Universidad de las Ciencias Informáticas

FACULTAD 6

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Video sensor para detección y reconocimiento de rostros

Autor: Breissy Peraza González

Tutores: Ing. Reinier Pupo Ruiz

Ing. Nilo T. Díaz Ales

La Habana, Junio de 2013

"Año del 55 Aniversario de la Revolución"

Declaración de autoría

Declaro ser autora 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 _____.

_____	_____
Firma del Autor	Firma del Tutor
Breissy Peraza González	Reinier Pupo Ruiz

Datos de contacto

Ing. Reinier Pupo Ruiz

Graduado *de* la Universidad de las Ciencias Informáticas (UCI) en el año 2009. Es Instructor Recién Graduado y ha impartido clases de Programación 2 en la Facultad 7 de la UCI. Pertenece al Departamento Señales Digitales del Centro de Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. Pertenece al proyecto Video Vigilancia (SURIA), donde es el responsable del módulo Video Sensores. Correo electrónico: rpupo@uci.cu

Ing. Nilo T. Díaz Ales

Graduado de la Universidad de las Ciencias Informáticas (UCI) en el año 2012. Recién Graduado, pertenece al Departamento Señales Digitales del Centro de Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. Pertenece al proyecto Video Vigilancia (SURIA), donde es el responsable del módulo Captura y Grabación. Correo electrónico: ntdiaz@uci.cu.

Agradecimientos

A mis padres, que han sido mi inspiración, estando a mi lado en cada momento, por ustedes he llegado aquí, porque me han dado todo lo que necesito, sobre todo su amor. Son lo más grande que tengo en la vida.

A mi hermana por siempre querer lo mejor para mí, por insistir aun cuando digo que no, por estar a mi lado no importa que, te amo.

A mi sobrino por ser el niño más bello del mundo, porque cada vez que me llama tía o me besa se me vuelca el corazón.

Un agradecimiento ESPECIAL a mi Pupo, que además de tutor por estos 3 años ha sido amigo, guía, compañero. Por soportarme, porque siempre ha estado ahí cuando lo necesito. Un gracias inmenso, no acabaría nunca de agradecerte.

A todos aquellos que vienen conmigo desde el 1er día de esta universidad y a los que se han sumado a lo largo de estos años. Adrián, Javier, Jorge, Yorbenys, Jose, Andris, Vivi, Maralys, Deymis, Doina, Orisel. Lieny. Disculpen aquellos que no he podido mencionar.

A las 4 fantásticas, por estos 5 años maravillosos, porque hemos compartido además de peleas, momentos difíciles y alegres. Hemos permanecido juntas, apoyándonos, sobrellevando los defectos de las otras. Gracias por aguantarme las peleas, son el mejor recuerdo que me llevo de la universidad.

Dedicatoria

A mi mami por su preocupación, por dedicarme su vida y ser su niña pequeña.

A mi papi por todo su arrojo, por transmitirme esa energía de seguir adelante.

A ustedes porque soy el fruto de todo su esfuerzo, y espero que se sientan orgullosos.

Los amo.

Resumen

Con el avance de las tecnologías de la información, el campo de la video vigilancia ha cobrado fuerza, numerosos han sido los sectores que han incorporado su uso. Así mismo se han incrementado y desarrollado las empresas que se dedican a la creación tanto del software, como las cámaras que utiliza. La Universidad de Ciencias Informáticas es uno de los centros vinculados directamente a la producción de software, dentro de la misma está el proyecto Video Vigilancia SURIA, cuya tarea principal es el procesamiento de imágenes y señales de video digital con el fin de optimizar y añadir nuevas prestaciones a los sistemas de vigilancia.

La presente investigación tiene como objetivo general desarrollar un video sensor para la detección y reconocimiento de rostros, ampliando las prestaciones al sistema de vigilancia SURIA. El mismo fue desarrollado utilizando la metodología RUP (Rational Unified Process) para guiar el proceso de desarrollo del software recurriendo a herramientas no propietarias. Se utiliza para la detección Viola and Jones, con la implementación que ofrece la biblioteca OpenCV, y 3 algoritmos para reconocimiento, utilizando también las facilidades de implementación de dicha biblioteca. Estos algoritmos son Eigenfaces, Fisherfaces y LBPH. Al sistema, se le realizaron pruebas de rendimiento, arrojando resultados positivos, logrando un porciento elevado de aciertos.

Palabras Clave:

Cámaras IP, detección, reconocimiento, video vigilancia

Abstract

With the advance of information's technology, the field of video surveillance has gained in strength; many sectors have been incorporated their use. Also, the companies engaged in the creation of both, the software and the cameras have been increased. The Universidad de las Ciencias Informáticas is one of the centers directly linked to the production of software, in it exists the project SURIA Video Surveillance System, it main task is the image and digital video signals processing in order to optimize and add new features to surveillance systems.

This research aims to develop a video sensor for the detection and recognition of faces, expanding benefits of SURIA surveillance system. It was developed using the RUP methodology (Rational Unified Process) to guide the software development process using non-proprietary tools. To detect the faces is used Viola and Jones, a implementation that gives the OpenCV library together other 3 algorithms for recognition, using also the implement facilities of that library. These algorithms are Eigenfaces, Fisherfaces and LBPH. The system was tested for performance, showing positive results, achieving a high percentage of hits.

Key words:

Detection, Eigenfaces, Fisherfaces, ip cameras, LBPH, recognition, video surveillance

Índice

Índice de figuras

Índice de tablas

Introducción

La primera generación de sistemas de vigilancia basada en video, utilizó señal y transmisión analógica. El análisis de la secuencia de video era realizado por una persona que atendía varios monitores a la vez. El material capturado por dichas cámaras se guardaba en cintas magnéticas o se sobrescribía periódicamente, lo cual limitaba el período de grabación que podía archivarse.

La segunda generación permitió centrar la atención del operador humano en situaciones de interés, debido al surgimiento de los primeros algoritmos de procesado de video y los métodos de compresión digital. Se basó en métodos de procesamiento y comunicación híbridos analógico-digitales, o completamente digitales.

Los sistemas actuales procesan el flujo de imágenes en tiempo real utilizando cámaras IP[1]. Este sistema de vigilancia permite que cada cámara sea independiente de otra en cuanto al tratamiento de la información captada, así el sistema es más confiable en estabilidad y la información que brinda. También cuentan con procesamiento inteligente autónomo, incrementando la seguridad, ejemplo de esto son los videos sensores.

"El video sensor no es más que una herramienta de análisis de video digital que ofrece información significativa proveniente de una secuencia de video" (Colomer, 2003). Existen diferentes videos sensores, entre los que se pueden mencionar, video sensor para detección de movimiento, video sensor para conteo de personas, video sensor para extracción de matrículas en vehículos. En esta rama, la actividad relacionada con la capacidad para establecer la identidad de los individuos ha cobrado gran importancia. Esta se lleva a cabo a través de un análisis de características faciales del sujeto en la imagen o fotograma en cuestión, comparándolas con otras almacenadas previamente en una base de datos. Este proceso puede actuar de dos modos; comparando una imagen de la cara con otra que se quiera saber la identidad, donde se confirma o se rechaza (verificación o autentificación de caras). Otra alternativa es comparar una imagen de un rostro desconocido con otras conocidas para permitir su identidad (Identificación o reconocimiento de caras).

[1] *"Las cámaras IP, son videocámaras de vigilancia que tienen la particularidad de enviar las señales de video y en muchos casos audio"* (Beltrán Mesias, febrero 2002)

Cuba, a pesar de ser un país tercermundista ha prosperado notoriamente en el desarrollo de la informática y las comunicaciones. Con el objetivo de seguir evolucionando en esta rama, en el año 2002 se funda la Universidad de las Ciencias Informáticas (UCI). En la UCI se encuentra el centro de Geoinformática y Señales Digitales (GEYSED), dentro del que se desarrolla el proyecto Video Vigilancia SURIA, el mismo cuenta con los módulos Recuperador, Grabador, Visor, el módulo central llamado Gestor y el módulo Analytics, que es el encargado del desarrollo de los video sensores.

SURIA necesita de un mecanismo que permita la detección y reconocimiento de rostros a partir de flujos de videos obtenidos desde las cámaras IP. Este permitiría una agilización de procesos y mayor control en los entornos que sean monitoreados. Se ha detectado que en el estudio de los procesos involucrados en la video vigilancia, los trabajadores de seguridad son incapaces de llevar el control exacto de lo que ocurre en varias cámaras simultáneamente, y a su vez examinar quién puede salir o entrar en cierto local. Es decir, se dificulta considerablemente reconocer e identificar a las personas que puedan tener acceso a cierto recurso y llevar el registro de estas.

A partir de la situación descrita anteriormente, surge el siguiente **problema a resolver**: ¿Cómo lograr un reconocimiento de rostros de personal registrado en el sistema utilizando flujos de videos obtenidos desde cámaras IP?

Se plantea como **objeto de estudio**: Procesos de detección y reconocimiento de rostros, y se enmarca como **campo de acción**: La detección y reconocimiento de rostros en flujos de videos obtenidos de cámaras IP en el sistema de Video Vigilancia SURIA.

Para solucionar el problema planteado se traza como **objetivo general**: Desarrollar un video sensor para la detección y reconocimiento de rostros que procese los flujos de video obtenidos de cámaras IP en el sistema de Video Vigilancia SURIA. Planteando como **idea a defender**: “La construcción y puesta en funcionamiento de un video sensor en el sistema SURIA, que proveerá al mismo de la capacidad de procesamiento inteligente de los flujos de videos, para detectar y reconocer rostros”.

Para el cumplimiento del objetivo trazado se definieron las **tareas de la investigación** que se proponen a continuación:

- Valorar los sistemas líderes en el campo de la video vigilancia.

- Identificar algoritmos y bibliotecas que permitan el procesamiento de los videos e imágenes digitales que puedan ser utilizados en el desarrollo del video sensor.
- Realizar la documentación asociada al proceso de desarrollo.
- Desarrollar video sensor para la detección y reconocimiento de rostros.
- Validar el resultado esperado a través de pruebas realizadas al video sensor.

Como **posible resultado** de esta investigación se obtendría un componente video sensor que se le integrará al Sistema de Video Vigilancia SURIA, aportándole al mismo mayor capacidad e inteligencia.

Para el desarrollo de las tareas de investigación es necesario aplicar métodos de investigación para entender, verificar, corregir y aplicar los conocimientos adquiridos, logrando una mejor organización y estructuración del trabajo investigativo.

Entre los métodos de investigación se encuentran los **Métodos Teóricos**, los cuales permiten estudiar las características del objeto de investigación que no son observables directamente (Conferencia_Tema_3_El_diseño_metodológico_de_la_investigación_científica, 2010). Entre los Métodos Teóricos a utilizar se encuentra el **Histórico-Lógico**, mediante el cual se realiza un estudio acerca de los antecedentes y tendencias actuales de los videos sensores, así como los conceptos y metodologías de desarrollo existentes para llevar a cabo su construcción.

A su vez se utiliza el **Analítico-Sintético,** el cual facilita mediante el análisis y la síntesis, la búsqueda de lo fundamental en la bibliografía consultada para poder efectuar una amplia investigación sobre los elementos que se relacionan con el objeto de estudio.

Por otra parte están los **Métodos Empíricos** que representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional (Conferencia_Tema_3_El_diseño_metodológico_de_la_investigación_científica, 2010), entre estos está la **observación**, en la misma se llevó a cabo un registro visual del trabajo en el proyecto de Video Vigilancia SURIA.

El contenido del presente trabajo de diploma está estructurado de manera que en su primera parte se definen los elementos teóricos que sustentan la investigación, analizando soluciones similares existentes en el mundo y Cuba. También se enuncian los conceptos que posibilitan un mayor entendimiento de lo

planteado en la situación problemática. Además se seleccionan las tecnologías y herramientas a utilizar para el desarrollo del sistema.

En su segunda parte se definen las clases de análisis y diseño de los casos de usos; así como los diagramas de secuencias, cumpliendo con las descripciones de los casos de uso. Se presenta la arquitectura del sistema, describiendo además las clases entidades y las controladoras del flujo de trabajo análisis y diseño.

En su tercera parte se explican los algoritmos utilizados para la implementación del video sensor y posteriormente, en su última parte los diagramas asociados a la etapa de implementación así como las pruebas realizadas y los resultados arrojados por estas.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el presente capítulo se muestra el estudio del arte realizado sobre el tema tratado. Se analizan las técnicas que utilizan sistemas existentes hoy día, las más usadas y eficientes, teniendo en cuenta las prestaciones de la computadora en la que se puede desplegar el sistema, o sea, métodos eficaces capaces de ser ejecutados en todo tipo de entornos, consumiendo el mínimo de recursos. También se tienen en cuenta las tecnologías y metodologías más utilizadas. Se definen conceptos, ventajas y desventajas que permitieron seleccionar las herramientas utilizadas.

1.2 Definiciones Generales

A continuación se muestran una serie de conceptos asociados al dominio del problema para un mayor entendimiento del mismo.

Video: Según (Vera) "*El vídeo, hace referencia a la captación, procesamiento, transmisión y reconstrucción por medios electrónicos de una secuencia de imágenes y sonidos que representan escenas en movimiento.*" Esta secuencia de imágenes puede o no estar acompañadas de sonido, y puede ser almacenada en diferentes formatos y resoluciones.

Imagen Digital: Según (Woods, 1992) *"una imagen puede ser definida como una función bidimensional f (m, n), donde m y n son coordenadas espaciales, y la amplitud de f en cualquier par de coordenadas (m, n) se llama intensidad o nivel de gris de la imagen en ese punto. Cuando m, n y los valores de amplitud de f son todos finitos y valores discretos, se le llama a la imagen, imagen digital."* Es una matriz bidimensional cuyo objetivo es recrear elementos imaginarios como el arte, o reales como el entorno mismo que rodea al hombre.

Píxel: Según (Morán, 2001-2013) ***"****Es el elemento más pequeño que forma la imagen. Para hacerse una idea visual de su naturaleza, la comparación con los mosaicos formados por pequeñas piedras de colores que conforman una imagen es siempre un ejemplo muy recurrente."* O sea, es el elemento básico que compone una imagen digital.

Detección de rostros: Según (Emami, 2010) es *"cuando una imagen es analizada para encontrar alguna cara, luego la imagen procesada obtiene el rostro para hacer más fácil el reconocimiento."* Existen diversas vías para la detección, pero todas con el mismo fin, revelar el rostro del individuo presente en la imagen, el cual puede ser enmarcado o extraído, dependiendo del algoritmo utilizado.

Reconocimiento de rostros: Según (Emami, 2010) es *"cuando la cara detectada y procesada previamente es comparada con una base de datos que contiene rostros, para decidir quién es la persona"*. También para verificar si el rostro detectado está contenido en la base de datos y dar respuesta positiva o negativa.

1.3 Procesos de detección y reconocimiento de rostros

A continuación se describen los procesos involucrados en el objeto de estudio, exponiendo lo fundamental de cada paso.

1.3.1 Descripción general

El procesamiento de imágenes digitales constituye el punto de partida para el análisis de flujos de video, este se define como un conjunto de técnicas y procesos para descubrir o hacer resaltar información contenida en una imagen usando como herramienta principal una computadora (Torres, 1996). Al examinar una imagen, esta puede ser modificada y a su vez se le pueden extraer características o trabajar simplemente con alguna región de dicha imagen.

La detección y reconocimiento de rostros se realiza a través de tres procesos fundamentales. Inicialmente se detecta un rostro, ya en la segunda fase se extraen las características del rostro detectado, para posteriormente en una tercera etapa confrontar esas características con las almacenadas previamente en el sistema. Dependiendo de esa comparación el sistema emite una respuesta del reconocimiento del rostro. (M. Delbracio C. Aguerrebere, 2006)

La detección es el paso básico y también el más sencillo en este procedimiento, en comparación con los restantes. Primeramente hay que detectar que en la imagen existe un rostro, teniendo la seguridad que hay una persona presente en dicho fotograma, se sigue la secuencia de pasos que la actividad de detección conlleva. Sin embargo, la eficacia de estos algoritmos puede verse afectada dependiendo de las condiciones del entorno en el cual se toma la secuencia de imágenes (Vázquez, enero 2012). Si estas son capturadas en un entorno controlado, o sea, condiciones aceptables donde exista un fondo conocido,

iluminación controlada, alta resolución, pose frontal, expresión neutra, sin oclusiones, sería más factible para el sistema ejecutar una correcta detección, y a su vez garantizar un mayor porcentaje de aceptación en la etapa de reconocimiento (M. Delbracio C. Aguerrebere, 2006). Ver figura 1.

Figura 1: Entorno controlado

Por el contrario, si existe alguna situación que desate que se trabaje en un entorno no controlado, como lo muestra la figura 2, el sistema tendría pocas posibilidades de detectar el rostro y podría fallar igualmente a la hora de reconocerlo. La adquisición de una imagen que esté afectada por las agravantes mencionadas, condiciona el resultado final del procesamiento, es por ello que son múltiples los algoritmos que van enfocados a la normalización de la imagen del rostro y a su detección en entornos variados (lossurveys de M-H. Yang, January 2002).

Figura 2: Entorno no controlado

En el caso de la presente investigación, se asumirá la adquisición de la imagen en situaciones controladas donde debe garantizarse que el ruido que afecta a la detección sea el mínimo. Como se propone aplicar el resultado obtenido a través de una aplicación de control de acceso, la persona que vaya a ser analizada está consciente de la presencia de la cámara y el objetivo del sistema, por lo cual debe colaborar para la obtención de una imagen nítida y se presente el rostro con el mínimo de accesorios o elementos que atenten contra el proceso de reconocimiento.

Según las técnicas para detección de rostros, estas se dividen en dos grupos:

- Métodos basados en rasgos faciales.
- Métodos basados en la imagen.

Métodos basados en rasgos faciales

Estas técnicas explotan propiedades aparentes de la cara, tal como el color de la piel y la geometría facial, buscan encontrar aquellas características presentes en cualquier rostro: ojos, cejas, labios, boca, mentón, líneas de contorno, entre otros. La detección de la cara se resuelve manipulando medidas de distancia, ángulos y áreas de los rasgos visuales en la imagen. Lo más importante en este tipo de técnicas es decidir qué rasgos de la cara interesan para su estudio.

Métodos basados en la imagen

En estas técnicas el objeto de estudio es la imagen misma. El conocimiento previo se incorpora implícitamente en esquemas de entrenamiento. Se trabaja directamente con una representación de la imagen a la que se le aplican algoritmos de entrenamiento y análisis. Aplican herramientas generales de reconocimiento de patrones para sintetizar un modelo a partir de un conjunto de imágenes de entrenamiento (M. Delbracio C. Aguerrebere, 2006).

También existen **enfoques híbridos**, estos métodos usan tanto la información local como la global para la detección, basándose en el hecho de que el sistema de percepción humano distingue tanto las características locales como globales del rostro. (MARTA LUCÍA GUEVARA, Junio de 2008)

Entre estas clasificaciones, los métodos basados en la imagen tienen mayor aceptación y efectividad. Dentro de este grupo se encuentran técnicas muy utilizadas, aportando grandes resultados a investigaciones como el método propuesto por Paul Viola y Michael Jones, es uno de los más usados hoy en día ya que ha permitido segmentar múltiples rostros en una imagen con tiempos de procesamiento

bajos. También las redes neuronales, para segmentar el rostro alcanzando porcentajes de detección entre 77,9% y 90,3% para las diferentes configuraciones de la red (MARTA LUCÍA GUEVARA, Junio de 2008).

La extracción de características es la etapa más compleja del proceso de reconocimiento facial. Aquí es donde se extraen las características importantes del rostro, para almacenarlas en la base de datos y posteriormente establecer la comparación de estos. Los algoritmos de reconocimiento de rostro, durante la extracción de características se basan en los descriptores para la representación de objetos de interés (W. Zhao, 2003).

Una variedad de métodos de reconocimiento han sido propuestos durante los últimos años. El reconocimiento de caras es un tema desafiante que ha capturado el interés de muchos grupos de investigadores en las más variadas áreas y disciplinas. Es por esto que la literatura en este tema es realmente vasta y diversa. Para este trabajo se toma la siguiente categorización:

Métodos basados en características geométricas: Estos se basan en los mismos descriptores que utilizan las personas, explícitamente los componentes del rostro: los rasgos faciales (nariz, boca, ojos), las propiedades y relaciones entre estos (áreas, distancias, ángulos). Estos métodos son eficientes cuando hay poca variabilidad de iluminación, pose, el sujeto no cuenta con exceso de prendas, además deben contar con imágenes de alta calidad. Estas características geométricas por sí solas son inadecuadas para el reconocimiento de rostros, ya que existe información como: la textura o la apariencia facial, por solo citar algunas, que tienen alto grado de detalles y son obviadas (Marcel, June 2007).

Por otra parte los métodos basados en la apariencia han sido los más dominantes en los últimos años. En estos se han centrado los mayores esfuerzos por parte de los investigadores, debidos a que se enfocan directamente en las intensidades de los píxeles u otras representaciones basadas en la imagen que logran una mejor descripción de la misma. Esto constituye un avance para los sistemas de reconocimiento de rostros (Jain, 2005).

Los métodos basados en apariencia se clasifican en dos tipos a partir de las técnicas y algoritmos que utilizan:

Métodos holísticos: utilizan toda la imagen de la cara como entrada al sistema de reconocimiento, siendo ésta la unidad básica de procesamiento.

Métodos basados en características locales: Se extraen las características locales, como ojos, nariz, boca, etc. Sus posiciones y estadísticas locales constituyen la entrada al sistema de reconocimiento.

También existen métodos híbridos que combinan técnicas holísticas y locales (Vázquez, enero 2012). Los métodos basados en características locales, comparados con los globales, son más apropiados para el reconocimiento efectivo de rostro con una sola imagen de muestra (M. Delbracio C. Aguerrebere, 2006).

Los métodos basados en la apariencia o características locales involucran cuatro pasos generales:

1. Partición en regiones locales.

2. Extracción de rasgos locales.

3. Selección de rasgos.

4. Clasificación.

Se reporta que la forma más simple y comúnmente utilizada para extraer la información del rostro es dividir la imagen en ventanas rectangulares las cuales pueden estar solapadas o no. Sin embargo, se han utilizado también otras formas como elipses y franjas (Reyes, 2008)

Una vez extraídas las características del rostro representadas en los descriptores se procede a clasificar los mismos. El objetivo de este proceso es retener solamente un subconjunto del vector de rasgos original, evitando la pérdida de información discriminativa. Este paso es necesario en dependencia de la forma de representar la información, algunos métodos de extracción de rasgos incluyen ellos mismos un paso de selección. Para la clasificación se utiliza comúnmente la combinación de clasificadores, esta se puede llevar a cabo de dos formas: concatenando en un solo vector los rasgos de las diferentes regiones o realizando la clasificación por separado en cada región y luego tomar la decisión final combinando el resultado. (M. Delbracio C. Aguerrebere, 2006)

Atendiendo a que la información que recogen estos métodos básicamente está definida por descriptores de imagen, y estos engloban los descriptores de información general, que muestran resultados en cuanto al área de reconocimiento de rostro. A continuación se procede a clasificarlos atendiendo a qué parte de la imagen procesada representan:

- Descriptores globales.
- Descriptores locales.

Descriptores globales: Son aquellos que toman la imagen como un todo y no utilizan información a priori o sobre el objeto a representar. Resumen el contenido de la imagen en un único vector o matriz de características. Poseen la ventaja de encapsular una gran cantidad de información de la imagen requiriendo una pequeña cantidad de datos para describirla. A pesar de su simplicidad, este tipo de descriptores han resultado ser ampliamente utilizados para diferentes tareas, debido entre otras cosas a su bajo coste computacional unido a prestaciones relativamente buenas. (M. Delbracio C. Aguerrebere, 2006).

Descriptores locales: Son los que extraen información de puntos o regiones particulares de la imagen, considerando el hecho de que se trabaja con un objeto dado. Actúan sobre regiones de interés, previamente calculadas o identificadas, construyendo un vector de características de esa región que tiene en cuenta la información contenida tanto en el punto de interés como en la región adyacente al mismo o vecindario. Son aplicables al área de reconocimiento de rostros los descriptores locales, ya que el rostro será tomado como el objeto a representar (M. Delbracio C. Aguerrebere, 2006).

Durante la **confrontación** de la información se comparan las características extraídas con las almacenadas de cada uno de los rostros, se presenta como la etapa final del proceso. Se basa en métodos de comparación de características que buscan encontrar esas diferencias o similitudes que existen entre los rostros que se analizan. En muchos de los casos se utilizan algoritmos de similitud como puede ser similitud entre vectores de Wavelets de Gabor (Jet), ponderados o por magnitud o algoritmos que buscan definir la distancia entre los vectores o grafos de características, como es el caso de la distancia del coseno. Dependiendo del modo de funcionamiento del sistema se compara contra el registro del mismo usuario (modo verificación), o contra los registros de todos los usuarios (modo identificación), como se mencionó anteriormente (Alés, Junio 2012).

1.4 Análisis de otras soluciones existentes

En Cuba se presentan un grupo de trabajos del Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV) que se especializa en investigaciones sobre diversas temáticas, entre ellos la detección de patrones y el reconocimiento de rostro como rama de esta área del conocimiento.

Uno de los trabajos más destacados es una tesis para doctor en ciencias técnicas, por Heydi Méndez Vázquez. Este trabajo tiene como título “Algoritmo de reconocimiento de rostros basado en la apariencia local para aplicaciones reales en condiciones variables de iluminación” (Vázquez, enero 2012). En el

algoritmo se evalúa la calidad de las imágenes de rostros respecto a la iluminación y sólo si se determina que están afectadas se pre-procesan antes de continuar el resto de las acciones.

El algoritmo propuesto incluye las etapas clásicas del proceso, incluyéndole esta nueva etapa para el análisis de la incidencia de la iluminación en el rostro. En caso de estar presente logra corregirla, logrando así mayores resultados para las posteriores acciones. El resto de los pasos son muy similares a las demás soluciones. En detección se emplea Viola and Jones, en extracción se utiliza Patrones Binarios Locales Multiresolución (MLBP) con la clasificación basada en análisis discriminante lineal (LDA) y en la etapa de confrontación utilizan una función de similitud que no se define en la investigación.

Es importante aclarar que esta investigación, aun cuando el algoritmo de reconocimiento de rostros propuesto es más eficaz que los existentes en condiciones de iluminación variable, relega otros pequeños detalles. Esta no tiene en cuenta otro tipo de variaciones drásticas en el rostro como la oclusión, envejecimiento, ángulos de desviación de la pose de más de 15º. Sin embargo puede ser muy útil en entornos donde los cambios de pose no sean drásticos y las condiciones normalmente se controlen, aunque pueda cambiar el escenario, y aquí llega a ser fundamental la luminosidad. Ejemplos muy prácticos para la implementación de este sistema es en chequeos migratorios, sistemas de control de acceso a lugares restringidos que pueden estar en interiores o al aire libre.

En el ámbito internacional, se encontraron diversos trabajos enfocados a la solución de problemas en el área del reconocimiento de rostro, aunque en muchos casos solo se planteaban temáticas específicas como la detección o la extracción de características. Es relevante para esta investigación el caso de un proyecto desarrollado en Uruguay denominado Proyecto Aguará (M. Delbracio C. Aguerrebere, 2006), que se especializa en la creación de un sistema biométrico para una aplicación de control de acceso, utilizando como característica biométrica una imagen digital del rostro de la persona. En la parte de detección se implementó un algoritmo de localización de caras en escenas a color, con resultados no aceptables. La representación de la cara fue enfocada a características locales. Se utilizó como descriptores los Wavelets de Gabor con el algoritmo de Reconocimiento de caras *elastic bunch graph matching* (EBGM). En la etapa de confrontación utilizan la similitud por magnitud. Fue de gran utilidad para la investigación, pero esta necesita utilizar tecnologías propias o herramientas que permitan su integración en sistemas comerciales del Departamento Señales Digitales.

También se analizó el proyecto VISOR-BASE (*Video sensor object request broker open architecture for distributed services*) (Francisco Martín Rico), financiado por la Unión Europea, que tiene como objetivo el desarrollo de la arquitectura VISOR, basada en CORBA[2]. O sea, desarrollar una herramienta distribuida basada en estándares, con la posibilidad de integrar video sensores especializados desarrollados por cualquier compañía. Es aquí donde entra la Universidad de Rey Juan Carlos, en Móstoles, España, la cual creó un Video-Sensor distribuido basado en CORBA para el reconocimiento de caras. La función de este software es verificar la identidad de una persona en el contexto de un sistema de control de acceso. Las entradas a este sistema son un PIN (*Personal information number*), que tendrá que introducir el usuario del sistema, y la imagen captada del mismo. El PIN será el índice por el que buscar en una base de datos que contienen las características faciales del sujeto. Por otro lado el sistema extraerá características faciales de la imagen tomada y las comparará con las almacenadas, dando un PCD *(Personal confidence degree*), un número que indica el grado de similitud. Esta solución utiliza dos métodos internos para la extracción de características faciales, como métodos globales, *Principal Component Analysis* (PCA), y locales, Filtro de Gabor. Una vez obtenida la información utilizan una Red Neuronal tipo Multi-Layer Perceptron Neuronal Network (MLP-NN) para el reconocimiento.

1.5 Principales herramientas y tecnologías a utilizar

Metodologías para el Desarrollo de Software

El proceso de desarrollo de software es complejo, si no se establece una serie de pasos a seguir, puede quedar comprometida la calidad del software, o que este no cumpla con las expectativas o requisitos planteados por el cliente. Al igual, si el cliente solicita un cambio de última hora, sería como empezar el proyecto de cero, lo que implicaría retraso en la entrega del mismo.

Para evitar esas situaciones y hacer más sencillo y organizado el trabajo de los desarrolladores se crearon las metodologías. Estas sirven de soporte para guiar el proceso de desarrollo del software, orientar a los desarrolladores al crear un nuevo producto. Estas establecen una serie de procedimientos, técnicas, herramientas y soporte documental para crear software de calidad. O sea, imponen un proceso disciplinado sobre el desarrollo de software en aras de lograr una mayor eficiencia y predictibilidad.

[2] Common Object Request Broker Arquitecture (CORBA). No es un software o aplicación, se utiliza para establecer una especificación de inter-operabilidad entre plataformas.

Debido a que los requisitos y complejidad de un software son tan variados con respecto a otros, ha dado lugar a que existan diferentes tipos de metodologías, estas pueden agruparse en dos grandes grupos:

Metodologías ágiles/ligeras: Estas muestran al cliente versiones parcialmente funcionales del software en desarrollo, en intervalos cortos, para que este pueda ir sugiriendo cambios y ser corregidos con prontitud. Son orientadas a la interacción con el cliente y el desarrollo incremental del software.

Metodologías pesadas: Estas establecen las actividades a desarrollar, herramientas a utilizar y notaciones que se emplearán. Requieren una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodología es más eficaz y necesaria cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear. Son orientadas al control de los procesos (Jacobson, 2000).

Proceso Unificado de Desarrollo de Software (Rational Unified Process, RUP)

RUP es un proceso de desarrollo de software, por lo cual puede ser considerado como el conjunto de actividades necesarias para transformar los requisitos del cliente en un producto de software. El sistema unificado es además un marco de trabajo genérico, que se puede especificar para diferentes sistemas de software, sin importar el tamaño, áreas de aplicación y tipos de organización. Este está basado en componentes, o sea, el software en construcción está conformado por componentes de software interconectadas a través de interfaces bien definidas.

Sin embargo, las tres principales características de RUP son:

- ***Dirigido por casos de uso****,* donde los casos de uso (fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante) definen lo que el usuario desea a partir de la captura de requisitos y la modelación del negocio.
- ***Centrado en la arquitectura,*** característica que está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema, la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. Deben implementarse en primer lugar los casos de uso base, o sea, aquellos que puedan ser de utilidad para el funcionamiento de otros.

- ***Iterativo e incremental,*** cada fase se divide en iteraciones, dividiendo el producto en pequeños proyectos para el desarrollo e incremento del mismo. Durante todo el proceso se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

RUP posee 4 fases en su ciclo de vida: Inicio, Elaboración, Construcción y Transición. Estas fases de trabajo poseen actividades las cuales son agrupadas en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos de apoyo. La figura 3 muestra el ciclo de vida de esta metodología (Jacobson, 2000).

Figura 3: Ciclo de vida de RUP (Banny Solano, 2010)

La metodología de desarrollo antes expuesta ha sido seleccionada para guiar todo el proceso de desarrollo de software del proyecto SURIA. Uno de los aspectos que se tuvo en cuenta para esta selección es el soporte que se le da a los expedientes de proyectos desarrollados para esta metodología por parte del equipo de calidad UCI, el cual se encarga de mantener actualizados dichos expedientes proporcionándole a los desarrolladores las plantillas de los distintos documentos a desarrollar.

Herramienta de modelado de los artefactos de RUP

A lo largo del ciclo de vida de desarrollo de un software, un conjunto de programas y ayudas dan asistencia a los programadores, analistas e ingenieros de software. Estas son las llamadas herramientas CASE (Ingeniería de software asistida por computadora). Para que las tareas de un proyecto sean organizadas y completadas de forma eficiente, se necesita de un mecanismo que se encargue de ello. El objetivo principal de estas herramientas es la automatización de esos procesos y facilitar la coordinación de eventos que necesitan ser tratados en el ciclo de vida de desarrollo del software.

Visual Paradigm 8.0

Es una herramienta profesional para el Lenguaje Unificado de Modelado (UML), que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Presenta la desventaja de que requiere mantener varios hilos del programa en proceso, y tiene problemas de integración con otras herramientas de desarrollo.

Lenguaje de Modelado (UML) 2.0

UML es un lenguaje que se centra en la representación gráfica de un sistema y a su vez indica cómo crear y leer los modelos.

Las funciones principales de este lenguaje son las siguientes:

- Visualizar: Permite mostrar un sistema de forma gráfica, facilitando que otra persona ajena a este pueda entenderlo.
- Especificar: Permite definir cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Para el desarrollo del video sensor se utiliza UML para lograr el diseño y la comunicación entre las clases, alcanzar un mayor entendimiento del problema, así como facilitar el proceso de implementación.

QtCreator 2.4 como entorno integrado de Desarrollo

QtCreator es un IDE (Entorno de desarrollo integrado) creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt. Es soportado por sistemas operativos como GNU/Linux, Mac OS X, Windows XP, Vista, Windows 7, por lo que es multiplataforma y software libre. Permite construir interfaces de usuario complejas de una forma visual y rápida, ya que incluye un editor de texto con autocompletado, diseñador de interfaces gráficas, gestión de proyectos, sistema de depuración e integración con sistemas de control de versiones.

QtCreator es un IDE para el desarrollo en C++ que cumple con las políticas del centro GEYSED y es utilizado en el proyecto SURIA, factor que favorece su selección.

C++ como Lenguaje de Programación

C++ es un lenguaje imperativo orientado a objetos derivado del C. Es una mejoría sobre muchas de las características de C, y proporciona capacidades de P.O.O. (Programación Orientada a Objeto) que aporta mucho para incrementar la productividad, calidad y reutilización del software. Este lenguaje es muy eficaz en cuanto a rapidez y uso de memoria en las aplicaciones que se obtienen. Las bibliotecas estándar de C++ proporcionan un conjunto extenso de capacidades de entrada/salida. En el diseño del C++ primó sobre todo la velocidad de ejecución del código.

C++ se ha seleccionado para la realización del video sensor ya que al ser uno de los lenguajes más rápidos en cuanto a ejecución, es de vital importancia a la hora de trabajar con flujos de video. Es uno de los más populares y utilizados a nivel mundial por lo que existe mucha bibliografía de apoyo. Además, la biblioteca a utilizar (OpenCV) utiliza C++ como lenguaje. Por las características antes mencionadas C++ resulta ser este el lenguaje más idóneo para realizar el video sensor.

OpenCV 2.4.3 como biblioteca a utilizar

OpenCV (*Open Source Computer Vision Library*) es una biblioteca de funciones en C y C++ desarrollado por Intel que proporciona un alto nivel de funciones para el procesado de imágenes, visión artificial, captura de video y visualización de imágenes. Es de código abierto, gratuita, multiplataforma (disponible para entornos MS Windows, Mac OS y Linux), está desarrollada bajo la licencia BSD (*Distribución de Software Berkeley*), rápida, de fácil uso y en continuo desarrollo. Estas bibliotecas permiten a los programadores crear aplicaciones poderosas en el dominio de la visión digital.

OpenCV, debido a sus características y capacidades para el procesamiento de imágenes digitales es muy utilizada hoy día en los sistemas de video vigilancia. Para la realización del video sensor para detección y reconocimiento de rostros se seleccionó esta biblioteca debido a que cuenta con funciones que facilitan el trabajo con rostro. También ayudó para su selección que es la utilizada en el proyecto Video Vigilancia SURIA.

FFMpeg como biblioteca

FFMpeg es un conjunto de bibliotecas y utilidades de código abierto cuyo propósito es la manipulación de archivos multimedia, principalmente vídeos. Puede grabar, convertir (transcodificar) y hacer streaming[3] de audio y vídeo. Para la realización del video sensor para detección y reconocimiento de rostros se seleccionó esta biblioteca con el objetivo de que capture el flujo de video de las cámaras ip y obtener los fotogramas que lo componen. Con el uso de esta se agiliza el proceso de captura del flujo de video con respecto a la captura que realiza OpenCV.

1.6 Conclusiones Parciales

Después de comprender la situación problemática, y analizados los procesos involucrados en el objeto de estudio se puede concluir que: el área de conocimiento relacionada con la detección y reconocimiento está sujeta a numerosas problemáticas, en cuanto al entorno en que se capturan las imágenes. De la calidad de las imágenes depende el resultado de los pasos implicados en el proceso.

Según la línea de este trabajo se percibieron diferentes soluciones que se acercan al resultado esperado, sin embargo existe conflicto entre las tecnologías o herramientas y su integración en sistemas comerciales del Departamento Señales Digitales. Estas soluciones presentan código cerrado y licencias privativas, por lo que imposibilita su integración al sistema SURIA.

Es importante resaltar la combinación eficiente de la utilización de QtCreator con C++ y OpenCV. QtCreator es un IDE para el desarrollo en C++, es software libre y cumple con las políticas del departamento. C++, al ser uno de los lenguajes más rápidos en cuanto a ejecución, es de vital importancia a la hora de trabajar con flujos de video. La biblioteca OpenCV está implementada en C/C++, por lo cual la fusión de estos elementos resulta fructífera para el desarrollo del video sensor para detección y reconocimiento de rostros.

[3] *El **streaming** es un término que hace referencia al hecho de escuchar música o ver vídeos sin necesidad de descargarlos, sino que se hace por fragmentos enviados secuencialmente a través de la red (como lo es Internet)* (Castro, 2013).

Capítulo 2: Análisis y diseño del sistema

2.1 Introducción

En este capítulo se realiza la propuesta inicial del sistema, describiendo el negocio en el que se enmarca el mismo. Se identifican los requisitos funcionales y no funcionales, con el fin de comprender las características del sistema, en función de procesamiento, rendimiento y sistemas operativos. Se puntualizan los casos de uso del sistema identificados, presentando la descripción de estos. También se definen las clases de análisis y diseño de los casos de uso; así como los diagramas de secuencia. Además, se presenta la arquitectura del sistema, y los patrones de diseño que se manejan para la implementación del sistema.

2.2 Modelo de Dominio

Siguiendo la descripción del problema a resolver, el proceso del negocio no está bien determinado, ya que no se logra ver claramente quiénes son las personas involucradas y aquellas que desarrollan las actividades en cada uno de estos procesos. Al no existir estos sujetos no se pueden identificar procesos que ocurren en el negocio y tampoco reconocer beneficiados, por lo tanto se realizará la modelación del dominio.

Un Modelo de Dominio es una representación visual de clases conceptuales u objetos reales en un dominio específico. Este consiste en un conjunto de diagramas de clases, sin definición de operaciones. Constituye una base de conocimiento con los principales conceptos asociados al desarrollo del sistema.

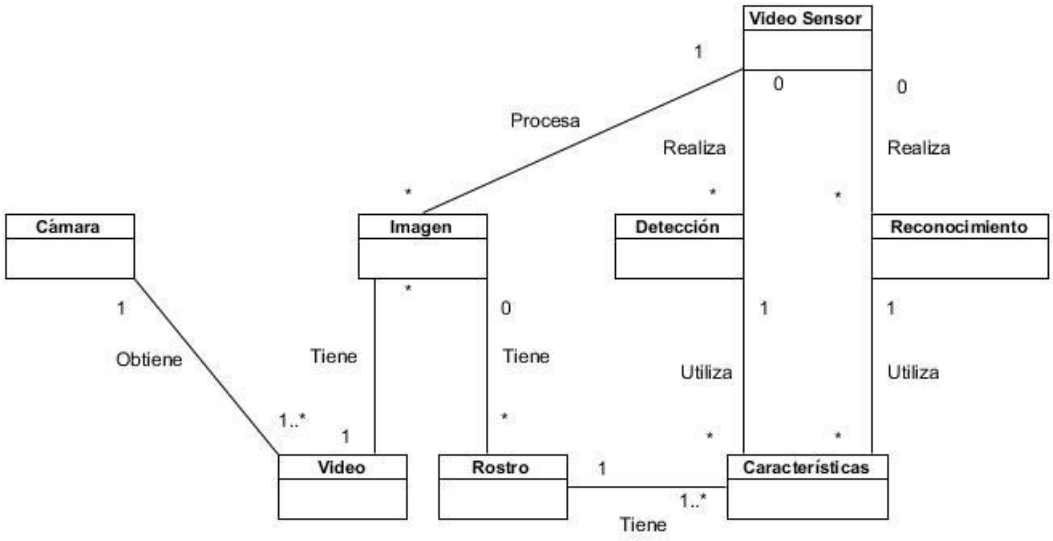

Figura 4: Diagrama del Modelo de Dominio

2.2.1 Descripción del Modelo de Dominio

En la figura anterior la cámara obtiene flujos de video. Un flujo de video está compuesto por una serie de imágenes, las cuales pueden contener rostros y a su vez son procesadas por el video sensor. El rostro tiene características, las cuales son utilizadas por los procesos de detección y reconocimiento que son realizados por el video sensor.

2.3 Especificación de los requisitos de software

A lo largo del proceso de desarrollo de software, lograr un entendimiento productivo entre clientes y el equipo de proyecto es de vital importancia. Para lograr esto se deben establecer los objetivos del producto. En aras de llegar a un consenso entre ambas partes involucradas, se establecen requisitos, estos pueden ser: funcionales y no funcionales. El cumplimiento de los mismos es directamente proporcional al resultado final.

2.3.1 Requisitos Funcionales

"Los Requerimientos Funcionales son condiciones o capacidades que el sistema debe cumplir, suficientemente buenas como para llegar a un acuerdo entre los clientes (incluyendo usuarios) sobre qué debe y qué no debe hacer el sistema." (Jacobson, 2000)

A continuación se muestran los requerimientos funcionales del sistema:

RF1. El sistema debe permitir capturar flujos de video proveniente de una cámara IP.

RF2. El sistema debe permitir detectar si en la imagen existe un rostro.

RF3. El sistema debe permitir actualizarse con un entrenamiento previo, de una serie de imágenes pertenecientes cada una de las personas involucradas.

RF4. El sistema debe permitir identificar la persona de la imagen que previamente debe haber sido registrada.

2.3.2 Requisitos no funcionales

"Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, además son aspectos importantes que el producto debe cumplir para lograr un producto atractivo, usable, rápido o confiable." (Pressman, 2005)

A continuación se muestran los requerimientos no funcionales del sistema:

RNF1. Rendimiento

RNF1.1 Se debe procesar el video en tiempo real.

Los fotogramas del flujo de video son re escalados a un tamaño de 320 x 240 pixeles. Aproximadamente el tiempo de procesamiento de un fotograma con estas características es de 145 milisegundos por lo que da un margen de procesar alrededor de 6 fotogramas en un segundo. Se obvian 5 fotogramas y se procesa uno, para cumplir con el procesado en tiempo real.

RNF2. Usabilidad

RNF2.1 El componente debe describir con claridad los parámetros de configuración del algoritmo.

En caso de no recibir los parámetros correctos se muestra un mensaje de ayuda explicando la forma de introducir los mismos.

RNF3 Fiabilidad

RNF3.1 La disponibilidad del componente será mayor del 95% sobre la totalidad del tiempo (24 horas de los 7 días a la semana)

En los parámetros anteriores no se tomaran en cuenta el tiempo fuera de servicio producido por afectaciones externas como:

- Mal estado constructivo
- Pérdida de conexión con las cámaras ya sea por falta de fluido eléctrico o error de hardware.
- Falta de clima en el local donde se encuentre el servidor de procesamiento.
- Falta de seguridad en el local donde se encuentre el servidor de procesamiento.

RNF4 Portabilidad

RNF4.1 El componente debe poder ser utilizado sobre diferentes plataformas como Windows, Linux y Mac OS.

Las bibliotecas OpenCV, FFmpeg y el lenguaje C++ son multiplataforma, por lo que se logra la portabilidad compilando el componente en la plataforma que se vaya a utilizar.

RNF4.2 La plataforma debe ser de una arquitectura de 32 bits.

Las bibliotecas de OpenCV fueron compiladas en un sistema operativo con una arquitectura de 32 bits, por lo que no se asegura el uso del componente en una arquitectura de 64 bits sea estable en su funcionamiento, debido a que pueden existir dependencias con la arquitectura de compilación.

RNF5 Hardware

RNF5.1 Se debe tener como mínimo 1Gb de RAM y procesador Pentium 4 a 3 GHz.

Si no se tiene esos requisitos mínimos de hardware la imagen se pixela y el resultado del algoritmo no es el correcto. Esto se demostró realizando pruebas en computadoras con menos prestaciones que las mínimas descritas.

RNF6 Restricciones del diseño

RNF6.1 Lenguaje: El lenguaje que se utilizará para el desarrollo del sistema será C++.

RNF6.2 Biblioteca: Las bibliotecas que se utilizarán serán OpenCV y FFMpeg.

2.4 Definición de los casos de uso

Según **Roger Pressman** *"Un caso de uso cuenta una historia estilizada de la manera en que un usuario final (el cual desempeña uno de varios papeles posibles) interactúa con el sistema en un conjunto específico de circunstancias. La historia puede ser un texto narrativo, un esquema de tareas o interacciones, una descripción basada en una plantilla o una representación por medio de diagramas. Sin importar su forma un caso de uso muestra el software o sistema desde el punto de vista del usuario final."* (Pressman, 2005) Estos proporcionan la entrada fundamental para el análisis, diseño y las pruebas del sistema. Facilitan un medio para que los desarrolladores, usuarios finales y trabajadores lleguen a una comprensión común del sistema propuesto.

2.4.1 Descripción de los actores

"Un actor es una idealización de una persona externa, de un proceso, o de un ente que interactúa con un sistema, un subsistema, o una clase. Un actor caracteriza las interacciones que los usuarios exteriores pueden tener con el sistema.

Cada actor participa en uno o más casos de uso. Interactúa con el caso de uso (y por lo tanto con el sistema o la clase que posee el caso de uso), intercambiando mensajes." (Jacobson, 2000) Este puede ser caracterizado suficientemente por un conjunto de atributos que definen su estado, su implementación interna no es relevante en el caso de uso.

Actor del Sistema	Descripción
Sistema Video Vigilante	Es el sistema con quien interactúa el componente (Módulo Analytics del sistema SURIA), es el que inicia el caso de uso Reconocer Rostro.

Tabla 1: Descripción del actor

2.4.2 Diagrama de Casos de Uso del Sistema

En la figura 5 se muestra el diagrama de casos de uso (CU) del sistema.

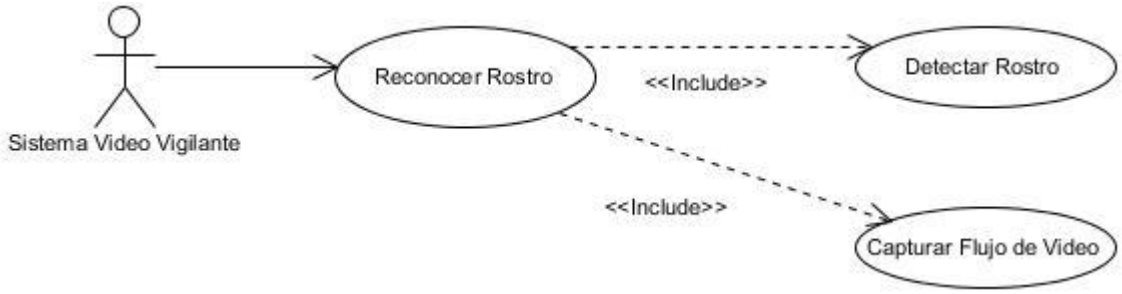

Figura 5: Diagrama de Casos de Uso del Sistema

2.4.3 Especificación de Casos de Uso

CU-1 Reconocer Rostro

Objetivo	Reconocer un rostro.	
Actores	Sistema Video Vigilante :(Inicia).	
Resumen	El caso de uso inicia cuando el Sistema Video Vigilante hace la petición de reconocer rostro. Este, a partir de un rostro detectado obtenido de un flujo de video de una cámara IP, indica si la persona pertenece o no al entrenamiento previo del sistema.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	Debe haber un rostro detectado.	
Poscondiciones	Ninguna	
Flujo de eventos		
Flujo básico: Reconocer Rostro.		
	Actor	**Sistema**

1.	Selecciona “Reconocer Rostro”.	
2.		Se analiza un fichero de entrenamiento de imágenes que es parámetro de la aplicación.
3.		Se verifica que exista suficiente información en el fichero y que éste es válido.
4.		Se crea un modelo del algoritmo a utilizar.
5.		Se cambia el tamaño de las imágenes para que todas tengan el mismo
6.		Se verifica que existe un entrenamiento previo.
7.		Se captura flujo de video.
8.		Se detecta un rostro.
9.		Se evalúa si el rostro detectado pertenece a alguna persona registrada.
10.		El caso de uso finaliza cuando muestra una imagen con el rostro de la persona y su nombre.
Flujos Alternos		
3a. Evento Insuficiente información en fichero		
	Actor	**Sistema**
1.		Si no existe suficiente información en el fichero se muestra un mensaje: “Insuficiente

		información para entrenar el sistema".
6a. Evento No existe entrenamiento previo		
	Actor	**Sistema**
1.		Si no existe entrenamiento previo el sistema se entrena con los datos obtenidos del fichero pasado por parámetro.
7a. Evento No se capturó flujo de video		
	Actor	**Sistema**
1.		Si el sistema no captura flujo de video muestra un mensaje "No se capturó flujo de video".
8a. Evento No se detectó rostro		
.	**Actor**	**Sistema**
1		Si el sistema no detecta rostro vuelve a empezar en el paso 7.

Relaciones	**CU Incluidos**	Capturar Flujo de Video. Ver CU Capturar Flujo de Video. Detectar Rostro. Ver CU Detectar Rostro.
	CU Extendidos	No procede.
Requisitos no funcionales	Ninguna.	
Asuntos pendientes	No procede.	

Tabla 2: Descripción del caso de uso "Reconocer Rostro"

CU-2 Capturar Flujo de Video

Objetivo	Capturar un flujo de video	
Actores	Sistema Video Vigilante	
Resumen	El caso de uso inicia cuando el Sistema Video Vigilante hace la petición de reconocer rostro. El sistema captura el flujo de video proveniente de la cámara IP, y extrae fotograma a fotograma.	
Complejidad	Media	
Prioridad	Crítico	
Precondiciones	Debe existir conexión a una cámara IP.	
Poscondiciones	Ninguna	
Flujo de eventos		
Flujo básico: Reconocer Rostro.		
	Actor	**Sistema**
1		Se obtiene la dirección de la cámara IP.
2		Se solicita a la cámara IP el flujo de video.
3		Se crea una estructura que referencie el flujo de video.
4		Se extrae un fotograma.

<table>
<tr><td colspan="4">Flujos Alternos</td></tr>
<tr><td colspan="4">2a. Evento Solicitar a la cámara dirección ip</td></tr>
<tr><td></td><td colspan="2">Actor</td><td>Sistema</td></tr>
<tr><td>1.</td><td colspan="2"></td><td>Si la cámara no brinda flujo de video se muestra un mensaje: “No se pudo obtener el flujo de video.</td></tr>
<tr><td rowspan="2">Relaciones</td><td>CU Incluidos</td><td colspan="2">No procede.</td></tr>
<tr><td>CU Extendidos</td><td colspan="2">No procede.</td></tr>
<tr><td>Requisitos no funcionales</td><td colspan="3">Ninguna</td></tr>
<tr><td>Asuntos pendientes</td><td colspan="3">No procede.</td></tr>
</table>

Tabla 3: Descripción del caso de uso “Obtener Flujo de Video”

CU-3 Detectar Rostro

Objetivo	Detectar un rostro
Actores	Sistema Video Vigilante
Resumen	El caso de uso inicia cuando el Sistema Video Vigilante hace la petición de reconocer rostro. El sistema verifica que exista un rostro en algún fotograma y extrae el mismo.
Complejidad	Media
Prioridad	Crítico

Precondiciones	Se debe haber capturado un frame previamente		
Poscondiciones	Ninguna		
Flujo de eventos			
Flujo básico: Reconocer Rostro.			
	Actor	**Sistema**	
1		Se procede a buscar los objetos presentes en la imagen.	
2		Se analiza si esos objetos representan un rostro.	
3		El caso de uso finaliza cuando se obtiene un rostro.	
Relaciones	**CU Incluidos**	No procede.	
	CU Extendidos	No procede.	
Requisitos no funcionales	Ninguna		
Asuntos pendientes	No procede.		

Tabla 4: Descripción del caso de uso "Detectar Rostro"

2.5 Análisis y Diseño

2.5.1 Descripción de la arquitectura

De acuerdo al *Software Engineering Institute* (SEI), la Arquitectura de Software se refiere a *"las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos."* (L. Bass, 2003) El concepto de arquitectura de software se refiere a la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo. La manera en que se organiza un sistema tiene un impacto directo sobre la capacidad de este para satisfacer los atributos de calidad del sistema, los que forman parte de los requisitos no funcionales del sistema.

El sistema SURIA está diseñado siguiendo una arquitectura de Pizarra. Esta consta de varios elementos funcionales conocidos como agentes y un instrumento de control conocido como pizarra. Los agentes normalmente son programas especializados en una tarea concreta, su comportamiento básico es examinar la pizarra, realizar su tarea y escribir sus conclusiones en la pizarra. Así, otro agente puede trabajar sobre los resultados arrojados por el primero.

La pizarra tiene un doble papel, por una parte, coordina a los distintos agentes y por otra, facilita su intercomunicación. Su estado inicial es una descripción del problema a resolver.

En el caso de SURIA, la pizarra es el gestor, el mismo es el módulo fundamental de la aplicación, mientras que los restantes módulos funcionan como agentes, encargándose de realizar tareas específicas. El módulo Analytics es uno de estos. Dicho módulo sirve de fachada entre el Gestor y los videos sensores.

El video sensor desarrollado, es un sistema con una arquitectura centrada en los flujos de datos, basada en el patrón "*pipe and filter*" (tuberías y filtros), donde los componentes presentes en el sistema responden a decisiones arquitectónicas para hacer cumplir requerimientos funcionales y no funcionales. Estos componentes están asociados a flujos de datos y refinamientos sucesivos. O sea, un conjunto de elementos denominados "filtros" conectados entre sí por "tuberías" y trasmiten datos desde un componente al siguiente.

Estos filtros se diseñan de tal modo que esperan un conjunto de datos en un determinado formato y obtiene como resultado otros datos de salida en un formato específico. Este patrón de arquitectura es particularmente efectivo a la hora de descomponer el problema en pasos independientes, reutilizar filtros, facilitar el mantenimiento, independencia entre filtros y ejecución concurrente de filtros.

En la figura 6 se muestra una representación de la arquitectura del video sensor para detección y reconocimiento de rostros basada en el patrón tuberías y filtros. Se muestra cada filtro y su funcionalidad, cómo los datos de entrada van siendo modificados por las actividades de cada filtro hasta la salida.

Figura 6: Arquitectura del video sensor

2.5.2 Patrones de Diseño

Los patrones de diseño brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Es necesario tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).Entre los patrones de diseño más utilizados se encuentran los patrones GRASP y los patrones GoF.

Los patrones GRASP (*General Responsibility Assignment Software Patterns*) son patrones generales para asignar responsabilidades y diseñar con éxito el software orientado a objetos. Para el desarrollo del video sensor se utilizaron los siguientes patrones pertenecientes a ese grupo:

Experto: Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Expresa simplemente que los objetos realizan operaciones relacionadas con la información que poseen.

Creador: Plantea la necesidad de asignarle a una clase la responsabilidad de crear una instancia de otra clase siempre y cuando agregue los objetos de la clase, los contenga, registre las instancias de estos objetos y los utilice específicamente.

Controlador: Se hace ventajoso llevar el control de la aplicación en una sola clase, esa es la idea del patrón controlador. Es un objeto en particular, en una aplicación, que lleva el manejo de la misma.

También se utilizan los patrones **Alta Cohesión** y **Bajo Acoplamiento**. El primero es la idea de tener las clases lo menos ligadas entre sí. Así, si se produce una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, fortaleciendo la reutilización, y disminuyendo la dependencia entre clases. El segundo indica que la información que contiene una clase debe de ser coherente y esté relacionada con la clase, esto facilita el cambio. Al realizar un cambio en una clase muy cohesionada, todos los métodos que se vean afectados estarán a la vista, en la misma clase.

2.6 Modelo de análisis

Durante el análisis, se analizan los requisitos que fueron descritos anteriormente con el objetivo de conseguir mayor comprensión y una descripción de los mismos que ayude a estructurar todo el sistema, incluyendo su arquitectura. Se puede considerar como una primera aproximación al modelo de diseño y es por tanto una entrada fundamental al mismo.

2.6.1 Diagrama de clases del análisis

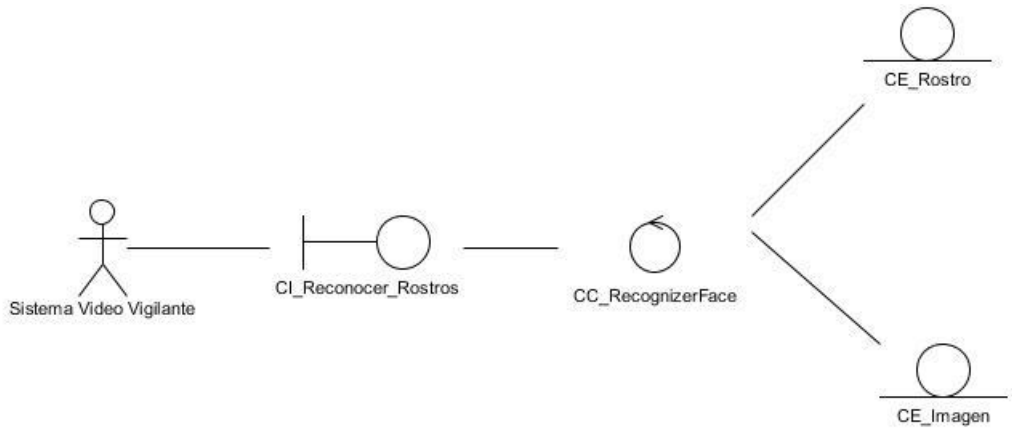

Figura 7: Diagrama de clases del análisis (Reconocer Rostro)

En la figura 7 se muestra el diagrama de clases del análisis para el CU Reconocer Rostro, se pueden observar las clases entidades CE_Rostro y CE_Imagen, son las clases persistentes con las que trabaja el algoritmo de reconocimiento que se ejecuta en la clase controladora CC_RecognizerFace que es la encargada de la secuencia de pasos para cumplir el objetivo que es el reconocimiento de rostros. También la clase interfaz CI_Reconocer_Rostro que es la que muestra la identidad del sujeto.

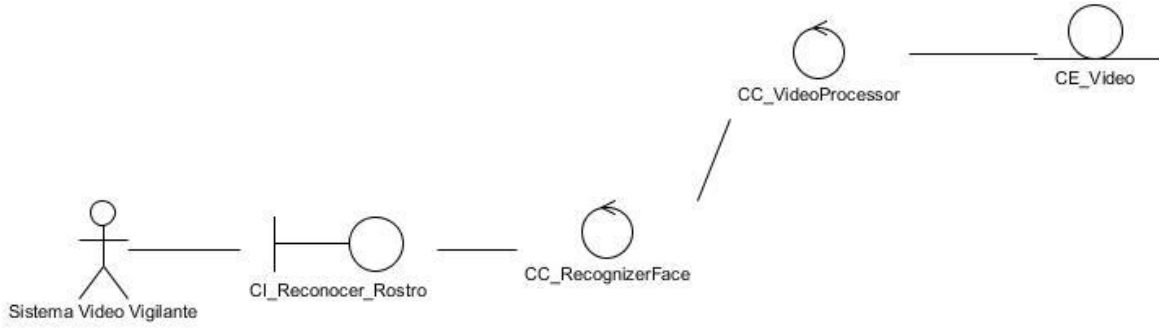

Figura 8: Diagrama de clases del análisis (Obtener Flujo de Video)

En la figura 8 se puede apreciar el diagrama de clases del análisis para el CU Obtener Flujo de Video, aquí aparece la clase entidad CE_Video, la cual es utilizada por la clase controladora CC_VideoProcessor que es la encargada de obtener el flujo de video y proporcionárselo a la clase controladora CC_RecognizerFace para lograr el reconocimiento de rostros.

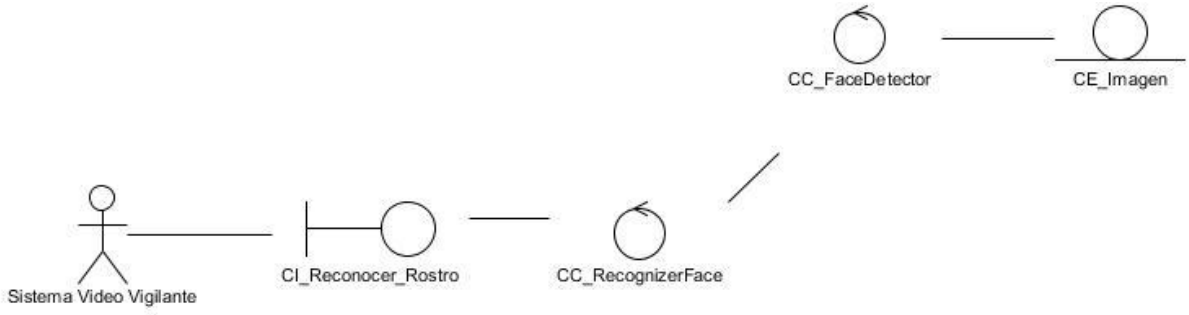

Figura 9: Diagrama de clases del análisis (Detectar Rostro)

En la figura 9 se muestra el diagrama de clases del análisis para el CU Detectar Rostro, aquí interviene la clase entidad CE_Imagen que la utiliza la clase controladora CC_FaceDetector, cuyo objetivo es devolverle un rostro a la clase CC_RecognizerFace para que esta reconozca el rostro.

2.6.2 Diagramas de interacción

Estos son diagramas que describen cómo grupos de objetos colaboran para lograr un objetivo. Lo que muestra este tipo de diagrama son objetos y los mensajes que se pasan entre ellos. Los diagramas de interacción capturan el comportamiento de un escenario o parte del sistema, estos pueden dividirse en varios grupos, diagramas de colaboración y de secuencia.

Diagramas de colaboración

Estos muestran cómo las instancias específicas de las clases trabajan en colectivo para conseguir un objetivo común. Implementa las asociaciones del diagrama de clases del análisis mediante el paso de mensajes de un objeto a otro. La secuencia de estos mensajes viene dada por el número de secuencia.

Diagramas de secuencia

Muestran las interacciones entre un conjunto de objetos, ordenadas según el tiempo en que tienen lugar. En estos diagramas los objetos que intervienen son instancias concretas de una clase que participa en la interacción. Un diagrama de secuencia representa una forma de indicar el período durante el que un objeto está desarrollando una acción directamente o a través de un procedimiento.

2.6.3 Diagramas de colaboración

En las figuras 10, 11 y 12 se muestran los diagramas de colaboración para los casos de uso Reconocer Rostro, Capturar Flujo de video y Detectar Rostro. Se observan los principales mensajes intercambiados entre las clases para realizar los casos de usos en cuestión.

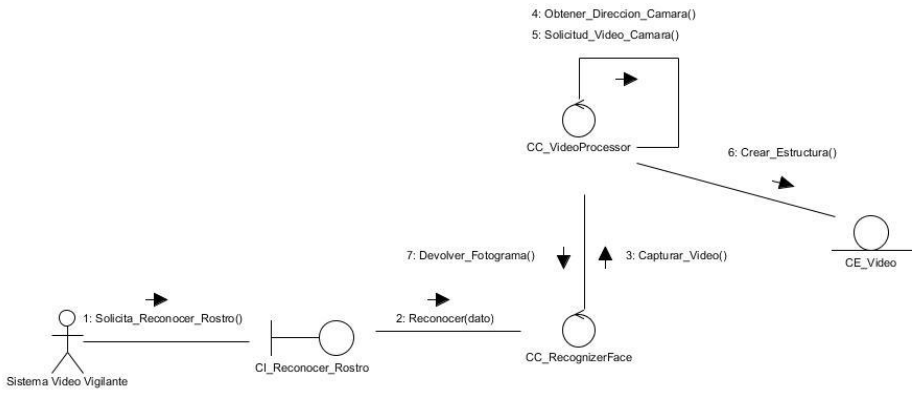

Figura 10: Diagrama de colaboración del CU Reconocer Rostro

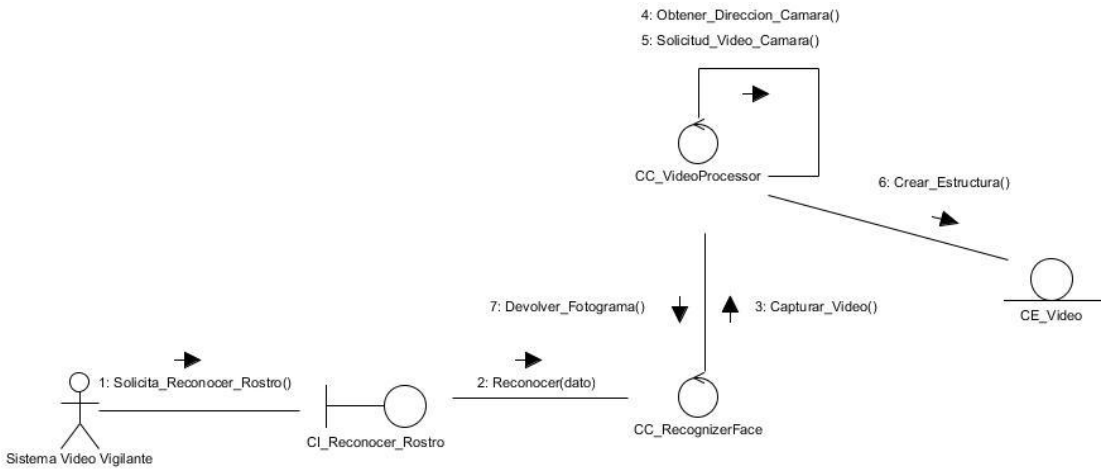

Figura 11: Diagrama de colaboración del CU Capturar Flujo de Video

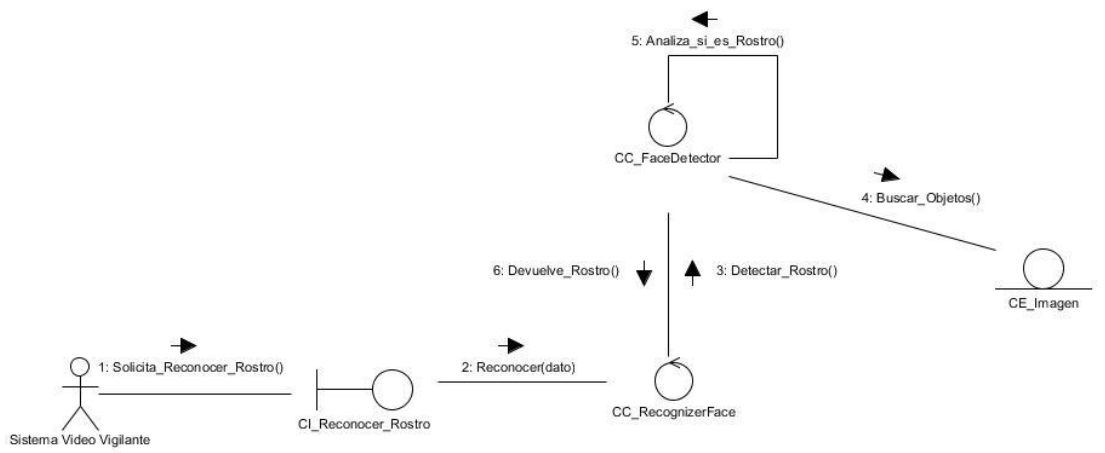

Figura 12: Diagrama de colaboración del CU Detectar Rostro

2.7 Modelo del diseño

El modelo de diseño es un modelo físico y concreto, da forma al sistema y debe ser mantenido durante todo el desarrollo del ciclo de vida del software. Se centra en la realización de los casos de uso a través de los requisitos, tanto funcionales como no funcionales. La entrada esencial a este modelo es el resultado del modelo de análisis, ya que este resultado proporciona una comprensión detallada de los

requisitos. El objetivo final del flujo de trabajo de diseño es producir un modelo lógico del sistema a implementar.

2.7.1 Diagrama de clase del diseño

Este tipo de diagrama representa la realización física de los casos de usos, centrándose en cómo los requisitos funcionales y no funcionales tienen impacto en el sistema. Describe gráficamente las especificaciones de las clases de software y las interfaces y contiene las definiciones de las entidades del software en vez de conceptos del mundo real.

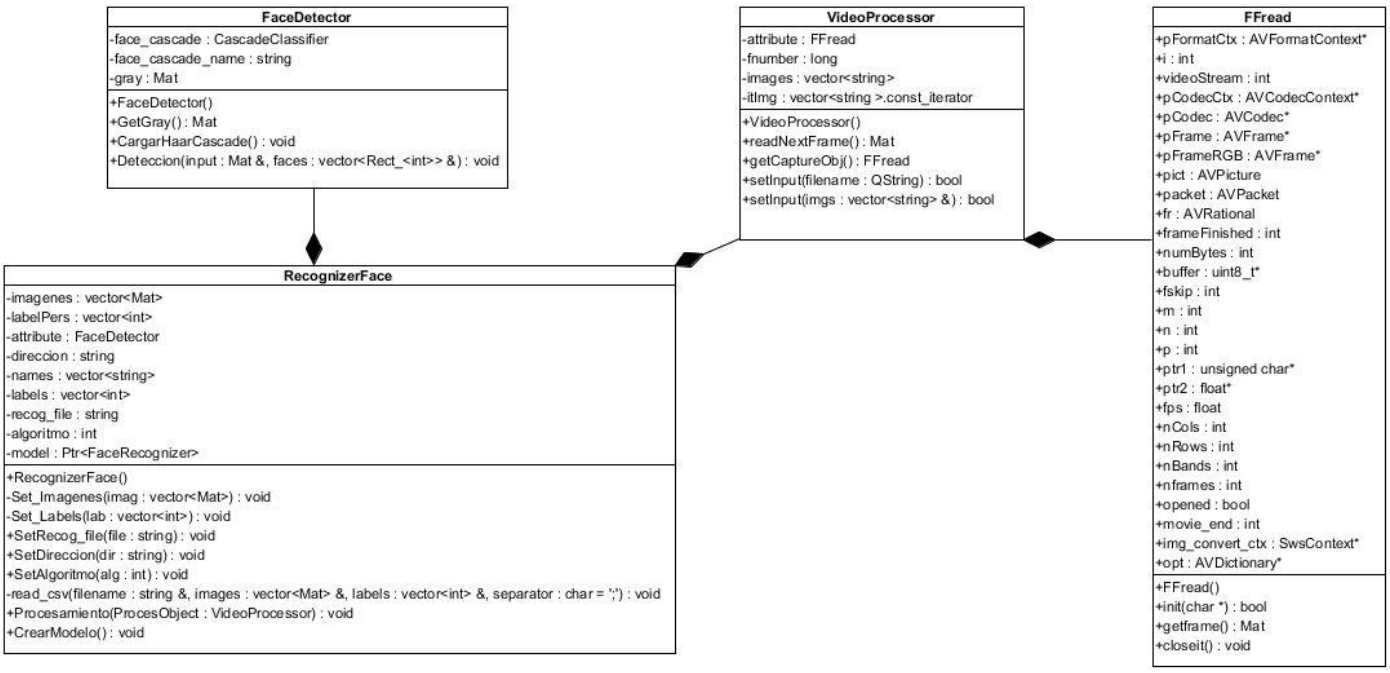

Figura 13: Diagrama de clases del diseño para el sistema propuesto

En la figura 13 se muestra el diagrama de clases del diseño para el video sensor para la detección y reconocimiento de rostros. Aquí intervienen las clases FFread que permite capturar un video y se lo va proporcionando a VideoProcessor frame a frame, según se necesiten, contiene toda la información necesaria para el trabajo con video. La clase VideoProcessor que es la que domina la información sobre cada frame que sea capturado del video y se los proporciona a RecognizerFace. Esta analiza cada imagen que se le brinda, buscando un rostro e identificándolo en caso de estar presente. La clase

FaceDetector es la encargada de devolverle a RecognizerFace el rostro que se encuentra en la imagen que esta le ofrece.

2.7.2 Diagrama de secuencia del diseño

En un diagrama de secuencia se indican los módulos o clases que forman parte del programa y las llamadas que se hacen en cada uno de ellos para realizar una tarea determinada. Estos son útiles para definir acciones que se puedan realizar en la aplicación. A continuación en las figuras 14, 15 y 16 se muestran los diagramas de secuencia para los casos de uso Reconocer Rostro, Capturar Flujo de Video y Detectar Rostro.

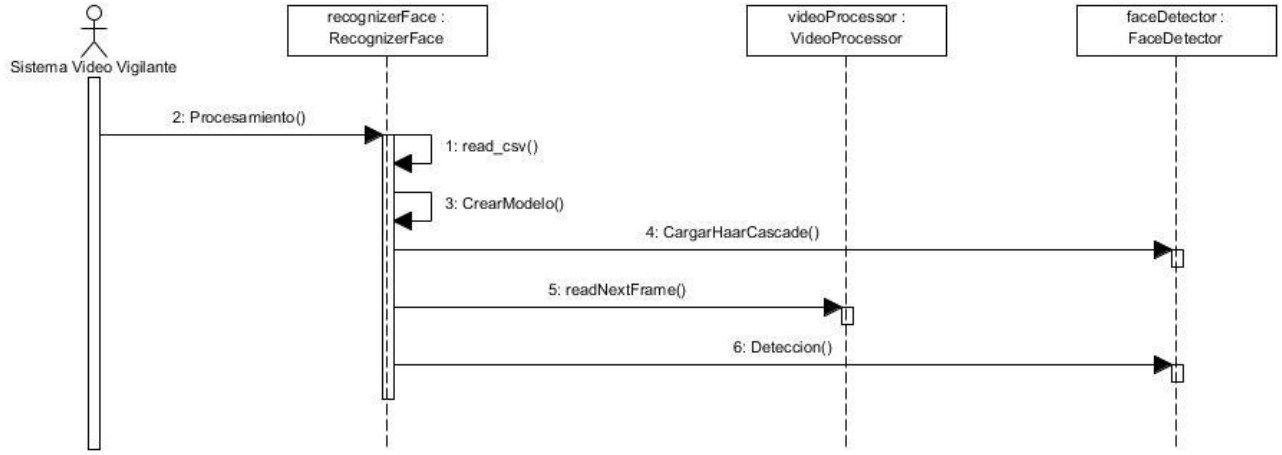

Figura 14: Diagrama de secuencia del CU Reconocer Rostro

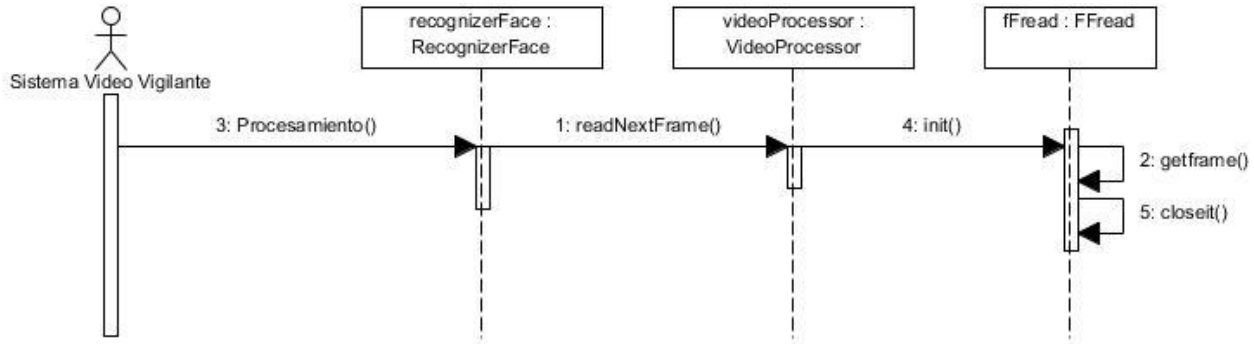

Figura 15: Diagrama de secuencia del CU Capturar Flujo de Video

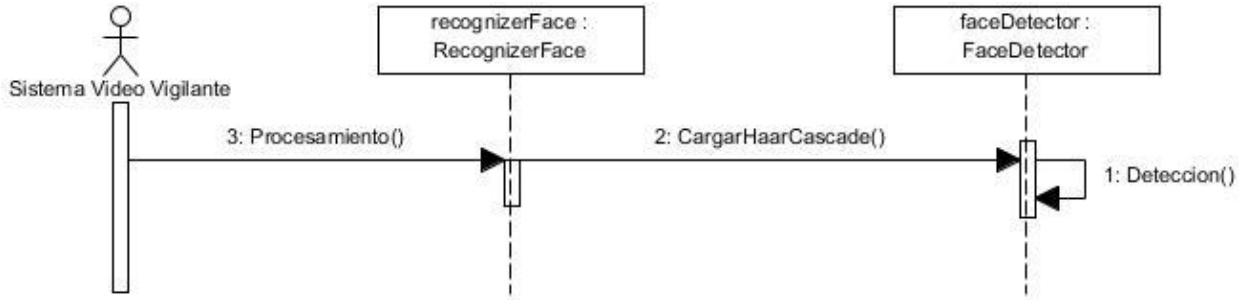

Figura 16: Diagrama de secuencia del CU Detectar Rostro

2.7.3 Descripción de las clases

Nombre	FaceDetector
Tipo de Clase	Control

Atributo	**Tipo**
face_cascade	CascadeClassifier
face_cascade_name	std::string
gray	cv::Mat

Para cada responsabilidad	
Nombre	GetGray()
Descripción	Devuelve el atributo gray que es una imagen en escala de grises
Nombre	CargarHaarCascade()
Descripción	Obtiene el HaarCascade de OpenCV para detectar rostro.
Nombre	Deteccion(cv:: Mat &input, vector<Rect_<int>>&faces)

Descripción	Detecta un rostro

Tabla 5: Descripción de la clase "FaceDetector"

Nombre	RecognizerFace
Tipo de Clase	Control

Atributo	**Tipo**
imagenes	std::vector<Mat>
labelPers	std::vector<int>
DetectorObject	FaceDetector
direccion	std::string
names	std::vector<string>
labels	std::vector<int>
recog_file	std::string
algoritmo	int
model	Ptr<FaceRecognizer>

Para cada responsabilidad	
Nombre	Set_Imagenes(std::vector<Mat>imag)
Descripción	Cambia el valor del atributo imagenes
Nombre	Set_Labels(std::vector<int> lab)

Descripción	Cambia el valor del atributo labelPers
Nombre	read_csv(const string &filename, std::vector<Mat>&images, std::vector<int>&labels, char separator=';')
Descripción	Lee un archivo txt y extrae los nombres y labels del mismo
Nombre	Procesamiento(VideoProcessorProcesObject);
Descripción	Reconoce rostros
Nombre	SetDireccion(stringdir)
Descripción	Cambia el valor del atributo direccion
Nombre	ComponerID(string path, int label)
Descripción	Compone un ID a partir de un atributo string
Nombre	SetRecog_file(string file)
Descripción	Cambia el valor del atributo recog_file
Nombre	SetAlgoritmo(intalg）
Descripción	Cambia el valor del atributo algoritmo
Nombre	CrearModelo()
Descripción	Crea un modelo del algoritmo de reconocimiento a utilizar

Tabla 6: Descripción de la clase "RecognizerFace"

Nombre	VideoProcessor

Tipo de Clase	Control
Atributo	**Tipo**
ffCapture	FFread
images	std::vector<std::string>
const_iteratoritImg	std::vector<std::string>::
fnumber	long
Para cada responsabilidad	
Nombre	readNextFrame()
Descripción	Devuelve un frame del video
Nombre	getCaptureObj()
Descripción	Devuelve el atributo ffCapture
Nombre	setInput(QStringfilename)
Descripción	Crea un arreglo de imágenes pasándole una dirección por parámetro
Nombre	setInput(conststd::vector<std::string>&imgs)
Descripción	Crea un arreglo de imágenes pasándole un vector de imágenes por parámetro.

Tabla 7: Descripción de la clase "VideoProcessor"

Nombre	FFread
Tipo de Clase	Control

Atributo	**Tipo**
*pFormatCtx	AVFormatContext
i	int
videoStream	int
*pCodecCtx	AVCodecContext
*pCodec	AVCodec
*pFrame	AVFrame
*pFrameRGB	AVFrame
pict	AVPicture
packet	AVPacket
fr	AVRational
frameFinished	int
numBytes	int
*buffer	uint8_t
fskip	int
m	int
n	int
p	int

*ptr1	unsignedchar
*ptr2	float
fps	float
nCols	int
nRows	int
nBands	int
nframes	int
opened	bool
movie_end	int
*img_convert_ctx	structSwsContext
*opt	AVDictionary
Para cada responsabilidad	
Nombre	init(char *)
Descripción	Inicia la captura
Nombre	Mat getframe()
Descripción	Devuelve un frame del video
Nombre	voidcloseit()
Descripción	Cierra la captura

Tabla 8: Descripción de la clase "FFread"

2.8 Conclusiones Parciales

En este capítulo se representó el sistema, definiendo el comportamiento del mismo a partir de los requisitos identificados. La visión del DCUS permitió de forma clara y sencilla mostrar los principales procesos a automatizar.

Para la obtención de la solución se realizó un análisis de los artefactos generados por cada flujo de trabajo de RUP. Cada artefacto sirvió de entrada al siguiente flujo facilitando la comprensión de la situación, acercándose cada vez más a la solución final. Los diagramas realizados permitieron dar una visión clara para la etapa posterior de implementación, de la dinámica de los procesos dentro de cada CU identificado a partir de los RF reconocidos anteriormente. Es importante destacar el uso de patrones, que garantiza la calidad del código fuente, puesto que al ser soluciones a problemas recurrentes, permiten probar el correcto funcionamiento de cada componente, clase o método implementado.

Capítulo 3: Algoritmos

3.1 Introducción

En este capítulo se explica en detalle los algoritmos utilizados para la implementación del video sensor para detección y reconocimiento de rostros. En la etapa de detección se utiliza Viola and Jones únicamente, recurriendo a los *Haar Cascade* de OpenCV. Para el reconocimiento se utilizaron 3 algoritmos (Fisherfaces, Eigenfaces, LBPH) de los cuales se ejecuta solo uno, pero es tarea del usuario el elegirlo.

3.2 Viola and Jones

El detector de objeto utilizado para la detección del rostro, fue propuesto inicialmente por Paul Viola (Jones, 2001) y mejorado por Rainer Lienhart (Maydt, septiembre 2002). OpenCV refiere a este detector como “Clasificador en Cascada” y viene con un conjunto de archivos pre-entrenados para la detección.

Un clasificador es entrenado con cientos de vistas de ejemplos de un objeto en particular (por ejemplo rostros, carros, aviones) llamados ejemplos positivos, son escalados al mismo tamaño y los ejemplos negativos, que son imágenes arbitrarias del mismo tamaño. Después de ser entrenado un clasificador, este puede ser aplicado a una región de interés de una imagen de entrada. El clasificador retorna 1 si la región es como el objeto que se quiere reconocer, si no retorna 0.

El clasificador se dice es en cascada ya que el resultado consiste en varios clasificadores simples que son aplicados subsecuentemente a una región de interés, hasta que en algún estado el candidato es rechazado o todos los estados son pasados. El clasificador es “*boosted*” ya que cada estado es complejo y son construidos a partir de clasificadores básicos usando uno de cuatro diferentes técnicas de *boosting* (*Discrete Adaboost, Real Adaboost, Gentle Adaboost y Logitboost*). Los clasificadores básicos son basados en árboles de decisión con al menos dos salidas. Las características basadas en *Haar* son la entrada a los clasificadores básicos y son calculadas como se describe a continuación.

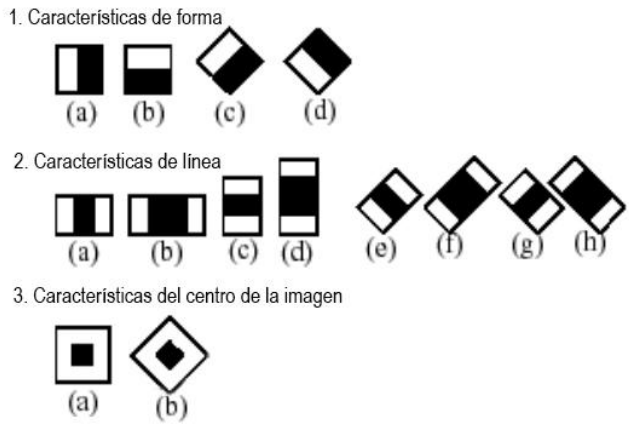

Figura 17: Características de Haar que utiliza el agoritmo

Las características usadas en un clasificador particular se especifican por su forma (1a, 2b, etc), posición dentro de la región de interés y la escala. Por ejemplo en el caso de la tercera característica de línea (2c) la respuesta es calculada por la diferencia entre la suma de los píxeles de la imagen según el rectángulo que cubre toda la característica (incluyendo las dos líneas blancas y la negra en el medio) y la suma de los píxeles de la imagen dentro de la línea negra multiplicada por tres para compensar la diferencia de áreas.

3.3 Fisherfaces

Para el reconocimiento una de las técnicas más empleadas es Fisherfaces, basada en LDA (*Linear Discriminant Analysis*). Este algoritmo realiza una reducción de dimensionalidad específica de clase y fue inventado por el gran estadista Sir R. A. Fisher. Con el fin de encontrar la combinación de características que mejor separa entre las clases, el análisis discriminante lineal maximiza el radio de la dispersión entre clases y dentro de las clases en vez de maximizar la dispersión total. La idea es simple: las clases semejantes se deben agrupar, mientras que las diferentes deben estar lo más lejos posible una de otra.

Descripción del algoritmo:

Sea X un vector aleatorio con las muestras tomadas de c clases:

$$X = \{X_1, X_{2,\ldots,}X_c\}$$

$$X_i = \{X_1, X_{2\ldots}, X_n\}$$

Las matrices de dispersión S_B y S_W son calculadas:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)\ (\mu_i - \mu)^T$$

$$S_W = \sum_{i=1}^{c} \sum_{X_j \in X_i} (X_j - \mu_i)(X_j - \mu_i)^T$$

Donde μ es la media total:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

El algoritmo clásico de Fisher ahora busca una proyección W, que maximiza el criterio de separabilidad de clase:

$$W_{opt} = arg\ max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

Según (Belhumeur, 1997), una solución para este problema de optimización está dada por la solución del *General Eigenvalue Problem*:

$$S_B v_i = \lambda_i S_W v_i$$

$$S_{W^{-1}} S_B v_i = \lambda_i v_i$$

Hay un problema pendiente de solución: El rango de S_W es como máximo $(N - c)$, con N muestras y c clases. En los problemas de reconocimiento de patrones el número de muestras N es casi siempre más pequeño que la dimensión de los datos de entrada (número de píxeles) por lo que la matriz de dispersión

S_W se convierte en singular (ver (Jain, 1991)). En el trabajo de (Belhumeur, 1997) esto fue resuelto mediante la realización de un análisis de componentes principales sobre los datos y la proyección de las muestras a un espacio de $(N - c)$ dimensiones.

El problema de optimización se puede reescribir como:

$$W_{pca} = arg\, max_W\, |W^T S_T W|$$

$$W_{fld} = arg\, max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

La matriz de transformación W, que proyecta una muestra en el espacio de $(c - 1)$ dimensión está entonces dada por:

$$W = W_{fld}^T W_{pca}^T$$

3.4 Eigenfaces

El problema con la representación de la imagen es su alta dimensionalidad. Una imagen en escala de grises de dos dimensiones $p\ x\ q$ llega a tener un espacio $m = pq$, por lo que una imagen de $100x100$ píxeles llega a ser un espacio dimensional de 10000 elementos. La pregunta es si todas esas dimensiones son de utilidad. Se puede tomar una decisión si hay alguna variación en la información, entonces lo que se está buscando son los componentes que brindan la mayor información.

Una técnica utilizada es Eigenfaces, basado en PCA (*The Principal Component Analysis*). Fue propuesta independientemente por Karl Pearson y Harold Hotelling, para convertir un conjunto de posibles variables correlacionadas en un conjunto menor de variables no correlacionadas. La idea es, que un conjunto de datos de alta dimensión a menudo es descrito por variables correlacionadas y por lo tanto sólo unas pocas dimensiones significativas representan la mayor parte de la información. El método PCA encuentra las direcciones con la mayor varianza en los datos, llamado componentes principales.

Descripción del algoritmo:

Sea $X = \{x_1, x_2, x_n\}$ un vector aleatorio con observaciones $x_i \in R^d$

Calcular la media μ

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Calcular la matriz de covarianza S

$$S = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T$$

Calcular los valores propios (*eigenvalues*) λ_i y vectores propios (*eigenvectors)* v_i de S

$$S_{vi} = \lambda_i v_i, i = 1, 2, \ldots, n$$

Ordenar los vectores propios descendientes por su valor propio. Los principales componentes k son los vectores propios correspondientes a los mayores k valores propios.

Los principales componentes k del vector X observados se dan entonces por:

$$y = W^T(x - \mu)$$

donde: $W = (v_1, v_2, \ldots, v_k)$

La reconstrucción desde PCA está dada por:

$$x = w_y + \mu$$

donde: $W = (v_1, v_2, \ldots, v_k)$

El método Eigenfaces hace el reconocimiento de rostros:

- Proyectando todos los datos de entrenamiento al subespacio PCA.
- Proyectando la imagen de macheo al subespacio PCA.
- Encontrando el vecino más cercano entre los datos de entrenamiento y la imagen de macheo.

3.5 Local Binary Patterns Histograms

Algunas investigaciones se concentran en extraer características locales de las imágenes, la idea es no mirar a la imagen entera como un vector de grandes dimensiones, pero describe solo características locales de un objeto. Las características que se extraen de esta manera van a tener implícita una baja dimensionalidad. Es una buena idea, pero se observará que la representación de la imagen no sufre solo de variaciones de la iluminación. Hay que pensar en elementos como el escalado, movimiento o rotación en las imágenes, el descriptor local debe ser al menos un poco robusto ante estas situaciones. La metodología de los Patrones Binarios Locales (*Local Binary Patterns*-LBP) tiene sus raíces en análisis de textura en dos dimensiones. La idea básica de LBP es de resumir la estructura local en una imagen comparando cada píxel con su vecindad. Toma un píxel como centro y umbraliza a sus vecinos. Si la intensidad del píxel central es mayor o igual que su vecino, entonces lo denota con 1 y 0 si no. Se obtendrá un número binario por cada píxel, como por ejemplo 11001111. Entonces con 8 píxeles de vecindad se tendrán 2^8 combinaciones posibles, llamados LBP aunque también se refiere como códigos LBP (Ver figura 18).

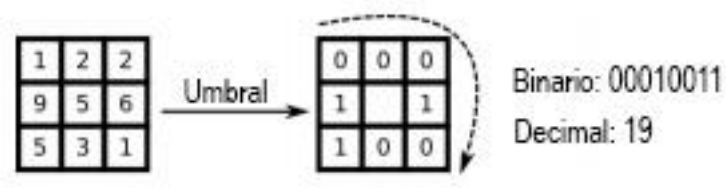

Figura 18: Ejemplo de obtención de código LBP

Descripción del algoritmo:

Una descripción más formal del operador LBP puede ser dada por:

$$LBP(x_c,y_c) = \sum_{p=0}^{P-1} 2^P \, S(i_p - i_c)$$

Con (x_c,y_c) como píxel central con intensidad i_c y i_n siendo la intensidad del píxel vecino, s es la función que se define como:

$$s(x) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & else \end{cases}$$

Esta descripción permite capturar detalladamente las características de la imagen. De hecho, los autores fueron habilitados para competir con los resultados del estado del arte para la clasificación de la textura. Poco después de que el operador se publicó, se denotó, que una vecindad fija falla al codificar detalles que difieran en escala. Así que el operador se extendió para usar una vecindad variable en (Ahonen, 2004). La idea es alinear un número arbitrario de vecinos en un círculo con un radio variable.

Para un punto dado (x_c,y_c) la posición del vecino (x_p,y_p), $p \in P$ puede ser calculada por:

$$x_p = x_c + R\ cos\left(\frac{2\pi p}{p}\right)$$

$$y_p = y_c - R\ sin\left(\frac{2\pi p}{p}\right)$$

Donde R es el radio del círculo y P es el número de los puntos de ejemplo.

El operador es una extensión de los códigos LBP originales, también es llamado LBP Extendido (También se refiere al LBP Circular). Si las coordenadas de un punto en un círculo no se corresponden con las coordenadas de la imagen, el punto se interpola. La ciencia de la computación tiene una rama de la interpolación inteligente, la implementación de OpenCV hace una interpolación bilineal:

$$f(x) \approx [1-x\ \ x]\begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix}\begin{bmatrix} 1-y \\ y \end{bmatrix}$$

Por definición, el operador LBP es robusto contra transformaciones en escala de grises monotónicas. Es fácil verificar esto tomando la imagen LBP de una imagen modificada artificialmente y se puede observar como luce la imagen (Ver figura 19).

Figura 19: Ejemplo de imagen LBP

Entonces, lo que queda por hacer es incorporar la información espacial en el modelo de reconocimiento de rostros. La representación propuesta por (Ahonen, 2004) es dividir la imagen LBP en m regiones locales y extraer un histograma de cada uno. El vector de características espacialmente mejorado se obtiene entonces mediante la concatenación de los histogramas locales (no uniéndolos). Estos histogramas se denominan Histogramas Patrones Binarios Locales.

3.6 Conclusiones Parciales

Luego de analizados los algoritmos utilizados para la detección y reconocimiento de rostros, se puede concluir que la biblioteca OpenCV facilita grandemente el trabajo con los mismos. La utilización de estos mediante la implementación que trae esta reduce la complejidad del código así como el tiempo de implementación. Estos son algoritmos muy utilizados actualmente ya que su efectividad ha sido probada en diferentes investigaciones, por lo que es abundante la bibliografía sobre los mismos. La popularidad y efectividad de dichos algoritmos, acompañado de las facilidades que brinda OpenCV son factores que favorecen la selección de estos sobre otros existentes.

Capítulo 4: Implementación y pruebas

4.1 Introducción

En este capítulo los elementos del modelo del diseño se implementan en términos de componentes. Además se realizan pruebas al sistema para lograr erradicar errores que puedan ser introducidos en la implementación del video sensor, y para comprobar que el producto final cumple con los requisitos establecidos en la primera fase de la investigación. Para lograrlo, se le realizan pruebas de eficiencia a la aplicación con el fin de comprobar el por ciento de efectividad de los algoritmos utilizados en el reconocimiento de rostros a partir de flujos de video obtenidos de cámaras IP en el sistema SURIA.

4.2 Diagrama de Componentes

Un componente es una pieza de software que describe un conjunto de servicios que son usados sólo a través de interfaces bien definidas. El diagrama de componentes describe las interacciones existentes entre los principales componentes que integran la solución, así como sus dependencias más significativas.

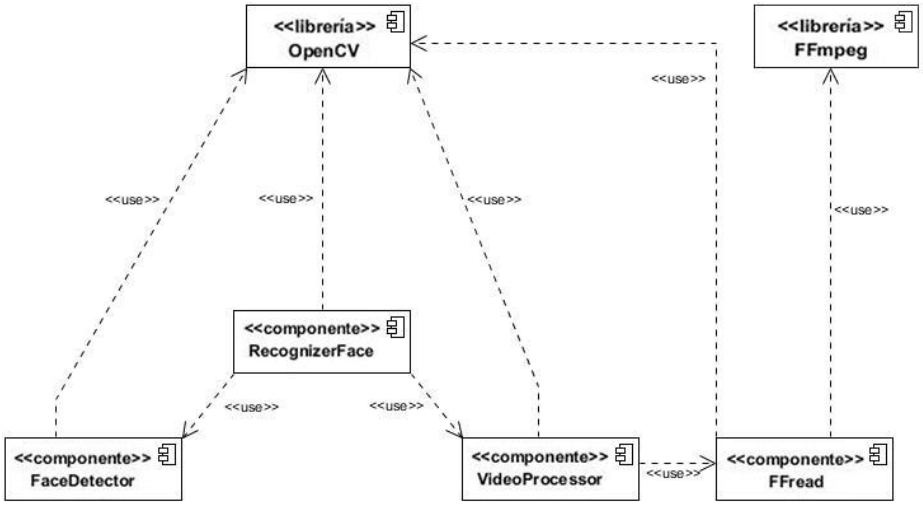

Figura 20: Diagrama de Componentes

El sistema cuenta con los componentes: FFMpeg, OpenCV, RecognizerFace, FaceDetector, VideoProcessor y FFread que se encargan de distribuir las funcionalidades necesarias para la realización del video sensor.

FFMpeg: Esta biblioteca se utiliza para el trabajo con video, específicamente para su captura.

OpenCV: Esta biblioteca se utiliza para el procesamiento de imágenes digitales.

RecognizerFace, FaceDetector, VideoProcessor y FFread: Tienen como objetivo brindar todas las funcionalidades del componente.

4.3 Diagrama de Despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Para el video sensor propuesto se cuenta con dos nodos principales: una computadora en la que se estará ejecutando la aplicación y un dispositivo, en este caso una cámara IP, capturando el flujo de video. Estos nodos se comunican mediante el Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP). Este protocolo representa todas las reglas de comunicación para Internet y se basa en la noción de dirección IP, es decir, en la idea de brindar una dirección IP a cada equipo de la red para poder enrutar paquetes de datos. (Ver Figura 21)

Figura 21: Diagrama de Despliegue

4.4 Pruebas de software

Las pruebas de software son el conjunto de técnicas que permiten determinar la calidad de un producto de software, estas se integran dentro de las diferentes fases del ciclo de vida del mismo. La calidad de un sistema de software es algo subjetivo, que depende del contexto y del objetivo que se pretenda conseguir. Para determinar dicho nivel de calidad se deben efectuar medidas o pruebas que permitan comprobar el grado de acatamiento con respecto a las especificaciones iníciales del sistema, o sea, los requisitos funcionales.

Para la ejecución de las pruebas al video sensor para detección y reconocimiento de rostros se realizarán pruebas de sistema para probar la eficacia de los algoritmos implementados en flujos de video obtenidos de cámaras IP en el sistema SURIA.

4.5 Resultados experimentales

Presicion y *Recall* son dos indicadores ampliamente utilizados para evaluar la exactitud y eficiencia de los sistemas de recuperación de información. Estos son definidos por las siguientes ecuaciones:

$$presicion\,rate = \frac{CC}{CC + Fp}$$

$$recall\,rate = \frac{CC}{CC + Fn}$$

Donde CC es el número de rostros que el sistema reconoció correctamente, Fp (Falsos Positivos) es todo lo que el sistema reconoce como rostro y no lo es y Fn (Falsos negativos) los reconocimientos que el sistema debería hacer y no los hace.

También está F que es una medida que combina $precisión$ y $recall$. Mientras mayor sea su valor, mejor se considerará el rendimiento del sistema.

$$F = 2 \times \frac{precision\,rate\ \times recall\,rate}{precision\,rate + recall\,rate}$$

Una vez obtenida F se puede calcular el porciento de error que tiene el sistema (E)

$$E = 1 - F$$

4.5.2 Resultados utilizando flujos de video en el sistema SURIA

En el sistema SURIA, para capturar las imágenes de rostros, se utilizan cámaras IP. Aquí el sujeto, mirando la cámara posa para que esta pueda grabarlo. También puede ir haciendo pequeñas rotaciones al rostro, en lo que el sistema lo identifica. Los resultados arrojados, utilizando el algoritmo Eigenfaces, se pueden observar en la siguiente tabla:

Parámetro	Valor
Conteos Correctos (CC)	897
Falsos Positivos (Fp)	40
Falsos Negativos (Fn)	247
$recall\,rate$	0.78
$presicion\,rate$	0.95
F	0.84
Error (E)	0.16

Tabla 9: Resultados utilizando flujos de video en el sistema SURIA y el algoritmo Eigenfaces

Para los algoritmos Fisherfaces y LBPH los resultados fueron los siguientes:

Parámetro	Valor
Conteos Correctos (CC)	848
Falsos Positivos (Fp)	39
Falsos Negativos (Fn)	305

$recall\ rate$	0.73
$presicion\ rate$	0.95
F	0.82
Error (E)	0.18

Tabla 10: Resultados utilizando flujos de video en el sistema SURIA y el algoritmo Fisherfaces

Parámetro	**Valor**
Conteos Correctos (CC)	885
Falsos Positivos (Fp)	40
Falsos Negativos (Fn)	265
$recall\ rate$	0.76
$presicion\ rate$	0.95
F	0.84
Error (E)	0.16

Tabla 11: Resultados utilizando flujos de video en el sistema SURIA y el algoritmo LBPH

4.6 Conclusiones Parciales

Con la realización de la implementación se demuestra que los algoritmos aplicados al procesamiento de imágenes digitales son factibles, esto se logra a través de pruebas de eficiencia para comprobar cuán eficaces son los algoritmos de reconocimiento de rostros utilizados. Se puede concluir que de los 3 algoritmos utilizados para el reconocimiento, Eigenfaces y LBPH demostraron ser los más estables en este tipo de entornos, obteniendo un 16% de error, mientras que Fisherfaces tuvo un 18%. Este porciento podría disminuir si las condiciones para la captura del flujo de video fuesen más favorables. Es decir, iluminación adecuada, sin variaciones drásticas de la pose, manteniendo mayormente pose frontal y una distancia relativa, aproximadamente 1m de la cámara al sujeto que se quiera reconocer.

Conclusiones generales

Para la realización del presente trabajo se llevó a cabo un estudio de las herramientas y tecnologías más utilizadas en los sistemas de video vigilancia actuales, así como del estado del arte de algunas aplicaciones de reconocimiento facial. En estos aspectos se identificó una alta tendencia a las tecnologías propietarias. Las aplicaciones de reconocimiento facial estudiadas fueron útiles para lograr una mayor comprensión de la problemática, sin embargo, al ser software propietario no se pudieron incluir en el presente trabajo.

La aplicación resultante de esta investigación fue estructurada siguiendo una arquitectura centrada en datos, implementando el patrón arquitectónico “Tuberías y Filtros”. Este hecho posibilitó la implementación de un componente con alto grado de reusabilidad del código. Dicho componente permite la detección y reconocimiento de rostros en un flujo de video obtenido de una cámara IP para el sistema de Video Vigilancia. El mismo cumple todos los requisitos especificados y se puede integrar a otros componentes y módulos del proyecto.

La biblioteca OpenCV facilitó la utilización de los algoritmos de detección y reconocimiento debido a las funcionalidades que posee para el trabajo con rostro. Usar esa biblioteca permitió reducir la complejidad del código así como el tiempo de implementación. Para detección se seleccionó Viola and Jones, mientras que para reconocimiento los algoritmos Fisherfaces, Eigenfaces y LBPH, dándole la posibilidad al usuario, que en este caso sería el módulo Analytics del sistema SURIA, de seleccionar el algoritmo que desea utilizar.

La aplicación fue probada utilizando flujos de video obtenidos desde cámaras IP en el sistema SURIA con el fin de comprobar eficiencia de los algoritmos de reconocimiento. Los resultados arrojados fueron positivos, logrando un gran porciento de aciertos por parte del sistema. Los algoritmos Fisherfaces y LBPH mostraron un error de 16%, mientras que Fisherfaces un 18% mostrando que el objetivo de la investigación fue cumplido.

Recomendaciones

Durante el desarrollo del sistema han surgido ideas, que aportarían al video sensor mayor robustez, por lo que se recomienda:

- Que las imágenes de entrenamiento del sistema sean tomadas en un ambiente controlado, con pose frontal, iluminación poco variable, imágenes nítidas, con calidad y un fondo conocido.
- Que el componente siga siendo probado en condiciones variables ambientales, pudiera ser en exteriores, interiores, con variación de pose y de ser posible en bases de datos internacionales para el reconocimiento de rostros.
- Para lograr un mayor porciento de aciertos de la aplicación y disminuir el porciento de error, las cámaras que utilizará el sistema deben ser ubicadas en locales iluminados y evitar el movimiento del rostro que se quiere reconocer, una posición frontal es la más recomendada.

Referencias Bibliográficas

Banny Solano. Banny's WebBlog. [En línea] 9 de febrero de 2010. http://bannysolano.wordpress.com/2010/02/09/rup-en-espanol/.

Francisco Martín Rico, Antonio Guzmán Sacristán, Enrique Cabello Pardos. *Video-Sensor distribuido basado en CORBA para el reconocimiento de caras.* Móstoles, España : Universidad Rey Juan Carlos.

Maydt, Rainer Lienhart and Jochen. *An Extended Set of Haar-like Features for Rapid Object Detection.* s.l. : IEEE ICIP, septiembre 2002. Vol. 1.

Jones, Paul Viola and Michael J. *Rapid Object Detection using a Boosted Cascade of Simple Features.* s.l. : IEEE CVPR, 2001.

Castro, Luis. About.com. [En línea] 2013. http://aprenderinternet.about.com.

L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice, 2nd Edition.* s.l. : Addison Wesley, 2003.

Pressman, Roger. *Ingenieria del Software un enfoque practico.* 2005.

Jacobson, Ivar, Booch, Grady and Rumbaugh, James. *El proceso unificado de desarrollo de software.* Madrid : Pearson educación, 2000. 84-78-29-036-2.

Alés, Nilo Tomás Díaz. *Procedimiento para el reconocimiento de rostro en sistemas de catalogación de audiovisuales.* La Habana : Universidad de las Ciencias Informáticas, Junio 2012.

Reyes, Heidi Méndez Vázquez y Edel García. *Estado actual de los métodos de reconocimiento automático de rostros basados en la apariencia local.* Ciudad Habana : CENATAV, 2008. 2072-6887.

Belhumeur, P. N., Hespanha, J., and Kriegman, D. *Eigenfaces vs. Fisherfaces: Recognition Using Class.* s.l. : IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997. 711–720..

Jain, Stan Z. Li and Anil K. *Handbook of facerecognition.* 2005.

Marcel, Sebastien. *A tutorial onfacerecognition. Switzerland : IDIAP.* June 2007.

W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. *Face recognition: A literature survey.* s.l. : ACM, 2003.

MARTA LUCÍA GUEVARA, JULIAN DAVID ECHEVERRY, WILLIAM ARDILA URUEÑA. *Faces Detection in Digital Images Using Cascade Classifiers.* s.l. : Universidad Tecnológica de Pereira, Junio de 2008. ISSN 0122-1701.

Jain, Raudys and A.K. *Small sample size effects in statistical pattern recognition: Recommendations for practitioneers.* s.l. : IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991. 252-264.

Ahonen, T., Hadid, A., and Pietikainen, M. *Face Recognition with Local Binary Patterns.* s.l. : Computer Vision- ECCV, 2004. 469–481.

lossurveys de M-H. Yang, D.J Kriegman, and N. Ahuja. *Detecting faces inimages: A survey.* s.l. : IEEE, January 2002.

M. Delbracio C. Aguerrebere, G. Capdehourat and M. Mateu. *reconocimiento automático de caras.* Uruguay : Proyecto Aguará, 2006. s.n.

Vázquez, Heydi Méndez. *Algoritmo de reconocimiento de rostros basado en la apariencia local para aplicaciones reales en condiciones variables de iluminación.* La Habana : CENATAV, enero 2012.

Torres, Alejandro Domínguez. *Procesamiento Digital de Imágenes.* México D.F. : Universidad Nacional Autónoma de México, 1996.

Emami, Shervin. [En línea] 2 de junio de 2010. [Citado el: 20 de 11 de 2012.] http://www.shervinemami.info/faceRecognition.html.

Morán, Iker. Que sabes de. [En línea] 2001-2013. http://quesabesde.com.

Woods, Rafael C González and Richard E. *Digital Image Processing.* 1992.

Vera, M. del Mar Sánchez. DISEÑO Y EVALUACIÓN DE MATERIALES EDUCATIVOS. [En línea] http://miuras.inf.um.es.

Conferencia_Tema_3_El_diseño_metodológico_de_la_investigación_científica. **Universidad de las Ciencias Informáticas.** La Habana : s.n., 2010.

Colomer, Antonio Albiol. *Seguimiento de objetos en secuencias de video.* s.l. : Valencia : s.n., 2003.

Beltrán Mesias, Cármen de las Mercedes. *Sistema de vigilancia en el tiempo real mediante cámaras IP, para el control de seguridad del Servicio Ecuatoriano de Capacitación Profesional - Centro de Formación Industrial Ambato.* febrero 2002.

Bibliografía Consultada

Cooper, James W. *The Design Patterns Java Companion.* 1998.

INTERSHARE, S.L. softonic. [En línea] 2003. http://ffmpeg.softonic.com/.

John Makhoul, Francis Kubala, Richard Schwartz, Ralph Weischedel. *PERFORMANCE MEASURES FOR INFORMATION EXTRACTION.* Cambridge : BBN Technologies, GTE Corp. MA 02138.

Gary Bradski, Adrian Kaehler. *Learning OpeCV.* Gravenstein Highway North, Sebastopol : O'Reilly Media, 2008. ISBN: 978-0-596-51613-0.

Low, Erik Hjelmas and Boon Kee. *Facedetection: A survey.ComputerVision and ImageUnderstanding.* 2001.

CORBA: Common Object Request Broker Arquitecture. **Saldivia, César Guerrero.** CC52N - 99/2.

Gamma, Erich, Helm, Richard and Johnson, Ralph. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995.

Documentación OpenCV. *The OpenCV Reference Manual.* November, 2012.

Cristiá, Maximiliano. *Catálogo Incompleto de Estilos Arquitectónicos.* s.l. : Facultad de Ciencias Exactas, Ingeniería y Agrimensura. Universidad Nacional de Rosario, 2006.

LIBRERÍA DE SEGMENTACIÓN DE ESTRUCTURAS EN IMÁGENES MÉDICAS DIGITALES BIDIMENSIONALES Y TRIDIMENSIONALES. **Alejandro Luis Ortega Díaz, Luis Carlos González Bárcenas, Filiberto López Palenzuela.** SLD075, La Habana : IX Congreso Internacional de Informática en Salud 2013, 2013.

Xiaoyang Tan, SongcanChen , Zhi-HuaZhou , Fuyan Zhang. *Facerecognitionfrom a single image per person: A survey.* 2006.

Ariel, M. A. *Detección y reconocimiento de caras.* 2011.

digia. Qt Creator IDE and Tools. [En línea] 20013. http://qt.digia.com/Product/Developer-Tools.

Phillips, H. Moon and P.J. *Analysis of pca-based face recognition algorithms.* Santa Barbara, California : s.n., June 1998.

James Short, Josef Kittler and Kieron Messer. *A Comparison of Photometric NormalisationAlgorithms for Face Verification.* s.l. : IEEE, 2004.

II SIMPOSIO PERUANO DE COMPUTACIÓN GRÁFICA Y PROCESAMIENTO DE IMÁGENES. *Reconocimiento de Rostros mediante Puntos Característicos Locales.* 2008.

GaryBradski. OpenCV. [En línea] http://opencv.willowgarage.com/wiki.

Jiménez, Carmen Virginia Gámez. *Diseño y Desarrollo.* Madrid : Universidad Carlos III de Madrid, 2009.

FERNANDO, C. S. M. *Tutorial Viola-Jones.* 2009.

IBM Corporation. *Rational Unified Process.* s.l. : IBM, 2007.

Company Headquarters. Visual Paradigm. [En línea] http://www.visual-paradigm.com.

Quiñones, Ernesto A. *INTRODUCCIÓN A POSTGRESQL.* s.l. : APESOL (Asociación Peruana de Software Libre).

Aldana, Ing. Edmis Deivis Semanat. *Sistema de Video Vigilancia.* 2009.

Laganière, Robert. *OpenCV 2 Computer Vision Application Programming Cookbook.* s.l. : Packt Publishing Ltd, 2011. ISBN 978-1-849513-24-1.

AmericaTI EIRL . *Ventajas y Desventajas: Comparación de los Lenguajes C, C++ y Java.* s.l. : AmericaTI.com, 2006.

Orallo, Enrique Hernández. *El Lenguaje Unificado de Modelado (UML).*

System España. *Enterprise Architect.* [En línea] Sparx Systems. http://www.sparxsystems.es.

INSTITUTO NACIONAL DE ESTADISTICA E INFORMATICA. *Herramientas CASE.* s.l. : Talleres de la Oficina de Impresiones de la Oficina Técnica de Difusión Estadística y Tecnología Informática del Instituto Nacional de Estadística e Informática (INEI), 1999. 875-99-OI-OTDETI-INEI.

Isaías Carrillo Pérez, Rodrigo Pérez González, Aureliano David Rodríguez Martín. *Metodologia de Desarrollo del Software.* 2008.

Larman, Craig. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Prentice Hall. 2da Edición.

Juan Arturo Barreto. Sistemas Biométricos. [En línea] 23 de enero de 2011. http://sistemasbiometricos26012011.blogspot.com/2011/01/reconocimiento-de-rostro.html.

Glosario

Componente: Es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos.

Frame: Imágenes que componen un video.

IDE: Es el acrónimo de los vocablos en inglés: **I**ntegrated **D**evelopment **E**nvironment, que equivalen en español a: Entorno de Desarrollo Integrado. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Método Holístico: Es la forma integral de observar un acontecimiento fenómeno o situación, enfoque holístico o globalizados donde una mismo fenómeno se observa y evalúa desde diversos parámetros.

OpenCV: Biblioteca abierta desarrollado por Intel para el procesamiento inteligente de imágenes y video.

Redes Neuronales: Son modelos bastante simplificados de las redes de neuronas que forman el cerebro, y al igual que este, intentan "aprender" a partir de los datos que se le suministran.

Sensor: Algoritmos que apoyan a los guardias de seguridad durante la vigilancia.

Sistema de Vigilancia: Grupo de redes a circuito cerrado controlados por un vigilante.

Umbral: Valor mínimo o máximo (establecido por un atributo característica o parámetro) el cual se usa como una cota de comparación o guía.