UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Título: “Desarrollo de un video sensor para la detección de objetos abandonados.”

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

Autor: Yuliet Hernández Expósito

Tutor: Ing. Reinier Pupo Ruiz

Junio 2011

Ciudad de La Habana

“Año del 53 Aniversario del Triunfo de la Revolución”

"El futuro de nuestra Patria tiene que ser necesariamente un futuro de hombres de ciencia, de hombres de pensamiento"

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los __ días del mes de __ del año 2011.

Autores:

__Yuliet Hernández Expósito__

Autor

Tutores:

_____Reinier Pupo Ruiz_____

Tutor principal

DATOS DE CONTACTO

Tutor:

Tutor principal: Ing. Reinier Pupo Ruiz.

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

Correo electrónico: rpupo@uci.cu

Dedicatoria

A mi familia que quiero y adoro tanto.

Gracias por hacerme la mujer que soy hoy.

Los quiero.

AGRADECIMIENTOS

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

Mi papá José que siempre me enseñó el valor del sacrificio y el trabajo.

A mi mamá Alicia, que me enseñó la perseverancia y a luchar por lo que se quiere.

Mi hermanito Danito que siempre contaba los días y los minutos para que pudiera graduarme.

A mi abuelita linda Elba que siempre me dio ánimos para seguir adelante.

A mi novio Yanier 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 mis primas Yaneysi y Elizabeth que esperaron deseosas este día.

A todas las personas que me brindaron su apoyo y ayuda.

A Lizandra Michelena que fue como una compañera de tesis para mí.

A mis amigas y amigos que siempre estaban preocupados por mí: Estela, Betty, René, Yiyi, Yamila, Yane y todos los demás que aunque no los mencione, no dejan de ser importantes para mí.

Gracias por ayudarme a llegar hasta aquí.

RESUMEN

El control que brindan los sistemas de video vigilancia a los interiores y alrededores de empresas e instituciones, les ha facilitado convertirse en una poderosa herramienta de seguridad. La presencia de cámaras integradas a estos sistemas es un hecho cotidiano en cualquier lugar de interés. Su evolución tecnológica ha posibilitado poder vigilar lugares casi imposibles de controlar.

El presente trabajo se desarrolló debido a la necesidad de añadir un video sensor al Sistema SURIA, posibilitando que este detecte objetos abandonados en un área determinada. Para su desarrollo se propone la utilización de librerías gratuitas como OpenCV, la cual cuenta con funciones factibles para el procesamiento inteligente del video, compatible con el marco de trabajo seleccionado (Visual Studio .Net 2010) y el lenguaje C++.

Como resultado, se ha creado un algoritmo que permita mejorar la seguridad en las instituciones del país. Su integración al Sistema RURIA constituye un gran aporte en cuanto a la seguridad y sirve de ayuda a que este pueda ser lanzado al mercado y competir con los demás sistemas.

Palabras claves: cámaras IP, flujos de videos, imagen digital, objetos abandonados, procesamiento inteligente, video sensor.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el mundo actual, las tecnologías han alcanzado un desarrollo vertiginoso a nivel mundial por lo que la informatización de la sociedad se ha convertido en un proceso imparable. Debido a la creciente exigencia del ser humano, investigadores y especialistas han creado día a día nuevos hardwares y softwares que son utilizados por las empresas para facilitar y agilizar el trabajo con las computadoras y la red. Uno de estos avances lo constituyen los sistemas de video vigilancia los cuales a través de los años han evolucionado enormemente.

En un comienzo se utilizaron estos sistemas en distintos países como Estados Unidos e Inglaterra. Poco a poco dada su importancia y aplicación, se fueron extendiendo a otros países. Aunque en un inicio lo que se utilizaba era la tecnología analógica, lo cual provocaba muchas molestias a los usuarios. Tiempo después vino la digitalización lo cual fue un enorme paso de desarrollo para los sistemas de video vigilancia. Con la ayuda del procesamiento inteligente de video, estos sistemas se han convertido en herramientas más eficientes y de gran calidad, aportando grandes ventajas a la seguridad al automatizar la toma de decisiones.

La necesidad de vigilancia de lugares públicos y privados actualmente ha cobrado un auge estremecedor, ya que se considera muy difícil controlar la entrada y salida de personas tanto en horario de trabajo como fuera de este. Por tal razón, una gran cantidad de especialistas e ingenieros se han dedicado a investigar sobre este tema, aportando día a día nuevas ideas. Se han creado importantes tecnologías que permiten enviar flujos de videos mediante computadoras por la red de manera continua y en tiempo real, así como cámaras que son capaces de ver en la oscuridad con rayos infrarrojos y otras que son resistentes al agua.

En Cuba actualmente, se utilizan estos sistemas fundamentalmente en lugares de gran riesgo de robo o atentados. Muchos hoteles de nuestro país cuentan con sistemas de seguridad por cámaras que controlan sus diferentes áreas.

Estos sistemas de video vigilancia tienen muchísimas aplicaciones tales como: vigilancia de establecimientos comerciales, bancos, oficinas, ayuntamientos, policía, edificios públicos, aeropuertos, monitoreo del tráfico en un puente, vigilancia en condiciones de absoluta oscuridad utilizando luz infrarroja, vigilancia en vehículos de transporte público (autobuses, trenes, barcos), vigilancia de los niños en el hogar,

en la escuela, parques, guarderías, etc. Las personas tienen grandes expectativas en estos nuevos sistemas de vigilancia por lo que se han desarrollado nuevas herramientas que permiten realizar reconocimiento facial, supervisión de tráfico, detección de comportamiento, es decir, si una persona se encuentra corriendo, merodeando o solo caminando por el área.

En el año 2002 se funda la Universidad de Ciencias Informáticas (UCI). Inmediatamente se comienzan a realizar actividades de investigación y desarrollo de software y se responsabiliza de una gran parte de proyectos nacionales que se llevan a cabo para informatizar el país. Una de las investigaciones anteriormente mencionadas que se realizan en este centro, es sobre el área de análisis y procesamiento de imágenes a través de flujos de video de seguridad. Esta se ha convertido en un tema muy interesante y utilizado desde muchos años atrás debido a su gran importancia y aplicación.

Actualmente en la UCI se puede hallar el Sistema de Video Vigilancia en la Facultad 6, perteneciente al departamento de Señales Digitales, donde se desarrollan herramientas entre las cuales se encuentra el video sensor para la Detección de Objetos Abandonados. Este permitirá mantener una estrecha vigilancia y evitar riesgos como por ejemplo el de que se produzca un atentado. Dicho sistema será utilizado a lo largo de todo el territorio cubano para la protección de sus empresas, áreas públicas, bancos, almacenes, aeropuertos, etc.

La **situación problemática** planteada será**:** Las personas son incapaces de llevar el control exacto de lo que ocurre en varias cámaras a la vez, por lo que es peligroso pasar por alto un detalle que puede conllevar a un hecho de alto riesgo. El proyecto Video Vigilancia SURIA necesita de un mecanismo que permita la detección automática de objetos abandonados 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 pueda detectar objetos abandonados a partir de flujos de videos obtenidos de cámaras IP?

La presente investigación tiene como **objeto de estudio:** el procesamiento inteligente de flujos de videos. A partir del objeto de estudio planteado anteriormente, se determina como el **campo de acción**: la detección de objetos abandonados en flujos de videos obtenidos de cámaras IP en el Sistema de Video Vigilancia SURIA. Para dar solución al problema, se propone como **objetivo general:** Desarrollar un video sensor que

procese los flujos de video de cámaras IP y detecte objetos que hayan sido abandonados en el entorno monitoreado.

Planteando como **idea a defender**: La construcción y puesta en funcionamiento de un Video Sensor dentro del Sistema SURIA logrará que este sea capaz de encontrar objetos que hayan sido abandonados.

Las **tareas de la investigación** planificadas para resolver el problema y alcanzar el objetivo planteado son:

1- Analizar los sistemas existentes en la actualidad.
2- Identificar algoritmos y librerías que permitan el procesamiento de los videos e imágenes digitales que puedan ser utilizados en el desarrollo del video sensor.
3- Realizar toda la documentación asociada al proceso de desarrollo.
4- Creación de un video sensor para la detección de objetos abandonados.
5- Creación de un demo con el objetivo de validar el resultado esperado.
6- Realizar pruebas al video sensor para verificar su funcionamiento.

Para dar solución a las tareas planificadas, se utilizan varios **Métodos Científicos**, para así garantizar la calidad de la investigación. Estos métodos se clasifican de dos formas: los Métodos Teóricos y los Métodos Empíricos.

Según (Universidad de las Ciencias Informáticas, 2010), "*los Métodos Teóricos permiten estudiar las características del objeto de investigación que no son observables directamente, facilitan la construcción de modelos e hipótesis de investigación, crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, etc.*", en otras palabras, los métodos teóricos no son más que aquellos métodos que posibilitan estudiar más bien lo teórico, la idea, el análisis del objeto, lo que no es observable directamente.

Según (Universidad de las Ciencias Informáticas, 2010), *"los Métodos Empíricos describen y explican las características fenomenológicas del objeto. Representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional."*, es decir, los métodos empíricos no son más que aquellos métodos que posibilitan estudiar lo práctico del objeto, lo que se observa y demostrar las cosas mediante ejemplos prácticos.

Entre los **Métodos Teóricos** se utilizará el Histórico-Lógico ya que este método posibilita que se realice un estudio del estado del arte, así como acerca de las metodologías a usar. También se hará uso del Analítico-

Sintético pues este método sirve para realizar un estudio de la bibliografía referente a los diferentes conceptos asociados al tema de detección de objetos abandonados. Además de la Modelación, para estudiar las relaciones y cualidades del objeto de estudio utilizando diagramas realizados con las herramientas de modelado escogidas y a través del lenguaje de modelado UML.

De los **Métodos Empíricos** se usará el Experimento ya que este método permite el estudio de un objeto que se escoja con las condiciones creadas anteriormente para el experimento y así demostrar si la herramienta funciona correctamente o no.

El presente trabajo está distribuido en **3 Capítulos**. El primer capítulo llamado Fundamentación Teórica, donde se incluye las herramientas de modelado, el lenguaje a utilizar, definiciones generales que ayudaran a entender el contenido del presente trabajo así como una breve explicación de la metodología a utilizar, las tendencias actuales y otros temas de interés. También se selecciona la plataforma, el lenguaje en que se desarrollará, la metodología y las herramientas que se utilizarán. El Capítulo 2: Análisis y Diseño, donde se identifican los requisitos que se deben tener en cuenta durante la implementación del video sensor para la detección de objetos abandonados. También se propondrá una solución de cómo se construirá el producto. El Capítulo 3: Implementación, aquí es donde se realiza la implementación del video sensor para la detección de objetos abandonados.

CAPÍTULO 1. **FUNDAMENTACIÓN TEÓRICA**

En este capítulo se hace referencia de un estudio del arte acerca del tema tratado tanto a nivel internacional como nacional. Se aborda sobre las tendencias actuales de las técnicas, tecnologías y metodologías para el desarrollo del sistema. Se definen conceptos y plantean ventajas y desventajas en las cuales se basa la selección de las tecnologías y técnicas.

1.1. Definiciones Generales

Imagen digital: Según (Web del Profesor, 2010), "*es cualquier imagen fija o en movimiento, que se capture en un medio electrónico y que se represente como un archivo de información leído como una serie de pulsos eléctricos.",* es decir, una imagen digital no es más que una representación bidimensional, producto del desarrollo de la Informática. Ver Ilustración 1.

Ilustración 1: Imagen digital

Píxel: Según (Web del Profesor, 2010), "*es la abreviatura de la expresión inglesa Picture Element (Elemento de Imagen) y es la unidad más pequeña que encontraremos en las imágenes compuestas por mapa de bits. El píxel es la unidad mínima en que se divide la retícula de la pantalla del monitor y cada uno de ellos tiene diferente color.",* es decir, es la menor unidad que compone una imagen en una computadora o cualquier medio electrónico, compuesta a su vez por mapas de bits. Ver Ilustración 2.

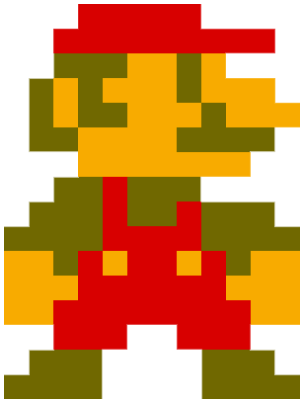


Ilustración 2: Pixel

Video: Según (Universidad de las Ciencias Informaticas, 2008/2009), "*Suele llamarse video a la captura, grabación, almacenamiento, y reconstrucción de una serie de imágenes y sonidos, las cuales representan escenas en movimiento".* No son más que una gran cantidad de marcos de las imágenes que se integran todas juntas para formar el video. Pueden o no estar acompañadas de sonido y se basan en píxeles.

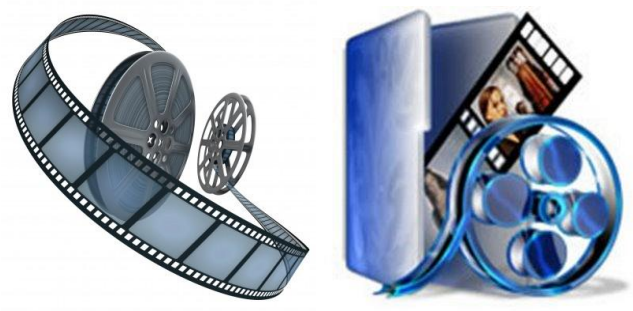


Ilustración 3: Video

Video inteligente: Según (Revista Negocios de Seguridad, 2010), "*el video inteligente permite convertir los datos de video utilizando técnicas de procesamiento de imágenes en información procesable que será analizada mediante la aplicación de algoritmos basados en Inteligencia Artificial y Computer Vision Systems, con el objetivo de tomar decisiones automatizadas."*, en otras palabras, un video inteligente no es más que aportar inteligencia al video a través de algoritmos que le permiten procesar las imágenes de manera más eficiente e inteligente, automatizando así la toma de decisiones. Ver Ilustración 4.

Ilustración 4: Video Inteligente

Videos sensores: Son algoritmos que le permiten al personal de guardia identificar objetos dentro de imágenes y hasta identificar objetos en movimiento si es necesario.

Ilustración 5: Videos Sensores

1.1.1. ¿Qué es un sistema de vigilancia?

Los sistemas de vigilancia que se utilizan actualmente, se basan fundamentalmente en una infraestructura física que da como resultado instalaciones con un gran número de cámaras, ya sean IP o cualquier otra alternativa. Esta gran cantidad de cámaras es controlada en un centro de control por un personal calificado para la realización de este trabajo.

1.1.2. Ventajas

Los sistemas de video vigilancia son muy utilizados alrededor de todo el mundo por las numerosas ventajas que aportan. Una de estas ventajas lo constituye el análisis inteligente de video que realizan en tiempo real.

Esto es muy importante ya que el flujo de video que se obtiene es más actualizado y si se diera un robo por ejemplo, se detectaría al momento, incrementando la posibilidad de capturar al ladrón. También se automatiza la toma de decisiones de los seres humanos lo cual resulta muy importante ya que el personal de seguridad debe estar atento a varias cámaras a la misma vez por lo que puede cometer un error de observación o quedarse dormido, provocando graves consecuencias. Otorga una mayor seguridad y rendimiento, así como una automatización de los oficiales de seguridad designados para la supervisión de las áreas. Además, se reduce en un gran porciento de los errores cometidos por los vigilantes.

1.2. Objeto de Estudio

Descripción General

El objeto de estudio de la presente investigación es el procesamiento inteligente de flujos de videos, para esto se utilizan técnicas como la de sustracción de fondo, segmentación de frente/fondo, entre otras. Dichas técnicas contribuyen en alto grado a desarrollar el procesamiento inteligente de video ya que ayudan a solventar problemas en las imágenes como el ruido y las sombras que pueden conllevar a confundir el proceso. Con la unión de varias de estas técnicas se puede lograr construir un video sensor capaz de llegar a decisiones automáticas, agilizando el proceso de seguridad en los sistemas.

Este proceso tiene varias etapas.

La primera etapa es donde se adquiere los frames o imágenes digitales desde el flujo de video proveniente de las cámaras IP. Una vez obtenida la imagen digital, se aplica un pre-procesado para de esta manera mejorar la calidad eliminando sombras y ruidos entre otros problemas que dificultan el procesamiento a través de algunas técnicas como las mencionadas anteriormente. Seguidamente se realiza la segmentación, la cual permite dividir la imagen en regiones de objetos y devuelve un conjunto de puntos por cada región que se encuentra en la imagen. Los algoritmos que se aplican durante esta etapa, permiten definir los objetos que se encuentran dentro de la imagen. Al finalizar este etapa se procede a representar las regiones que son de interés, ya sea por región interna o por contorno. Al contar con las regiones de interés, se realiza la descripción para de esta manera extraer los datos que los caracterizan, es decir, se recibe un objeto y se devuelven las características asociadas al mismo. Seguidamente se realiza el reconocimiento a través del cual se obtienen los rasgos característicos de cada objeto y se les asocia una categoría predeterminada. Finalmente se procede a la interpretación a través de la cual se asocia un significado o acción al conjunto que han sido identificados.

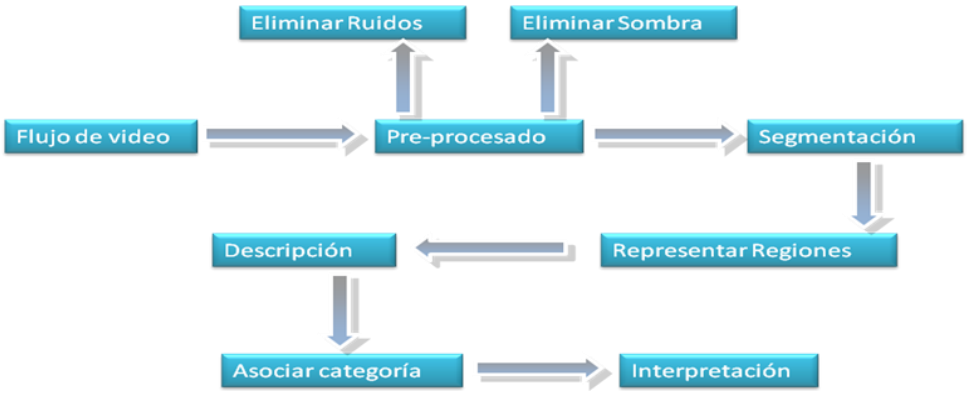


Ilustración 6: Análisis del procesamiento

1.3. Tendencias y Tecnologías actuales

El análisis automático de videos de seguridad cobra más auge cada día que transcurre por lo que un mayor número de personas se dedican a investigar sobre esta área. Estas herramientas le proporcionan al personal de seguridad la posibilidad de automatizar la vigilancia a través del procesamiento inteligente de video. Problemas como la detección de objetos abandonados cobran cada vez más importancia actualmente, debido a los peligros potenciales que representan no contar con un sistema capaz de encontrar estos problemas. Hoy día las empresas disponen de una amplia y más numerosas cantidad de herramientas y medidas organizativas que le permiten una mejora y automatización de la vigilancia y seguridad de sus locales y pasillos.

Anteriormente, los sistemas de video vigilancias eran muy comunes encontrarlos solo en los bancos, aeropuertos e instituciones empresariales, pero ya no es así. Con el incremento de las nuevas tecnologías, se ha incorporado de forma masiva en los hogares. Muchas personas cuentan con la seguridad de estos sistemas instalados en sus hogares. Las tendencias del mercado en cuanto a los sistemas de vigilancias, sobre todo los caseros, están demostrando día a día que su futuro es muy prometedor. Cuanto más confiable sea la tecnología, más solicitada será.

Cuando comenzaron a aplicarse estos sistemas en los hogares eran mucho más costosos y complicados. Sin embargo ahora, muchas personas tienen instalados en sus hogares estos sistemas, confiando en la seguridad que les proporciona.

Existen actualmente muchos algoritmos de procesamiento inteligente de imágenes utilizados para desarrollar estos videos sensores y que utilizan técnicas como la de Sustracción de Fondo y la técnica de modelado de fondo por mezcla gaussiana entre otras más y muy eficientes. Por lo general las imágenes tienen una cierta cantidad de ruido, píxeles con valores incorrectos por errores en el proceso de captura por lo que se eliminan las sombras y ruidos utilizando la técnica de segmentación de fondo/figura y un post-procesado del resultado.

1.4. Análisis de Otras soluciones

Como resultado del análisis y estudio de varias bibliografías, se han identificado varios software que utilizan técnicas para el análisis inteligente de video para la detección de objetos abandonados y otras facilidades.

IVA de Bosch

Está basado en el procesamiento inteligente de video, introduciendo un nivel más alto en la automatización en el control de CCTV. Este sistema puede advertir inmediatamente al operador cuando un objeto ha sido abandonado, entre otras facilidades. Utiliza algoritmos muy avanzados que permiten solucionar el problema de la vigilancia. Además de ser fácil de implementar, controlar y configurar. No cuenta con un servidor central lo cual permite eliminar el peligro que representa de un único punto de fallo. Tampoco necesita de un hardware centralizado, sistemas operativos o servidores de análisis. Su principal desventaja se basa en que es una solución privativa y presenta un alto precio en el mercado.

Ilustración 7: IVA de Bosch

XTRALIS ADPRO PRESIDIUM

Este video sensor de análisis inteligente de video para exteriores provee soluciones de análisis o procesado de imágenes en tiempo real lo cual le permite detectar objetos abandonados, obteniendo sus datos para posteriormente generar los eventos que se necesitan o las alarmas. Su funcionamiento se basa en el análisis de contenido de imágenes de videos, utilizando algoritmos que posibilitan la detección de objetos, su movimiento y su aprendizaje. No necesita de hardware adicional y funciona sobre su propia plataforma basada en Linux lo cual otorga un gran nivel de seguridad y estabilidad en el funcionamiento. Otra de sus

ventajas es que está optimizado en exteriores, lo cual permite minimizar los efectos adversos como son la nieve, viento, etc. Se puede programar remotamente sobre las redes IP, líneas telefónicas RTC, RDSI, etc.… a través de un entorno amigable basado en Windows. Sus principales desventajas son que su adquisición es muy costosa.

Ilustración 8: XTRALIS ADPRO PRESIDIUM

SIEMENS SISTORE CX ODR

Este sistema de sensor de vídeo permite detectar objetos retirados o abandonados. El área de aplicación cubre desde la detección de objetos de arte retirados en museos hasta la supervisión de áreas donde no deben ser colocados objetos tal como salidas de emergencia, secciones de carretera, etc. Al utilizar el SISTORE CX es posible grabar y transmitir vídeo en formato de MPEG4 y detectar si ha habido manipulación. Es de fácil actualización usando la configuración del software y tiempo de conmutación controlada entre los programas de alarma a través del software de IVM.

Su fundamental desventaja yace en que es un software privativo, por lo que solo se puede usar introduciendo una clave al instalarlo que se otorga solo al comprarlo en una gran suma de dinero.

Ilustración 9: SIEMENS SISTORE CX ODR

Como se puede observar, existen muchos sensores en el mundo que realizan esta funcionalidad pero se encuentran privatizados y desarrollados bajo plataformas no libre, lo cual representa un problema para su

utilización en nuestro país. Es por esto que se busca una mejor solución al crear un componente que pueda trabajar bajo plataforma libre como Linux y que no presente licencia privativa.

1.5. Metodologías de Desarrollo

Una metodología de desarrollo no es más que un marco de trabajo que recoge un conjunto de pasos y procedimientos que se deben seguir para estructurar, controlar y planificar el proceso de desarrollo de un software. Actualmente existen muchísimas metodologías que brindan soluciones a los desarrolladores. Su utilización posee una enorme importancia a la hora de gestionar de forma eficiente un proyecto. Pueden estar clasificadas de dos maneras, como metodologías ágiles o como pesadas.

1.5.1. Proceso Unificado de Desarrollo de Software (RUP)

Es un proceso de desarrollo de software y junto con el lenguaje Unificado de Modelado (UML), se ha convertido en la metodología más utilizada para el análisis, diseño e implementación. Su ciclo de vida organiza las tareas en fases e iteraciones. Es una metodología robusta y que genera varios artefactos durante su ciclo de desarrollo. Se aconseja fundamentalmente para proyectos de gran tamaño.RUP se caracteriza por estar: Dirigido por casos de Uso, Centrado en arquitectura, Iterativo e incremental.

El Proceso Unificado está equilibrado por ser el producto final de tres décadas de desarrollo y uso práctico. Su desarrollo como producto sigue un camino desde el Proceso Objectory (primera publicación en 1987) pasando por el Proceso Objectory de Rational (publicado en 1997) hasta el Proceso Unificado de Rational (publicado en 1998). En este camino de desarrollo ha tenido la influencia mayoritaria de dos grandes métodos: el Método de Ericsson y el Método de Rational.

El Proceso Unificado de Rational (RUP), es un proceso de ingeniería de software planteado por Kruchten (1996) cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecido. Cubre el ciclo de vida y desarrollo de software.

El Proceso Unificado de Racional consta de cuatro fases o etapas:

1. **Comienzo o Inicio:** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
2. **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

3. **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios release (liberaciones) del producto que han pasado las pruebas. Se ponen estos release (liberaciones) a consideración de un subconjunto de usuarios.
4. **Transición:** El release(liberaciones) ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Posee los siguientes flujos de trabajo de desarrollo:

1. **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
2. **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
3. **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
4. **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
5. **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.

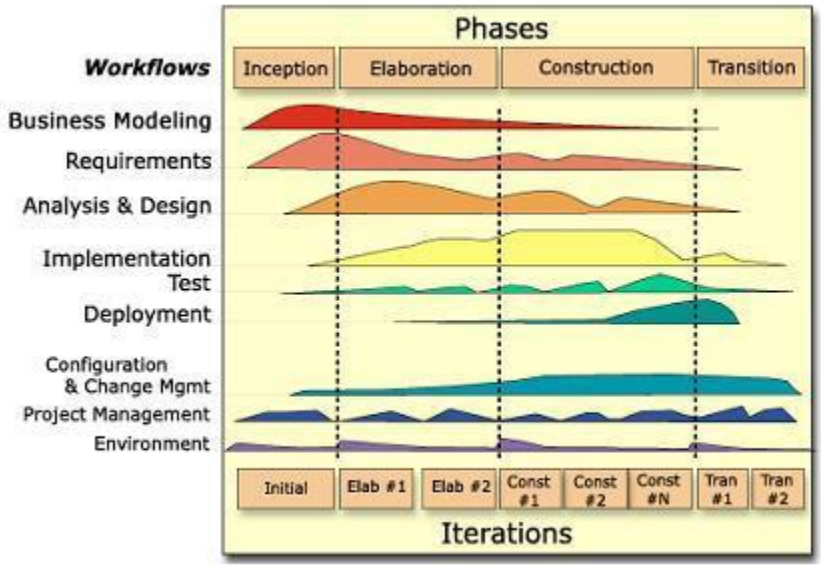


Ilustración 10: Flujo de trabajo RUP

1.5.2. ¿Por qué utilizar RUP?

Esta metodología ayuda a construir un software de alta calidad, desarrollado en el tiempo planificado, pero que además satisface la necesidad de ser elaborado de una forma más acelerada. Además de que es necesario enfocarse en trabajar de forma organizada, donde se controle y documente todo lo relacionado con el proyecto en cuestión y puedan eliminarse los riesgos que podrían presentarse durante el desarrollo del mismo, lo cual no podría lograrse sin el empleo de una metodología eficaz que se adapte a las características propias del software que se esté desarrollando. Por todo esto se ha escogido la metodología RUP para el desarrollo del video sensor.

1.6. Herramientas de Modelado

1.6.1. Enterprise Architect 7.0

Enterprise Architect 7.0 (EA) es una potente herramienta de modelado en UML bajo plataforma Windows. Provee lo último en desarrollo de sistemas, administración de proyectos y análisis de negocio. Posibilita que se abarque todo el ciclo de vida del video sensor, incluso desde el levantamiento de los requisitos del mismo y el análisis, diseño y demás etapas.

1.6.2. ¿Por qué utilizar Enterprise Architect 7.0?

Se utiliza esta herramienta ya que es muy flexible y completa por lo que resulta muy beneficiosa a la hora del análisis y diseño del componente. Además, permite a los miembros del proyecto trabajar bajo plataforma Windows lo cual es una cuestión muy necesaria para el componente. Posee una gran velocidad, buen rendimiento y estabilidad, posibilitando la obtención de buenos resultados. También porque se ha autorizado su uso para el proyecto de Video Vigilancia SURIA. Posee soporte para diagramas estructurales, diagramas de comportamiento y diagramas extendidos, con los cuales se trabajarán durante la fase de análisis y diseño. Permite que se realice Ingeniería Inversa lo cual es muy útil para los desarrolladores ya que agiliza el trabajo. A pesar de ser propietaria se ha autorizado su uso para el Sistema.

1.7. Biblioteca

No son más que un conjunto de subprogramas que contienen códigos y datos que brindan servicios a otros programas y ayuda a los programadores a desarrollar el software. No necesita ser modificado y el código que contienen, se añaden al programa principal cuando se genera.

1.7.1. ¿Qué es OpenCV?

Según (Universidad Javeriana Bogotá, 2009), *"OpenCv viene de las siglas Open Source Computer Vision Library, es una librería abierta desarrollado por Intel. Esta librería proporciona un alto nivel funciones para el procesado de imágenes. Estas librerías permiten a los programadores crear aplicaciones poderosas en el dominio de la visión digital. OpenCV ofrece muchos tipos de datos de alto-nivel como juegos, árboles, gráficos, matrices, etc.*

OpenCv permite, Operaciones básicas, Procesado de imágenes y análisis, Análisis estructural, Análisis de movimiento, Reconocimiento del modelo, Reconstrucción 3d y calibración de la cámara, Interfaz gráfica y adquisición, etc.

OpenCV implementa una gran variedad de herramientas para la interpretación de la imagen. Es compatible con Intel ImageProcessing Library (IPL) que implementa algunas operaciones en imágenes digitales. A pesar de primitivas como binarización, filtrado, estadísticas de la imagen, pirámides, OpenCV es principalmente una librería que implementa algoritmos para las técnicas de la calibración (Calibración de la Cámara), detección de rasgos, para rastrear (Flujo Óptico), análisis de la forma (Geometría, Contorno que Procesa), análisis del movimiento (Plantillas del Movimiento, Estimadores), reconstrucción 3D (Transformación de vistas), segmentación de objetos y reconocimiento (Histograma, etc.).

El rasgo esencial de la librería junto con funcionalidad y la calidad es su desempeño. Los algoritmos están basados en estructuras de datos muy flexibles, acoplados con estructuras IPL; más de la mitad de las funciones ha sido optimizada aprovechándose de la Arquitectura de Intel."

1.7.2. Estructura de OpenCV

Estas bibliotecas se dividen en cinco grandes grupos:

1. CXCORE: donde se encuentran las estructuras y algoritmos básicos que usan las demás funciones tales como: suma, media, operaciones-binarias.
2. CV: donde están implementadas las funciones principales de procesamiento de imágenes.
3. HighGUI: todo lo relacionado a la interfaz gráfica de OpenCV y las funciones que permiten importar imágenes y video (actualmente ffmpeg, cvcam, entre otros)
4. ML: que cuenta con algoritmos de aprendizaje, clasificadores y demás.

5. CvAux: con funciones experimentales.

1.7.3. ¿Por qué utilizar OpenCV?

Ha sido seleccionada gracias a que en cuanto a análisis de movimiento y seguimiento de objetos, ofrece funcionalidades que permiten el procesamiento inteligente de imágenes. Incorpora funciones básicas para modelar el fondo para su posterior sustracción, generar imágenes de movimiento MHI (MotionHistoryImages o Historia de Imágenes en Movimiento) para determinar donde hubo movimiento y en qué dirección, algoritmos de flujo óptico y otros. Implementa algoritmos para las técnicas de la calibración (Calibración de la Cámara), detección de rasgos, análisis de la forma (Geometría, Contorno que Procesa), segmentación de objetos y reconocimiento (Histograma, entre otros.).

1.8. Lenguaje de Programación

1.8.1. C++

Fue creado a mediado de 1980 por Bjarne Stroustrup bajo el nombre en un principio de C with classes. No fue hasta 1983 que se le llamó C++. El objetivo de su creación era el de extender el conocido lenguaje C permitiéndole a los programadores trabajar con objetos. Poco a poco se le fueron añadiendo facilidades como la programación estructurada y la orientada a objetos. Es un lenguaje imperativo, versátil, potente.

Ha eliminado también algunas de sus desventajas aunque mantiene muchas otras como por ejemplo: sigue muy ligado al hardware subyacente, manteniendo una baja potencia para la programación a bajo nivel. Debido a esto, se le han añadido elementos que le posibilitan la programación a alto nivel también. Permite programar desde sistemas operativos hasta compiladores, sistemas de Bases de Datos, procesadores de textos, entre otros.

Una de las cualidades de este lenguaje de programación es el de que permite redefinir los operadores y crear nuevos tipos que se comporten como tipos fundamentales. Aunque introduce nuevos operadores y palabras claves para manejar clases, algunas de sus extensiones tienen aplicación fuera del contexto de programación orientada a objetos.

1.8.2. ¿Por qué utilizar C++?

Se escogió este lenguaje pues al utilizar C++, se puede realizar una aplicación distribuible sin necesidad de pagar licencia. Además, es un lenguaje multi-nivel, es decir, con él se puede programar de forma directa

tanto hardware (según el sistema operativo que se utilice) como para crear operaciones de tipo Windows los cuales presentan una misma interfaz. Es versátil y posee las ventajas del lenguaje C en cuanto a flexibilidad, eficiencia y concisión, lo cual sería aprovechado en el rápido desarrollo de la aplicación. Presenta la facilidad de que es portable, es decir, un código creado en este lenguaje se puede compilar en cualquier sistema operativo sin necesidad de cambiarle prácticamente el código fuente. Un componente creado con este lenguaje obtiene mejores resultados en cuanto a la velocidad de trabajo. Además, presenta librerías y funciones óptimas para el tratamiento de imágenes, facilitando el procesamiento inteligente de video.

1.9. Herramientas de desarrollo seleccionadas

1.9.1. Visual Studio .Net 2010

Es un Entorno de Desarrollo Integrado (IDE) hecho para las diferentes versiones de sistemas operativos de Windows. Está capacitado para soportar varios lenguajes como Visual C#, Visual C++, Visual Basic .Net, Visual J# y ASP.Net. Posee nuevas características que mejoran la herramienta como la capacidad para utilizar múltiples monitores, además de la posibilidad de desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo. También, posee una edición que compila las características de todas las ediciones comunes de Visual Studio: Team Studio, Professional, Test (Visual Studio Ultimate).

Según (Microsoft Corporation, 2010), *"…permite a un modelo de programación más moderno mediante la adición de núcleo C++ características del lenguaje de la próxima C++0x estándar y el reacondicionamiento de la biblioteca estándar para aprovechar las nuevas características del lenguaje. Hay nuevas bibliotecas de programación paralelas y herramientas para simplificar la creación de programas paralelos."*

1.9.2. ¿Por qué utilizar Visual Studio .Net 2010?

Se ha escogido este IDE ya que soporta el lenguaje C++ y la biblioteca OpenCV. También porque es la versión más reciente de esta herramienta por lo que posee ventajas por encima de la versión del 2005, la cual era la más utilizada y factible para este tipo de aplicaciones. Algunas de estas ventajas lo constituyen el soporte para aplicaciones de Windows 7, lo cual es uno de sus mejores logros y un mejor rendimiento de manera general.

Conclusiones

Como se pudo apreciar, en el mundo existen muchos videos sensores que permiten la detección de objetos abandonados pero todos son soluciones privativas por lo que su adquisición es muy costosa. Este componente al estar construido con el lenguaje C++, se considera una aplicación distribuible que no necesita pagar licencia alguna lo cual es una de sus mayores ventajas. El estudio del procesamiento inteligente de video e imágenes digitales ha permitido encontrar técnicas que posibilitan cumplir con el objetivo general planteado. Además, con la ayuda de las herramientas y tecnologías descritas anteriormente, se puede obtener un producto de buena calidad y fiabilidad.

CAPÍTULO 2. ANÁLISIS DE LA SOLUCIÓN

2.1. Introducción

En este capítulo se abordarán los temas relacionados con los temas del análisis. Inicialmente se aborda los requisitos funcionales y no funcionales que debe cumplir el componente, así como el modelo de dominio a partir de los indicadores que se seleccionaron y los casos de usos con las descripciones correspondientes de cada uno.

2.2. Propuesta de sistema

La solución que se plantea es la de desarrollar un componente que permita integrarse en un futuro a un sistema, contribuyendo a la vigilancia y sirviendo de apoyo a los guardias de seguridad. El componente tendría como entrada flujos de videos obtenidos de cámaras IP, las cuales se ubicarán en todas las zonas de interés y de donde tengan acceso a la red. El algoritmo procesaría esta entrada de video y detectaría en caso de que haya, un objeto que haya sido abandonado en el área monitoreada.

2.3. Modelo de Dominio

2.3.1. Conceptos asociados al Modelo de Dominio.

Video Sensor: algoritmo de identificación de objetos.

Objeto: algo que se puede observar y carece de autonomía.

Características: cualidades de un objeto.

Video: secuencia de imágenes a gran velocidad.

Cámara: dispositivo de captura de videos.

Seguimiento: capacidad de observar algo durante un tiempo determinado.

2.3.2. Descripción.

La cámara brinda un video, el cual tiene puede tener varios objetos o ninguno. Estos presentan características que le permiten al Video Sensor procesar el video y detectar los objetos que han sido abandonados.

2.3.3. Modelo de Dominio

Los procesos que se enmarcan en el flujo de trabajo actual, no complementan la realización de un modelo completo del negocio, por lo que se decidió crear un modelo de dominio como base de conocimiento donde se capturan los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema.

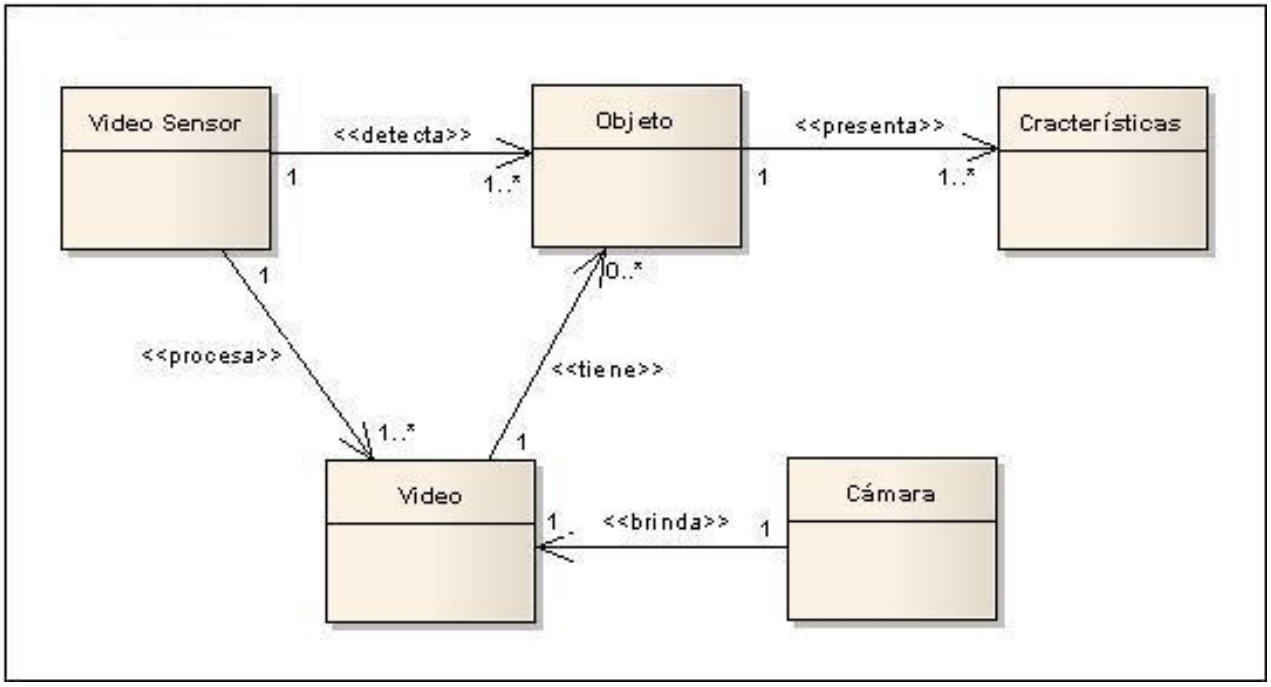


Ilustración 11: Modelo de Dominio

2.4. Requisitos Funcionales

Los requisitos funcionales son aquellos que definen la función que el sistema va a llevar a cabo. De manera general para el video sensor en desarrollo se identificaron los siguientes requisitos:

RF# 1: Capturar el flujo de video proveniente de la cámara.

RF# 2: Extraer fotogramas del flujo de video.

RF# 3: Modelar frente del video.

RF# 4: Modelar fondo del video.

RF# 5: Eliminar el ruido.

RF# 6: Procesar sombras.

RF# 6.1. Detectar sombras.

RF# 6.2. Eliminar sombras.

RF# 7: Determinar los objetos que se mantienen estáticos.

RF# 8: Identificar los objetos estáticos que son personas.

RF# 9: Determinar si están abandonado los objetos encontrados.

RF# 10: Emitir alarma.

2.5. Requisitos No Funcionales

2.5.1. Requisitos de usabilidad

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

2.5.2. Requisitos de fiabilidad

- Disponibilidad:
  - El componente debe estar funcionando las 24 horas del día durante toda la semana.

2.5.3. Restricciones de Diseño

2.5.3.1. Lenguaje

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

2.5.3.2. Librería

- La librería que se utilizará será OpenCV.

2.5.4. Rendimiento

- Se debe procesar el video en tiempo real.
- Se debe ajustar el tamaño del fotograma con la cantidad de estos por segundos.
- Optimizar el uso de la memoria RAM.

2.5.5. Portabilidad

- El componente debe poderse utilizar por diferentes plataformas como Windows y Linux.

- La plataforma debe ser de arquitectura de 32 bit.

2.5.6. Hardware

- Se debe tener como mínimo 1 GB de RAM y procesador PENTIUM 4 a 3GHZ.

2.6. Definición de los Casos de Usos y actores

2.6.1. Definición de los actores

Un actor del negocio es cualquier individuo, grupo, organización, máquina o sistema de información externo que interactúa con el negocio. El término actor significa el rol que algo o alguien juega cuando interactúa con el negocio para beneficiarse de sus resultados. De acuerdo con esta idea un actor del negocio representa un tipo particular de usuario del negocio más que un usuario físico, ya que varios usuarios físicos pueden realizar el mismo papel en relación al negocio, pueden ser instancias de un mismo actor. A continuación, se muestra en la tabla el actor identificado y su correspondiente descripción.

Actor	Descripción
Sistema Video Vigilante 	Es el sistema con quien interactúa el componente. Es el que inicia el caso de uso Detectar objetos abandonados.

Tabla 1: Definición de los actores

2.6.2. Definición de Casos de usos

De acuerdo a los requisitos funcionales definidos anteriormente, se extrajeron los siguientes casos de usos:

CU# 1	Extraer frente
Actor	Sistema Video Vigilante
Propósito	Dividir el video en frente y fondo para obtener todos los objetos de la escena que se encuentran.

Resumen	Se sustrae el fondo para de esta manera obtener los objetos que se encuentren.
Precondiciones	Se han obtenido los fotogramas de la escena.
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 CU8. En caso contrario, se continúa el procesamiento.
Poscondición	Se obtienen todos los objetos del fotograma.

Tabla 2: Descripción del CU Extraer frente

CU# 2	Discriminación Persona/Objeto.
Actor	Sistema Video Vigilante
Propósito	Clasificar los blob en personas u objetos.
Resumen	Se clasifican los blobs en personas u objetos.

Precondiciones	Se obtuvieron los objetos estáticos.
Referencia	RF# 8
Prioridad	Crítico
Flujo normal de Eventos	
Acción del Actor	**Respuesta del sistema**
	1. Se extraen las características. 2. Se calcula el radio y el porcentaje de píxeles. 3. Si dicho porcentaje es menor que un cierto umbral, el blob es clasificado como persona
	3. Si dicho porcentaje es mayor que un cierto umbral, el blob es clasificado como objeto.
Poscondición	Se obtienen los blobs que son objetos.

Tabla 3: Descripción de CU Discriminación Persona/Objeto

CU# 3	Emitir alarma
Actor	Sistema Video Vigilante
Propósito	Dar una alarma de que se ha detectado un objeto abandonado.
Resumen	Se emite una alarma y selecciona el objeto que ha sido abandonado mediante un cuadro de selección.
Precondiciones	Se han obtenido los objetos abandonados.
Referencia	RF# 10
Prioridad	Crítico
Flujo normal de Eventos	

Acción del Actor	Respuesta del sistema
	1. Se dibuja un cuadro de selección alrededor del objeto detectado como abandonado.
Poscondición	Se emite la alarma.

Tabla 4: Definición de CU Emitir alarma

CU# 4	Detectar Objetos Abandonados
Actor	Sistema Video Vigilante
Propósito	Localizar en la imagen los objetos que se encuentren abandonados
Resumen	Se captura un video y a través de un procesamiento inteligente de video determina si el objeto está abandonado y de ser así, emite una alarma.
Precondiciones	Se han obtenido los objetos estáticos.
Referencia	RF# 9, RF# 1, RF# 2.
Prioridad	Crítico
Flujo normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1. El actor solicita la detección de objetos abandonados en un flujo de video proveniente de una cámara IP.	2. Se captura el flujo de video proveniente de la cámara. 3. Los fotogramas son extraídos del flujo de video. 4. Se modela el fondo y el frente del video. (Ver descripción del CU-3) 5. Es eliminado el ruido. (Ver descripción del CU-3) 6. Se chequea si se procesan las sombras. (Ver descripción del CU-8) 7. Son determinadas los objetos que se mantienen

	estáticos. (Ver descripción del CU-4) 8. Se identifican los objetos estáticos que son personas. (Ver descripción del CU-5) 9. El sistema determina si el objeto es abandonado. (Ver descripción del CU-7) 10. Se emite una alarma. (Ver descripción del CU-6)
Flujo Alternativo Paso 6	
	7. Si se va a procesar sombras, se ejecuta CU-8. En caso contrario, se continúa el procesamiento.
Poscondición	Se obtienen los objetos abandonados de la escena.

Tabla 5: Definición de CU Detectar Objeto Abandonado

CU# 5	Procesar Sombras
Actor	Sistema Video Vigilante
Propósito	Eliminar si existen las sombras en los objetos.
Resumen	El Caso de Uso inicia cuando el sistema necesita 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 de frente del fotograma.
Referencia	RF# 6
Prioridad	Crítico
Flujo normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1. El Sistema Video Vigilante hace la petición de	2. De los objetos detectados en la escena, se

procesar sombras.	analiza cuáles pixeles pertenecen a las sombras. 3. Se eliminan de los objetos, los pixeles que son sombras.
Poscondición	Los objetos que se encuentran en el frente no cuentan con sombra.

Tabla 6: Definición de CU Procesar sombras

2.6.3. Diagrama de Casos de Usos

Los Casos de Uso son parte del análisis, de forma que ayudan a describir qué es lo que es sistema debe hacer. Indican 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. En la imagen que aparece a continuación se muestra el diagrama de Casos de Usos del sistema correspondiente al componente en cuestión.

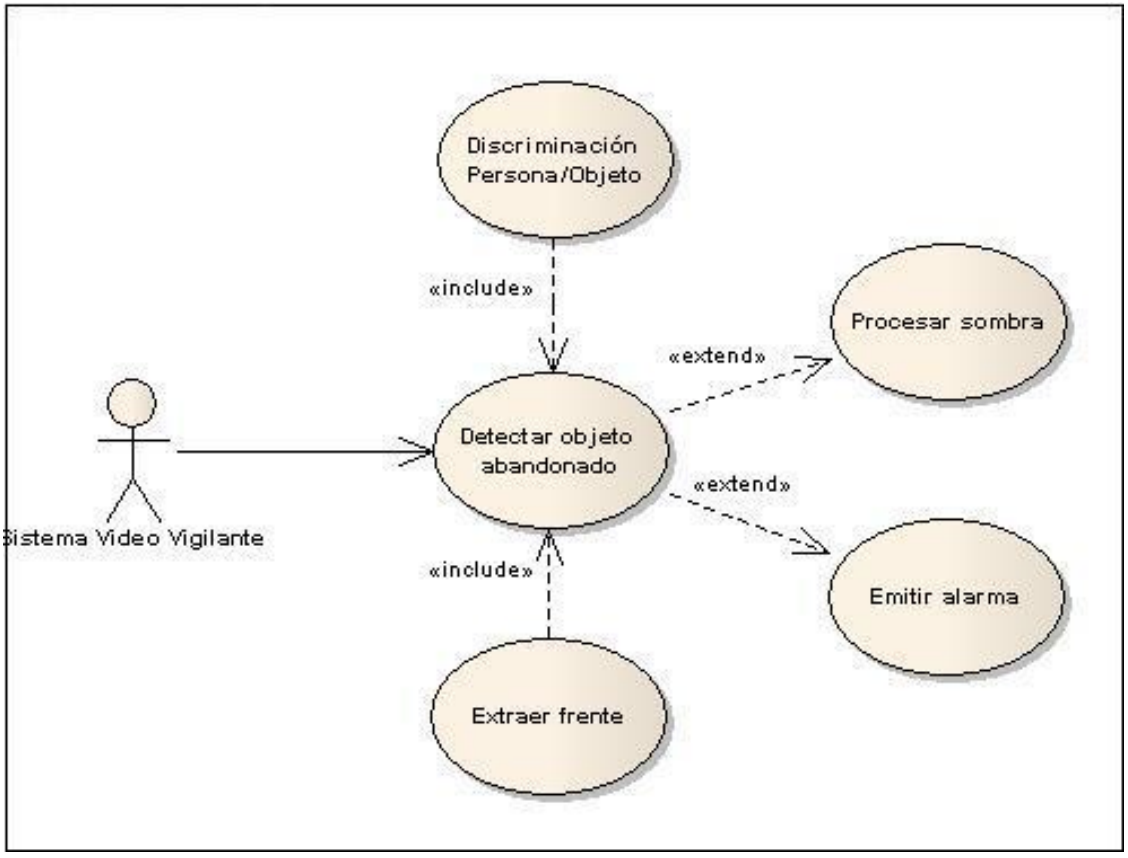


Ilustración 12: Diagrama de Casos de Usos

Conclusiones

Las condiciones o capacidades mencionadas durante el capítulo, permitirán que el componente en desarrollo obtenga un resultado satisfactorio. A través, de la descripción detallada de los Casos de Usos, se puede concluir que se ha logrado una mejor comprensión del funcionamiento del sistema internamente y sus flujos normales de eventos así como los alternos en caso de que ocurran. 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

En este capítulo se abordará el tema de los patrones que se decidieron utilizar para el desarrollo de la solución propuesta. Se definen también las clases de análisis y diseño del componente. Se realizan los diagramas de secuencias en el análisis y diseño que responden a los requerimientos funcionales planteados anteriormente. Finalmente se describe las clases Interfaz, Controladora y Entidades presentes.

3.1. Descripción de la Arquitectura

La arquitectura de software es la organización principal del sistema que incluye los componentes, las relaciones que existen entre ellos. Su diseño y evolución, está relacionada con el diseño y la implementación de estructuras de software de alto nivel.

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión los requisitos reales, haciendo más fácil el trabajo con los sistemas y mucho más simple su mantenimiento. Un patrón de diseño es un conjunto de directrices y principios estructurados que describen un problema común y entregan una buena solución ya probada a la que le dan un nombre. Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces, introduciendo el trabajo con patrones de diseño para asignar responsabilidades.

"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." (Four, 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." (Four, 2009)

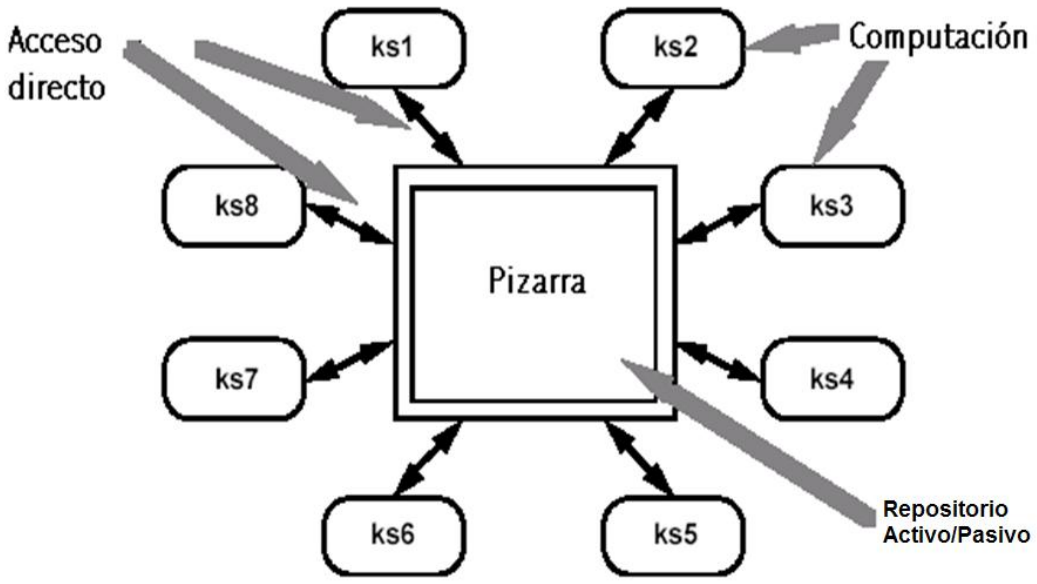


Ilustración 13: Arquitectura Pizarra

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 mencionados anteriormente que compone el Sistema de Video Vigilancia SURIA es el módulo Suria Analytic. Este se encarga de 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.

Se utiliza el patrón Fachada con la intención de *"Proporcionar una interfaz unificada para un conjunto de interfaces en un subsistema, haciéndolo más fácil de usar"* (Universidad de Sevilla, 2011).

El sistema utiliza el patrón fachada porque el módulo Suria Analytic sirve de fachada entre el gestor y los componentes, el Suria recibe todas las peticiones del gestor y le devuelve respuestas, en otras palabras, el Gestor nunca se va a comunicar directamente con los componentes. 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 varios protocolos de transporte, formatos de serialización entre otros. (Ver figura 3.1)

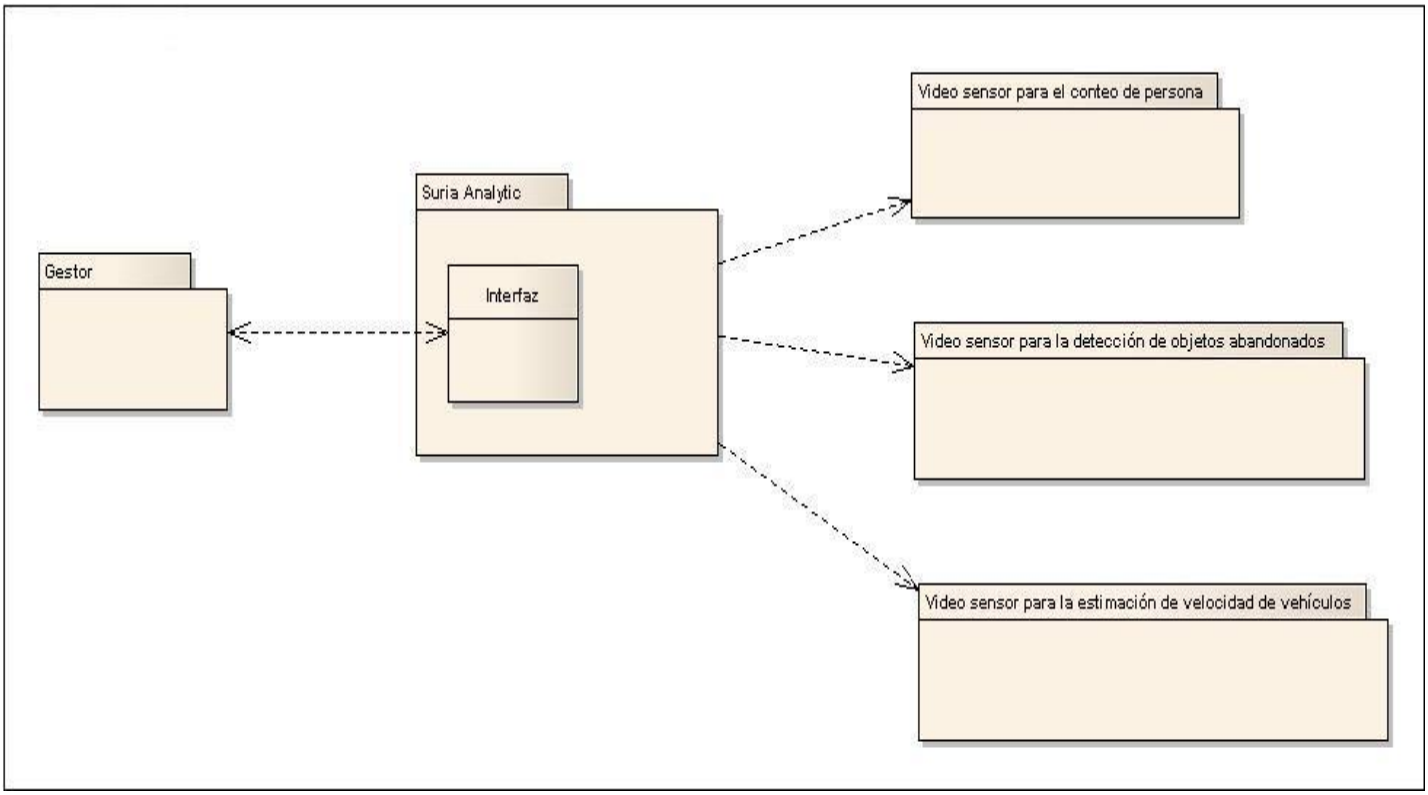


Ilustración 14: Arquitectura del Sistema de Video Vigilancia SURIA

Durante el desarrollo del componente se utilizaron una serie de patrones de diseño como son los Patrones GRASP (General Responsibility Assignment Software Patterns):

El patrón **Experto** que según (González, 2010), "*Consiste en asignar la responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir dicha responsabilidad."* Esto se ve en la clase Análisis ya que es la encargada de asignar las responsabilidades dentro del sistema, es decir, posee los atributos necesarios para la selección de instancias de acuerdo con el algoritmo.

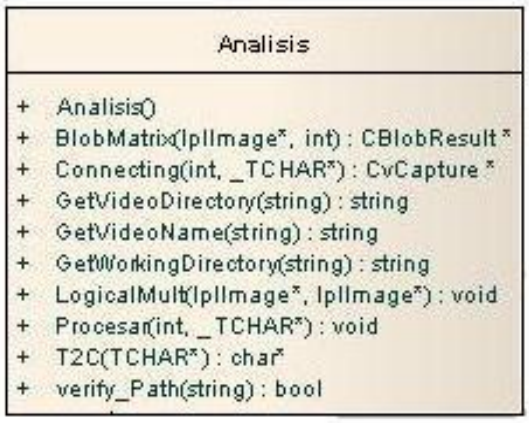


Ilustración 15: Patrón Experto

También se utiliza el patrón **Alta Cohesión** que no es más que aquel que *"…caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme."* según (González, 2010) y el patrón **Bajo Acoplamiento** que *"es la medida de qué tan fuertemente está conectada una clase con las demás (es decir, cuántas clases conoce y necesita). Una clase con bajo acoplamiento no depende de muchas otras clases."* (González, 2010)

En la figura de a continuación, se muestra una imagen donde se aprecia la aplicación del patrón Alta Cohesión ya que no se le asignan todas las responsabilidades a la clase Análisis, sino que los métodos y atributos necesarios para darle la funcionalidad a la clase Análisis, lo tienen las clases Sombra, Frente, Blobs, Persona y MixData3.

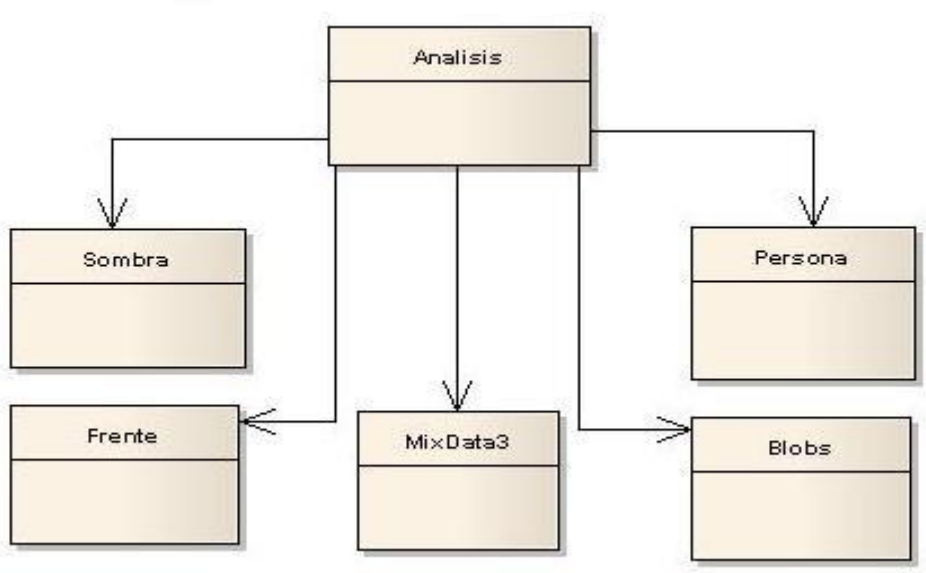


Ilustración 16: Alta Cohesión

El patrón Bajo Acoplamiento se ve en la siguiente figura ya que la clase Análisis es la responsable de ejecutar el algoritmo, utiliza funcionalidades de la clase Frente la cual a su vez se acopla al conocimiento de la estructura MixData3, por lo que no es necesario que la clase Análisis se relacione con la estructura MixData3 para poder acceder a sus funcionalidades.

Ilustración 17: Patrón Bajo Acoplamiento

El patrón **Creador** es el que *"…guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea muy común en sistemas orientados a objetos. Permite que el bajo acoplamiento sea soportado, lo que implica bajo mantenimiento y altas oportunidades de reutilización."* (Zayas, 2010).

En la imagen que se muestra a continuación, se aprecia el patrón creador ya que la clase Análisis contiene objetos de la clase Frente, por lo que es capaz de asumir la responsabilidad de crear instancias de la clase Frente.

Ilustración 18: Patrón Creador

El patrón **Controlador** no es más que aquel que "*Delega la responsabilidad de controlar el flujo de eventos del sistema en varias clases con las que tiene una alta cohesión. Por lo que no existe una sola clase controladora que realice todas las actividades garantizando de esta forma mejorar las validaciones y seguridad en el sistema."* (González, 2010).

En la figura que se aprecia seguidamente se puede ver el patrón Controlador ya que la clase Análisis es la encargada de ejecutar el algoritmo utilizando funcionalidades de las demas clases: Blobs, Frente, Sombra, Persona y la estructura MixData3.

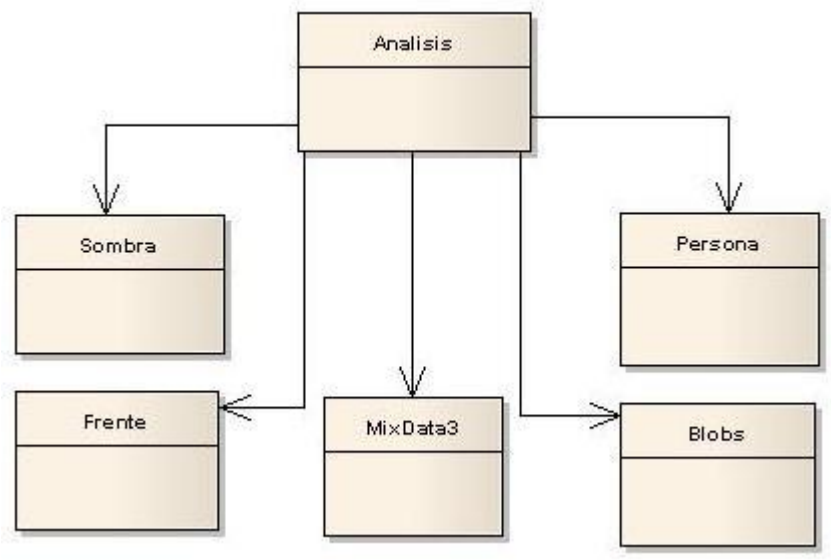


Ilustración 19: Patrón Controlador

3.2. Modelo de análisis

El modelo de análisis presenta las siguientes características:

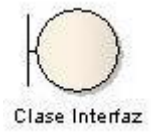
- Modelo conceptual (abstracción del sistema y permite aspectos de la implementación).
- Genérico respecto al diseño (aplicable a varios diseños).

- Tres estereotipos conceptuales sobre las clases: Control, Entidad e Interfaz.
- Dinámico (no muy centrado en la secuencia).
- Bosquejo del diseño del sistema, incluyendo su arquitectura.
- Puede no estar mantenido durante todo el ciclo de vida del software.
- Define una estructura que es una entrada esencial para modelar el sistema, incluyendo la creación del modelo de diseño.

3.2.1. Clases del análisis

Las clases del análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos.

Una clase del análisis participa en relaciones, aunque esas relaciones son más conceptuales que sus contrapartidas de diseño e implementación. Las clases del análisis siempre encajan en uno de tres estereotipos básicos: de interfaz, de control o de entidad y que se mostrarán con más detalles en la siguiente tabla:

Clase	Descripción
	Se utilizan para modelar la interacción entre el sistema y sus actores (usuarios y sistemas externos). Esta interacción a menudo implica recibir (y presentar) información y peticiones de (y hacia los usuarios) y los sistemas externos.
	Coordinan la realización de uno o unos pocos CU, coordinando las actividades de los objetos que implementan la funcionalidad del CU: - Delegan trabajo a otros objetos. - Definen el flujo de control y transacciones dentro de un CU. En principio, se define una clase de control por CU

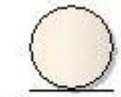
	Modelan información que posee una vida larga y que es a menudo persistente. Modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso del mundo real.

Tabla 7: Clases del análisis

3.2.1.1. Diagramas de clases del análisis

En la siguiente ilustración se puede ver el diagrama de clases del análisis para el caso de uso Detectar Objeto Abandonado. La clase control Análisis es la que maneja todos los pasos que se van a ejecutar durante el procesamiento y contiene las otras clases control: CC_Persona, CC_Alarma, CC_Frente y CC_ Sombra. Se pueden apreciar dos clases entidades: CE_Imagen y CE_Objeto, entre las cuales existe una relación de composición ya que una imagen tiene objetos.

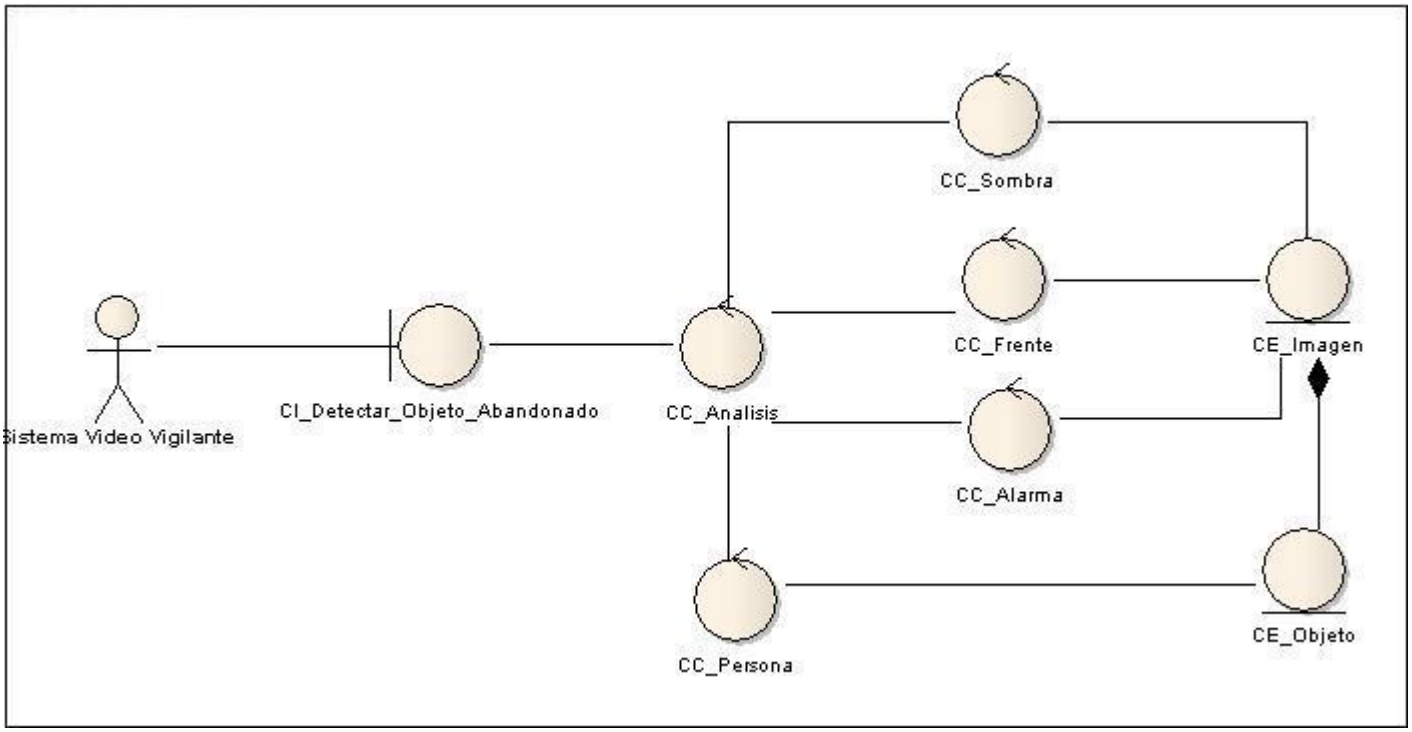


Ilustración 20: Diagramas de clases del análisis

3.2.2. Diagramas de Colaboración

Un diagrama de colaboración es una forma alternativa al diagrama de secuencia de mostrar un escenario. En los diagramas de colaboración se muestran las relaciones entre objetos, se crean enlaces entre ellos y

se añaden mensajes a esos enlaces. El nombre del enlace debe denotar el propósito del objeto que invoca en la interacción con el objeto invocado.

3.2.2.1. Diagrama de colaboración para el CU Emitir Alarma

En la ilustración se observa el diagrama de colaboración para el caso de uso Emitir Alarma. Se encuentran los mensajes fundamentales que son intercambiados entre las clases para el caso de uso que se está analizando. Se obtiene como resultado una alarma.

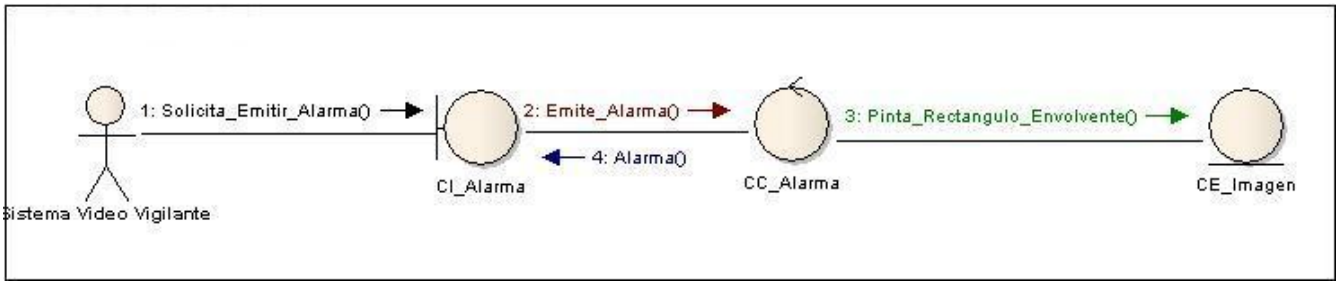


Ilustración 21: Diagrama de colaboración para el CU Emitir Alarma

3.2.2.2. Diagrama de colaboración para el CU Extraer Frente

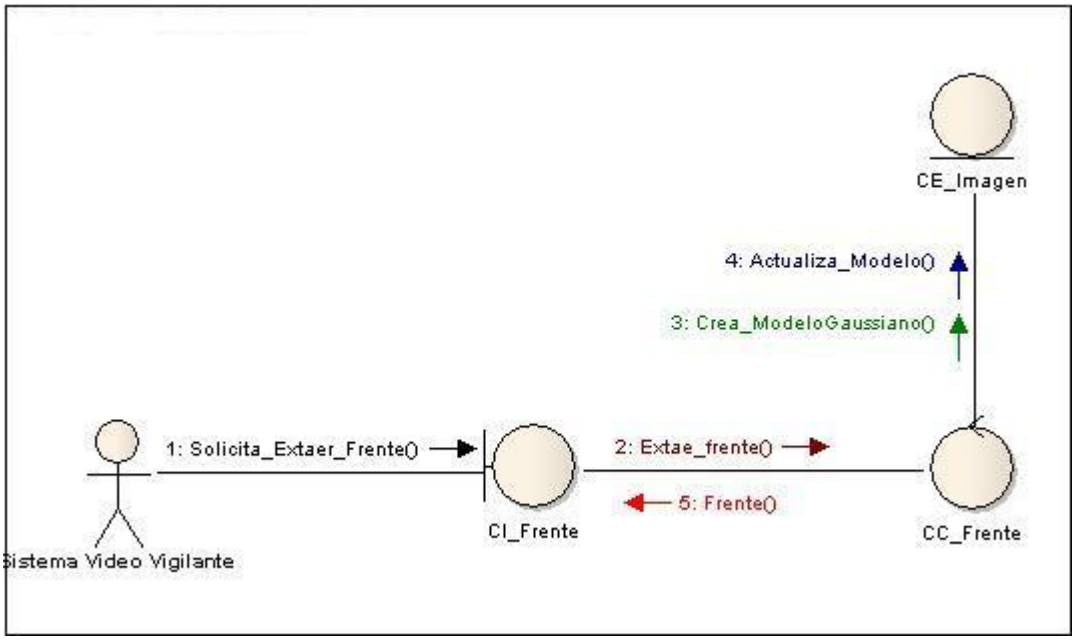


Ilustración 22: Diagrama de colaboración para el CU Extraer Frente

En esta figura se aprecia el diagrama de colaboración para el caso de uso Extraer Frente. Se ven los mensajes fundamentales que son intercambiados entre las clases. Como resultado se obtiene un frente.

3.2.3. Diagrama de colaboración para el CU Procesar sombras

En la siguiente imagen se puede ver el diagrama de colaboración para el caso de uso Procesar Sombras. Se aprecian los mensajes fundamentales que son enviados entre las clases para el caso de uso en cuestión. Finalmente se obtiene la imagen sin sombras de la escena.

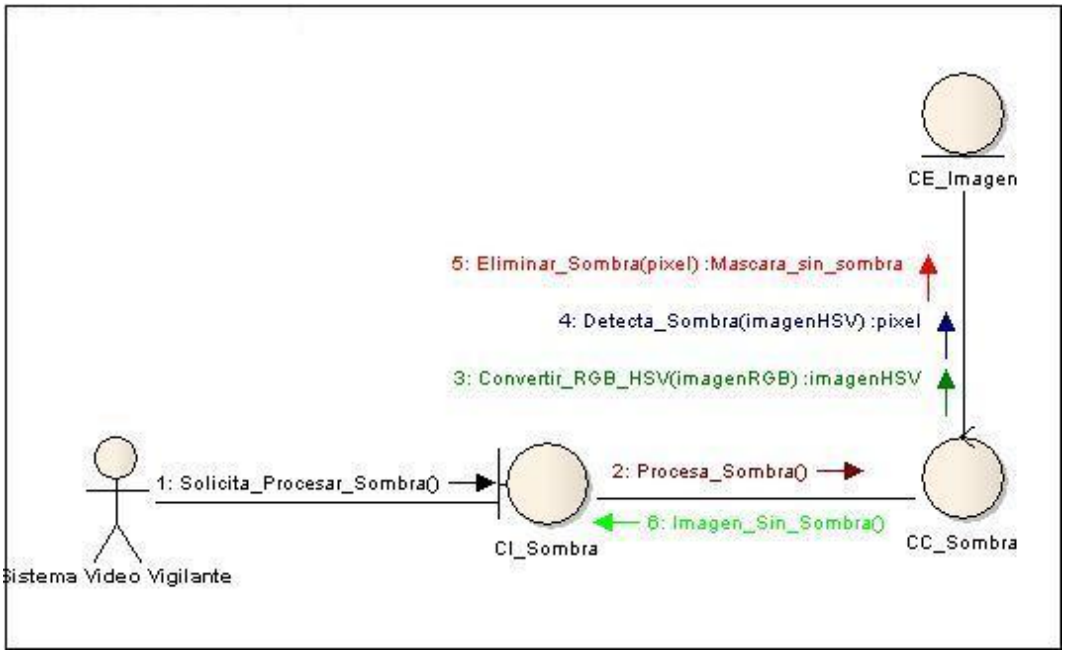


Ilustración 23: Diagrama de colaboración para el CU Procesar sombras

3.3. Modelo de Diseño

Un modelo de diseño no es más que un conjunto de diagramas que describen el diseño lógico. Comprende los diagramas de clases, diagramas de interacción, diagramas de paquetes, etcétera.

Presenta las siguientes características:

- Modelo físico (plano de la implementación)
- No genérico, específico para una implementación.
- Cualquier número de estereotipos (físicos) sobre las clases, dependiendo del lenguaje de implementación
- Dinámico (centrado en las secuencias).

- Manifiesto del diseño del sistema, incluyendo su arquitectura (una de sus vistas).
- Debe ser mantenido durante todo el ciclo de vida del software.
- Da forma al sistema mientras que intenta preservarla estructura definida por el modelo de análisis lo más posible.

3.3.1. Clases del diseño

Una clase de diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema y tienen operaciones, parámetros, atributos, tipos, etc.; necesarios para su implementación en el lenguaje de programación elegido.

3.3.2. Diagrama de clases del diseño

Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema. Principalmente, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas.

Los diagramas de clases también son la base para un par de diagramas relacionados: los diagramas de componentes y los diagramas de despliegue. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

En la siguiente ilustración, se puede observar el diagrama de clases del diseño. Se encuentran en el diagramas siete clases de las cuales una de ellas como se puede ver que es una estructura auxiliar. La clase Análisis es la que se encargará de interactuar con las demás clases, dirigiendo el flujo de acciones dentro del algoritmo. Esta posee todos los métodos necesarios para responder a los requisitos e interactuar con las otras clases presentes. Presenta funciones que posibilitarán detectar cada objeto que se encuentre estático dentro de la escena, apoyándose en los datos que contiene la clase blob. Es por esto que presenta una relación de composición con la clase blob, puesto que esta clase va a trabajar con una lista de objetos de dicha clase.

La clase Blobs es la que contiene toda la información necesaria de los objetos (blobs) que serán procesados. Posee datos importantes como el radio y compactibilidad. La clase Sombra es la que tiene los métodos que permitirán detectar los píxel que forman parte de las sombras y eliminarlos de la imagen que

se va a procesar, todo esto siempre y cuando se encuentren sombras. La clase Frente tiene los métodos necesarias para separar el frente del fondo de la imagen. La clase Persona cuenta con todo lo necesario para clasificar un objeto y saber si un blob (Objeto) detectado es una persona o no. La clase Alarma cuenta con las funciones necesarias para enviar una alarma en caso de producirse la detección de un objeto abandonado.

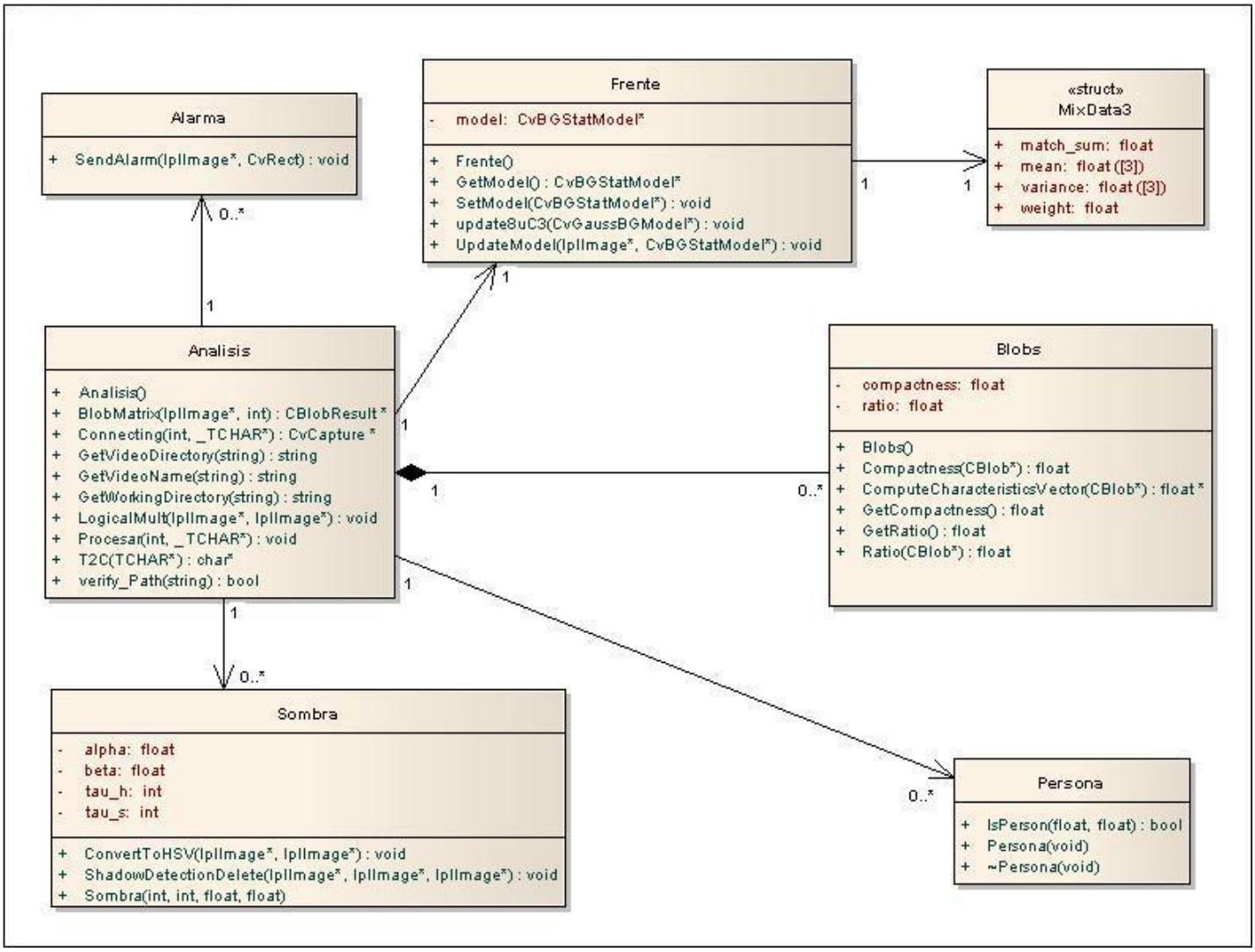


Ilustración 24: Diagrama de clases del diseño

3.3.3. Diagrama de interacción del diseño

Un diagrama de interacción consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Los diagramas de secuencia destacan el orden temporal de los mensajes. Los diagramas de colaboración destacan la organización estructural de los objetos que envían y reciben mensajes.

3.3.3.1. Diagrama de secuencia para el CU Extraer Frente

En la ilustración que se aprecia a continuación, se encuentra el diagrama de secuencia para el caso de uso Extraer Frente. Aquí se muestra la interacción que existe entre las clases Frente y Analisis, así como los mensajes enviados entre ellas.

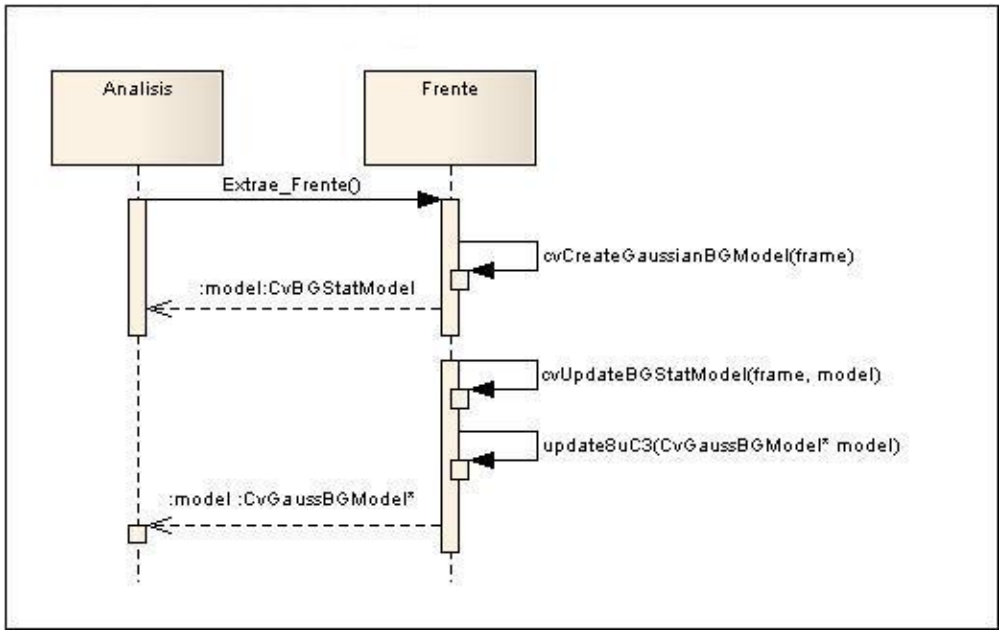


Ilustración 25: Diagrama de secuencia para el CU Extraer Frente

3.3.3.2. Diagrama de secuencia para el CU Enviar Alarma

En la figura que se observa seguidamente, se representa el diagrama de secuencia para el caso de uso Enviar Alarma. Como se puede ver, se muestra la interacción que existe entre las clases Alarma y Analisis y los mensajes que son intercambiados entre ellas.

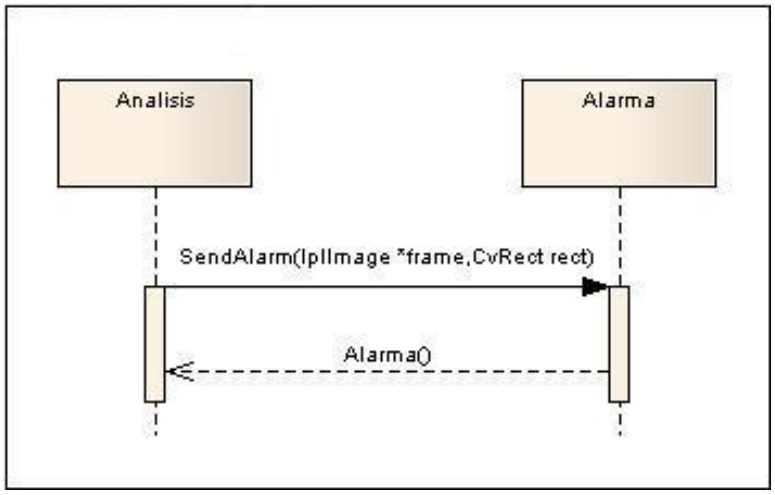


Ilustración 26: Diagrama de secuencia para el CU Enviar Alarma

3.3.3.3. Diagrama de secuencia para el CU Procesar Sombra

En la siguiente imagen, se encuentra el diagrama de secuencia para el caso de uso Procesar sombra, en el cual se muestra la interacción que existe entre las clases Sombra y Analisis, así como los mensajes enviados entre ellos.

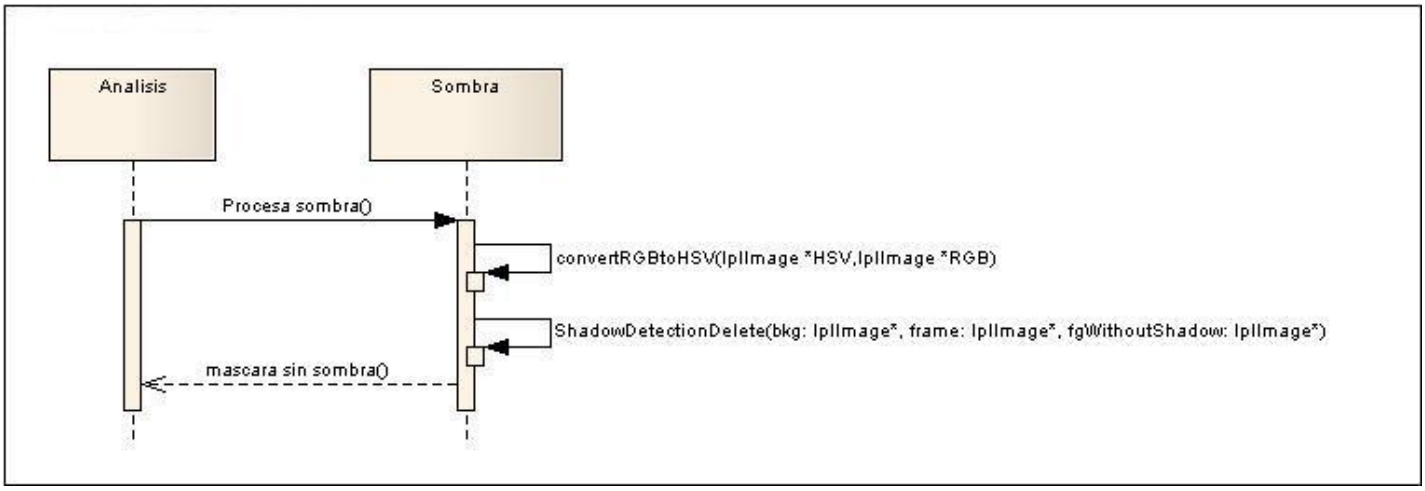


Ilustración 27: Diagrama de secuencia para el CU Procesar Sombra

3.3.3.4. Diagrama de secuencia para el CU Discriminación Persona/Objeto

En la ilustración que se aprecia a continuación, se encuentra el diagrama de secuencia para el caso de uso Discriminación Persona/Objeto. Se muestra la interacción que existe entre las clases Analisis, Persona y Blobs, así como los mensajes enviados entre ellos.

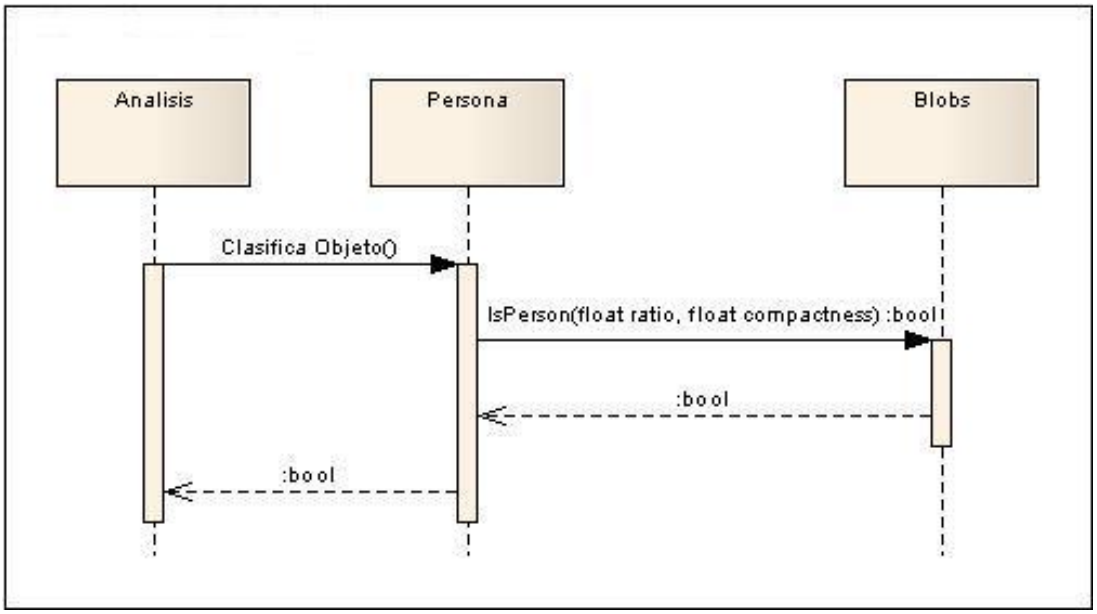


Ilustración 28: Diagrama de secuencia para el CU Discriminación Persona/Objeto

3.3.4. Descripción de las clases

En este epígrafe se expone una descripción más detallada de las clases presentes en el diagrama, se especifica el tipo de clase, sus atributos y métodos y cuáles son sus funcionalidades.

Nombre	Sombra
Tipo de clase	Control
Atributo	**Tipo**
tau_h	int
tau_s	int
alpha	float
beta	float
Para cada responsabilidad	
Nombre	ConvertToHSV(IplImage* RGB,IplImage* HSV)
Descripción	Convierte la imagen que se le pasa por parámetro de RGB a espacios.

Nombre	ShadowDetectionDelete(IplImage *bkg, IplImage *frame, IplImage *maskWithoutShadow)
Descripción	Elimina las sombras de la imagen.

Tabla 8: Descripción clase Sombra

Nombre	Analisis
Tipo de clase	Control
Atributo	**Tipo**
Para cada responsabilidad	
Nombre	Procesar(argc: int, argv[]:_TCHAR*)
Descripción	Hace un submuestreo de las imágenes del video y detecta los objetos que están abandonados.
Nombre	BlobMatrix (Image: IplImage*, blobMinSize: int)
Descripción	Devuelve una lista de los objetos que se encuentran en la imagen y que no son más pequeños que el mínimo tamaño indicado por parámetro.
Nombre	GetWorkingDirectory(string road)
Descripción	Devuelve la dirección del ejecutable de la aplicación.
Nombre	verify_Path(string road)
Descripción	Verifica si existe o no una dirección.
Nombre	GetVideoName(string road)
Descripción	Devuelve el nombre del video.
Nombre	GetVideoDirectory(string road)
Descripción	Devuelve la dirección del video.

Nombre	Connecting(int argc, _TCHAR* argv[])
Descripción	Permite que se realice la conexión a la cámara.
Nombre	T2C(source: TCHAR*)
Descripción	Convierte el tipo de dato TCHAR a CHAR.

Tabla 9: Descripción clase Analisis

Nombre	Blobs
Tipo de clase	Entidad
Atributo	**Tipo**
compactness	float
ratio	float
Para cada responsabilidad	
Nombre	GetCompactness(): float
Descripción	Obtiene la compactibilidad del objeto.
Nombre	GetRatio(): float
Descripción	Obtiene el radio del objeto.
Nombre	Ratio(object: CBlob *): float
Descripción	Calcula el radio del objeto.
Nombre	Compactness(object: CBlob *): float
Descripción	Calcula la compactibilidad del objeto.
Nombre	ComputeCharacteristicsVector(CBlob * currentBlob): float*
Descripción	Devuelve las características del objeto.

Tabla 10: Descripción clase Blobs

Nombre	Persona	
Tipo de clase	Control	
Atributo		**Tipo**
Para cada responsabilidad		
Nombre	IsPerson(float: ratio, float: compactness):bool	
Descripción	Clasifica si el objeto que tiene estas características es una persona o no.	

Tabla 11: Descripción clase Persona

Nombre	Alarma	
Tipo de clase	Control	
Atributo		**Tipo**
Para cada responsabilidad		
Nombre	SendAlarm(frame*: IplImage, CvRect rect*)	
Descripción	Envía una alarma, dibujando un cuadro envolvente sobre el objeto detectado.	

Tabla 12: Descripción clase Alarma

Nombre	Frente	
Tipo de clase	Control	
Atributo		**Tipo**
model		CvBGStatModel*

Para cada responsabilidad	
Nombre	UpdateModel(image: IplImage*, model: CvBGStatModel*):void
Descripción	Actualiza el modelo gaussiano.
Nombre	SetModel(p_model: CvBGStatModel*):void
Descripción	Asigna el valor al modelo gaussiano.
Nombre	GetModel():CvBGStatModel*
Descripción	Obtiene el modelo gaussiano.
Nombre	update8uC3(CvGaussBGModel* model):void
Descripción	Actualiza el fondo de una imagen de tres canales.

Tabla 13: Descripción clase Frente

Conclusiones

La realización del Modelo de Análisis ha sido muy importante para la construcción del componente ya que en este flujo se refinan y estructuran los requisitos obtenidos en el Capítulo 2. Esto ayuda a profundizar en el dominio de la aplicación, lo que permite una mayor comprensión del problema a la hora de modelar la solución. La construcción del Modelo de Diseño, ha permitido que se haga un refinamiento del análisis, teniendo en cuenta los requisitos no funcionales, en otras palabras: ¿Cómo cumple el sistema sus objetivos? El diseño es suficiente para que el sistema pueda ser implementado sin ambigüedades. Por otro lado, 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 componentes.

CAPÍTULO 4. **IMPLEMENTACIÓN**

En el siguiente capítulo se presentan todos los elementos relacionados al flujo de trabajo de implementación, desarrollando lo necesario para obtener el componente. Se representa el diagrama de despliegue y se muestra el diagrama de componentes. También las descripciones de cada uno de los componentes.

4.1. Análisis del algoritmo

Con la ayuda del procesamiento digital de imágenes, se pueden identificar y clasificar objetos (blobs) y zonas que se encuentren dentro de las imágenes que se procesan. En este epígrafe se exponen una serie de técnicas que se utilizan para el procesamiento de imágenes para de esta manera lograr los objetivos propuestos. Para detectar los objetos y procesar las imágenes se han definido una serie de pasos con el fin de obtener las regiones que son de interés, es decir, los blobs. Los objetos que se encuentren estáticos en la escena serán detectados y señalados, utilizando sus características. Vale destacar que el algoritmo se ha apoyado en la utilización de dos importantes librerías: *cvblobslib* y *OpenCV*. Estas librerías ofrecen una gran gama de funcionalidades que facilitan el procesamiento inteligente de imágenes y que se utilizan en el algoritmo.

4.1.1. Capturar video

El primero de todos los pasos es el de capturar el video que se va a procesar, es decir, se va a referenciar el flujo de video que va a procesarse desde una dirección obtenida desde una cámara IP. Para esto se realizan una serie de pasos que dan lugar a la obtención del video. Inicialmente, se obtiene la dirección del video de donde se va a extraer el flujo de video. Luego, con esta dirección de video se hace una petición del flujo a la cámara IP. Si la cámara brinda un flujo, se crea una estructura que referencia al flujo de video. En caso de que no se pueda obtener el flujo, entonces se emitiría un mensaje de error informándole al usuario del problema ocurrido.

4.1.2. Extraer fotogramas

Un video está formado por un número consecutivo de imágenes. En este paso se obtendrían los fotogramas, es decir, las imágenes contenidas en el video para poder procesarlas y extraer los datos necesarios para el procesamiento y realizar así la detección de los objetos abandonados que se encuentren

dentro de ellas. Para ello se utilizaría una de las funciones de OpenCV: *cvQueryFrame* la cual extrae una imagen del video.

4.1.3. Modelar fondo con Mezclas Gaussianas (MoG)

Para comenzar el procesamiento se separa el frente (foreground) y el fondo (background) con la ayuda del método basado en las propiedades del Modelo de Mezcla de Gaussiana (MoG). Este se basa en utilizar K Gaussiana por cada píxel de la imagen que se procesa, a través de una desviación típica, una media y un factor de peso. La desviación típica indica lo que se puede alejar el valor medio más probable de dicho píxel, es decir, la media, por cada uno de los lados y el peso que determina la importancia de la Gaussiana. La de mayor peso sería la de mayor importancia y la suma de los pesos de una Gaussiana sería 1.

Primeramente se inicializan los valores de la media, desviación típica y peso por lo que en este caso, se inicializa la media de una de las Gaussianas (Ej. Gaussiana i=1) a la primera imagen y el resto de le dan valores aleatorios. Ya que la suma de los pesos de la Gaussiana debe ser 1, el porcentaje de distribución o peso (W) para K = 1 debe ser un valor muy próximo a 1 y para el resto se le asignarán valores muy pequeños, próximos a 0 todos.

Para obtener los píxeles que pertenecen al fondo de la imagen, se calculará la diferencia de los K modelos posibles. Si se obtiene algún parecido con alguna distribución entonces el píxel se marca como fondo y se actualizan los valores de esta distribución mediante una media móvil.

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

$$\Downarrow$$

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

Donde el factor $\rho = \alpha\ Pr\ (I_t(x,y) / \mu_{i,t-1}(x,y), \sigma^2_{i,t-1}(x, y))$, α va a ser un parámetro que desde el inicio de la ejecución se va a poner como predefinido, $I_t(x, y)$ es el valor del píxel en el instante t y $\mu_{i,t}(x, y)$ es el valor

de la media del pixel en el instante t. El resto de los píxeles que no sean clasificados como background, serán marcados como foreground.

Una de las grandes ventajas que posee este método es que no tolera oclusiones ya que si se produce una oclusión delante de un objeto, se crearía una nueva Gaussiana de Foreground para dicho objeto, perdiendo el seguimiento que se había realizado anteriormente.

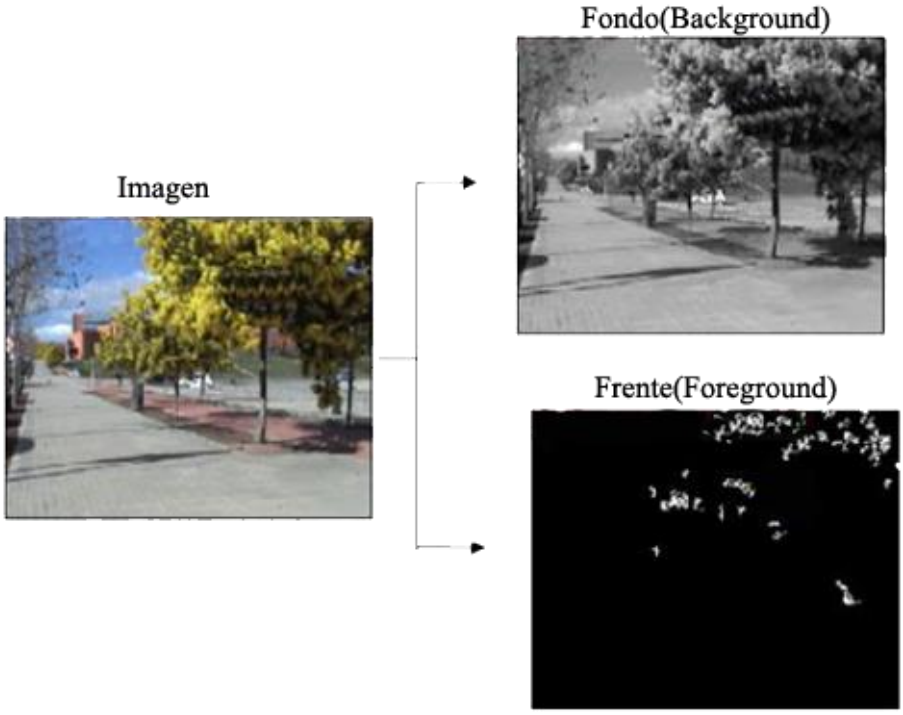


Ilustración 29: Separación frente-fondo

4.1.4. Detectar y eliminar sombras. Espacio de color HSV

Es necesario eliminar también las sombras generadas por la interacción de la iluminación con los objetos. Estas no forman parte del foreground ni del background. Esto se hace utilizando el espacio de color HSV (Tono (Hue), Saturación (Saturation) y Valor (Value)) en vez del espacio de color RGB utilizado usualmente, con el objetivo de detectar los píxeles que pertenecen a las sombras.

El espacio HSV es un subconjunto visible del espacio original con valores válidos de RGB. La razón principal de esta selección radica en la separación entre los valores de intensidad y color que se produce en el espacio de color HSV, trabajando con los 3 mapas de colores: Saturación, Tono y Valor. La Saturación

no es más que la relación entre el color y el Brillo, es decir, la pureza del color, el Tono: similitud que existe entre el Rojo, Amarillo, Verde o Azul o una combinación de ellos y el Valor que no es otra cosa que mayor o menor iluminación incidente, en otras palabras, es la intensidad de luz de un color.

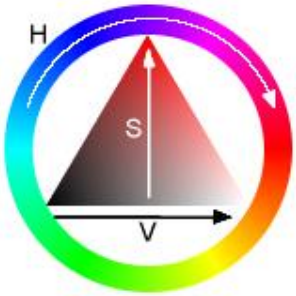


Ilustración 30: Espacio de color HSV

Por todo lo anteriormente dicho, la imagen a la que se le va a procesar las sombras, será convertida a HSV primeramente, siendo:

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

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

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

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

$$H = 360^{o} - H1, \quad si\ B > G$$

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

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

Luego de aplicar las fórmulas anteriores, se obtendría una imagen en HSV. Después de la conversión de RGB a HSV, se calcularán los tres mapas de colores mencionados anteriormente: Saturación (S), Tono (H) y Valor (V), con el objetivo de detectar cuáles píxeles de la máscara foreground obtenida pertenecen a la sombra. El primer paso es utilizar el valor V para filtrar los posibles píxeles que pueden o no pertenecer a las sombras. La decisión de que si un píxel pertenece o no a la sombra se basa en la siguiente fórmula:

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

Donde β y α son constantes que muestran que entre la imagen actual (I^v_t) y la de fondo (B^v_t), los valores de V no superan ni rebajan un determinado umbral, así como los valores de S y H tampoco lo superan. Los cálculos que se realizan se hacen píxel a píxel que en este caso sería el píxel de la posición (x,y). Finalmente, los que sean clasificados como sombra, serán eliminados de la máscara obteniendo un foreground sin sombras.

4.1.5. Eliminar ruido (blobs pequeños)

Algunos de estos blobs obtenidos son muy pequeños como para ser considerados importantes en la escena, por lo que se eliminarán las regiones donde el área sea más pequeña que un umbral determinado. A cada imagen se calculará un porcentaje aproximado del tamaño medio del rectángulo envolvente (*Bounding Box*) y los que el tamaño sea menor que dicho umbral, se eliminarán de la lista final de regiones con la ayuda de un filtro proporcionado por la librería *cvblobslib*.

De esta manera también se elimina el ruido, quitando las impurezas de la imagen que pueden ocasionar problemas durante el procesado. Esto es muy importante y necesario para el procesado ya que su presencia puede hacer que se produzcan errores y hasta generar nuevas regiones en el frente.

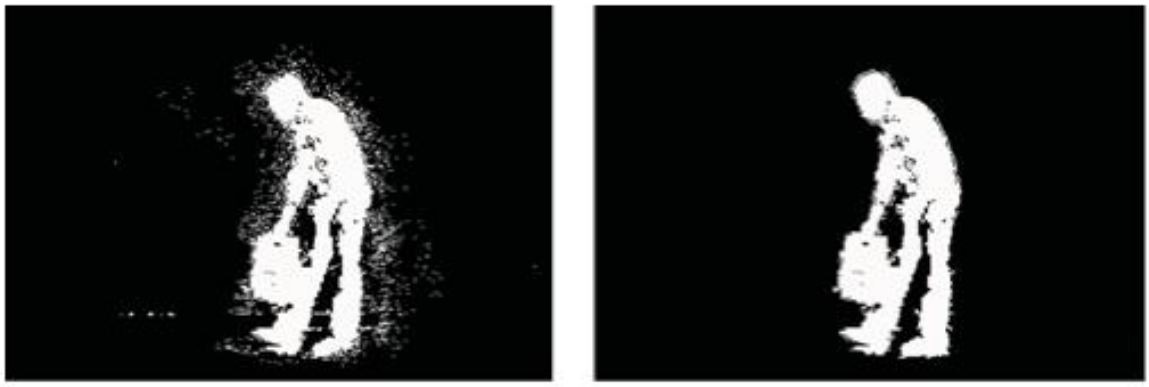


Ilustración 31: Eliminación del ruido

4.1.6. Extraer características de los blobs

Para realizar el procesamiento de las imágenes se necesita contar con una serie de características que poseen cada uno de los objetos obtenidos. Por tal razón se recopilan una serie de datos de cada blob (objeto). Uno de estos es la compactibilidad: porcentaje de píxeles con valor 1(en blanco) que contiene.

Otra de las características que se debe tener de los objetos es el radio que es la relación entre el ancho y el alto del objeto: R = W/H. También se debe contar con el valor del rectángulo envolvente del objeto. Para ello se utilizarán funciones de OpenCV como *CvRect*. Esto se hará ya que permite almacenar los valores del rectángulo envolvente de los objetos, almacenándolos según los valores (x, y, ancho, alto).

4.1.7. Submuestreo de máscaras binarias de foreground

Este paso es muy importante ya que permitirá localizar las regiones estáticas dentro de las imágenes. El número de máscaras binarias para el muestreo en este caso es de 6 aunque este valor puede variar. El tiempo escogido para clasificar un objeto como abandonado es de 30 segundos, ya que se considera que un objeto que esté estático durante todo este tiempo, es porque está abandonado. Se toma este valor y se multiplica por el rate que no es más que la cantidad de imágenes por segundos, obteniendo así la cantidad de imágenes totales con que se contaría al cabo de este tiempo. Como se quieren escoger solo 6 imágenes, se debe tomar un rango por lo que se divide este valor resultante entre 6 para determinar el rango con que se van a escoger los fotogramas. Contando ya con el intervalo se ejecutaría el submuestreo, tomando cada una de las 6 imágenes escogidas utilizando el intervalo calculado.

Los 50 primeros fotogramas se utilizan para modelar el fondo. Una vez obtenidos se comienza el submuestreo. Aplicando la técnica de Modelo de Mezclas Gaussianas (MoG), se extraen las 6 máscaras binarias de foreground cuyos píxeles que tengan valor 255 pertenecerán al foreground (frente) y los de valor 0 al background (fondo). Luego, se efectúa una multiplicación lógica píxel a píxel de cada una de las 6 máscaras binarias de foreground que se han obtenido, para contar así con una imagen final de objetos estáticos S. Esto se hará de la siguiente manera:

$$S=M_1 * M_2 * M_3 * M_4 * M_5 * M_6$$

El valor M, son las imágenes del frente del muestreo después de actualizar el modelo de fondo con cada muestra. Finalmente, se puede afirmar que un objeto en dicha máscara final que tenga valor 1 o 255, se

considera una región que ha permanecido estática durante los 30 últimos segundos. Por tanto, se puede considerar como un objeto abandonado y se pasaría a las siguientes etapas del análisis del algoritmo.

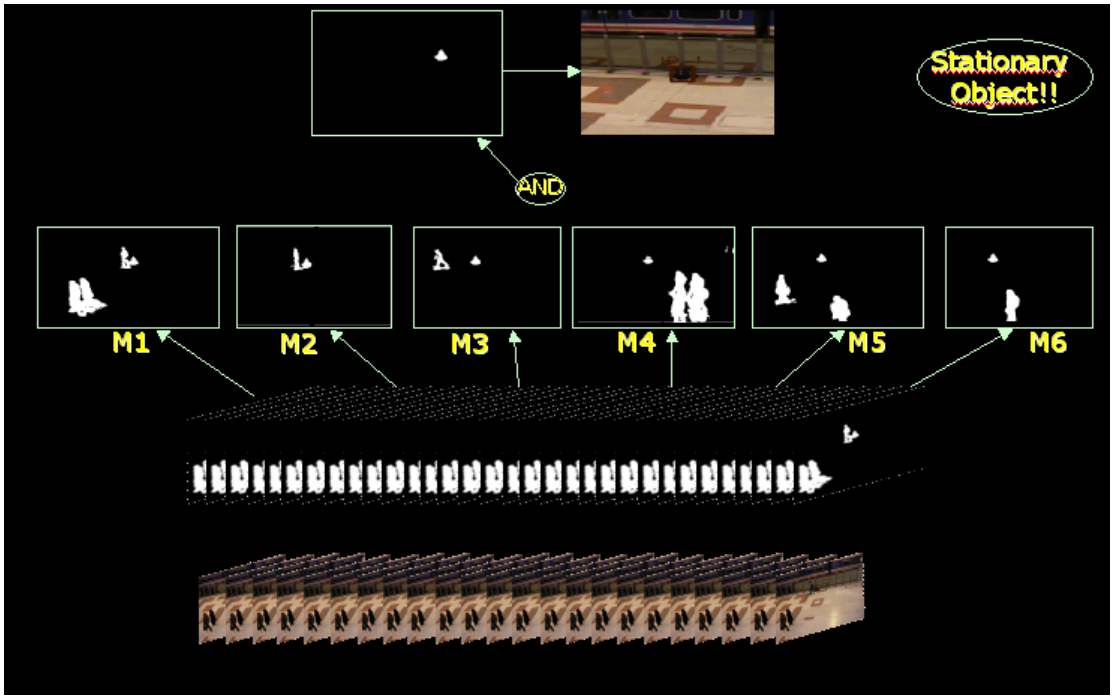


Ilustración 32: Submuestreo de máscaras

4.1.8. Obtención de blobs (Objetos)

Luego se extraen los blobs que se encuentran en la escena. Esto se lleva a cabo haciendo uso de la librería *cvblobslib*, permitiendo extraer los componentes que sean conexos en la máscara foreground. El algoritmo que emplea es "*Connected Component Algorithm using 8 neighborhood*" y según (Sciencie, 2005), *"funciona mediante el escaneo de una imagen, píxel por píxel (de arriba a abajo y de izquierda a derecha) con el fin de identificar relacionada regiones de píxeles, es decir, las regiones de los píxeles adyacentes que comparten el mismo conjunto de valores de intensidad V."*

Este algoritmo se basa en un análisis que realiza para cada uno de los píxeles y una comparación que hace con los píxeles vecinos, reuniendo los que tengan valores similares, obteniendo de esta forma todas las regiones conexas. A cada una de estas regiones se le asignará una etiqueta característica de cada cual. También se calculan los *Bounding Box* o rectángulos englobantes de cada uno de los *blob*s que han sido seleccionados.

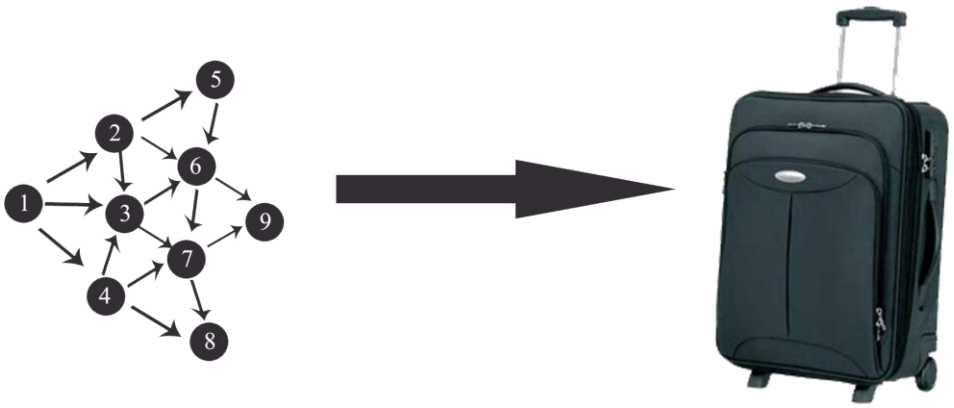


Ilustración 33: Algoritmo de Componentes Conexos.

4.1.9. Discrimación Persona/Objeto

Antes que todo vale aclarar que clasificar un blob como un objeto no es una tarea sencilla ya que una persona puede adoptar múltiples posturas y formas, cambiándolas a cada momento, es decir, no tiene un modelo fijo. También está el problema de la ropa y los accesorios como por ejemplo: mochilas, gorras, sombrillas e incluso hasta el peinado. Todos estos factores hacen más difícil la clasificación.

Para conocer si un blob que ha sido detectado es una persona o un objeto, se utiliza la combinación de dos algoritmos. El primer algoritmo se basa en calcular el radio del blob y según su valor resultante entonces se considerará como objeto o persona.

El segundo algoritmo se basa en calcular el porcentaje de píxeles con valor 1(en blanco) o compactibilidad que estén dentro del rectángulo envolvente. Si su valor es mayor que cierto umbral determinado previamente, entonces el blob es objeto sino es persona. Dicho umbral *p* se ha determinado de manera empírica y su valor es alrededor al 70% y el 75%.

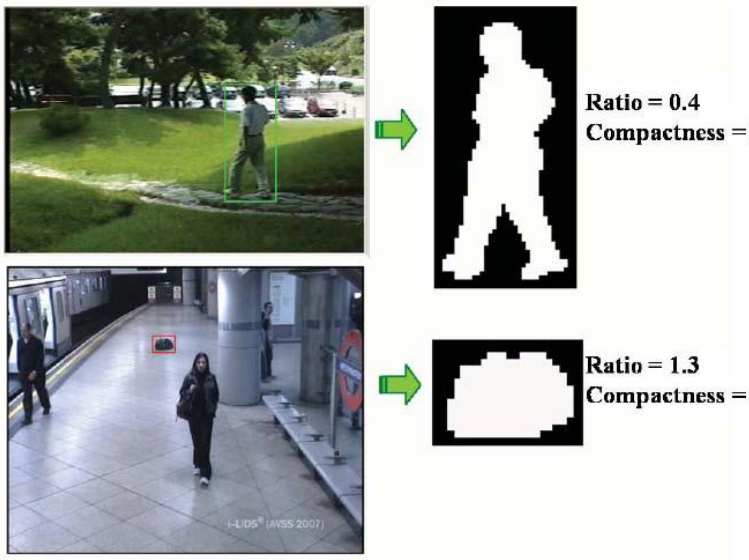


Ilustración 34: Clasificación Persona/Objeto

4.1.10. Enviar alarma

Si finalmente se detecta un objeto abandonado en la escena entonces se emite una alarma, pintando un rectángulo envolvente de color azul sobre el objeto detectado con ayuda de funciones de OpenCV como *CvRect* y *cvDrawRect*. Ésta última, con la ayuda de las coordenadas almacenadas en la variable de tipo *CvRect* es quien se encarga de dibujar el rectángulo alrededor del objeto encontrado.

4.2. Modelo de implementación

4.2.1. Diagrama de componente

Según (Microsoft Corporation, 2011), *"Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces."*

Según (Universidad de Castilla-La Mancha, 2011), *"Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, etc."*

A continuación se muestra el diagrama de componente del sensor.

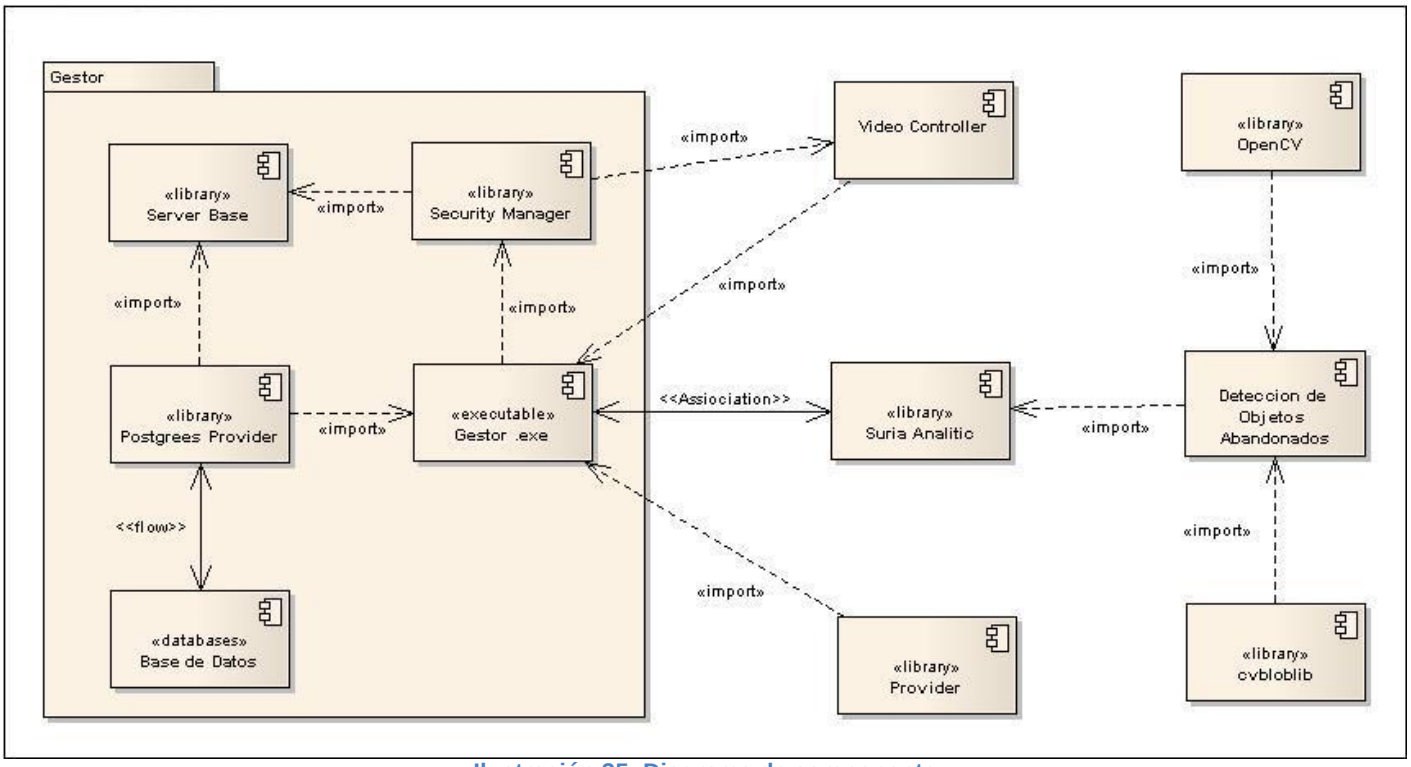


Ilustración 35: Diagrama de componente

4.2.2. Diagrama de Despliegue

Seguidamente se muestra el diagrama de despliegue del componente. Aquí se ve como el componente Detección de objetos abandonados está junto con otros componentes, integrado al sistema Suria Analytic, el cual a su vez se conecta al Gestor.

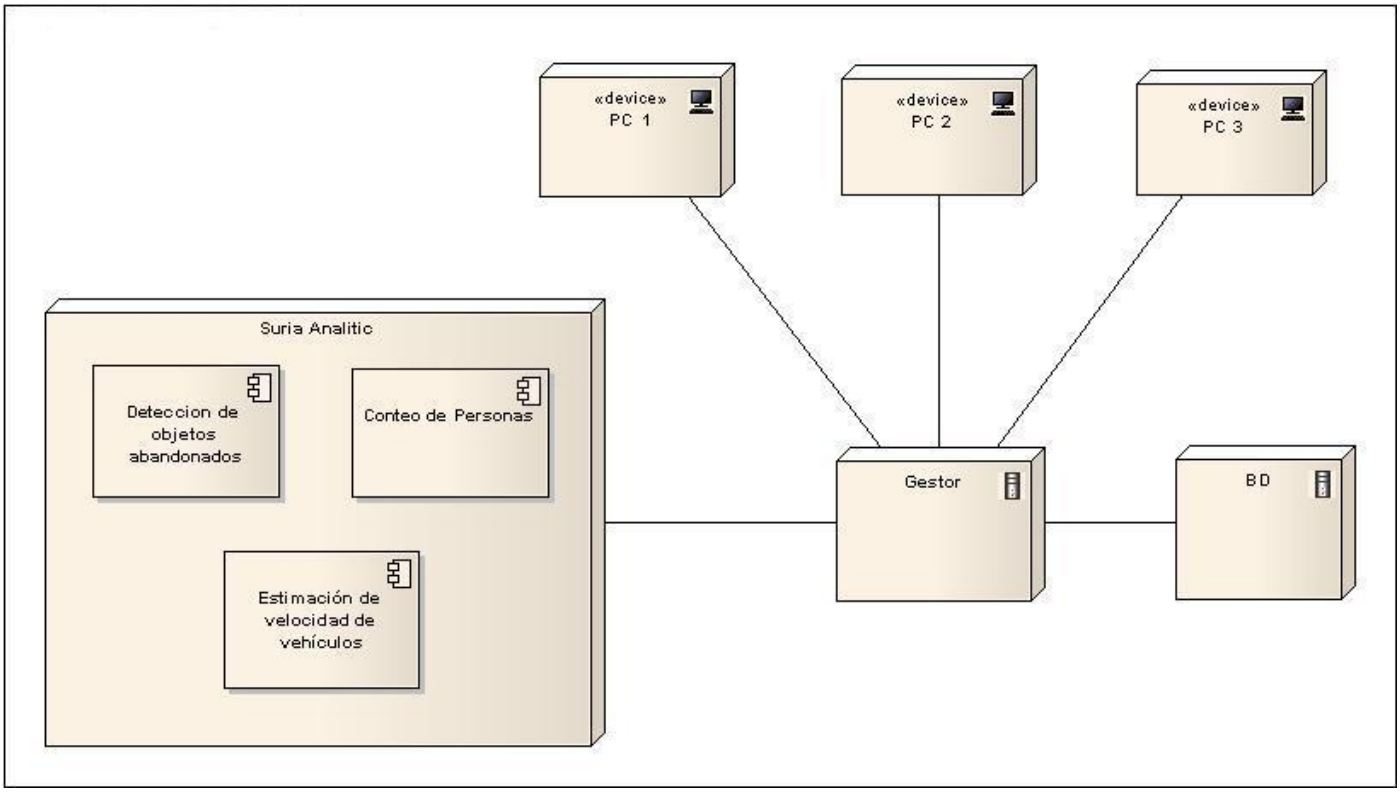


Ilustración 36: Diagrama de Despliegue

4.3. Pruebas al sistema

4.3.1. Falsos Positivos y falsos negativos.

Finalizada la fase de implementación de nuestro componente, es el momento de comprobar su eficacia. A continuación se pueden apreciar las pruebas realizadas al software para validar la calidad obtenida, tomando como base dos videos. Para esto se hará un estudio de los falsos positivos y falsos negativos de estos dos videos lo que permitirá evaluar su calidad.

En el primer análisis, el sistema detectó dicho objeto y envió una alarma correctamente por lo que hay un 100% de factibilidad o grado de cumplimiento respecto a las funcionalidades del software. En el segundo ocurrió lo mismo pero el sistema no logró detectar el objeto por lo que hay un 50% de factibilidad. Esto ocurrió debido a que el video se filmó con una cámara digital, provocando problemas en el procesamiento.

En el análisis se considera como: FN son los objetos que el algoritmo debería haber detectado pero no hizo y como FP aquello que detectó pero que no es un objeto estático. En este caso puede ser una persona que se encuentre sin moverse en un punto determinado de la escena, sombras, ruido, etc.

A continuación, se muestran dos tablas con los porcentajes de falsos positivos y falsos negativos de los algoritmos:

Tomando en cuenta el video sensor para detección de objetos estáticos, el índice de precisión (presicion rate) cuantifica el total de objetos detectados satisfactoriamente por el sistema. El recall rate es el porciento de objetos detectados correctamente en el total de objetos que debió encontrar el sistema. Formalmente están definidos por las siguientes ecuaciones:

$$presicion\ rate = DC/DC+Fp \qquad recall\ rate = DC/TO$$

Donde DC es el número de objetos detectados correctamente por el sistema, *TO* es el total de objetos que el sistema debió encontrar y *Fp* (Falsos Positivos) es todo lo que el sistema detecta que no es un objeto como: la sombra, la iluminación, una persona, el ruido, etc. Para evaluar experimentalmente el método propuesto de detección de objetos abandonados se realizó la grabación de 2 videos en diferentes ambientes y posiciones de las cámaras. En la Tabla se muestra detalladamente los resultados experimentales obtenidos. Ver (University of Glasgow, 2010).

Parámetro	Valor
Detecciones Correctas(DC)	2
Total de objetos(TO)	3
Falsos Positivos	0
Falsos Negativos	1
recall rate	66%
presicion rate	100%

Tabla 14: Resultados

Hay un 66% de factibilidad o grado de cumplimiento respecto a las funcionalidades del software pues solo se logra detectar este por ciento de objetos detectados correctamente.

4.3.2. Pruebas de Unidad

Para realizar las pruebas de unidad al software se ha escogido las Prueba de Caja Blanca, pues no posee interfaz. La técnica a emplear será Camino Básico. Esta técnica posibilita obtener una medida de la complejidad lógica del diseño procedimental y utiliza esa medida como guía para la definición de un conjunto básico de caminos de ejecución, de los cuales se obtienen los Casos de Prueba, que garantizan la ejecución de cada sentencia del programa al menos una vez, durante las pruebas.

4.3.2.1. Prueba Caja Blanca: Camino Básico

Paso #1: Grafo de flujo asociado: Clase Sombra. Método ShadowDetectionDelete

```
{
1  IplImage *hsvBkg = cvCreateImage(cvGetSize(bkg),8,3);
1  this->ConvertToHSV(bkg,hsvBkg);
1  IplImage *hsvImage = cvCreateImage(cvGetSize(bkg),8,3);
1  this->ConvertToHSV(image,hsvImage);
1  int dh = 0;
2  for(int i=0;i<hsvImage->height;i++)
   {
3      for(int j=0;j<hsvImage->width;j++)
       {
4          CvScalar imageValue;
4          CvScalar bkgValue;
4          imageValue.val[0]= (float)((uchar *)(hsvImage->imageData + i*hsvImage->widthStep))[j*hsvImage->nChannels +
4          imageValue.val[1]= (float)((uchar *)(hsvImage->imageData + i*hsvImage->widthStep))[j*hsvImage->nChannels +
4          imageValue.val[2]= (float)((uchar *)(hsvImage->imageData + i*hsvImage->widthStep))[j*hsvImage->nChannels +
4          bkgValue.val[0]= (float)((uchar *)(hsvBkg->imageData + i*hsvBkg->widthStep))[j*hsvBkg->nChannels + 0];
4          bkgValue.val[1]= (float)((uchar *)(hsvBkg->imageData + i*hsvBkg->widthStep))[j*hsvBkg->nChannels + 1];
4          bkgValue.val[2]= (float)((uchar *)(hsvBkg->imageData + i*hsvBkg->widthStep))[j*hsvBkg->nChannels + 2];
4         CvScalar sf;
4          sf.val[0] = (float)((uchar *)(fgWithoutShadow->imageData + i*fgWithoutShadow->widthStep))[j];
5          if(sf.val[0] == 255)
           {
6              dh = abs(imageValue.val[0]-bkgValue.val[0]);
7              if(dh <= tau_h)
               {
8                  if( imageValue.val[1] - bkgValue.val[1] <= tau_s)
                   {
9                      if(imageValue.val[2]/bkgValue.val[2] >= alpha && imageValue.val[2]/bkgValue.val[2] <= beta)
                       {
10                         ((uchar *)(fgWithoutShadow->imageData + i*fgWithoutShadow->widthStep))[j] = 0;
                       }
                   }
               }
           }
11     }
12 }
13 cvReleaseImage(&hsvBkg);
13 cvReleaseImage(&hsvImage);
}
```

Ilustración 37: Técnica Camino Básico. Paso #1.

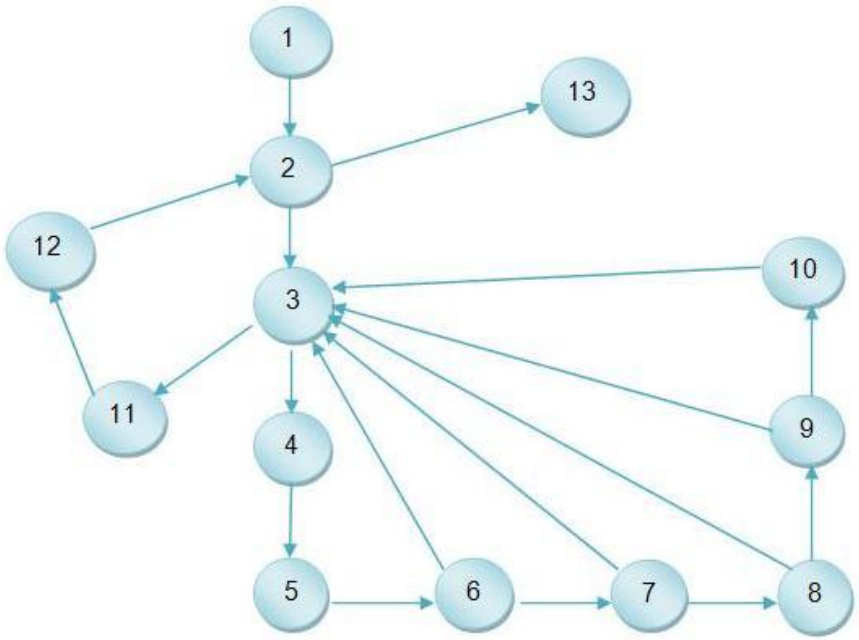


Ilustración 38: Técnica Camino Básico. Paso #1.

Paso #2: Complejidad Ciclomática

Proporciona una medición cuantitativa de la complejidad lógica de un programa, define el número de caminos independientes del conjunto básico de este y brinda el límite superior para el número de pruebas a realizar que aseguren la ejecución de todas las sentencias al menos una vez. La Complejidad Ciclomática se denota por *V(G),* NA: número de aristas y NN: número de nodos.

V (G) = NA - NN + 2 CC = 18 - 13 + 2 = 7

Paso #3: Determinar un conjunto de Caminos Básicos Independientes

Camino #1: 1, 2, 13

Camino #2: 1, 2, 3, 4, 5, 3, 11, 12, 13

Camino #3: 1, 2, 3, 4, 5, 6, 7, 3, 11, 12, 13

Camino #4: 1, 2, 3, 4, 5, 6, 7, 8, 3, 11, 12, 13

Camino #5: 1, 2, 3, 4, 5, 6, 7, 8, 9, 3, 11, 12, 13

Camino #6: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 3, 11, 12, 13

Camino #7: 1, 2, 3, 11, 12, 13

Paso #4: Casos de Prueba

Caja Blanca. Método ShadowDetectionDelete. Caso de Prueba #1	
Camino:	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 3, 11, 12, 2, 13
Caso de Prueba:	Probando la función ShadowDetectionDelete
Entrada:	Para un valor de bkg = IplImage, image = IplImage y para un valor d fgWithoutShadow = IplImage.
Resultado:	Detectar y eliminar las sombras que se encuentren en la imagen.

Tabla 15: Paso #4: Casos de Prueba

Paso #1: Grafo de flujo asociado: Clase Persona. Método IsPerson

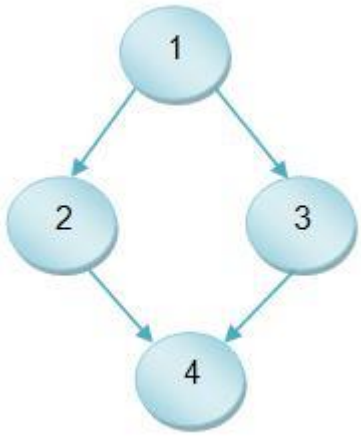


Ilustración 39: Técnica Camino Básico. Paso #1.

```
bool  Persona::IsPerson(float ratio, float compactness)
{
    1 if (ratio <= 0.4 && compactness >= 30 && compactness <= 55)
      {
    2   return true;
    3 }
    4 return false;
}
```

Ilustración 40: Técnica Camino Básico. Paso #1.

Paso #2: Complejidad Ciclomática

V (G) = NA - NN + 2 = 4 - 4 + 2 = 2

Paso #3: Determinar un conjunto de Caminos Básicos Independientes

Camino #1: 1, 2, 4

Camino #2: 1, 3, 4

Caja Blanca. Método IsPerson. Caso de Prueba #2	
Camino:	1, 2, 4
Caso de Prueba:	Probando la función IsPerson
Entrada:	Para un valor de ratio = float y un valor de compactness = float.
Resultado:	El tipo de dato es Persona.

Tabla 16: Caja Blanca. Método IsPerson. Caso de Prueba #2

Caja Blanca. Método IsPerson. Caso de Prueba #3	
Camino:	1, 3, 4
Caso de Prueba:	Probando la función IsPerson
Entrada:	Para un valor de ratio = float y un valor de compactness = float.
Resultado:	El tipo de dato es Objeto.

Tabla 17: Caja Blanca. Método IsPerson. Caso de Prueba #3

No del camino	Caso de Prueba	Objetivo	Resultado
6	Probando la función ShadowDetectionDelete	Detectar y eliminar las sombras que se encuentren en la imagen.	Satisfactorio
1	Probando la función IsPerson	Determina si es una persona o no un objeto.	Satisfactorio
2	Probando la función IsPerson	Determina si es una persona o no un objeto.	Satisfactorio

Tabla 18: Resultados de las pruebas de Caja Blanca

Conclusiones

Como se puede observar, con la ayuda de las diferentes técnicas de procesamiento inteligente de video como el Modelo de Mezclas Gaussianas y el submuestreo de píxeles, se puede desarrollar un componente que pueda detectar objetos abandonados en estancias monitoreadas. Estas técnicas son de gran utilidad ya que su utilización permite que se le otorgue las funcionalidades necesarias al componente como el de separar los fotogramas del video y modelar el frente (foreground) y el fondo (background) de los mismos. Además, se ha logrado un mejor entendimiento de cómo funciona el algoritmo a través de la presentación de las diferentes técnicas de procesamiento inteligente de video. También se ha brindado una visión general de cómo quedó finalmente implementada la aplicación.

CONCLUSIONES GENERALES

Actualmente existen muchos videos sensores que permiten la detección de objetos abandonados pero todos son software privativos por lo que para poder usarlos hay que pagar una gran suma de dinero que no todos poseen. Esta solución, a diferencia de estos componentes, se considera una aplicación distribuible que no necesita pagar licencia alguna ya que está construido con C++. El incorporamiento del procesamiento inteligente de video e imágenes digitales al componente, ha permitido utilizar técnicas como el Modelo de Mezclas Gaussianas, el Algoritmo de Componentes Conexos y el submuestreo de píxeles, que posibilitan cumplir con las funcionalidades requeridas por el componente. Estas técnicas son de gran utilidad ya que su utilización permite que se le otorgue las funcionalidades necesarias al componente como el de separar los fotogramas del video y modelar el frente (foreground) y el fondo (background) de los mismos.

Además, con la ayuda de las herramientas y tecnologías descritas, se pudo obtener un producto de buena calidad y fiabilidad. Además, se ha logrado a través de un estudio de las diferentes técnicas de procesamiento inteligente de video, un mejor entendimiento de cómo se realiza el funcionamiento del algoritmo. También se ha obtenido una visión general de cómo quedó implementada la aplicación. Finalmente, durante la investigación se obtuvo un componente que permite la detección de objetos abandonados, utilizando las técnicas de procesamiento inteligente de video mencionadas anteriormente.

La incorporación de este componente al Sistema de Video Vigilancia SURIA significará un gran aporte ya que el sistema podrá detectar objetos abandonados en los entornos que monitorea. Con este Video Sensor, ahora el sistema podrá procesar flujos de videos provenientes de cámaras IP, utilizando técnicas de procesamiento inteligente de video. También constituye un paso de avance para los guardias de seguridad ya que les servirá de apoyo y ayuda a la hora de la vigilancia de los locales y áreas.

RECOMENDACIONES

El autor del presente trabajo de diploma recomienda:

- Continuar el estudio de métodos y algoritmos para realizar el procesamiento inteligente de imágenes.
- Continuar la optimización del componente para lograr un mejor tiempo de procesamiento.
- Utilizar el componente propuesto dentro del Sistema SURIA.
- Mejorar la captura del flujo de video.
- Migrar el componente a software libre.

REFERENCIAS BIBLIOGRÁFICAS

Martín, Sonsoles Herrero, (2009), Análisis comparativo de técnicas de segmentación de secuencia de video basado en el modelado del fondo, 2009

APEXNET, (2011), APEXNET, http://www.apexnet.com.ar/index.php/product/viewProducts/24/sl=0

Carrera, Antonio Collazos, (2009), Detección de Objetos Abandonados en Estancias Controladas, Madrid, 2009

DEV, (2010), DEV, [Online] 2010, http://dev.mysql.com/doc/refman/5.0/es/blob.html

Benito, Pablo Cervera, (2010), Integración de información de movimiento en la segmentación de secuencia de video basada en el modelado de fondo, 2010.

Microsoft Corporation, (2011), Developer, [Online] 2011, http://msdn.microsoft.com/es-es/library/dd409390.aspx

Microsoft Corporation, (2010), Visual Studio Magazine, [Online] Microsoft, 01 04, 2010, http://visualstudiomagazine.com/articles/2010/04/01/the-evolution-of-visual-c-in-visual-studio-2010.aspx

González, Danays Rodríguez Martínez y Marilys Valiente, (2010), PGST: "Módulo para la gestión de configuración del equipamiento tecnológico en la Universidad de las Ciencias Informáticas", La Habana, 2010

Zayas, Kalianny Laffita Nicot y Yaima Mariño, (2010), Propuesta de Algoritmos para la Reducción de Instancias, Ciudad de la Habana, 2010.

Sciencie, School of Electronic Engineering and Computer. (2005). Queen Mary. University of London. [Online] 04 2005. http://www.eecs.qmul.ac.uk/~phao/CIP/Labs/handouts/connected.pdf

Revista Negocios de Seguridad, (2010), Intekio, [Online] 2010, http://www.intekio.com/material/Informe_VI_rnds.pdf

Scribd, (2011), [Online] 2011, www.scribd.comwww.scribd.comhttp://www.scribd.com/doc/2050925/metodologias-de-desarrollo-software

Four, Edmis Deivis Semanat Aldana y Leonor Verdecia, (2009), Sistema de Video Vigilancia, Ciudad de La Habana, Cuba, 2009

Martín, Marcos, (2002), Técnicas Clásicas de Segmentación de Imagen, 2002.

John Makhoul, Francis Kubala, Richard Schwartz, Ralph Weischedel, (2011), PERFORMANCE MEASURES FOR INFORMATION EXTRACTION, 2011

Universidad de Castilla-La Mancha, (2011), [Online] 2011, Departamento de Sistemas Informáticos, http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf

Universidad de las Ciencias Informaticas, (2009), Conferencia: "Características del video. Conceptos", La Habana, Cuba, 2009.

Universidad de las Ciencias Informáticas, (2010), Conferencia_Tema_3_El_diseño_metodológico_de_la_investigación_científica, La Habana, Cuba, 2010.

Universidad de Sevilla, (2011), Patterns: Facade y Template Method, [Online] 2011, http://www.google.com.cu/url?sa=t&source=web&cd=7&ved=0CD4QFjAG&url=http%3A%2F%2Fwww.lsi.us.es%2Fdocencia%2Fget.php%3Fid%3D1310&rct=j&q=patron%20fachada&ei=4TehTbHnN6Tj0gHl2IShBQ&usg=AFQjCNET9_ABqmgkj0vPiLiaWv_VTugkeA&cad=rja

Universidad Javeriana Bogotá, (2009), Opencvjaveriana, [Online] Web de la Universidad Javeriana Bogotá, http://opencvjaveriana.wikispaces.com/

University of Glasgow, (2010), University of Glasgow, [Online] 2010, http://www.dcs.gla.ac.uk/Keith/Chapter.7/Ch.7.html

Pechuán, Luis Miralles, 2007, Uso de ontologías para guiar el proceso de segmentación de imágenes. 2007

Web del Profesor, (2010), Web del Profesor, [Online] 2010, http://webdelprofesor.ula.ve/humanidades/raymond/computacion2/archivos/ImagDig.pdf

BIBLIOGRAFÍA CONSULTADA

Martín, Sonsoles Herrero, (2009), Análisis comparativo de técnicas de segmentación de secuencia de video basado en el modelado del fondo, 2009

Microsoft Corporation, (2011), Developer, [Online] 2011, http://msdn.microsoft.com/es-es/library/dd409390.aspx

Revista Negocios de Seguridad, (2010), Intekio, [Online] 2010, http://www.intekio.com/material/Informe_VI_rnds.pdf

Microsoft Corporation, 2010, Visual Studio Magazine, [Online] Microsoft, 01 04, 2010, http://visualstudiomagazine.com/articles/2010/04/01/the-evolution-of-visual-c-in-visual-studio-2010.aspx

González, Danays Rodríguez Martínez y Marilys Valiente, (2010), PGST: “Módulo para la gestión de configuración del equipamiento tecnológico en la Universidad de las Ciencias Informáticas”, La Habana, 2010

Zayas, Kalianny Laffita Nicot y Yaima Mariño, (2010), Propuesta de Algoritmos para la Reducción de Instancias, Ciudad de la Habana, 2010.

Four, Edmis Deivis Semanat Aldana y Leonor Verdecia, (2009), Sistema de Video Vigilancia, Ciudad de La Habana, Cuba, 2009

Universidad de las Ciencias Informaticas, (2009), Conferencia: "Características del video. Conceptos", La Habana, Cuba, 2009.

University of Glasgow, (2010), University of Glasgow, [Online] 2010, http://www.dcs.gla.ac.uk/Keith/Chapter.7/Ch.7.html

Universidad de las Ciencias Informáticas, (2010), Conferencia_Tema_3_El_diseño_metodológico_de_la_investigación_científica, La Habana, Cuba, 2010.

Web del Profesor, (2010), Web del Profesor, [Online] 2010, http://webdelprofesor.ula.ve/humanidades/raymond/computacion2/archivos/ImagDig.pdf

ANEXOS

Anexo 1: Diagrama de colaboración del CU Discriminación Persona/Objeto

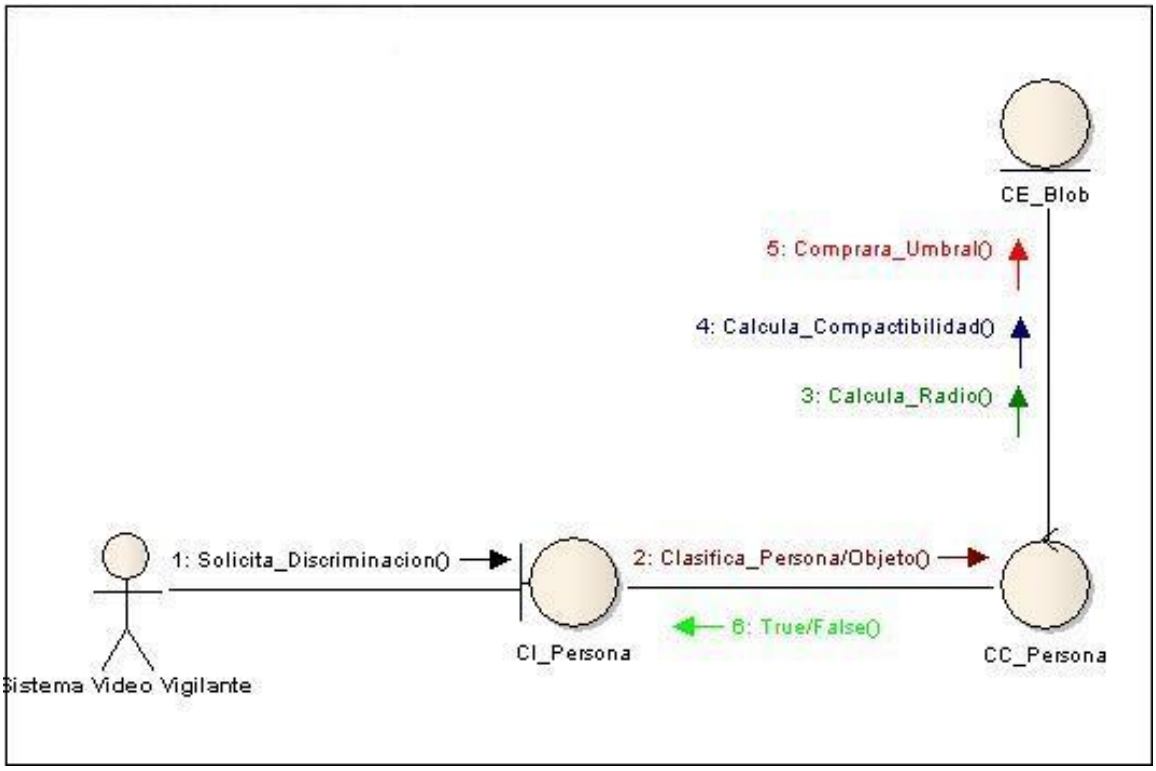


Ilustración 41: Diagrama de colaboración del CU Discriminación Persona/Objeto

Como pudieron ver en el diagrama de colaboración para el caso de uso Procesar Sombras, se aprecian los mensajes fundamentales que son intercambiados entre las clases y finalmente se obtienen los objetos sin sombras de la escena.

Anexo 2: Diagrama de secuencia para el CU Discriminación Persona/Objeto

En la ilustración que se aprecia a continuación, se encuentra el diagrama de secuencia para el caso de uso Discriminación Persona/Objeto. Se muestra la interacción que existe entre las clases Persona y Blob, así como los mensajes enviados entre ellos.

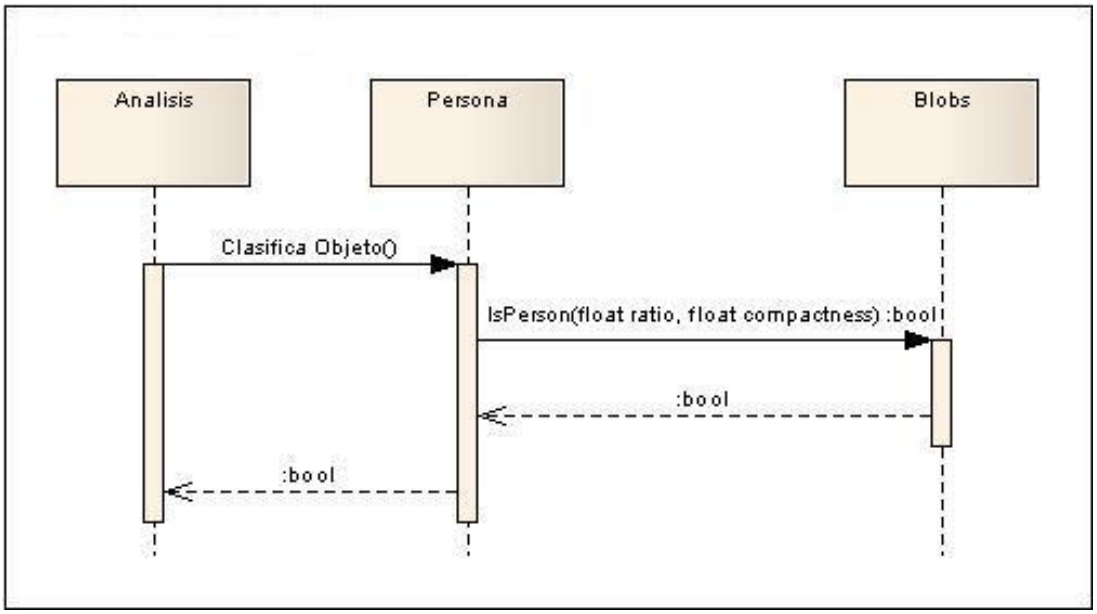


Ilustración 42: Diagrama de secuencia para el CU Discriminación Persona/Objeto

GLOSARIO DE TÉRMINOS

Actor: Cualquier individuo, grupo, organización, máquina o sistema de información externo que interactúa con el negocio.

Componente: cualquiera de los elementos que podemos insertar en una ficha, tanto si su función es visual como si no lo es (por supuesto un componente es también un objeto).

Biblioteca: Colección de clases y funciones.

Frame: Una de las imágenes que compone un video.

Fotogramas: Imágenes contenidas en un video.

GRASP: General Responsibility Assignment Software Patterns.

Herramienta de modelado: Herramientas que permiten la creación de entidades, propiedades y relaciones de objetos.

IDE: Entorno de desarrollo integrado.

Imagen Digital: Representación bidimensional producto del desarrollo de la Informática.

Iterativo: Se ejecuta mediante ciclos.

Metodología: Marco de trabajo que recoge un conjunto de pasos y procedimientos que se deben seguir durante el desarrollo de un software.

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.

RUP: Proceso Unificado de Desarrollo de Software.

Rate: Cantidad de imágenes por segundos en un video.

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.