Universidad de las Ciencias Informáticas

Facultad 6

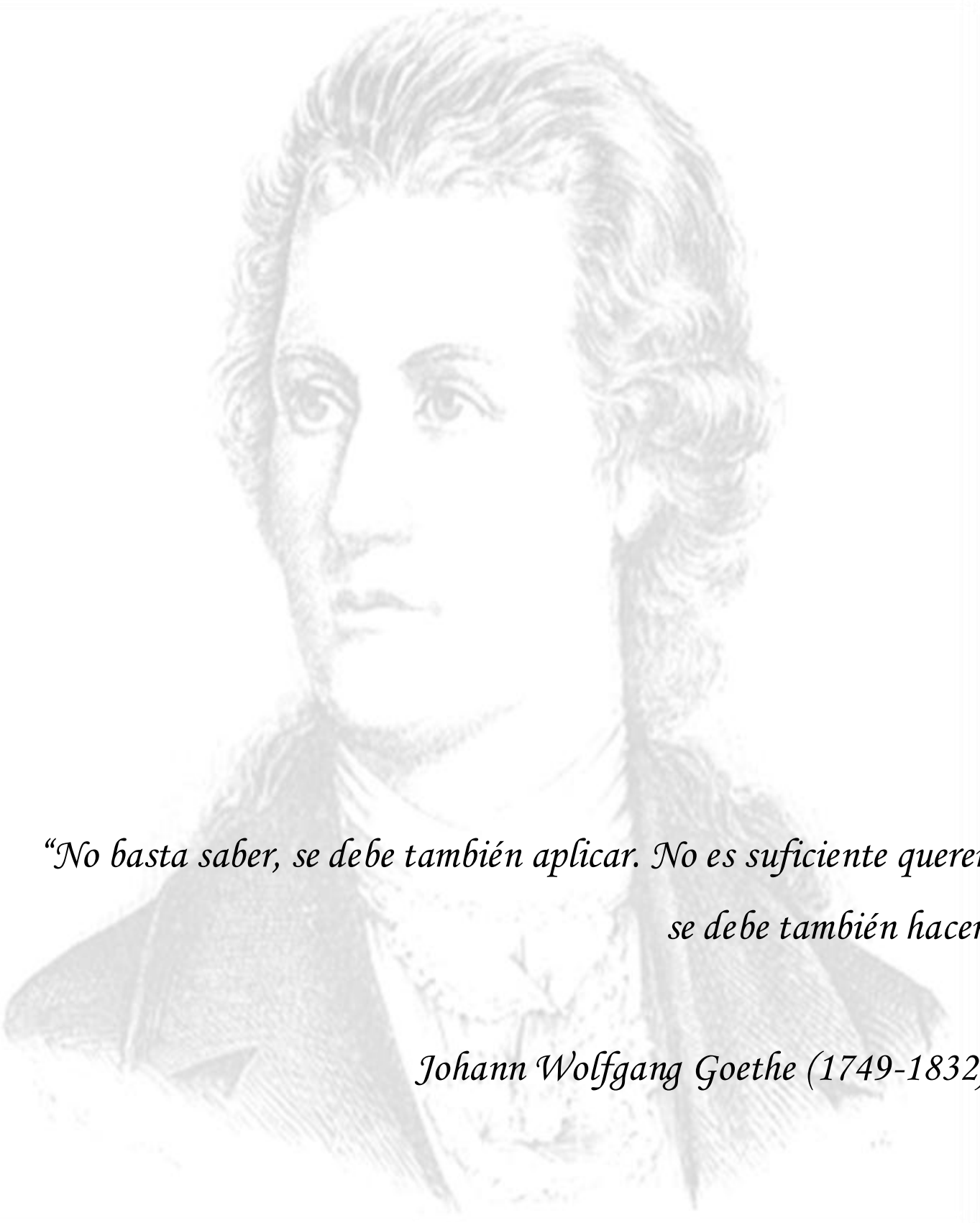
Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Desarrollo de un video sensor para el conteo de personas.

Autor: **Lisandra Michelena Rodríguez.**

Tutor: **Ing. Reinier Pupo Ruiz.**

La Habana, Cuba.
Año del 53 Aniversario del Triunfo de la Revolución.
Junio de 2011.



"No basta saber, se debe también aplicar. No es suficiente querer, se debe también hacer"

Johann Wolfgang Goethe (1749-1832)

DEDICATORIA.

Dedico este trabajo a mi padre, a mi madre, a mi hermano, a mis tíos, a mis abuelos, en fin a toda mi familia, la cual me ha apoyado en todo momento de mi vida y me ha alentado a seguir estudiando para culminar esta etapa de tantos años de estudio. Pero más que nada se la dedico a mi abuela Mamima, ese ser especial que aunque ya no esté conmigo me cuida desde el cielo.

AGRADECIMIENTOS.

Quiero agradecer primeramente a mi familia que es la razón de mi vida, especialmente mi hermano Santiago: mi niño lindo.

A mi padres Santiago y Jorge que siempre me enseñaron el valor del sacrificio y el trabajo.

A mis madres Nancy y Clara, que me enseñaron la perseverancia y a luchar por lo que se quiere.

A mis abuelos Blanca y Ángel que siempre me dieron ánimos para seguir adelante.

A mi Cosi (Ale) que me ayudó, me apoyó en todo momento y siempre estuvo ahí para mí.

A mi tutor Reinier Pupo un GRACIAS especial, pues sin él no estuviera aquí hoy.

A Yuliet Expósito que fue como una compañera de tesis para mí.

A mis viejitos lindos de la UCI Massiel y Victor que me ayudaron todo el tiempo.

A mi amigo incondicional Alejandro Orgelio Hernández por ayudarme y guiarme estos 5 años aquí en la UCI.

A todas las personas del proyecto Video Vigilancia Suria que me brindaron su apoyo y ayuda.

A mis amigas y amigos que siempre estaban preocupados por mí: Jany, Mónica, Saili, Yanet Gómez, Yamilé y todos los demás que aunque no los mencione, no dejan de ser importantes para mí.

Gracias por ayudarme a llegar hasta aquí.

DECLARATORIA DE AUTORÍA.

Declaro ser autor de la presente tesis y autorizo a la Universidad de las Ciencias Informáticas (UCI) y al Centro de Video Vigilancia Suria a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2011.

Lisandra Michelena Rodríguez.

_____

Firma del Autor.

Ing. Reinier Pupo Ruiz.

_____

Firma del Tutor

DATOS DE CONTACTO.

TUTOR:

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.

e-mail: rpupo@uci.cu

RESUMEN

En estos momentos Cuba se encuentra inmersa en la informatización de sectores que son el pilar fundamental de su economía y sociedad. Para ello, el gobierno cubano ha vinculado un sin número de empresas y universidades al desarrollo de soluciones informáticas que son necesarios para llevar a cabo tan importante labor. La Universidad de Ciencias Informáticas es uno de los centros vinculados directamente a la producción de software y en ella se encuentran grupos de proyectos que trabajan con el objetivo de dar solución a problemas reales existentes dentro y fuera del país. Uno de estos proyectos es el Proyecto de 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.

Este Trabajo de Diploma ha sido elaborado con el fin de ampliar prestaciones al sistema de vigilancia SURIA agregando un componente video sensor que sea capaz de realizar conteos de personas a través de cámaras IP. Para ello se plantea la construcción del video sensor utilizando la metodología RUP (Rational Unified Process), se lleva a cabo un estudio acerca de sistemas similares existentes en el mercado actual y se plantean las herramientas necesarias para su construcción.

La puesta en práctica de este trabajo trae como beneficio económico el aporte de una nueva funcionalidad al sistema SURIA, de manera que este pueda competir con otros sistemas existentes en el mercado.

Palabras claves: Cámaras IP, conteos de personas, imágenes, procesamiento, video digital, video sensor.

INDICE

Índice de Figuras.

Índice de Tablas.

INTRODUCCIÓN

Con el fin de proteger sus intereses, recursos y a sí mismo, el hombre, ya desde la antigüedad estudiaba la manera de establecer mecanismos de seguridad. Al pasar el tiempo, y debido a la evolución tecnológica, este comenzó a sentir la necesidad de reducir la aplicación de recursos humanos en la vigilancia y protección, y llenar el vacío existente a causa de esta reducción con dispositivos electrónicos cuyas funcionalidades superaban ampliamente las capacidades de la mente y la visión humana. Hoy en día existen mecanismos altamente sofisticados, entre los que se encuentran las tarjetas de seguridad, los scanners biométricos, las alarmas electrónicas y las cámaras de video vigilancia, garantizándose de esta manera una adecuada protección de empresas, instituciones, hogares, bancos, estaciones y aeropuertos.

Desde sus inicios los sistemas de video vigilancia han evolucionado considerablemente. En la década de 1970 se empleaban videograbadores, que tenían como soporte los casetes, estos sistemas poseían una autonomía de sólo un par de horas y además, requerían grandes espacios organizados para almacenar las cintas de respaldo de seguridad. Como consecuencia de lo antes mencionado era necesario tener una especie de biblioteca, lo que hacía muy compleja la búsqueda. Se requería de una persona para que se dedicara a observar las cámaras y mantener los equipos encendidos, y de otra para encontrar imágenes almacenadas, teniendo que revisarlas de forma completa.

Los sistemas actuales, por el contrario, obtienen la información en tiempo real desde cámaras IP a través de redes de alta velocidad. Dichos sistemas cuentan con varias partes fundamentales, como 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."* (Albiol Colomer,Antonio , 2003)

Algunos usos de los videos sensores son: video sensor para detección de movimiento, video sensor para detección de rostros, video sensor para conteo de personas, video sensor para extracción de matrículas en vehículos entre otros, estos utilizan el reconocimiento de patrones y el procesamiento de imágenes para brindar un producto de alto valor agregado.

En la Facultad 6 de la Universidad de las Ciencias Informáticas se encuentra el Centro de Geoinformática y Señales Digitales (GEYSED), donde se desarrolla un sistema de Video Vigilancia que cuenta con cuatro módulos: Recuperador, Grabador, Visor y el módulo central, llamado Gestor. Este sistema brinda funcionalidades de gestión de la información, recuperación, grabación y visualización en tiempo real de los flujos de videos obtenidos desde cámaras IP. Para él se desarrollan otras funcionalidades que le

aportan inteligencia al sistema, ya que se procesan los flujos de videos en tiempo real para mejorar la vigilancia.

La **situación problemática** planteada será**:** Debido a que los guardias de seguridad son incapaces de llevar el control exacto de lo que ocurre en varias cámaras a la vez, el proyecto Video Vigilancia SURIA necesita de un mecanismo que permita el conteo de personas 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.

A partir de la situación descrita anteriormente, surge el siguiente **problema a resolver:** ¿Cómo incorporar procesamiento inteligente al sistema de video vigilancia SURIA para que permita contar personas a partir de flujos de videos obtenidos desde cámaras IP?

Se plantea como **objeto de estudio:** El procesamiento inteligente de flujos de videos y se enmarca como **campo de acción:** El conteo de personas 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 el conteo de personas 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 proveerá al mismo de la capacidad de un procesamiento inteligente de los flujos de videos que permita contar personas”.

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

1. Analizar sistemas de video sensor para el conteo de personas en la actualidad.
2. Caracterizar la metodología y herramientas de desarrollo a utilizar en la construcción del video sensor.
3. Elaborar toda la documentación asociada a la metodología de desarrollo seleccionada.
4. Implementar un componente que procese los flujos de video y cuente las personas.
5. Implementar un componente que muestre los resultados del procesamiento realizado por el componente.
6. Realizar las pruebas al componente.

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. 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 utilizará 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, además permite definir los requisitos necesarios de este proceso que incluye los subprocesos: adquirir, reprocesar, segmentar, representar, describir y reconocer imágenes digitales.

Por último se empleará el método de **Modelación** para estudiar nuevas relaciones y cualidades del objeto de estudio haciendo uso de diagramas obtenidos por medio de las herramientas de modelado y con el Lenguaje Unificado de Modelación (UML), donde se acumula la información de los artefactos generados de todas las etapas por las que pasa el software: dominio, requerimientos, diseño e implementación. Este método es el más importante en el proceso de construcción de software, pues con su aplicación se logra un mejor entendimiento del problema de estudio.

Por otra parte, entre los **Métodos Empíricos** se encuentra 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 manera que en su primera parte se definen los elementos teóricos que sustentan la investigación y se analizan soluciones similares existentes en el mundo. Además se enuncian los conceptos que posibilitan un mejor entendimiento de lo planteado en la situación problemática. Se seleccionan las tecnologías y herramientas a utilizar en el desarrollo del sistema y se fundamenta la elección del lenguaje de programación escogido, así como la Metodología de Desarrollo.
A continuación, se ofrece el modelo de dominio de la aplicación, conjuntamente con la especificación de los requerimientos funcionales y no funcionales y el modelo de casos de usos del sistema.

Posteriormente se muestran los resultados obtenidos en el desarrollo del proceso de diseño del sistema, así como los diagramas que fueron necesarios para obtener una mayor claridad a la hora de elaborar la solución que se propone.

Por último se realiza el modelo de implementación, el cual está compuesto por su respectivo diagrama de despliegue y por su diagrama de componentes. Además de realizar la validación para comprobar que el sistema cumple con los requerimientos funcionales y que presenta una óptima calidad.

CAPÍTULO 1: Fundamentación Teórica.

1.1 Introducción.

El presente capítulo es una presentación de los conceptos y aspectos teóricos asociados al dominio del problema planteado, y su objetivo es proporcionar un mejor entendimiento de los elementos presentes en la investigación. Por otra parte contiene un análisis de las principales soluciones de video sensores existentes en el mundo, cuya finalidad es el conteo de personas y finalmente se exponen, la metodología y plataforma de desarrollo de software así como la herramienta de diseño y modelado a emplear.

1.2 Conceptos asociados al dominio del problema.

Para lograr un mayor entendimiento del trabajo a continuación se reflejan conceptos asociados al dominio del problema.

Flujos de Video.

"*Un video streaming es un vídeo en la web que se reproduce al mismo tiempo que los datos llegan al ordenador a través de Internet. A diferencia del video por Internet para descargar, el video streaming comienza a reproducirse una vez que los datos comprimidos se reciben de un determinado buscador de videos, y elimina la preocupación de virus que pueden acompañar a las descargas."* (Bernal, 2010)

El streaming es la transferencia de contenido audiovisual por la red desde un servidor hacia sus clientes. Este permite aligerar la descarga y ejecución de audio y vídeo (AV), de manera que se pueden escuchar y visualizar la información contenida en los archivos en tiempo real, esto es ideal para las cámaras IP que se utilizan en sistemas de video-vigilancia, que en caso de no utilizar streaming, sería necesario descargar completamente los archivos de AV para luego visualizar su contenido dificultando el monitoreo y procesamiento inteligente en tiempo real.

Imagen digital

La imagen digital es un producto del desarrollo de la informática que tiene como antecesor a la fotografía, (que toma como punto de partida un objeto del mundo real) y a la pintura, (donde la imagen ha sido creada por un artista), en la imagen digital podemos ver incluidos los dos hechos, la originalidad de la imagen cuando es tomada por primera vez, y luego el resultado de compresiones, optimizaciones, filtrados y otros procesos que forman parte del arte digital.

Algunos especialistas como **Rafael C González y Richard E. Woods,** consideran que "*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*." (González, et al., 1992)

De acuerdo con las valoraciones de **Rafael C. González y Richard E. Woods** se llega a la conclusión que una 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. Las imágenes digitales son muy utilizadas hoy en día, hasta el punto de haber extinguido casi en su totalidad a la fotografía tradicional. Esta es aplicable en casi todos los campos, entre ellos: la fotografía, el arte, publicidad, comercio, medicina, investigación, educación y otras. Su alta aplicabilidad está dada principalmente por ser fácil de procesar, manejar, clonar y transportar. En la figura 1 se muestra un ejemplo de una imagen digital.

Figure 1 *Imagen digital.*

Video digital

"El video hace referencia a la captación, procesamiento, transmisión y reconstrucción de una secuencia de imágenes y sonidos que representan escenas en movimiento." (Fortis, 2005)

El video digital se puede definir como la representación digital de la señal de vídeo analógico, es una secuencia de imágenes o frames que son almacenadas y reproducidas a una velocidad y en un orden determinado (Figura 2). Las secuencias o videos pueden o no estar vinculadas a un fichero de audio y se aplican en diversas ramas como la medicina, arte digital, educación, control de tránsito, agencias de seguridad entre otros.

Figure 2 *Video digital.*

Cámara IP

Una cámara IP puede ser descrita como la combinación de una cámara y una computadora en una sola unidad, la cual captura y transmite imágenes en vivo a través de una red, habilitando a usuarios autorizados a ver, almacenar y administrar el video sobre una infraestructura de red estándar basada en el protocolo IP (Ver Figura 3).

Figure 3 *Cámaras IP.*

1.3 Objeto de Estudio.

El objeto de estudio de esta investigación está centrado en el procesamiento inteligente de imagen y video digital.

1.3.1 Descripción General.

Alejandro Domínguez Torres, plantea*: "Al conjunto de técnicas y procesos para descubrir o hacer resaltar información contenida en una imagen usando como herramienta principal una computadora se le conoce como Procesamiento de Imágenes Digitales (PID)."* (Torres, 1996)

El procesamiento de imágenes digitales comprende los procesos que: reciben como entrada una imagen y retornan otra imagen, reciben como entrada una imagen y retornan rasgos de objetos presentes en la imagen, reciben como entrada un conjunto de rasgos y le asocian una categoría predefinida.

Pasos a seguir en el procesamiento de imágenes.

El primer paso consiste adquirir la imagen digital, para ello se necesita una cámara IP que aporte los flujos de video a los cuales se le extraen las imágenes digitales que lo componen.

Una vez que se obtiene la imagen digital, esta es preprocesada con el objetivo de ser mejorada y restaurada para así garantizar que la fase final del procesamiento tenga mayores posibilidades de éxito. El preprocesamiento consiste en aplicar algoritmos para mejorar el contraste, suprimir ruido y aislar regiones que por sus características indiquen la existencia real una persona u objeto.

El paso siguiente es la segmentación, su objetivo es dividir la imagen en regiones de objetos que la forman y retorna un conjunto de puntos por cada región encontrada. Aplicar algoritmos de segmentación en la solución propuesta a desarrollar tiene como objetivo obtener las áreas que definen una o un grupo de personas dentro de una imagen digital.

Luego de realizar una correcta segmentación se procede a representar las regiones de interés, esta representación se puede realizar de dos maneras, por contorno o por región interna.

Se realiza la descripción después de haber representado las regiones de interés, con el objetivo de extraer los datos que las caracterizan, se recibe un objeto y se retornan los rasgos asociados a él.

Posteriormente se lleva a cabo el reconocimiento, el cual se encarga de obtener los rasgos característicos de cada uno de los objetos y les asocia una categoría predefinida, y para finalizar se pone en práctica la interpretación cuya misión es asociar un significado o acción al conjunto de objetos reconocidos.

1.4 Análisis de otras soluciones existentes.

Como resultado de la consulta y estudio de materiales bibliográficos, ha sido posible la identificación de un software cuya funcionalidad está basada en técnicas de análisis de imagen para el conteo de personas, estimación de afluencias y control visual de operaciones de terminales de venta, dicho software se nombra **Análisis de Vídeo: Conteo de personas**, es un excelente programa tecnológicamente hablando, con funcionalidades basadas en las necesidades de cada cliente, además posee equipos **PeCo** altamente capacitados que realizan conteo de personas basados en el análisis de video. Son equipos dotados de sensores inteligentes que analizan las imágenes de las cámaras de CCTV (Closed Circuit Television) situadas cenitalmente sobre las zonas deseadas.

Dentro de las características de este producto se destacan: el funcionamiento con cámaras estándar sin necesidad de un software ni de hardware adicional, posee un alto porcentaje de precisión registrando con una fiabilidad del 95% todos los visitantes de una zona determinada, diferenciando entradas y salidas (bidireccional) y el paso de varias personas simultáneamente. Presenta datos de conteo para el control de operaciones, esto permite saber cuántos visitantes tiene un establecimiento dado, las horas y días de mayor afluencia, entre otras. Todo lo anterior posibilita tener un mayor conocimiento de un negocio específico y de esta manera tomar decisiones que aumenten su rentabilidad.

Por otra parte se conoce que existe un sistema llamado **SENSOURCE**, el mismo incluye tecnología de fácil expansión y actualización. Es un sistema para el análisis y detección de tráfico de clientes, que cuenta con un contador de personas el cual incorpora tecnología moderna para la detección, colección de datos y reportes. Esta aplicación hace uso de una tecnología avanzada llamada **PC-THI60** utilizada en la proyección de imágenes termales para contar el tráfico de visitantes desde una vista superior en una puerta de entrada y salida. Dicha tecnología permite rastrear continuamente a la gente que entra o sale al mismo tiempo a través del procesando de las imágenes termales correspondientes a las personas que se mueven debajo del lente de la cámara.

Otro video sensor para el conteo de personas es el iTally está formado de un pequeño procesador de video que está conectado a un pequeño sensor instalado en el techo de la entrada o cualquier zona de paso. Este sistema detecta automáticamente y cuenta personas según pasan por su campo de visión. Está preparado para hacer seguimiento bidireccional o sea, tanto de las entradas como de las salidas y es capaz de realizar correctamente su trabajo en un ambiente concurrido.

1.5 Tendencias y tecnologías actuales en el desarrollo del software.

1.5.1 Metodologías para el Desarrollo de Software.

El proceso de desarrollo de software es riesgoso y difícil de controlar, si no se cuenta con una guía puede llegar a ocurrir que los clientes y desarrolladores queden inconformes con los resultados finales. Cuando un sistema es pequeño muchas veces lo común es trazar un plan subyacente mínimo donde se separa rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función se determina un tiempo aproximado de desarrollo. A medida que los proyectos a desarrollar aumentan su envergadura, se hace más difícil la estimación y gestión del mismo.

Muchas veces se diseña el software de una manera rígida, específicamente con los requerimientos que el cliente solicita. Una vez terminado dicho software tiene que someterse a una larga fase de pruebas, de tal manera que cuando el cliente en dicha etapa solicita un cambio, trae consigo un atraso en el proyecto, pues casi siempre esto implica tener que implementar desde cero funcionalidades que no existían hasta ese momento.

Para evitar estos incidentes se ha tenido una alternativa desde hace mucho: las Metodologías. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software en aras de lograr una mayor eficiencia y predictibilidad. Desarrollan un proceso detallado con un fuerte énfasis en la planificación inspirado por otras disciplinas de la ingeniería. En forma de resumen se puede decir que son un conjunto de procedimientos y técnicas que ayudan a la documentación para el desarrollo de productos software. Actualmente se pueden encontrar dos vertientes en el mundo de las metodologías de desarrollo de software, estas vertientes son las metodologías pesadas o tradicionales y las metodologías ágiles.

"Las metodologías no ágiles son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo; llamadas también metodologías tradicionales o clásicas, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema." (Letelier, et al.)

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

"El Proceso Unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto". (Jacobson, 2000)

RUP tiene tres características esenciales está **dirigido por casos de uso**, donde los casos de uso 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, e **iterativo e incremental,** donde cada fase se divide en iteraciones, dividiendo el producto en pequeños proyectos para el desarrollo e incremento del mismo. Durante todo el proceso de desarrollo de software se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

La Figura 4 muestra el ciclo de vida de esta metodología. 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 como de apoyo.

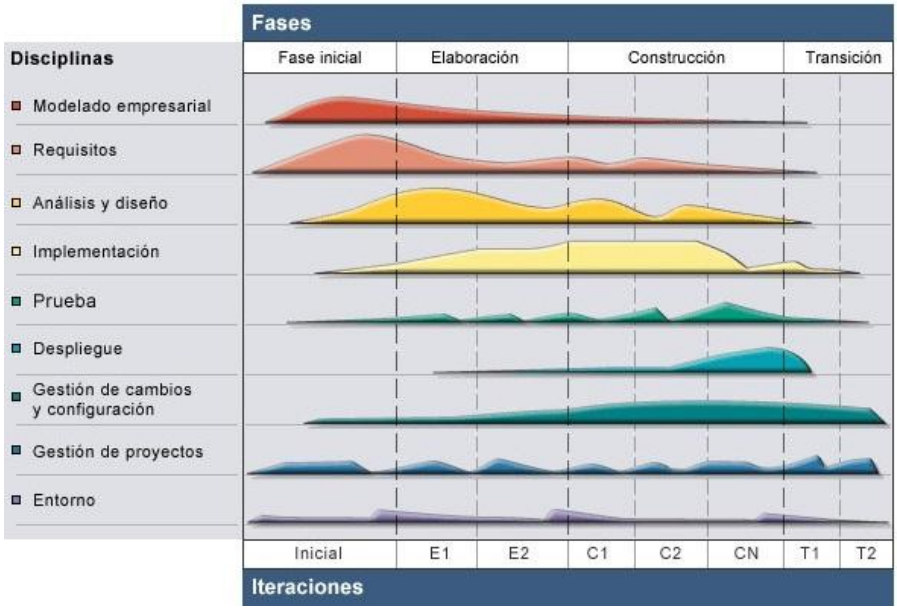


Figure 4 *Fases y flujos de trabajo de RUP.*

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.

Por otra parte se tuvo en consideración la gestión de requisitos, aspecto de vital importancia para la construcción de cualquier software, debido a que dicha gestión permite identificar de una manera más eficiente las necesidades y deseos de los usuarios; lo cual es esencial. Según el planteamiento de Jacobson, Booch y Rumbaugh en su libro El Proceso Unificado de Desarrollo de Software: *"Un sistema software ve la luz para dar servicio a sus usuarios. Por tanto, para construir un sistema con éxito debemos conocer lo que sus futuros usuarios necesitan y desean"*. (Jacobson, 2000).

Además la metodología RUP es de gran importancia como apoyo en el desarrollo del software, debido a varias prestaciones que brinda como son: provee alta calidad del resultado de los sistemas, aumenta la productividad de los desarrolladores mediante el acceso a las bases de conocimientos y herramientas. Presenta herramientas de apoyo en cada fase del proyecto. Se centra en la producción y tiene un alto potencial de organización.

1.5.2 Herramienta de modelado de los artefactos del RUP.

Enterprise Architect 7.5

Para modelar todos los artefactos de RUP se ha seleccionado Enterprise Architect 7.5 como herramienta de modelado, por ser una plataforma de diseño, administración y colaborativa, basada en UML 2.1 y estándares relacionados. También por ser una herramienta ágil, intuitiva y extensible, con poderosas características para dominios específicos totalmente integrados. Un aspecto de vital importancia que se tuvo en cuenta para dicha selección es que ofrece salida de documentación flexible y de alta calidad. Además posee todas las funcionalidades del Visual Paradigm. Desde los inicios el sistema de Video Vigilancia SURIA utiliza esta herramienta de modelado ya que fue seleccionada por el arquitecto del proyecto por sus importantes características.

1.5.3 Lenguaje de programación C++

"C++ es un lenguaje de programación basado en C que soporta directamente conceptos de la Orientación a Objetos. Retiene los recursos de bajo nivel y la eficiencia de C". Además, elimina algunas dificultades presentes en el C original. Es considerado un lenguaje híbrido ya que en él coexisten la programación estructurada y los conceptos de la POO." (Mora, 1997)

Para llevar a cabo la creación del video sensor para el conteo de persona se ha seleccionado a C++ como lenguaje de programación por su rapidez, portabilidad y documentación en el procesamiento de imágenes. Es un lenguaje versátil, potente y general. C++ presenta una gran versatilidad es decir, es un lenguaje de

propósito general, por lo que se puede emplear para resolver cualquier tipo de problema. Está estandarizado y un mismo código fuente se puede compilar en diversas plataformas. Es uno de los lenguajes más rápidos en cuanto ejecución.

1.5.4 Entorno de Desarrollo Integrado.

Para llevar a cabo la creación del video sensor que permita el conteo de personas se ha seleccionado a Visual C++.Net perteneciente al Entorno de Desarrollo Integrado (IDE) Visual Studio.Net, que se encuentra incluido dentro de la Plataforma de Ejecución Intermedia de Microsoft.Net con el objetivo primario de garantizar una total compatibilidad entre SURIA y el nuevo componente en desarrollo.

La Plataforma de Ejecución Intermedia de .Net a pesar de ser de licencia propietaria es reconocida a nivel internacional como la más completa si de desarrollar se trata, esto se debe a que es totalmente orientada a objetos y permite la creación de aplicaciones y servicios con la garantía de ser altamente robustos, seguros, compatibles con el sistema operativo Windows y entre ellos.

Esta posee un conjunto de bibliotecas de funcionalidades y controles reutilizables (Class Library), con una enorme cantidad de componentes ya programados listos para ser consumidos por otras aplicaciones. Además, permite desarrollar aplicaciones en Visual Basic, C#, C++, y Java# con una total comunicación entre sí.

1.5.5 Librería a utilizar.

OpenCV (Open Source Computer Vision Library) es la librería seleccionada para la creación del video sensor para el conteo de persona, es de código abierto para visión por computadoras que ha sido optimizada y proyectada la construcción de aplicaciones en tiempo real. Es independiente del hardware y del sistema operativo. Lee, escribe y adquiere genéricamente las imágenes y videos. Provee interfaces de optimización para procesadores Intel y además permite el procesamiento básico de imágenes así como el análisis de movimiento y reconocimiento de objetos.

1.6 Conclusiones.

Las soluciones existentes en la actualidad referente a los sistemas de video vigilancia que incorporan conteo de personas son eficientes pero limitadas, además de ser propietarias son altamente costosas y no todas las personas pueden tener acceso a esta herramienta. El empleo de la metodología de desarrollo RUP, y una buena selección de las herramientas Case facilitan todo el proceso de desarrollo de software garantizando una alta calidad del mismo. La función del lenguaje de programación C++ junto con el entorno de desarrollo integrado escogido, y una buena selección de la librería a utilizar, permiten la creación de un video sensor para el conteo de persona con la garantía de ser altamente robusto y seguro.

CAPÍTULO 2: Características del Sistema.

2.1 Introducción.

En este capítulo se da a conocer la propuesta de solución para desarrollar un video sensor que cuente personas provenientes de una cámara IP. En aras de describir la solución se realiza un modelo de dominio y la especificación de los requisitos funcionales y no funcionales que debe presentar el sistema a construir. Además se determinan los casos de uso del sistema y los actores que interactuarán con este, elaborándose una descripción de cada uno de los mismos.

2.2 Sistema Propuesto.

Se propone desarrollar un video sensor que adquiera flujos de video provenientes desde un grupo de cámaras IP a través de la red, procese digitalmente el video y cuente las personas que transitan frente a dichas cámaras. Este componente debe ser sencillo de integrar a sistemas de vigilancia en desarrollo como SURIA con el objetivo de dotarlos de una herramienta que les permita realizar este tipo de procesamiento.

2.3 Modelo de Dominio.

El modelo de negocio presenta un sistema, en este caso el negocio, desde la perspectiva de su uso y esquematiza como proporciona valor a sus usuarios, que son: clientes y socios. En este caso no se puede brindar un valor a clientes y socios porque no existen, al no existir no se puede identificar procesos que ocurren en el negocio ni tampoco se pueden identificar beneficiados y mucho menos personas que lo identifiquen. Por estas razones se ha decidido crear un modelo de dominio, el cual se centra en una parte del modelo de negocio.

Un modelo de dominio es una representación de las clases conceptuales del mundo real. Es una base de conocimiento con los principales conceptos asociados al desarrollo del sistema. Puede considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio.

2.3.1 Conceptos Fundamentales

Para un mejor entendimiento del Diagrama de Modelo de Dominio conformado en éste punto se proporciona un marco conceptual con las definiciones identificadas, las cuales son:

Video sensor: Algoritmo de identificación de objetos.

Persona: Objeto localizado en el video.

Características: Rasgos que identifican al objeto.

Seguimiento: Registro de movimientos del objeto a través del tiempo.

Cámara: Dispositivo de captura de imagen y video.

Video: Archivo que almacena los flujos de video de una cámara comprimidos en un formato de uso común.

2.3.2 Diagrama del Modelo de Dominio

Diagrama Conceptual del Modelo de Dominio

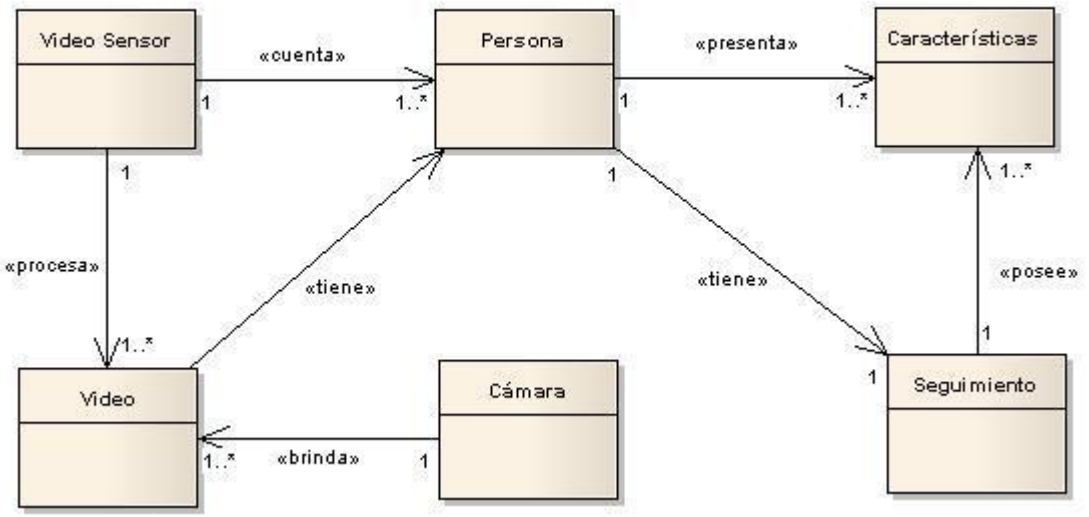


Figure 5 *Diagrama Conceptual del Modelo de Dominio*

Descripción del modelo de dominio: La cámara brinda un flujo de video, el cual puede tener varias personas o ninguna. Esta persona presenta características que se utilizan para el seguimiento de la misma, y le permite al video sensor contar las personas. (Ver figura 5)

2.4 Especificación de los Requisitos del Software

2.4.1 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 extraer los fotogramas del flujo de video.

RF3. El sistema debe permitir modelar el fondo del video.

RF4. El sistema debe permitir extraer el frente del video.

RF5. El sistema debe permitir eliminar ruido en fotogramas.

RF6. El sistema debe permitir procesar los píxeles pertenecientes a sombras en los fotogramas del flujo de video.

RF6.1 El sistema debe permitir detectar sombras en fotogramas.

RF6.2 El sistema debe permitir eliminar sombras en fotogramas.

RF7. El sistema debe permitir hacer seguimiento de objeto.

RF8. El sistema debe permitir contar la cantidad de personas.

2.4.2 Requisitos No Funcionales

Según **Roger Pressman**, *"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 enuncian, separados en categorías, los diferentes requisitos no funcionales que el componente debe satisfacer.

A continuación se muestran los requerimientos no funcionales del sistema:

- **RNF1**. **Rendimiento**

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

RNF1.2 Se debe ajustar el tamaño del fotograma con la cantidad de estos por segundos.

RNF1.3 El máximo de memoria RAM que debe de consumir el sistema es de 300 mb.

- **RNF2. Usabilidad**

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

- **RNF3 Fiabilidad**

RNF3.1 El componente puede estar funcionando a 24 horas, tiempo completo.

- **RNF4 Portabilidad**

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

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

- **RNF5 Hardware**

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

- **RNF6 Restricciones del diseño**

RNF6.1 Lenguaje:

- El lenguaje que se utilizará para el desarrollo del sistema será C++.

RNF6.2 Librería:

- La librería que se utilizará será OpenCV.

2.5 Definición de los casos de uso

"Los Casos de Uso no son parte del diseño (el cómo), sino parte del análisis (qué). De forma que al ser parte del análisis ayudan a describir qué es lo que es sistema debe hacer. Los Casos de Uso son qué hace el sistema desde el punto de vista del usuario. Es decir, describen un uso del sistema y cómo este interactúa con el usuario". (Gracia, 2011)

2.5.1 Definición de los actores

Ivar Jacobson es de la opinión que, *"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. La implementación interna de un actor no es relevante en el caso de uso; un actor puede ser caracterizado suficientemente por un conjunto de atributos que definen su estado.

Los actores pueden ser definidos en jerarquías de generalización, en las cuales una descripción abstracta del actor es compartida y aumentada por una o más descripciones específicas del actor. Un actor puede ser un ser humano, otro sistema informático, o un cierto proceso ejecutable. Se dibuja a un actor como una persona pequeña con trazos lineales y el nombre debajo de él". (Jacobson, 1998)

Actor del Sistema	Descripción
Sistema Video Vigilante	Es el sistema con quien interactúa el componente, es el que inicia el caso de uso Contar persona.

2.6 Diagrama de Casos de Uso del Sistema

En la figura 6 se muestra el diagrama de casos de uso del sistema.

Diagrama de Casos de Uso del Sistema

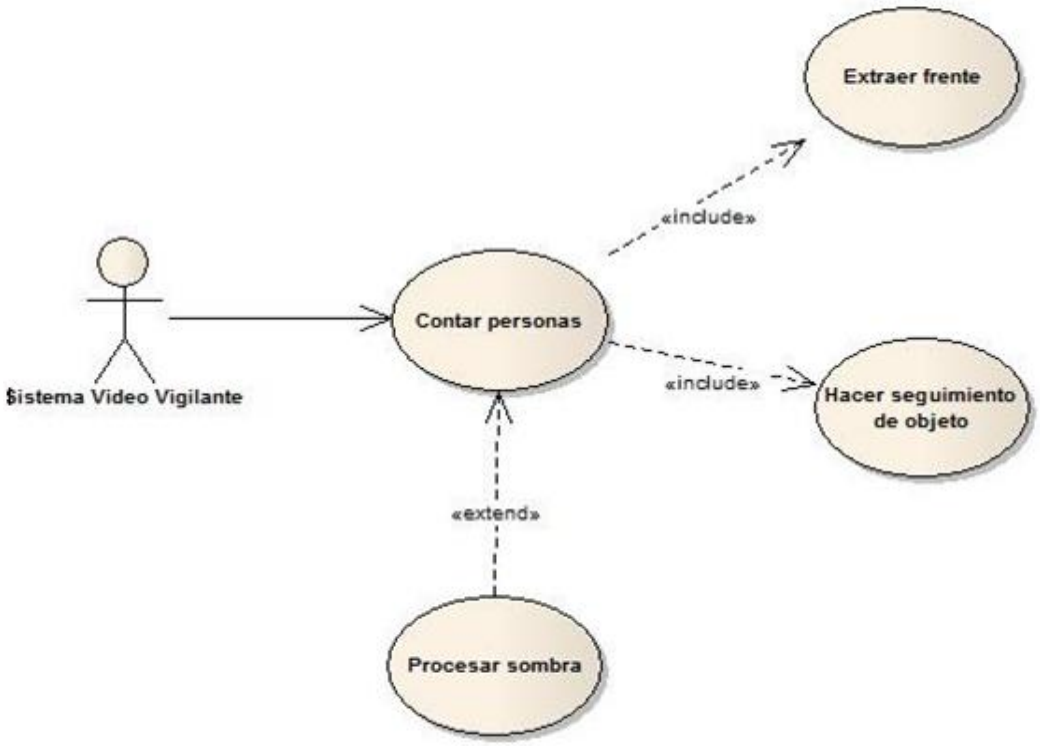


Figure 6 *Diagrama Casos de Uso del Sistema.*

2.6.1 Listado de casos de uso

CU-1	Contar personas
Actor	Sistema Video Vigilante
Descripción	El caso de uso inicia cuando el Sistema Video Vigilante hace la petición de contar personas, el sistema capturara los flujos de videos obtenidos desde la cámara IP. Posteriormente extrae de él los fotogramas, modela el fondo y extrae el frente para obtener los objetos en la escena. Seguidamente elimina el ruido y la sombra en caso de ser necesario. Por último realiza el seguimiento del objeto y lo cuenta.

Referencia	RF#8
Prioridad	Crítico

Tabla 1 *Descripción del caso de uso "Contar personas"*

CU-2	Extraer frente
Actor	Sistema Video Vigilante
Descripción	El caso de uso inicia cuando el Sistema Video Vigilante hace la petición de contar personas, el sistema modela el fondo de los fotogramas obtenidos a partir del flujo de video capturado. Se obtiene la máscara de frente del objeto, se elimina el ruido y en caso de ser necesario la sombra. Obteniéndose así los objetos de los fotogramas.
Referencia	RF# 3, RF# 4, RF# 5
Prioridad	Crítico

Tabla 2 *Descripción del caso de uso "Extraer frente"*

CU-3	Procesar sombra
Actor	Sistema Video Vigilante
Descripción	El caso de uso inicia cuando el Sistema Video Vigilante solicita que se procesen las sombras presentes en la escena, el caso de uso permite detectar las partes que son sombras pertenecientes a los objetos en la imagen y eliminarlas de la misma.
Referencia	RF#6
Prioridad	Crítico

Tabla 3 *Descripción del caso de uso "Procesar sombra"*

CU-4	Hacer seguimiento de objeto.
Actor	Sistema Video Vigilante
Descripción	El caso de uso inicia cuando el Sistema Video Vigilante hace la petición de contar personas, se chequea que los objetos aparecidos en la escena se muevan uniformemente para llevar un registro de su dirección o desplazamiento.
Referencia	RF#7
Prioridad	Crítico

Tabla 4 *Descripción del caso de uso "Hacer seguimiento de objeto."*

2.7 Descripción extendida de los casos de uso

CU- 1	Contar personas
Actor:	Sistema de Video Vigilancia
Propósito:	Contar personas a través de un flujo de video obtenido desde cámaras IP.
Resumen:	El Sistema de Video Vigilancia solicita el conteo de personas en un flujo de video, se realiza un proceso inteligente obteniéndose el seguimiento de la persona y el conteo de la misma.
Precondiciones:	El Sistema Video Vigilante solicita contar personas.
Referencia:	RF# 1, RF# 2, RF# 8
Prioridad:	Crítico
Flujo normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1. El caso de uso inicia cuando el actor solicita el conteo de personas en un flujo de video proveniente de una cámara IP.	2. Se obtiene la dirección del video 3. Con la dirección del video se hace una petición del flujo de video a la cámara. 4. Si la cámara brinda flujo se crea una estructura que referencia el flujo de video. 5. Se extraen los fotogramas del flujo de video. 6. Se extrae el frente del video. (Ver descripción del CU-2) 7. El sistema chequea si se procesan las sombras. 8. Se realiza el seguimiento de la persona. (Ver descripción del CU-4) 9. El caso de uso finaliza cuando cuenta la persona.
Flujo Alternativo Paso 4	
	4. Si la cámara no brinda flujo de video se muestra un mensaje de error:”No se pudo obtener el flujo de video.”
Flujo Alternativo Paso 7	
	7. Si se va a procesar sombras, se ejecuta CU3. En caso contrario, se continúa el procesamiento.

Poscondiciones:	Se contaron los objetos que entraron y salieron en la escena.

Tabla 5 *Descripción extendida del caso de uso "Contar personas."*

En el Anexo 1 se muestran las demás descripciones detalladas de los casos de uso del sistema. Se muestra el flujo de eventos que ocurren entre el actor y el sistema.

2.8 Conclusiones

Con el apoyo de los requisitos funcionales y la obtención de los casos de usos se logra una mejor comprensión del funcionamiento del sistema internamente. . Al analizar el modelo de dominio, se puede afirmar que se ha logrado capturar el conocimiento del área de negocio necesario para desarrollar el modelo de diseño del componente, basándose en las descripciones que realiza de los conceptos del negocio del mismo.

CAPÍTULO 3: Análisis y Diseño del Sistema.

3.1 Introducción.

En este capítulo se construye la propuesta de solución para llevar a cabo la elaboración del componente. Dicha propuesta incluye los modelos de análisis y diseño, los cuáles dejarán sentadas las bases para comenzar con la implementación de la primera propuesta de solución del sistema. Esto se logrará a través de los diferentes diagramas de clases e interacción que incluyen dichos modelos. Además se explican el cúmulo de patrones del diseño empleados para garantizar la viabilidad y calidad requeridas por el sistema a construir.

3.2 Descripción de la Arquitectura.

La arquitectura de software, está relacionada con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.

"El sistema Suria está diseñado siguiendo una Arquitectura base en forma de pizarra, en su variante de tablero de control. Ésta desacopla el sistema en componentes denominados agentes autónomos, los cuales son independientes en la realización atómica de su funcionalidad. Pero dependen de una entrada de información externa, que es provista por otros agentes, y a su vez, producen un resultado que puede ser entrada de otros agentes." (Aldana, 2009)

"Cada agente se rige por interfaces estándares que permiten cambios en el ambiente externo al agente sin que éste sufra cambios en su funcionamiento interno. Todo el funcionamiento de los agentes autónomos, está coordinado por un elemento central, denominado Repositorio Activo, el cual entrega y recibe información de los agentes y coordina su funcionamiento." (Aldana, 2009). En el caso del sistema de video vigilancia Suria, el Gestor cumple con la función de Repositorio activo al que se pueden conectar diferentes agentes autónomos.

Uno de los agentes autónomos que compone el sistema de video vigilancia Suria es el módulo Suria Analytic. Su función es gestionar el procesamiento inteligente de video, a través de video sensores como: video sensor para el conteo de persona, video sensor para la detección de objetos abandonados y video sensor para la estimación de velocidad de vehículos. El módulo Suria Analytic sirve de fachada entre el Gestor y los video sensores a través de la tecnología .Net Remoting. Esta tecnología permite la

comunicación entre objetos mediante canales para enviar y recibir mensajes de aplicaciones remotas, usando diferentes protocolos de transporte, formatos de serialización entre otros. (Ver figura 7)

Los videos sensores son los que realizan todo el procesamiento inteligente de video, poseen un estilo arquitectónico llamado Pipes & Filters (Tuberías y Filtros), donde cada componente tiene un conjunto de entradas y un conjunto de salidas. Un componente lee un flujo de datos en la entrada y produce un flujo de datos diferente en su salida. Esto es logrado aplicando una transformación local al flujo de entrada mientras este se lee, de tal forma que el flujo de salida empieza antes que se consume todo el flujo de entrada. Este patrón divide la tarea de un sistema en varios pasos de procesamiento secuenciales. Estos pasos están conectados por el flujo de datos a través del sistema, cada paso del procesamiento está encapsulado en un componente de filtro. Los datos pasan a través de las tuberías que son los conectores que sirven como conductos para transmitir las salidas de un filtro a las entradas de otro.

Este patrón de arquitectura es particularmente efectivo a la hora de descomponer el problema en pasos independientes, reutilizar filtros, facilitar el mantenimiento, independencia y ejecución concurrente de filtros. Este patrón tiene algunas restricciones, como que los filtros deben ser entidades independientes: en particular no deben compartir estados con otros filtros. Los filtros no conocen la identidad del filtro de donde proviene el flujo que reciben como entrada y tampoco la identidad del filtro a donde llega el flujo de salida.

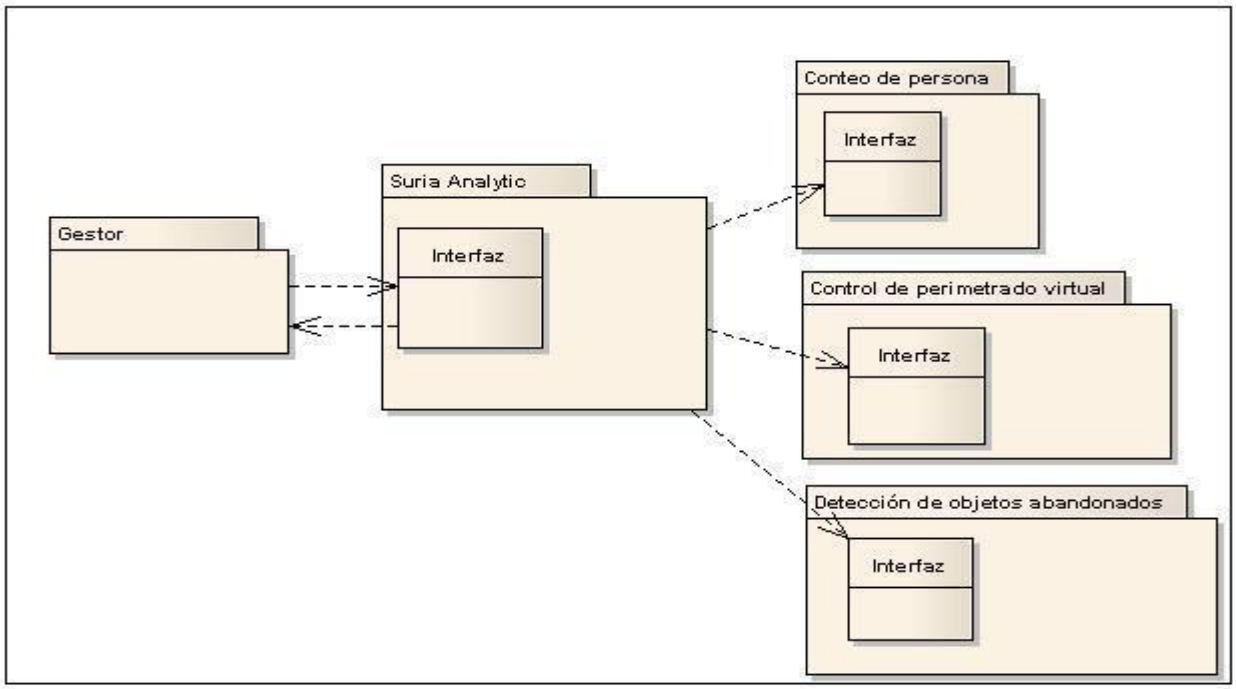


Figure 7 *Vista Lógica del Módulo Suria Analytic.*

3.3 Patrones de diseño.

"Un patrón es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos. Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema." (Larman, C, 1999)

Los patrones de diseño ayuda a como se debe estructurar las clases y objetos para guiar todo el procesos de creación de software. Componen soluciones concretas a problemas que se presentan durante el diseño de una aplicación. 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 resposabilidades para diseñar con éxito el software orientado a objetos. En el desarrollo del sistema se aplicaron los patrones: Experto, Creador, Bajo Acoplamiento, Alta Cohesión y Controlador.

El patrón Experto es el que le asigna la responsabilidad a la clase experta, es la que contiene la información para dar cumplimiento a ella misma. El patrón Creador decide que clase es la responsable de la construcción de objetos es decir, la que le puede asignar los datos al constructor. El patrón Bajo Acoplamiento asigna responsabilidades a otras clases para tratar de reducir las dependencias al mínimo, lo más que se pueda para facilitar la reutilización de código. El patrón Alta Cohesión asigna responsabilidades a clases o subsistemas que estén altamente relacionadas, es decir si estos elementos no poseen una gran cantidad de trabajo poseen alta cohesión. El patrón Controlador es el que decide que clase es la responsable de recibir o manejar un evento del sistema.

"Los patrones de diseño GoF (Gang of Four) se clasifican en patrones creacionales, estructurales y de comportamiento. Los creacionales resuelven problemas relativos a la creación de objetos, mientras los estructurales ofrecen solución a problemas relativos a la composición de objetos. Por su parte, los patrones de comportamiento resuelven los problemas referentes a la interacción entre los objetos." (Cooper, 1998)

El módulo Suria Analytic utiliza el patrón estructural Fachada, el cual proporciona una interfaz unificada que representa un subsistema y permite acceder a funcionalidades de un conjunto de clases.

3.4 Modelo de análisis

"El análisis proporciona una visión general del sistema que puede ser más difícil de obtener mediante el estudio de los resultados del diseño y la implementación, debido a que contienen demasiados detalles". (Jacobson, 2000)

De esta manera definen uno de los propósitos del análisis los creadores del libro El Proceso Unificado de Desarrollo de Software. El modelo de análisis le da una vista panorámica a los desarrolladores de las funciones que se implementarán y la idea general de cómo hacerlo. Durante este flujo de trabajo se generan una serie de artefactos entre los cuales se encuentran las clases del análisis.

3.4.1 Clases del análisis

En la realización del modelo de análisis se utilizan los siguientes tipos de clases:

- **Clases Interfaz**

Figure 8 Clase Interfaz

"Las clases interfaz se utilizan para modelar la interacción entre el sistema y sus actores (es decir, usuarios y sistemas externos). Esta interacción a menudo implica recibir (y presentar información) y peticiones de (y hacia) los usuarios y los sistemas externos". (Jacobson, 2000). Ver Figura 8.

- **Clases de Control**

Figure 9 Clase de Control

"Las clases de control representan coordinación, secuencia, transacciones, y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto. Las clases de control también se utilizan para representar derivaciones y cálculos complejos, como la lógica del negocio, que no pueden asociarse con ninguna información concreta, de larga duración, almacenada por el sistema". (Jacobson, 2000). Ver Figura 9.

- **Clases Entidad**

Figure 10 *Clase Entidad*

"Las clases de entidad se utilizan para modelar información que posee una vida larga y que es a menudo persistente. Las clases de entidad modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real, o un suceso del mundo real". (Jacobson, 2000) Ver Figura 6.

3.4.2 Diagrama de Clases

En la figura 11 se muestra el diagrama de clases del análisis para el caso de uso Contar personas. Se muestra la clase interfaz Contar personas que es la que se encargará de comunicarse con la clase control Procesamiento que manejará todos los pasos que se van a ejecutar en las clases controladoras Procesar sombra, Extraer frente y Hacer seguimiento, las cuáles manipulan todas las rutinas para trabajar con las clases entidades Imagen y Objeto. Éstas últimas representan la información de larga vida con que se va a interactuar, tienen relación de composición ya que la imagen está compuesta por objetos.

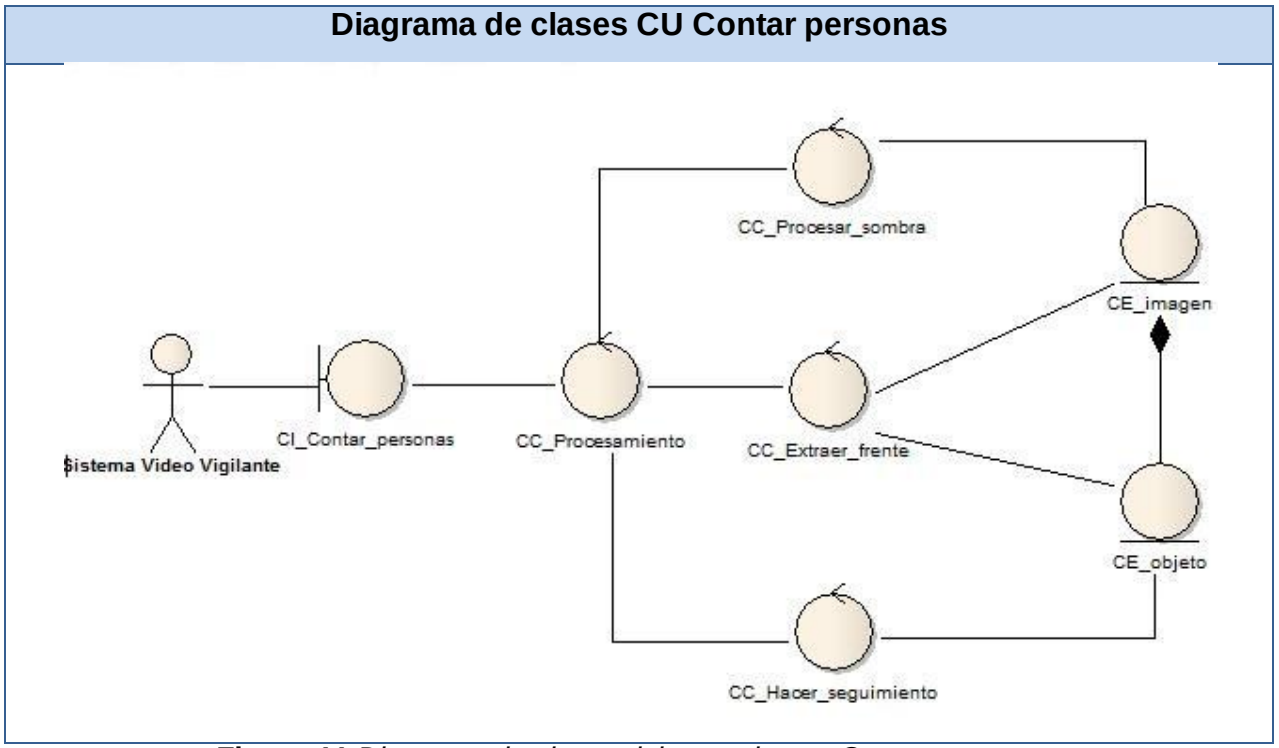


Figure 11 *Diagrama de clases del caso de uso Contar personas.*

3.4.3 Diagramas de Interacción

Analizando el razonamiento de **Craig Larman,** *"El UML contiene diagramas de interacción que explican gráficamente como los objetos interactúan a través de mensajes para realizar las tareas".* (Larman, C, 1999). Estos diagramas le muestran a los desarrolladores la manera en que las clases interactúan unas con otras y los mensajes que se envían. Según UML existen dos tipos de diagramas de interacción, estos son los:

Diagramas de colaboración:

Un diagrama de colaboración es un diagrama de clases que contiene roles de clasificador y roles de asociación en lugar de sólo clasificadores y asociaciones. Los roles de clasificador y los roles de asociación describen la configuración de los objetos y de los enlaces que pueden ocurrir cuando se ejecuta una instancia de la colaboración.

Diagramas de secuencia:

Un diagrama de secuencia representa una interacción como un gráfico bidimensional. La dimensión vertical es el eje de tiempo, que avanza hacia abajo de la página. La dimensión horizontal muestra los roles de clasificador que representan objetos individuales en la colaboración.
Cada rol de clasificador se representa mediante una columna vertical-línea de vida. Durante el tiempo que existe un objeto, el rol se muestra por una línea discontinua. Durante el tiempo que dura una activación de un procedimiento en el objeto, la línea de vida se dibuja como una línea doble.
Se muestra un mensaje como una flecha desde la línea de vida de un objeto a la del otro. Las flechas se organizan en el diagrama en orden cronológico hacia abajo.

A continuación se muestra los diagramas de colaboración elaborados para la construcción de la solución del sistema.

3.4.4 Diagramas de colaboración.

En las figuras 12, 13 y 14 se muestran los diagramas de colaboración para los casos de usos Procesar sombra, Extraer frente y Hacer seguimiento de objeto. Se observan los principales mensajes intercambiados entre las clases para realizar los casos de usos en cuestión.

Diagrama de colaboración CU Procesar sombra.

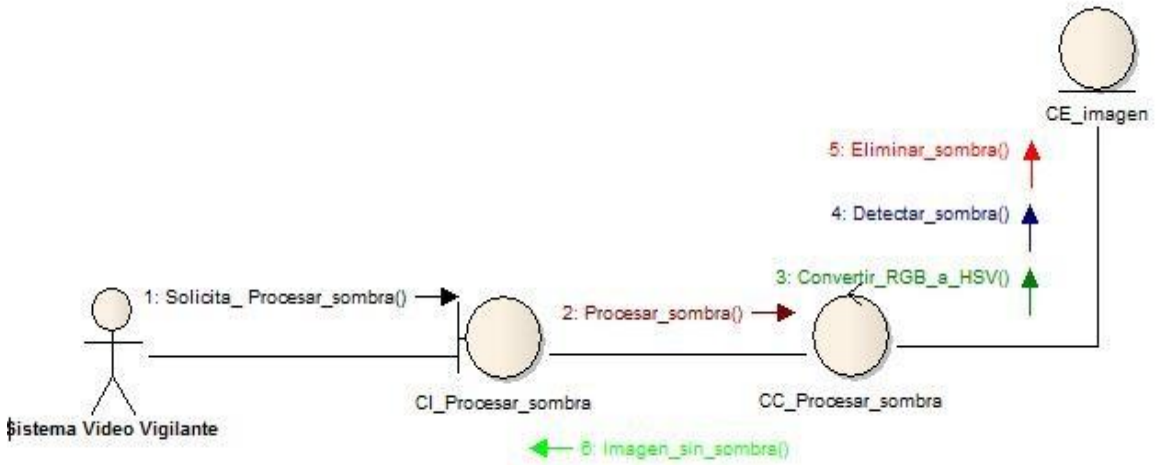


Figure 12 *Diagrama de colaboración del caso de uso Procesar sombra.*

Diagrama de colaboración CU Extraer frente.

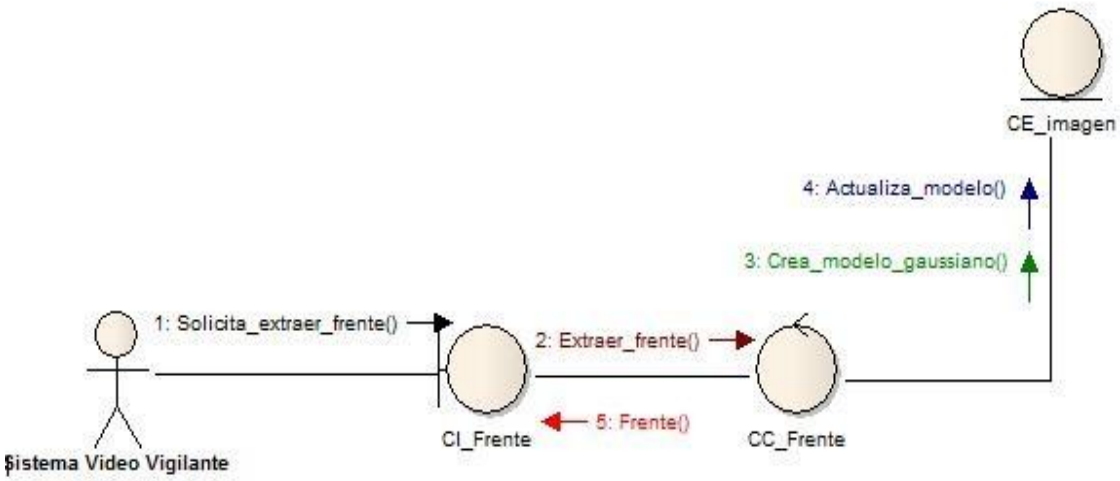


Figure 13 *Diagrama de colaboración del caso de uso Extraer frente.*

Diagrama de colaboración CU Hacer seguimiento de objeto.

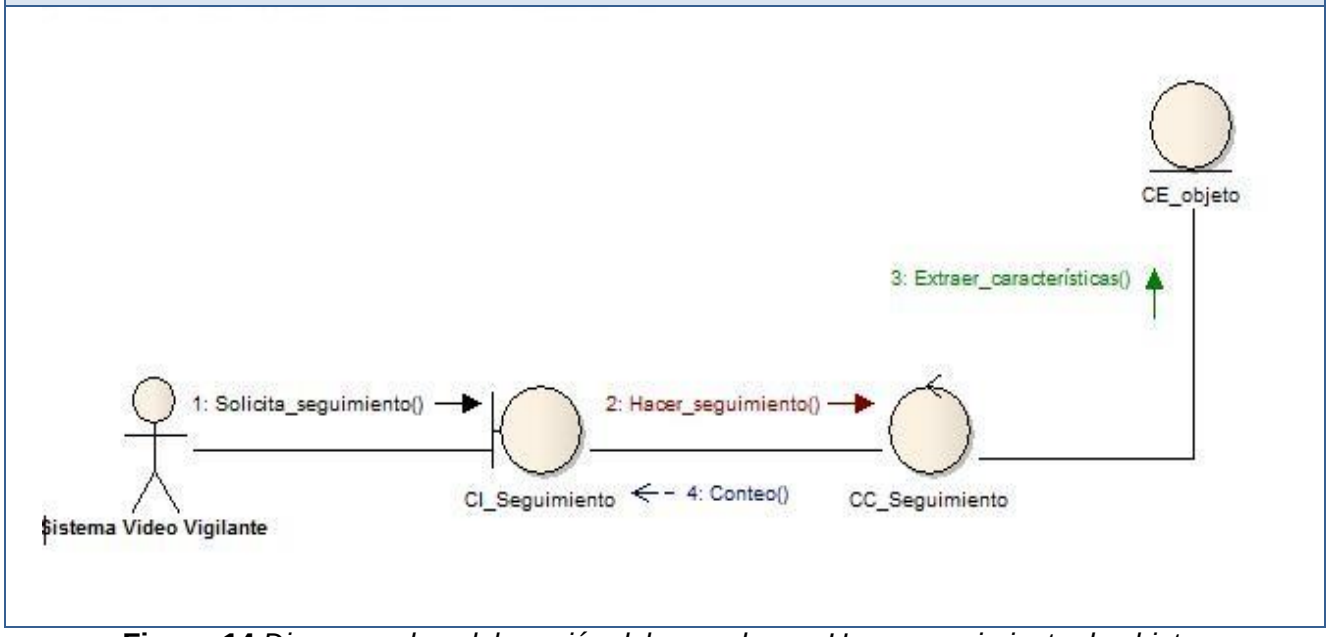


Figure 14 *Diagrama de colaboración del caso de uso Hacer seguimiento de objeto.*

3.5 Modelo del diseño

"El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar". (Jacobson, 2000)

El modelo de diseño se centra en la realización de los casos de uso a través de los requisitos, tanto funcionales como no funcionales. Una entrada esencial en este modelo es el resultado del modelo de análisis, este resultado proporciona una comprensión detallada de los requisitos, y lo más importante es que impone una estructura del sistema que se debe conservar lo más fielmente posible cuando se le da forma al sistema. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación.

3.5.1 Clase del diseño

"Una clase del diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema. Lo cual significa que el lenguaje utilizado para especificar una clase del

diseño es lo mismo que el lenguaje de programación, esto conlleva a que las operaciones, atributos, tipos y demás son especificados utilizando la sintaxis del lenguaje de programación elegido". (Jacobson, 2000)

Diagrama de clases del diseño

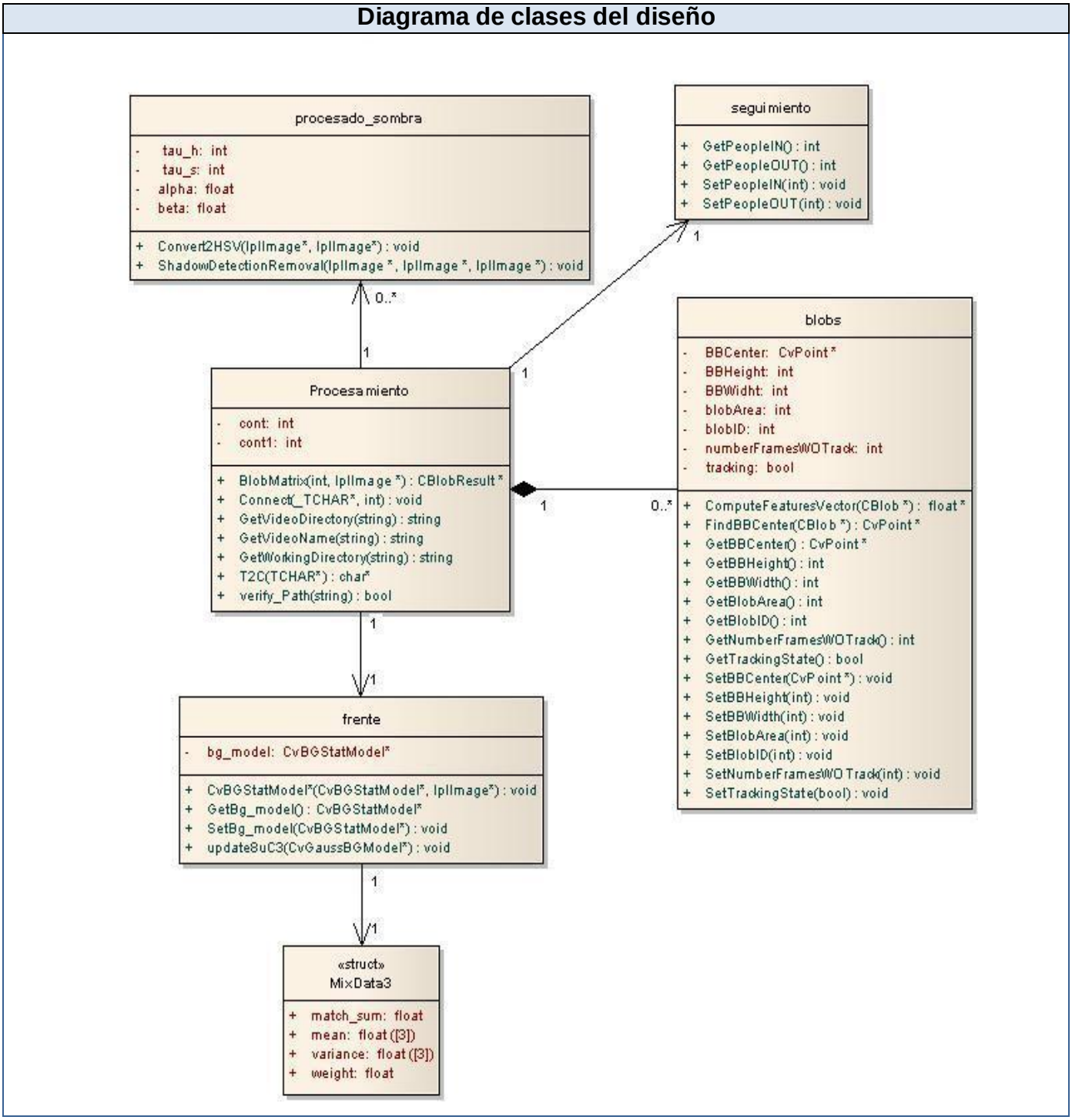


Figure 15 *Diagrama de clases del diseño para el sistema propuesto.*

La clase Procesamiento es la encargada de controlar las funcionalidades del componente. Posee los métodos necesarios para cumplir con los requisitos y además interactúa con las clases Blobs, Procesado de Sombra, Frente y Seguimiento que sirven de apoyo para el procesamiento de las imágenes. (Ver figura 15)

La clase Blobs es la que guarda los datos necesarios de los objetos que serán procesados, ella contiene datos importantes como centro, ancho, alto, área y seguimiento que son de vital importancia para procesar la imagen y obtener los rasgos característicos del objeto, lo cuales son muy importantes ya que la clase Seguimiento los utiliza para realizar el seguimiento a los objetos y así contarlos. La clase Procesado de Sombra es la encargada de eliminar las sombras de los objetos detectados en la escena. Y la clase Frente posee métodos para modelar el fondo de las imágenes y así obtener los objetos en movimiento que aparecen en la escena.

3.5.1.1 Diagrama de secuencia del diseño

La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Si se considera el “interior”, se tendrá algún objeto del diseño que recibe el mensaje del actor. Después el objeto de diseño llama a algún otro objeto, y de esta manera los objetos implicados interactúan para realizar y llevar a cabo el caso de uso. En el diseño es preferible representar esto con diagramas de secuencia ya que el centro de atención principal es el encontrar secuencias de interacciones detalladas y ordenadas en el tiempo. A continuación en las figuras 16, 17 y 18 se muestran los diagramas de secuencia para los casos de usos Procesar sombra, Extraer frente y Hacer seguimiento de objeto.

Diagrama de secuencia CU Procesar sombra.

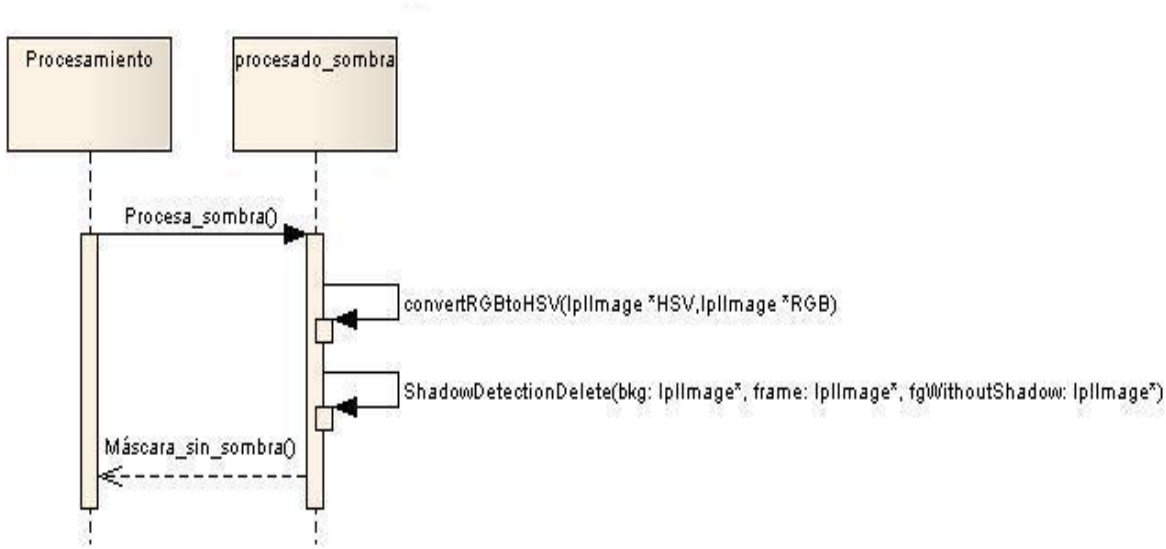


Figure 16 *Diagrama de secuencia del caso de uso Procesar sombra.*

Diagrama de secuencia CU Extraer frente.

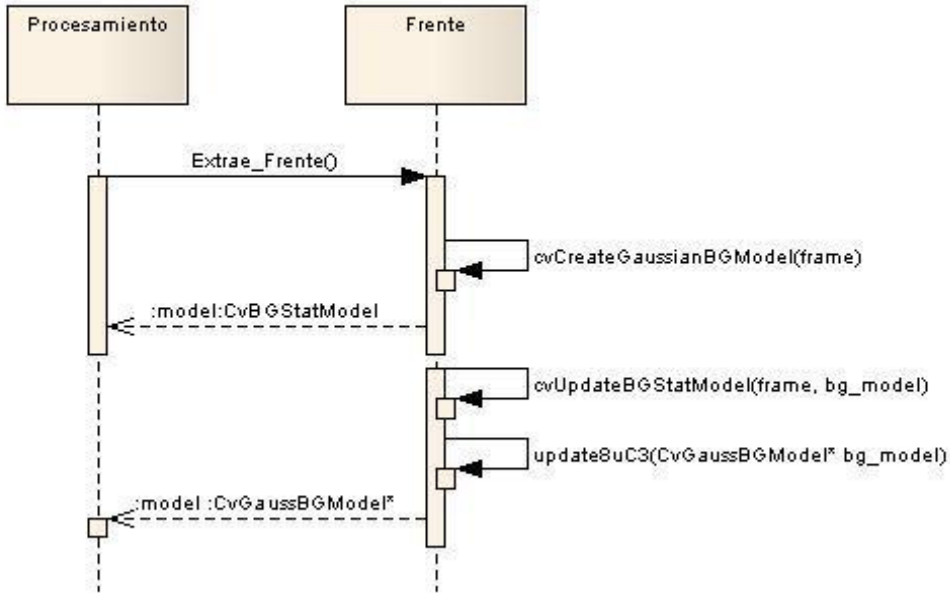


Figure 17 *Diagrama de secuencia del caso de uso Extraer frente.*

Diagrama de secuencia CU Hacer seguimiento de objeto.

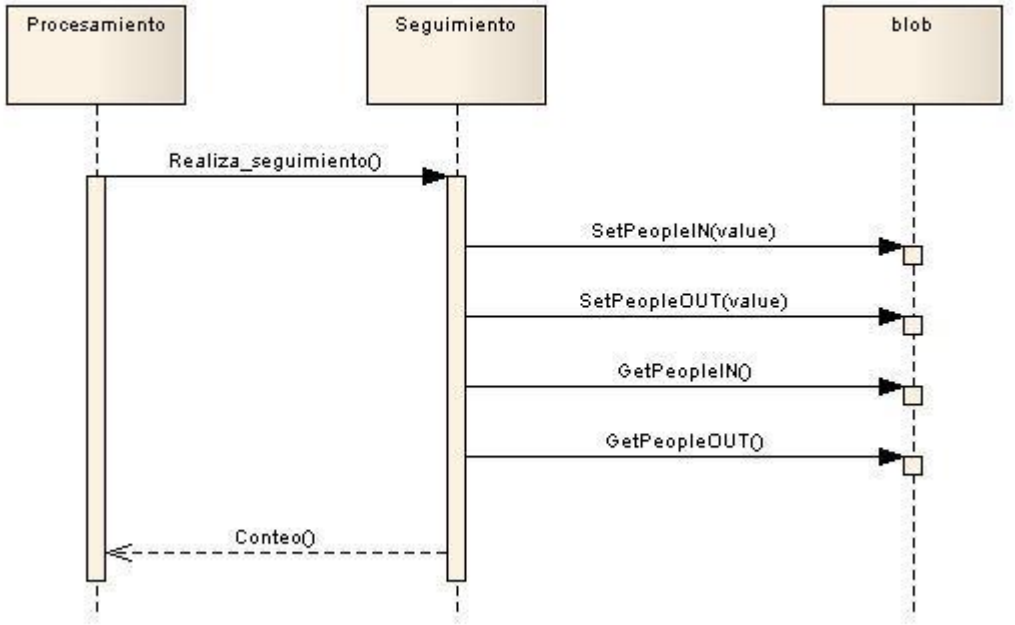


Figure 18 *Diagrama de secuencia del caso de uso Hacer seguimiento de objeto.*

3.5.2 Descripción de las clases.

Nombre	Procesamiento
Tipo de clase	Control
Atributo	**Tipo**
cont	int
cont1	int
Para cada responsabilidad	
Nombre	Procesar(int argc, _TCHAR *argv[])
Descripción	Devuelve la cantidad de personas.
Nombre	BlobMatrix(IplImage * Image, int blobMinSize)
Descripción	Devuelve una lista de objeto.
Nombre	GetWorkingDirectory(string path)
Descripción	Devuelve la dirección del ejecutable de la aplicación.

Nombre	verify_Path(string path)
Descripción	Verifica si existe o no una dirección.
Nombre	GetVideoName(string path)
Descripción	Devuelve el nombre del video.
Nombre	GetVideoDirectory(string path)
Descripción	Devuelve la dirección del video.
Nombre	Connect(int argc, _TCHAR* argv[])
Descripción	Permite la conexión a la cámara.
Nombre	T2C(TCHAR* source)
Descripción	Convierte de un tipo de dato a otro.

Tabla 9 *Descripción de la clase "Procesamiento."*

Nombre	Blobs
Tipo de clase	Entidad
Atributo	**Tipo**
blobID	int
BBWidht	int
BBHeight	int
blobArea	int
numberFramesWOTrack	int
BBCenter	CvPoint *
tracking	bool
Para cada responsabilidad	
Nombre	GetBlobID()
Descripción	Devuelve el id del objeto.
Nombre	SetBlobID(int pBlobID)
Descripción	Asigna el id del objeto.
Nombre	GetBBWidth()
Descripción	Devuelve el ancho del bounding box(rectángulo que define el objeto)
Nombre	SetBBWidth(int pBBWidth)
Descripción	Asigna el ancho del bounding box(rectángulo que define el objeto)
Nombre	GetBBHeight()

Descripción	Devuelve el alto del bouding box
Nombre	SetBBHeight(int pBBHeight)
Descripción	Asigna el alto del bouding box.
Nombre	GetBlobArea()
Descripción	Devuelve el área del objeto.
Nombre	SetBlobArea(int pBlobArea)
Descripción	Asigna el área del objeto.
Nombre	GetNumberFramesWOTrack()
Descripción	Devuelve el número de frames que un objeto lleva sin seguimiento.
Nombre	SetNumberFramesWOTrack(int pNFWOTrack)
Descripción	Asigna el número de frames que un objeto lleva sin seguimiento.
Nombre	GetTrackingState()
Descripción	Devuelve el estado del seguimiento del objeto. Si está en seguimiento o no.
Nombre	SetTrackingState(bool pTrackingState)
Descripción	Asigna el estado del seguimiento del objeto.
Nombre	GetBBCenter()
Descripción	Devuelve las coordenadas x,y del centro del bouding box
Nombre	SetBBCenter(CvPoint *center)
Descripción	Asigna las coordenadas x,y del centro del bouding box
Nombre	ComputeFeaturesVector(CBlob * currentBlob)
Descripción	Devuelve las características del objeto.
Nombre	FindBBCenter(CBlob *currentBlob)
Descripción	Buscar el centro del objeto

Tabla 10 *Descripción de la clase "Blobs."*

Nombre	Procesado_sombra
Tipo de clase	Control
Atributo	**Tipo**
tau_h	int
tau_s	int
alpha	float

beta	float
Para cada responsabilidad	
Nombre	Convert2HSV(IplImage* RGBImage,IplImage* HSVImage)
Descripción	Convierte de RGB a espacio de color HSV.
Nombre	ShadowDetectionRemoval(IplImage *bkg, IplImage *frame, IplImage *fgWithoutShadow)
Descripción	Elimina la sombra del objeto.

Tabla 11 *Descripción de la clase "Gestor de Sombra."*

Nombre	Seguimiento
Tipo de clase	Control
Atributo	**Tipo**
peopleIN	int
peopleOut	int
id	int
Para cada responsabilidad	
Nombre	SetPeopleIN(int value)
Descripción	Cuenta las personas que entran
Nombre	SetPeopleOUT(int value)
Descripción	Cuenta las personas que salen
Nombre	GetPeopleIN()
Descripción	Devuelve las personas que entran
Nombre	GetPeopleOUT()
Descripción	Devuelve las personas que salen

Tabla 12 *Descripción de la clase "Seguimiento."*

Nombre	Frente
Tipo de clase	Control
Atributo	**Tipo**
bg_model	CvBGStatModel*
Para cada responsabilidad	
Nombre	UpdateModel(IplImage* frame, CvBGStatModel* bg_model)
Descripción	Actualiza el fondo de una imagen.

Nombre	SetBg_model(CvBGStatModel* pbg_model)
Descripción	Separar el frente del fondo de una imagen aplicando el modelo de mezclas de Gaussianas.
Nombre	update8uC3(CvGaussBGModel* bg_model)
Descripción	Actualiza el fondo de una imagen de 3 canales.

Tabla 13 *Descripción de la clase "Frente."*

3.6 Conclusiones

La realización del modelo de análisis permite una mayor comprensión del problema a la hora de modelar la solución, ya que en este flujo se refinan y estructuran los requisitos obtenidos en el capítulo 2. El modelo de diseño es la base fundamental para la implementación, se enfoca en cómo va a estar estructurado e implementado el software. El establecimiento de la línea base de la arquitectura para el componente, posibilita una mayor comprensión del mismo y hace que aumenten las posibilidades de reutilizar tanto la arquitectura como el componente.

CAPÍTULO 4: Implementación del Sistema.

4.1 Introducción.

En general en este capítulo se presenta una muestra de los principales pasos del algoritmo para la construcción del componente como son: modelar el fondo, conversión de colores, eliminar sombra, obtener blobs, eliminar ruido y realizar seguimiento de objetos. Se exponen los diagramas de componente y de despliegue, así como los casos de prueba encargados de comprobar el correcto funcionamiento de la aplicación.

4.2 Análisis del algoritmo

El procesamiento de digital de imágenes es utilizado para identificar y clasificar los blobs, que no son más que los objetos y regiones de mucha o poca relevancia dentro de una imagen digital. En esta sección se muestra la técnica o pasos a seguir para realizar un correcto procesamiento de manera que se obtenga el resultado deseado.

El procesamiento comienza con la captura del video proveniente de las cámaras IP, al cual se le extraen los fotogramas o imágenes estáticas, de manera que se pueda aplicar los algoritmos indicados a continuación en el mismo orden en que aparecen.

- **Modelar el fondo a través de mezclas gaussianas.**

Primeramente se realiza una etapa de modelación de fondo con el objetivo de separar el frente del fondo de una imagen, y para ello, se aplica el modelo de mezclas de Gaussianas (MoG).

"El modelo MoG se basa en usar K Gaussianas por píxel, que se modelan a través de una media (μ), una desviación típica (σ) y un factor de peso (ω). La media indica el valor más probable de dicho píxel para cada píxel, la desviación típica lo que se puede alejar dicho valor medio por cada uno de los lados, y el peso indica que Gaussiana es la de mayor importancia. Destacar que la suma de los pesos de una Gaussiana para cada píxel es igual a 1." (Gómez, 2009)

El primer paso para aplicar MoG es inicializar los parámetros: media, desviación y factor de peso, los cuales representan el fondo de la imagen. Luego se inicializa la media de una de las Gaussianas en 1 a la primera imagen (por Ejemplo: la Gaussiana i=1), el resto de medias tendrá valores aleatorios. La primera Gaussiana modelará el pixel en la primera imagen, en la cual el peso w para k=1 debe ser un valor

próximo a 1, y para el resto será un valor próximo a 0, debido a que la suma de los pesos de las Gaussianas es igual a 1.

Para obtener los píxeles pertenecientes al fondo de la imagen digital, se obtiene una imagen diferencia entre los K modelos posibles; en caso de encontrarse algún parecido con respecto a alguna distribución, es decir, si existe una diferencia de c (2-3) veces el valor estimado de la desviación correspondiente a la distribución, entonces el píxel se marca como fondo y se actualizan los parámetros de dicha distribución a través de una media móvil:

$$\exists i \in [1, K] / \left| I_t(x, y) - \mu_{i,t}(x, y) \right| \leq c\,\sigma_{i,t}(x, y)$$

$$\Downarrow$$

$$\begin{cases} \mu_{i,t+1}(x, y) = \rho I_t(x, y) + (1-\rho)\mu_{i,t}(x, y) \\ \sigma^2_{i,t+1}(x, y) = \rho\left(I_t(x, y) - \mu_{i,t}(x, y)\right)^2 + (1-\rho)\sigma^2_{i,t}(x, y) \\ w_{i,t+1}(x, y) = w_{i,t}(x, y) \end{cases}$$

El factor $\rho = \alpha\, Pr\,(I_t(x,y) / \mu_{i,t-1}(x,y), \sigma^2_{i,t-1}(x,y))$, α es un parámetro predefinido al inicio de la ejecución, $\mu_{i,t}(x, y)$ es el valor de la media del píxel en el instante *t* e $I_t(x, y)$ es el valor del píxel en el instante *t* . Los píxeles que no sean marcados como background (fondo) serán marcados como foreground.

A continuación, se muestra en la figura 19 un ejemplo sobre el funcionamiento de dicho método:

Figure 19 *Ejemplo del método de mezclas de Gaussianas.*

- **Conversión de color de RGB a HSV.**

Luego de realizar la etapa de modelación de fondo, se lleva a cabo la etapa de eliminación de sombra pues una vez que se han separado el fondo y el frente de la imagen digital suelen aparecer sombras pertenecientes a los objetos que clasificados como parte del frente de la imagen. La eliminación de sombras se logra con efectividad aplicando el modelo de color HSV como una alternativa a los clásicos espacios de color como RGB.

HSV separa la intensidad y el color que se encuentra dentro de su espacio utilizando tres mapas de colores:

- **Tonalidad (Hue):** Representa el tipo de color. Es un grado de ángulo de 0 a 360º ó de 0 a 100%. Cada valor corresponde a un color: 0 es rojo, 60 amarillo y 120 es verde.
- **Saturación (Saturation):** Representa la distancia al eje de brillo blanco-negro. Los valores van de 0 a 100%.
- **Valor del color (Value):** Representa el brillo del color, es decir la altura en el eje blanco-negro. Los valores van de 0 a 100%.

A continuación se muestra la fórmula aplicada para obtener una imagen en el espacio de color HSV.

$$H_1 = \cos^{-1}\left(\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right)$$

$$M = \max(R,G,B)$$

$$m = \min(R,G,B)$$

$$H = H1, \quad si\ B \le G$$

$$H = 360º - H1, \quad si\ B > G$$

$$S = \frac{M-m}{M}$$

$$V = \frac{M}{255}$$

- **Eliminación de sombra**

Luego de obtener la imagen en espacio de color HSV, se calculan los tres mapas de colores: Tonalidad (H), Saturación (S) y Valor (V) con el propósito de detectar en la máscara de Foreground los píxeles pertenecientes a sombras. Primeramente, se utiliza el valor V entre la imagen de fondo y la imagen actual para filtrar los posibles píxeles pertenecientes a sombras. Posteriormente se calculan los valores H y S con el objetivo de obtener el cambio de cromaticidad entre la imagen actual y la imagen de fondo y de esta

forma determinar que píxel pertenece o no a la sombra. A continuación se describe la fórmula del algoritmo a aplicar.

$$SP_t(x,y) = \begin{cases} 1 & if\ \alpha \leq \frac{I_t^V}{B_t^V} \geq \beta \\ & \wedge \|I_t^S(x,y) - B_t^S(x,y)\| \leq \tau_S \\ & \wedge \|I_t^H(x,y) - B_t^H(x,y)\| \leq \tau_H \\ 0 & otherwise \end{cases}$$

"Donde α y β son constantes que marcan que la relación entre los valores de V entre la imagen actual (I^V_t) y la imagen del fondo (B^V_t) no superan ni rebajan un cierto umbral, y la diferencia entre los valores tanto de S (I^S_t y B^S_t) como de H (I^H_t y B^H_t) tampoco superan un cierto umbral. Los cálculos se hacen píxel a píxel (en el caso de dicho ejemplo, píxel para la posición (x,y))." (Gómez, 2009)

Para concluir, los píxeles clasificados como fondo se eliminan de la máscara anterior obteniéndose una nueva máscara sin sombras. Esta figura 20 muestra una máscara con sombra detectada, lista para eliminar.

Figure 20 *Ejemplo de eliminación de sombras.*

- **Obtención de los Blobs**

Tras la etapa de eliminación de sombra, se lleva a cabo una etapa de obtención de los blobs. Para realizar dicha tarea se utiliza la librería cvblobslib, la cual devuelve un listado de todos los blobs en la escena. Internamente, esta librería utiliza el algoritmo de componentes conexas para obtener los objetos y algunos

descriptores como área, perímetro, momentos y rectángulo envolvente (Bounding Box). En la siguiente figura 21 se muestra un ejemplo de obtención de blobs.

Figure 21 *Ejemplo de rectángulos englobantes*

- **Eliminación del ruido**

Existen algunos elementos dentro de la imagen que no deben ser detectados como blobs debido a su tamaño. Este problema se puede solucionar eliminando las regiones cuya área en píxeles es más pequeña que un umbral determinado. Para cada imagen, dicho umbral es calculado como un porcentaje del tamaño medio del rectángulo envolvente de un blob. Los rectángulos envolventes obtenidos cuyo tamaño sea menor al de dicho umbral, serán eliminados de la lista final de regiones consideradas como parte de la máscara de objetos. Se puede ver un ejemplo de este proceso en la figura 22.

Figure 22 *Ejemplo de eliminación de ruido*

- **Seguimiento de objetos**

Esta etapa realiza un seguimiento sobre los blobs obtenidos en la imagen cuyo ruido ya ha sido eliminado. Esta consiste en buscar semejanzas entre los blobs de la imagen anterior y la imagen actual de la serie de imágenes del video, ya procesadas.

Para ello se realiza un seguimiento a los blobs en las imágenes consecutivas con el fin de saber si un blob es el mismo que se encuentra en la imagen anterior de la secuencia de imágenes. Este seguimiento se realiza calculando la distancia entre el centro del rectángulo envolvente y de las áreas. La fórmula para realizar este cálculo se muestra a continuación:

$$d=\sqrt{(X2-X1)^2+(Y2-Y1)^2}$$

Se tiene un delimitador que puede ser vertical o horizontal cuando la persona pase por él automáticamente cuenta la persona, tanto las que salen como las que entra. En la figura 23 se muestra el proceso.

Figure 23 *Ejemplo de seguimiento de objetos*

4.3 Diagrama de Componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño, algunos estereotipos estándar de componentes son los siguientes:

- <<executable>> es un programa que puede ser ejecutado en un nodo.

- <<file>> es un fichero que contiene código fuente o datos.

- <<library>> es una librería estática o dinámica.

- <<document>> es un documento.

- <<table>> es una tabla de una base de datos.

- <<databases>> es una base de datos.

En la figura 24 se muestra a continuación el diagrama de componentes del sistema de Video Vigilancia Suria.

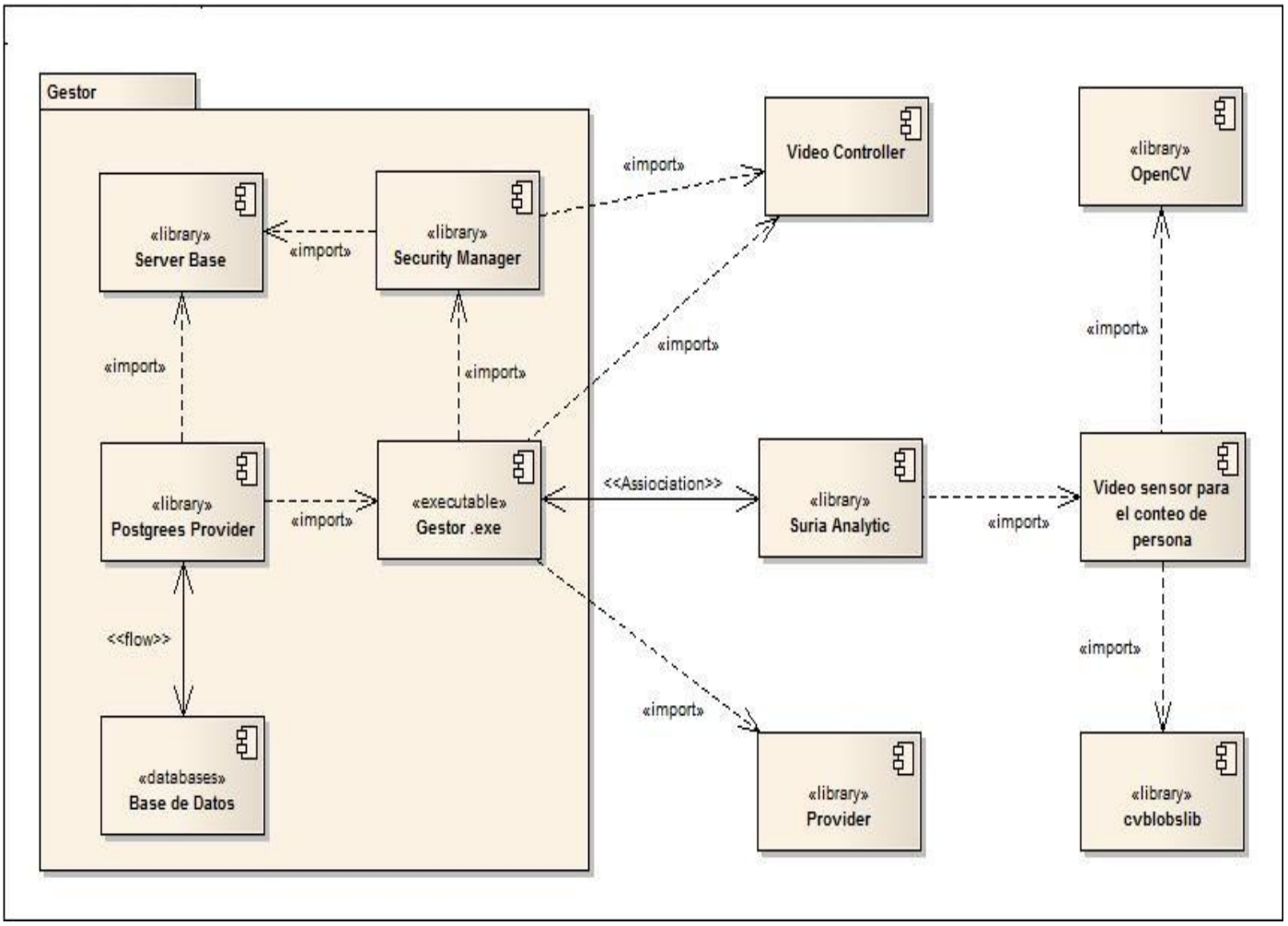


Figure 24 *Diagrama de componente del sistema Suria.*

4.4 Diagrama de Despliegue.

"Los diagramas de despliegue muestran la disposición física de los distintos nodos que entran en la composición de un sistema y el reparto de los programas ejecutables sobre estos nodos". (Visconti, et al.)

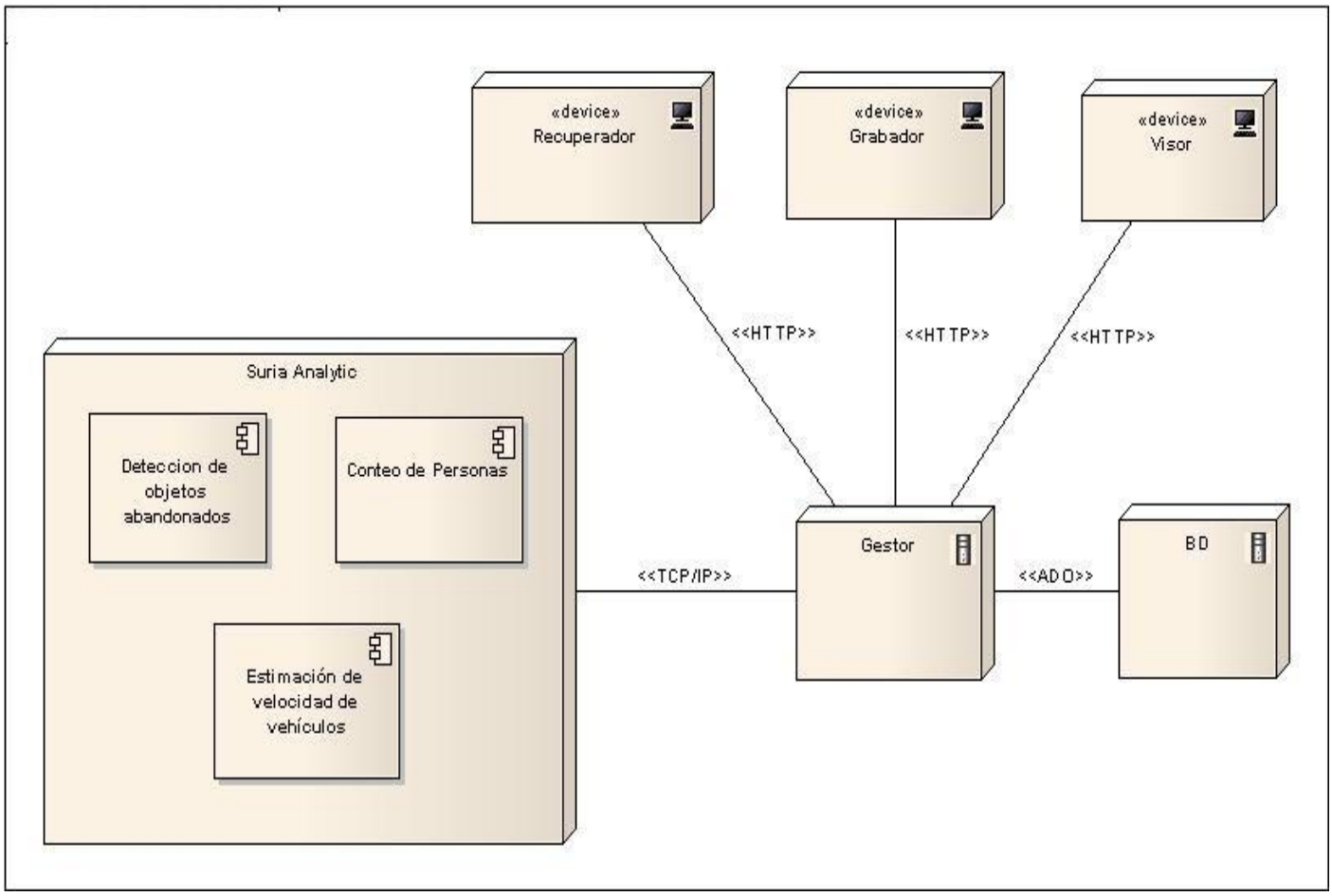


Figure 25 *Diagrama de despliegue del sistema.*

En la figura 25 se muestra el diagrama de despliegue de la aplicación que está representado por 6 nodos. Uno de ellos es el servidor Gestor, este se comunica con el módulo Suria Analytic a través del protocolo TCP/IP, también establece comunicación con el servidor de Base de Datos haciendo uso del protocolo ADO, y su vez se comunica con tres computadoras a través del protocolo HTTP.

4.5 Pruebas.

4.5.1 Pruebas de unidad.

Para realizar pruebas de unidad se selecciona el método de caja blanca, usando específicamente la técnica del camino básico. Con estas pruebas es posible evaluar el funcionamiento de la estructura interna de las funcionalidades del sistema, para verificar y revelar la calidad del mismo.

"Camino básico es un método que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un

conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizarán que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa." (Pressman, 2005)

Los pasos del diseño de pruebas a seguir mediante el camino básico son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo de flujo.
3. Se determina el conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

1. Grafo de flujo.

"Grafo de flujo o grafo del programa: este representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. (Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control)." (Pressman, 2005) Para la elaboración del mismo se utiliza los tres elementos siguientes:

- Nodos: representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión (bifurcación).

- Aristas: líneas que unen dos nodos.

- Regiones: áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región más.

2. Complejidad Ciclomática

"Para contar el número de ciclos diferentes que se siguen en un fragmento de código de un programa, habiendo creado una rama imaginaria desde el nodo de salida al nodo de entrada se utiliza la Complejidad Ciclomática (Cyclomatic Complexity), esta es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Es una de las métricas de software más ampliamente aceptada, ya que ha sido concebida para ser independiente del lenguaje". (Pressman, 2005)

El resultado obtenido en el cálculo de la complejidad Ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo:

1ra vía V (G)= Número de Regiones.

2da vía V (G)= Número de Aristas – Número de Nodos + 2

3ra vía V (G)= Número de Nodos Predicados +1

3. Camino independiente

Un camino independiente es cualquier camino del programa que introduce un nuevo conjunto de sentencias. El conjunto de caminos independientes de un grafo no es único. No obstante, a continuación, se muestran algunas heurísticas (término con el que nos referimos al método o procedimiento usado en la investigación o en el descubrimiento de algo) para identificar dichos caminos:

a) Elegir un camino principal que represente una función válida que no sea un tratamiento de error. Debe intentar elegirse el camino que atraviese el máximo número de decisiones en el grafo.
b) Identificar el segundo camino mediante la localización de la primera decisión en el camino de la línea básica alternando su resultado mientras se mantiene el máximo número de decisiones originales del camino inicial.
c) Identificar un tercer camino, colocando la primera decisión en su valor original a la vez que se altera la segunda decisión del camino básico, mientras se intenta mantener el resto de decisiones originales.
d) Continuar el proceso hasta haber conseguido tratar todas las decisiones, intentando mantener como en su origen el resto de ellas.

4. Derivación de casos de prueba:

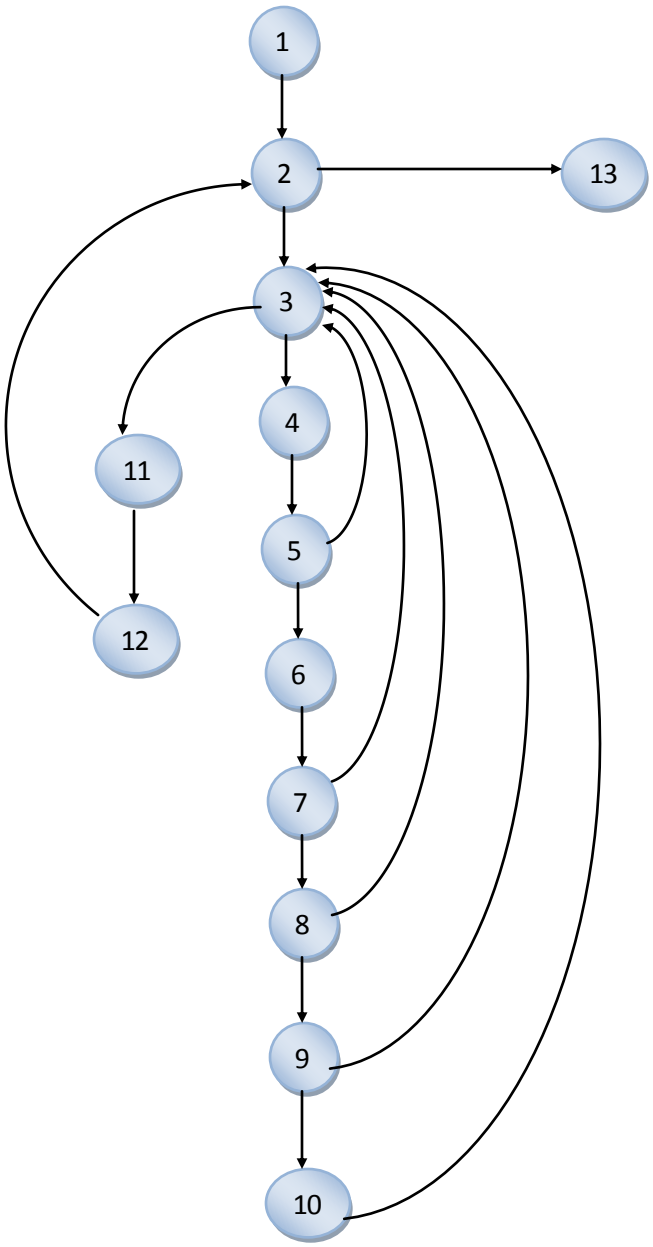
Luego de tener elaborados los Grafos de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

Pruebas de unidad para el caso de uso Procesar sombra.

1- Grafo de flujo.

```
void Procesado_Sombra::ShadowDetectionRemoval(IplImage *bkg, IplImage *frame, IplImage *fgWithoutShadow)
{
    1 IplImage *hsvBkg = cvCreateImage(cvGetSize(bkg),8,3);
    1 this->Convert2HSV(bkg,hsvBkg);
    1 IplImage *hsvFrame = cvCreateImage(cvGetSize(bkg),8,3);
    1 this->Convert2HSV(frame,hsvFrame);
    1 int dh = 0;
    2 for(int i=0;i<hsvFrame->height;i++)
    {
        3 for(int j=0;j<hsvFrame->width;j++)
        {
            4 CvScalar frameValue;
            4 CvScalar bkgValue;
            4 frameValue.val[0]= (float)((uchar *)(hsvFrame->imageData + i*hsvFrame->widthStep))[j*hsvFrame->r
            4 frameValue.val[1]= (float)((uchar *)(hsvFrame->imageData + i*hsvFrame->widthStep))[j*hsvFrame->r
            4 frameValue.val[2]= (float)((uchar *)(hsvFrame->imageData + i*hsvFrame->widthStep))[j*hsvFrame->r
            4 bkgValue.val[0]= (float)((uchar *)(hsvBkg->imageData + i*hsvBkg->widthStep))[j*hsvBkg->nChannels
            4 bkgValue.val[1]= (float)((uchar *)(hsvBkg->imageData + i*hsvBkg->widthStep))[j*hsvBkg->nChannels
            4 bkgValue.val[2]= (float)((uchar *)(hsvBkg->imageData + i*hsvBkg->widthStep))[j*hsvBkg->nChannels
            4 CvScalar sf;
            4 sf.val[0] = (float)((uchar *)(fgWithoutShadow->imageData + i*fgWithoutShadow->widthStep))[j];
            5 if(sf.val[0] == 255)
            {
                6 dh = abs(frameValue.val[0]-bkgValue.val[0]);
                7 if(dh <= tau_h)
                {
                    8 if( frameValue.val[1] - bkgValue.val[1] <= tau_s)
                    {
                        9 if(frameValue.val[2]/bkgValue.val[2] >= alpha && frameValue.val[2]/bkgValue.val[2] <= beta)
                        {
                            10 ((uchar *)(fgWithoutShadow->imageData + i*fgWithoutShadow->widthStep))[j] = 0;

                        }
                    }
                }
            }
        }11
    }12
    13 cvReleaseImage(&hsvBkg);
    13 cvReleaseImage(&hsvFrame);
}
```



2- Cálculo de la Complejidad Ciclomática

V (G) = Numero de Aristas – Número de Nodos + 2

V (G) = 18 – 13 + 2

V (G) = 7

3- Caminos Independientes:

1: 1-2-13

2: 1-2-3-4-5-3-11-12-2-13

3: 1-2-3-4-5-6-7-3-11-12-2-13

4: 1-2-3-4-5-6-7-8-3-11-12-2-13

5: 1-2-3-4-5-6-7-8-9-3-11-12-2-13

6: 1-2-3-4-5-6-7-8-9-10-3-11-12-2-13

7: 1-2-3-11-12-2-13

Pruebas de unidad para el caso de uso Extraer frente.

1- Grafo de flujo.

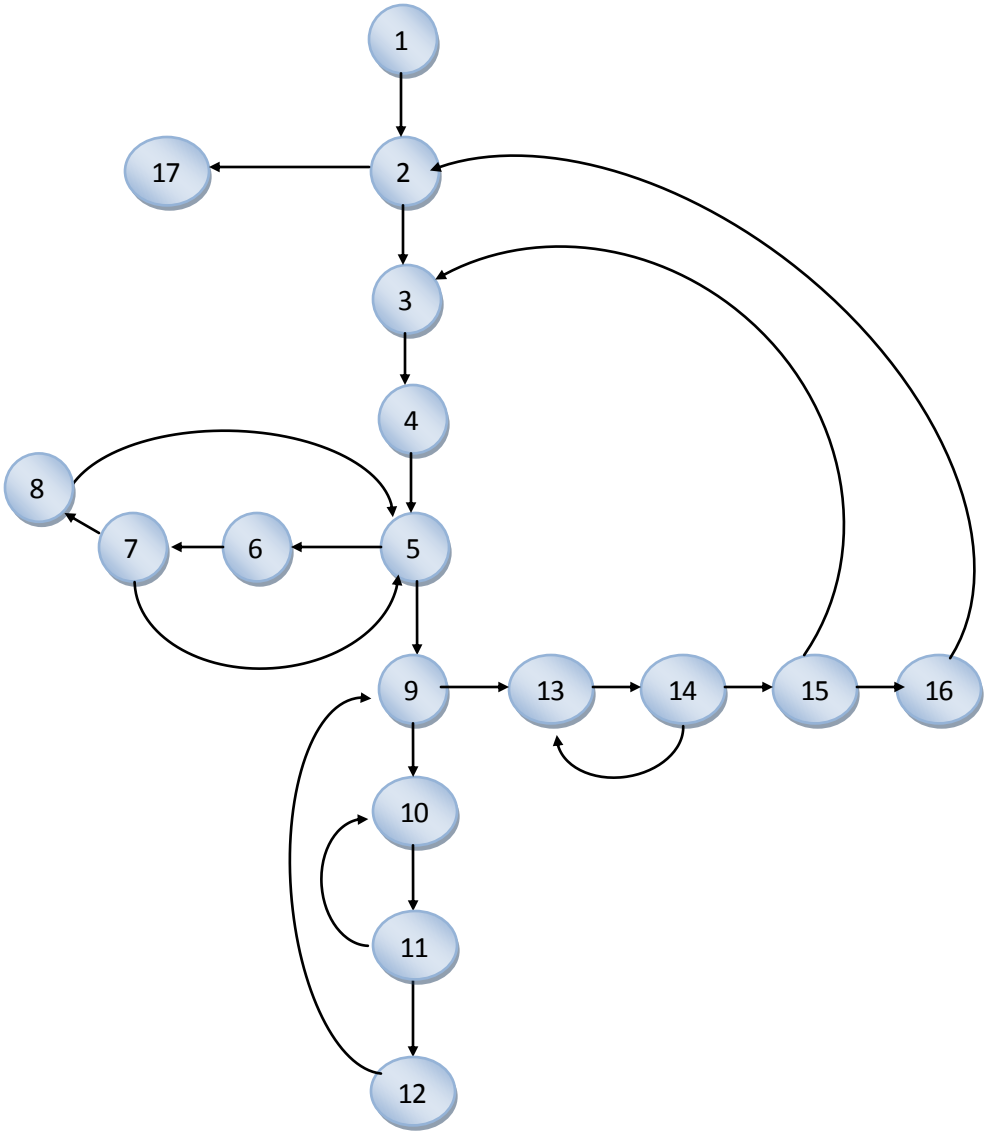
```
void Frente::update8uC3(CvGaussBGModel* bg_model)
{
    1 int K = bg_model->params.n_gauss;
    1 float T = bg_model->params.bg_threshold;
    1 int nchannels = bg_model->background->nChannels;
    1 int height = bg_model->background->height;
    1 int width = bg_model->background->width;
    1 MixData3 *g_point = (MixData3 *) ((CvMat*)(bg_model->g_point))->data.ptr;
    1 MixData3 *mptr = g_point;

    2 for(int y=0; y<height; y++)
    {
        3 for (int x=0; x<width; x++, mptr+=K)
        {
            4 int pos = bg_model->background->widthStep*y + x*nchannels;
            4 float mean[3] = {0.0, 0.0, 0.0};
            4 float wsum = 0.0;
            4 int kForeground = K;

            5 for(int k=0; k<K; k++)
            {
                6 wsum += mptr[k].weight;
                7 if (wsum > T)
                {
                    8 kForeground = k+1;
                    8 break;
                }
            }
```

```
9  for(int k=0; k<kForeground; k++)
   {
10     for(int m=0; m<nchannels; m++)
       {
11         mean[m] += mptr[k].weight * mptr[k].mean[m];
       }
   }12

13 for(int m=0; m<nchannels; m++)
   {
14     bg_model->background->imageData[pos+m] = (uchar) (mean[m]/wsum);
   }
}15
}16
}17
```



2- Cálculo de la Complejidad Ciclomática

V (G) = Numero de Aristas – Número de Nodos + 2

V (G) = 22 – 17 + 2

V (G) = 7

3- Caminos Independientes

1: 1-2-17

2: 1-2-3-4-5-6-7-5-9-13-14-15-16-2-17

3: 1-2-3-4-5-6-7-5-9-10-11-12-9-13-14-15-16-2-17

4: 1-2-3-4-5-6-7-8-5-9-13-14-15-16-2-17

5: 1-2-3-4-5-6-7-8-5-9-10-11-12-9-13-14-15-16-2-17

6: 1-2-3-4-5-6-7-5-9-10-11-12-9-13-14-15-16-2-17

7: 1-2-3-4-5-6-7-5-9-13-14-15-16-2-17

Resultados obtenidos para los casos de usos.

Nombre CU: Procesar sombra.

Para el Camino 6: 1-2-3-4-5-6-7-8-9-10-3-11-12-2-13

Caso de Prueba: Probando la función ShadowDetectionRemoval.

Resultado: Detectar y eliminar las sombras que se encuentren en la imagen.
.

Nombre CU: Extraer frente.

Para el Camino 5: 1-2-3-4-5-6-7-8-5-9-10-11-12-9-13-14-15-16-2-17

Caso de Prueba: Probando la función update8uC3.

Resultado: Actualizar el fondo de una imagen de 3 canales.

No del camino.	Caso de Prueba.	Objetivo.	Resultado.
6	Probando la función ShadowDetectionRemoval.	Detectar y eliminar las sombras que se encuentren en la imagen.	Satisfactorio.
5	Probando la función update8uC3.	Actualizar el fondo de una imagen de 3 canales.	Satisfactorio.

4.6 Resultados experimentales.

"El índice de precisión y detección son dos criterios comúnmente usados para evaluar la eficiencia en los sistemas de recuperación de información". (Yingzi and Chein-I, 2003). Desde el punto de vista del conteo de personas, el índice de precisión (presicion rate) cuantifica el total de personas correctamente contadas en el total de personas contadas por el sistema. Siendo el recall rate el porciento de personas contadas correctamentes en el total de personas que el sistema debió contar. Formalmente están definidos por las siguientes ecuaciones:

$$presicion\ rate = \frac{CC}{CC+Fp}\ recall\ rate = \frac{CC}{TP}$$

Donde CC es el número de personas que el sistema contó correctamente, *TP* es el total de personas que el sistema debió contar y *Fp* (Falsos Positivos) es todo lo que el sistema cuenta que no es persona como: la sombra, la iluminación, el ruido, etc.

Para evaluar experimentalmente el método de conteo de persona propuesto se grabaron 2 videos en diferentes ambientes y posición de las cámaras. En la Tabla se muestra detalladamente los resultados experimentales obtenidos.

Parámetro	Valor
Conteos Correctos(CC)	19
Total de personas(TP)	22
Falsos Positivos	0
Falsos Negativos	3
recall rate	86%
presicion rate	100%

Tabla 6 *Resultados experimentales*

Hay un 86% de factibilidad o grado de cumplimiento respecto a las funcionalidades del software pues solo se logra detectar este por ciento de personas contadas correctamente. Es insuficiente la cantidad de videos grabados para la realización de las pruebas por la escasez de cámaras IP, por lo que el trabajo no es del todo óptimo

4.7 Conclusiones

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 unidades para inspeccionar y verificar la calidad y eficiencia de los requisitos especificados. La validación de resultados experimentales demuestra que hay un índice de precisión 100% factible y un 86% de recall.

CONCLUSIONES GENERALES

Actualmente existen muchos sistemas de video vigilancia que incorporan videos sensores para el conteo de personas, pero son propietarias y altamente costosas. Con el incorporamiento del procesamiento inteligente de imágenes y videos digitales se implementó un sistema que permite contar personas a partir de flujos de videos obtenidos desde cámaras IP en el sistema de Video Vigilancia Suria. Se utilizaron técnicas de gran utilidad para lograr la solución final, como el Modelo de Mezclas Gaussianas para modelar el frente (foreground) y el fondo (background) de los videos. También con la utilización de la valiosa librería para el preocesamiento de imágenes OpenCV, se logró obtener los objetos presentes y hacerles un seguimiento para contarlos.

Es de gran importancia este sistema porque posibilita la contabilización automática de personal en los entornos monitoreados que pueden ser: tiendas, estadios y calles. Ya sea para tener constancia de la cantidad de clientes o para llevar estadísticas de que horarios son más concurridos. El sistema posee la calidad y eficiencia requerida para cumplir con los requisitos especificados luego de aplicarles algunas pruebas para su funcionamiento.

Se obtuvo un componente confiable y configurable dependiendo de las necesidades del cliente, que después de las pruebas realizadas mostró un índice de precisión de 100% y un recall de 86 %. Con esto se evidencia que el sistema puede mejorar en cuanto al procesamiento de las sombras que aparezcan en la escena y en cuanto a la discriminación persona-objeto.

RECOMENDACIONES

La autora del trabajo de diploma recomienda:

- Mejorar el seguimiento de los objetos para manejar eficientemente las oclusiones parciales o totales que puedan ocurrir.
- Analizar estadísticamente la eliminación de sombras para ajustar los parámetros a la mayor cantidad de ambientes posibles.
- Investigar recursos y herramientas que permitan capturar el flujo de video de las cámaras IP de manera más rápida y estable.
- Capturar el flujo de video de las cámaras IP con la herramienta o recurso seleccionado.

TRABAJOS CITADOS

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

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

—. **Yingzi, Du and Chein-I, Chang. 2003.** 2003, Journal of Electronic Imaging.

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

Gómez, Álvaro Bayona. 2009. *Detección de Objetos Abandondos/Robados en secuencias de video-seguridad.* 2009.

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

Gracia, Joaquin. 2011. *IngenieroSoftware.* [Online] 2011. http://www.ingenierosoftware.com.

Jacobson, I., G. Booch, and J. Rumbaugh. 1998. *El lenguaje Unificado de Modelado, Manual de referencia.* s.l. : Addison Wesley, 1998.

—. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Larman, C. 1999. *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.* 1999.

Letelier, Patricio and Penadés, Maria del Carmen. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [Online] [Cited: Noviembre 10, 2010.] www.willydev.net/descargas/masyxp.pdf.

Martín, Sonsoles Herrero. 2009. *ANÁLISIS COMPARATIVO DE TÉCNICAS DE SEGMENTACIÓN DE SECUENCIAS DE VÍDEO BASADAS EN EL MODELADO.* 2009.

Mora, Miguel Katrib. 1997. *Programación orientada a objeto en C++.* 1997.

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

Torres, Alejandro Dominguez. 1996. *Procesamiento Digital de Imágenes.* México : s.n., 1996.

Visconti, Marcello and Astudillo, Hernán. *Fundamentos de Ingeniería de Software.*

BIBLIOGRAFÍA CONSULTADA

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

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

Automated system for text detection in individual video images. **Yingzi, Du and Chein-I, Chang. 2003.** 2003, Journal of Electronic Imaging.

—. **Yingzi, Du and Chein-I, Chang. 2003.** 2003, Journal of Electronic Imaging.

Ciberaula. 2006. Patrones de Diseño en aplicaciones Web con Java J2EE. *Ciberaula-Java.* [Online] 2006. [Cited: Abril 10, 2010.] http://java.ciberaula.com/articulo/diseno_patrones_j2ee/.

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

Datacyl. *Datacyl Solutions.* [Online] [Cited: Octubre 1, 2010.] http://www.datacyl.com/sistemas-televigilancia-videovigilancia-valladolid.php.

2009. Digital Screens for digital signage. [Online] 2009. [Cited: Octubre 17, 2010.] http://www.dsdigitalsignage.com/admin/_venco/archivos/medidor/0000008/itally%20brochure_espa%C3%B1ol.pdf.

GeoVision. *GeoVision.* [Online] [Cited: Octubre 1, 2010.] http://www.geovision.com.tw/english/3_1_Video.asp.

Gómez, Álvaro Bayona. 2009. *Detección de Objetos Abandondos/Robados en secuencias de video-seguridad.* 2009.

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

Gracia, Joaquin. 2011. *IngenieroSoftware.* [Online] 2011. http://www.ingenierosoftware.com.

Jacobson, I., G. Booch, and J. Rumbaugh. 1998. *El lenguaje Unificado de Modelado, Manual de referencia.* s.l. : Addison Wesley, 1998.

—. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Larman, C. 1999. *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.* 1999.

Letelier, Patricio and Penadés, Maria del Carmen. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [Online] [Cited: Noviembre 10, 2010.] www.willydev.net/descargas/masyxp.pdf.

Martín, Sonsoles Herrero. 2009. *ANÁLISIS COMPARATIVO DE TÉCNICAS DE SEGMENTACIÓN DE SECUENCIAS DE VÍDEO BASADAS EN EL MODELADO.* 2009.

2010. MAXCO SECURITY. [Online] 2010. [Cited: Octubre 17, 2010.] http://www.maxcosecurity.net/conteopersonas-1.html.

Molina, R. *Introducción al Procesamiento y Análisis de Imágenes Digitales.*

Mora, Miguel Katrib. 1997. *Programación orientada a objeto en C++.* 1997.

Moreno, Francisco. 2000. *Introducción a la POO.* 2000.

2010. Pixelco Blog. [Online] 2010. [Cited: Diciembre 2, 2010.] http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/.

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

Sistemas de Seguridad, Vigilancia y Monitoreo Remoto. [Online] [Cited: Noviembre 19, 2010.] http://www.gscssoftware.com/teccamaraip.htm.

2010. Sparx Systems. [Online] 2010. [Cited: Diciembre 2, 2010.] http://www.sparxsystems.com.ar/index.html.

Torres, Alejandro Dominguez. 1996. *Procesamiento Digital de Imágenes.* México : s.n., 1996.

Visconti, Marcello and Astudillo, Hernán. *Fundamentos de Ingeniería de Software.*

VISUAL TOOLS, S.A. [Online] [Cited: Octubre 17, 2010.] http://www.visual-tools.com/productos/analisis-de-video-conteo-de-personas-y-control-pos.

ANEXOS

Anexo 1. Descripción extendida de los casos de uso

CU- 2	Extraer frente
Actor:	Sistema Video Vigilante
Propósito:	Obtener los objetos nuevos que aparecen en la escena.
Resumen:	Se sustrae el fondo para obtener los objetos en movimiento que aparecen en la escena.
Precondiciones:	Se obtuvieron los fotogramas para procesar.
Referencia:	RF# 3, RF# 4, RF# 5
Prioridad:	Crítico
Flujo normal de Eventos	
Acción del Actor	**Respuesta del sistema**
	1. Se crea un modelo de fondo gaussiano. 2. Se actualiza el modelo de fondo. 3. Se obtiene la máscara de frente. 4. Se chequea si se procesan las sombras. 5. Se segmenta la imagen para obtener los objetos. 6. Los objetos muy pequeños son eliminados.
Flujo Alternativo Paso 4	
	7. Si se va a procesar sombras, se ejecuta CU5. En caso contrario, se continúa el procesamiento.
Poscondiciones:	Se obtienen los objetos de los fotogramas.

Tabla 7 *Descripción extendida del caso de uso "Extraer frente."*

CU-3	Procesar Sombra.
Actor:	Sistema de Video Vigilancia
Propósito:	Eliminar las sombras de los objetos en la imagen.
Resumen:	El sistema solicita que se procesen las sombras presentes en la escena, el caso de uso permite detectar las partes que son sombras pertenecientes a los objetos en la imagen y eliminarlas de la misma.
Precondiciones:	Se ha obtenido la máscara del frente de la imagen.
Referencia:	RF# 6
Prioridad:	Crítico

Flujo normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1. El caso de uso inicia cuando el Sistema de Video Vigilancia hace la petición de procesar sombras.	2. De los objetos detectados en la escena, se analiza cuáles píxeles pertenecen a las sombras. 3. El caso de uso finaliza cuando se eliminan de los objetos, los píxeles que son sombras.
Poscondiciones:	Los objetos del frente no poseen sombras.

Tabla 8 *Descripción extendida del caso de uso "Procesar Sombra."*

CU- 4	Hacer seguimiento de objeto.
Actor:	Sistema Video Vigilante
Propósito:	Verificar que el objeto se mueva a través del tiempo.
Resumen:	Se chequea que los objetos aparecidos en la escena se muevan uniformemente para llevar un registro de su dirección o desplazamiento.
Precondiciones:	Se obtuvieron los objetos de los fotogramas.
Referencia:	RF# 7
Prioridad:	Crítico

Flujo normal de Eventos	
Acción del Actor	**Respuesta del sistema**
	1. Se verifica que los objetos nuevos aparecidos en la imagen ya estén siendo seguidos. 2. Los objetos en seguimiento que traspasen un

	delimitador son contados como salientes o entrantes.
Flujo Alternativo Paso 1	
	1. Si los objetos nuevos no están siendo seguidos se comienza su seguimiento.
Flujo Alternativo Paso 2	
	2. Si los objetos no pasan el delimitador se procesa el siguiente fotograma y se ejecuta el paso 1 de nuevo.
Poscondiciones:	Se contaron los objetos que entraron y salieron en la escena.

Tabla 9 *Descripción extendida del caso de uso "Hacer seguimiento de objeto."*

GLOSARIO

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

C++: Versión de C orientada a objetos creada por Bjarne Stroustrup. C++ se ha popularizado porque combina la programación tradicional en C con programación orientada a objetos.

Frame: Imágenes que compone un video.

Fotogramas: Imágenes contenidas en un video.

HTML: Acrónimo inglés **H**yper**T**ext **M**arkup **L**anguage (Lenguaje de Marcado Hipertextual), lenguaje de marcación diseñado para estructurar texto y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web.

Herramientas CASE: CASE es el acrónimo de los vocablos en inglés: Computer Aided Software Engineering, que en español equivalen a: Ingeniería de Software Asistida por Computadoras. Estas herramientas son útiles en todos los aspectos del ciclo de vida de desarrollo del software: realización de modelos y diagramas, diseño de proyectos, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

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.

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

Píxel: Menor unidad que compone una imagen en un medio electrónico.

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.

Video: Número consecutivo de imágenes y sonido que representan escenas en movimiento.