennis Ritchie.** *The C Programming Language.* 1978.

47. **Stroustrup, Bjarne.** *The c++ programming language .* s.l. : IEEE Software - SOFTWARE, 1986.

48. **Quesada, Alma Evangelina.** [En línea] [Citado el: 2013 de 5 de 15.] http://www.freewebs.com/almaquesada/portafolio.htm.

49. Zona Qt. [En línea] [Citado el: 2013 de 5 de 17.] http://www.zonaqt.com/tutoriales/tutorial-b%C3%A1sico-de-qt-4.

50. ECURED. [En línea] 2010. [Citado el: 17 de 5 de 2013.] http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

51. Qt Creator. *Ecured.* [En línea] [Citado el: 20 de 2 de 2013.] http://www.ecured.cu/index.php/Qt_Creator.

52. **Larman, Craig.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Prentice Hall, 2008.

53. **Beatriz Ayala, Claudia Marcela Ramírez, Lina María Ocampo.** *LA INGENIERÍA DE REQUERIMIENTOS APLICADA AL DESARROLLO DE SISTEMA DE INFORMACIÓN.* 2006.

54. **Kim Hamilton, Russel Miles.** *Learning UML 2.0.* s.l. : O´Reilly, 2006. ISBN.

55. **Cavenago, Adriana Sandra Almeira y Vanina Perez.** *Arquitectura de Software: Estilos y Patrones.* Trelew : s.n., 2007.

56. **Cesar de la Torre Llorente, Unai Zorrilla Castro, Miguel Angel Ramos Barroso, Javier Calvarro Nelson.** *Guía_de_Arquitectura_N-Capas_orientada_al_Dominio_con_.NET_4.0.* s.l. : Krasis Press, 2010.

57. **S. Somasegar, Scott Guthrie, David Hill.** *Microsoft Application Archiitecture Guide, patterns & practices.* s.l. : MICROSOFT, 2009.

58. **KevinZhang.** *Design Patterns, Elements of Reusable Object-Oriented Software.* 1994.

59. **Astudillo, Marcello Visconti y Hernán.** *Fundamentos de Ingeniería de Software.* s.l. : Universidad Técnica Federico Santa María, 2008.

60. **Rojas, M.C. Juan Carlos Olivares.** *Patrones de Diseño.* s.l. : ITM, 2010.

61. **Marca Huallpara Hugo Michael, Quisbert Limachi Nancy Susana.** Diagrama de Despliegue, ANALISIS Y DISEÑO DE SISTEMAS . Universidad Salesiana Bolivia, 20010.

62. **García, Ing. Marisol Lara.** *3 UNIDAD. PRUEBAS DE SOFTWARE.* Lima : s.n., 2013.

63. *Validation, Verification and Testing of Computer Software.* **W. Richards Adrion, Martha A. Branstad, John C. Cherniavsky.** s.l. : CS780, 2006.

64. **Rodríguez, Luis Angel Pupo.** *Implementación de un componente para la separación automática de segmentos de videos integrado al Sistema de Captura y Catalogación de Medias.* s.l. : Universidad de las Ciencias Informaticas, 2011.

65. Revisiones de código y estándares de codificación. *msdn.* [En línea] msdn. [Citado el: 24 de 2 de 2013.] http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

66. *MSDN.* [En línea] Microsoft. [Citado el: 15 de 4 de 2013.] http://msdn.microsoft.com/es-es/library/dd409390.aspx.

67. **Pressman, Roger S.** *Software Engineering: A Practitioner's Approach (Hardback).* s.l. : McGraw Hill Higher Education, 2000. 9780073655789.

68. **Baber, J.** *Shot boundary detection from videos using entropy and local descriptor .* 2011.

69. **Yuchou Chang, D. J. Lee, Yi Hong, and James Archibald.** *Unsupervised Video Shot Detection Using.* 200.

70. *Diccionario Enciclopédico Vox 1.* s.l. : Larousse Editorial, S.L., 2009 .

71. **Herbert Bay, Tinne Tuytelaars, y Luc Van Gool.** *SURF: Speeded Up Robust Features.* 2004.

72. ECURED. [En línea] [Citado el: 8 de 12 de 2012.] http://www.ecured.cu/index.php/Algoritmo.

73. *Collins English Dictionary – Complete and Unabridged.* s.l. : HarperCollins Publishers, 2003.

74. Word Reference. [En línea] [Citado el: 10 de 5 de 2013.] http://www.wordreference.com/es/translation.asp?tranword=cluster.

75. **Szyperski, Clemens.** Component Software: Beyond Object Oriented Programming. 2002.

76. **Pérez, M.** *Arquitectura para Ambientes CASE Integrados.* 1999.

77. **Laganière, Robert.** *OpenCV 2 Computer Vision Application Programming Cookbook.* BIRMINGHAM - MUMBAI : PACKT, 2010.

78. **Daniel Lélis Baggio, Shervin Emami, David Millán Escrivá, Khvedchenia Ievgen, Naureen Mahmood, Jason Saragih y Roy Shilkrot.** *Mastering OpenCV with Practical Computer Vision Projects.* BIRMINGHAM - MUMBAI : PACKT, 2012.

79. XML-RPC. [En línea] [Citado el: 20 de 5 de 2013.] http://xmlrpc-c.sourceforge.net/.

80. **ZeroC, Inc.** *Ice Manual.* 2013.

81. SOAP Specifications - W3C. [En línea] [Citado el: 20 de 5 de 2013.] www.w3.org/TR/soap/.

ANEXOS

Anexo 1 Diagrama de Clases de Diseño

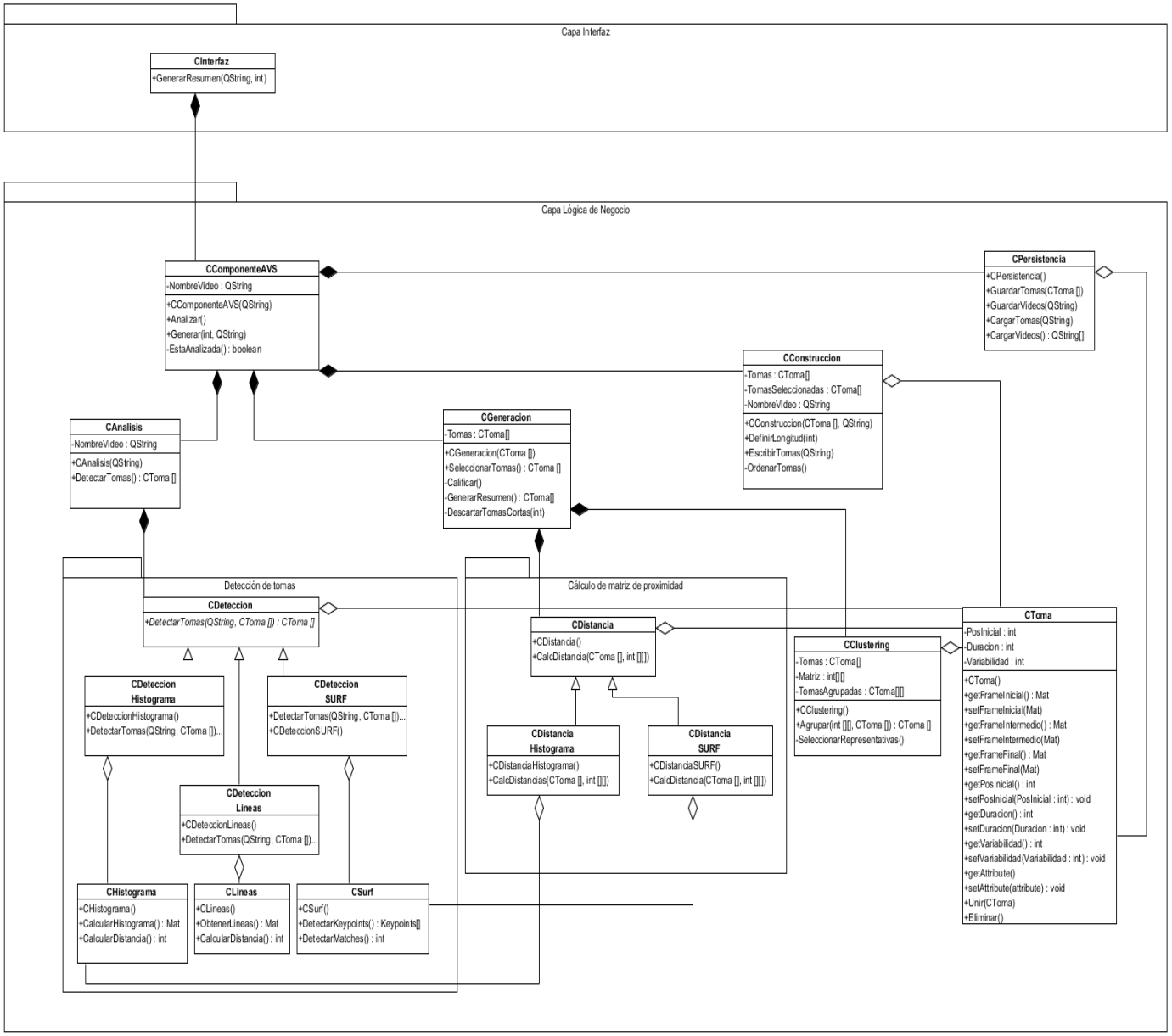

Figura 15. Diagrama de Clases de Diseño completo.

Anexo 2 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. (65)

Para efectuar la implementación del componente se utiliza el estándar de codificación establecido por el proyecto SCTV perteneciente al centro GEYSED de la UCI. A continuación se puntualizan las características adoptadas de este estilo.

Identificadores:

- Deberán tener un nombre significativo para que por su simple lectura, pueda conocerse su función, sin tener que consultar manuales o hacer demasiados comentarios.
- Para nombres que se usen con frecuencia o para términos largos, se recomienda usar abreviaturas estándar para que éstos tengan una longitud razonable. Si usa abreviaturas deben manejar la misma lógica en todo el programa.
- Cada identificador de variable o procedimiento deberá ser precedido por la abreviación del tipo de dato de que es la variable.
- Los atributos deben comenzar con letra minúsculas y los métodos deben comenzar con letra mayúsculas.
- Los parámetros deben comenzar con la letra p y con nombre lo más similar posible al atributo que se refiere.

Organización Visual del Programa

Generales

- No manejar en los programas más de una instrucción por línea.
- Declarar las variables en líneas separadas
- Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.

Sangrías

- Las sangrías tendrán una longitud de tres espacios.

- Para las llaves que definen el cuerpo de una función, sangre un nivel.

  Ejemplo: void Funcion ( )

  {

  //Instrucciones de la función

  }

Líneas y espacios en blanco

- Insertar una línea en blanco antes y después de una declaración de datos que aparezca entre instrucciones ejecutables.

  a = b + c;

  // línea en blanco

  int f; //declaración entre instrucciones

  // línea en blanco

  f = a;

- Las declaraciones de datos dentro de una función, deberán ir al inicio y separadas de las instrucciones ejecutables de la función por medio de una línea en blanco.
- Deben incluirse espacios en ambos lados de los operadores binarios.

  Ejemplo:

  y = 50 + 15 - x;

- Es posible distribuir una instrucción grande sobre varias líneas. Si lo hace, seleccione puntos de ruptura que tengan sentido, como después de una coma en el caso de una lista, o después de un operador en el caso de una expresión larga.
- Sangre todas las líneas subsecuentes.

Ejemplo:

```
cout << “Ejemplo de ruptura de una instrucción en más de una”

<< ”línea de comandos”;
```

- Los operadores unarios (++, --, etc.) deben ponerse junto a sus operandos, sin espacios intermedios.
- Antes y después de cada estructura de control deberá poner una línea en blanco.

GLOSARIO

AVS: Automatic Video Summarization, elaboración de resúmenes automáticos.

Clustering: Proceso de agrupar objetos similares, por su uso frecuente en la documentación consultada tanto en inglés como en español.

Dendograma: Representación gráfica en forma de árbol que ilustra el proceso de agrupación en un análisis de clúster.

Grafos: un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas que pueden ser orientados o no.

Falso positivo: Referido en la investigación como una detección de un corte de tomas erróneo.

h.264: Codificador de video.

MPEG-4: Contenedor de video.

Resúmenes embebidos: Resumen largo del cual pueden ser extraído resúmenes de distintas longitudes.

Rendimiento: Velocidad con la que se ejecuta un algoritmo en proporción a las características de hardware de la máquina en la que se ejecuta.

Umbral: Limite numérico que se establece para clasificar los resultados de determinadas operaciones.

Umbralizar: Proceso de utilizar un umbral para filtrar resultados numéricos.

Video bajo demanda: Contenido audiovisual que se visualiza desde la posición deseada al tener acceso al mismo en la red.

Video en vivo: Contenido de video cuya ejecución es constante y por tanto solo se puede visualizar la posición que tenga en ese momento al acceder al mismo a través de la red.