Universidad de las Ciencias Informáticas

FACULTAD 6

TÍTULO: VIDEO SENSOR PARA LA DETECCIÓN DE MANIPULACIÓN DE CÁMARAS DE VIDEO VIGILANCIA.

Trabajo de Diploma para optar por el título de

Ingeniero en ciencias informáticas

Autor: Lisandra Matos Servera

Tutor: Ing. Reinier Pupo Ruiz

Año 55 de la Revolución

DECLARACIÓN DE AUTORÍA

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

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

Lisandra Matos Servera

_____
Firma del Autor

Reinier Pupo Ruiz

_____
Firma del Tutor

Datos de Contacto

Graduado de la Universidad de las Ciencias Informáticas (UCI) en el año 2009. Es especialista general y ha impartido clases de Programación 2, Procesamiento Digital de Imágenes I y II y Video y Sonido Digital en las facultades 6 y 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

Le dedico esta tesis a mi familia, que me apoyó mucho en estos cinco años y sin ellos, no hubiera sido tan fácil mi estadía en esta escuela.

Agradecimientos

Les agradezco en primer lugar a mis padres por su apoyo, su comprensión y sus esfuerzos.

Al resto de mi familia que convive conmigo que siempre se preocupó por mí, a mi abuela que si suspendía lloraba conmigo y si aprobaba se reía de mí porque ya había llorado por gusto dos días antes, a mi prima, mi tía, Serance que me soporta cada día y a mi bella sobrina.

A mis hermanos Manolo, Larytza y Yamilet que me ayudaron mucho.

A mi tutor que Reinier Pupo Ruiz por su ayuda incondicional.

A mis amigas y amigos de Bauta que si digo nombres se me ponen celosos.

A mis amistades de la escuela que son unas cuantas, más de las que esperaba, porque soy desesperante.

A la Revolución cubana que me dio los medios para convertirme en una profesional y poder brindarle mis servicios a la patria.

Resumen

En la Universidad de Ciencias Informáticas (UCI) se desarrolla el proyecto Video Vigilancia Suria, llevado a cabo por el Centro Geoinformática y Señales Digitales (GEYSED). El objetivo de este trabajo es proporcionarle a Suria una herramienta capaz de detectar cuando una cámara ha sido movida del ángulo de vigilancia, tapada o desenfocada, enviar las alarmas necesarias al sistema y así evitar brechas en la seguridad del entorno. Para darle solución a este problema se utilizaron algoritmos que procesan las imágenes y video digital capturados por las cámaras. Los algoritmos implementados utilizan funciones de la biblioteca OpenCV como: la sustracción de fondos en los tres tipos de detección y más específico, el cálculo del histograma en el algoritmo de detección de tapado, la extracción de bordes en el algoritmo de detección de desenfoque y comparación de pixeles en el algoritmo de detección de movimiento. Se realizaron pruebas de rendimiento para comprobar la eficiencia de los algoritmos.

PALABRAS CLAVE

Desenfoque, manipulación, movimiento, sensor, tapado, video.

Tabla de Contenidos

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTOS BASICOS PARA EL DESARROLLO DEL PROCESAMIENTO DE IMAGEN Y VIDEO DIGITAL 5

1.1 Introducción. 5
1.2 Definiciones generales. 5
1.3 Objeto de estudio. 9
1.3.1 Descripción general. 9
1.4 Análisis de soluciones existentes. 10
1.5 Herramientas y tecnologías a utilizar. 13
1.5.1 Metodología de desarrollo. 13
1.5.2 Lenguaje de modelado. 15
1.5.4 Lenguaje de Programación 16
1.5.5 Biblioteca a utilizar. 17
1.5.6 Entorno de desarrollo 18
1.6 Conclusiones 20

CAPÍTULO 2: ANÁLISIS DE LAS CARACTERISTICAS DEL SISTEMA 21

2.1 Introducción. 21
2.2 Sistema propuesto. 21
2.3 Modelo de dominio. 21
2.3.1 Conceptos fundamentales. 21
2.3.2 Diagrama del Modelo de dominio. 22
2.4 Especificación de los requisitos del Software. 23
2.4.1 Requisitos Funcionales 23
2.4.2 Requisitos no funcionales 23
2.5 Definición de Casos de Uso. 25
2.5.1 Definición de actores 25
2.5.2 Listado de los casos de uso. 25

2.6 Diagrama de Casos de Uso. 25
2.6.1 Descripción de los Casos de Uso del Sistema 26
2.7 Arquitectura a utilizar y patrones de diseño seleccionados. 27
2.7.1 Arquitectura. 27
2.7.2 Patrones de Diseño. 28
2.8 Conclusiones 30
CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA 31
3.1 Modelo de Análisis. 31
3.1.1 Diagrama de clases de Análisis. 32
3.2 Diagramas de Interacción. 34
3.2.1 Diagramas de Comunicación de los Caso de Uso. 34
3.3 Modelo de Diseño. 37
3.3.1 Diagrama de clases del diseño. 37
3.3.2 Diagrama de interacción del diseño 39
3.4 Conclusiones. 43
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA 44
4.1 Introducción 44
4.2 Análisis de los algoritmos. 44
4.2.1 Modelado de fondo. 44
4.2.2 Detectar desenfoque del lente. 46
4.2.3 Detectar movimiento de la cámara. 47
4.2.4 Detectar tapado del lente de la cámara. 49
4.3 Modelo de implementación. 51
4.3.1 Diagrama de Componentes. 51
4.3.2 Diagrama de despliegue. 52
4.4 Pruebas de Software. 52
4.4.1 Prueba de Falsos Positivos y Falsos Negativos. 53
4.4.2 Resultados experimentales. 54
4.5 Conclusiones. 56
CONCLUSIONES 57

RECOMENDACIONES 58

TRABAJOS CITADOS 59

BIBLIOGRAFÍA 61

ANEXOS 63

Anexo 1: Descripciones de los casos de Uso 63

GLOSARIO 67

Tabla de Ilustraciones y Diagramas

Ilustración 1: Imagen Digital 6
Ilustración 2: Píxel 6
Ilustración 3: Video 7
Ilustración 4: Procesamiento inteligente de video para la detección de objetos perdidos 8
Ilustración 5: Procesamiento inteligente para la detección de robo 8
Ilustración 6: Procesamiento inteligente para la detección perimetral 8
Ilustración 7: Análisis del procesamiento. 10
Ilustración 8: Captura de pantalla del sistema de análisis de video de Intellivision 11
Ilustración 9: Cámara Vivotek IP7160 12
Ilustración 10: Interfaz de Cisco® Video Analytics 13
Ilustración 11: Flujos de trabajo RUP (Chacón, 2006) 15
Ilustración 12: Representación gráfica de la Arquitectura de diseño del Video Sensor. 28
Ilustración 13: Fondo del fotograma. 45
Ilustración 14: Fotograma. 46
Ilustración 15: Imagen actual y bordes extraídos. 47
Ilustración 16: Modelo de fondo y bordes extraídos. 47
Ilustración 17: Modelo de fondo anterior. 48
Ilustración 18: Modelo de fondo actual. 48
Ilustración 19: Imagen actual y su histograma. 49
Ilustración 20: Modelo de fondo y su histograma. 49

Diagrama 1: Diagrama conceptual del Modelo de dominio 22
Diagrama 2: Diagrama de Casos de Uso del Sistema 25
Diagrama 3: Diagrama de clases de Análisis CU Tratar video. 32
Diagrama 4: Diagrama de clases de Análisis CU Capturar flujo de video y extraer fotograma. 32
Diagrama 5: Diagrama de clases de Análisis CU Procesar fotograma. 33
Diagrama 6: Diagrama de Clases de Análisis CU Detectar Manipulación. 33
Diagrama 7: Diagrama de Comunicación CU Tratamiento de video. 35
Diagrama 8: Diagrama de Comunicación CU Captura de flujo y extracción de fotograma. 35

Diagrama 9: Diagrama de Comunicación CU Procesamiento de fotograma. 35
Diagrama 10: Diagrama de Comunicación CU Detectar Manipulación sección Detectar desenfoque de la cámara. 36
Diagrama 11: Diagrama de Comunicación CU Detectar Manipulación sección Detectar movimiento de la cámara. 36
Diagrama 12: Diagrama de Comunicación CU Detectar Manipulación sección Detectar tapado de la cámara. 37
Diagrama 13: Diagrama de Clases del Diseño 39
Diagrama 14: Diagrama de secuencia CU Capturar de flujo de video y extraer fotograma 40
Diagrama 15: Diagrama de secuencia CU Procesar fotograma 40
Diagrama 16: Diagrama de secuencia CU Detectar manipulación sección Detectar desenfoque de cámara. 41
Diagrama 17: Diagrama de secuencia CU Detectar manipulación sección Detectar tapado de cámara. 41
Diagrama 18: Diagrama de secuencia CU Detectar manipulación sección Detectar el movimiento de la cámara. 42
Diagrama 19: Diagrama de componentes del Video Sensor para la detección de manipulación. 52
Diagrama 20: Diagrama de despliegue del Video Sensor para la detección de manipulación. 52

> Video Sensor para la detección de manipulación de cámaras de video vigilancia.

INTRODUCCIÓN

En las primeras etapas de los sistemas de video vigilancia se utilizaba la tecnología analógica que provocaba molestias a los usuarios al tener que monitorear por largos periodos de tiempo los medios a proteger, se utilizaban los casetes como forma de almacenamiento. Para la revisión de estos como evidencia tenían que observarse las grabaciones completamente, lo que hacía muy compleja la búsqueda de hechos específicos en la grabación.

Como se expresa en la revista “Video analytics and preemptive surveillance” el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha permitido la digitalización de los sistemas de vigilancia, esto posibilita que se comience a realizar un procesamiento de imágenes para extraer información relevante del video obtenido, lo que aporta mejores maneras de detección de incidentes. En la actualidad la información es obtenida en tiempo real desde cámaras que envían los flujos de video por la red, a estas se puede acceder mediante una dirección IP, por lo que se les denominan cámaras IP. (Video analytics and preemptive surveillance, 2010).

El análisis inteligente del video es parte fundamental de un sistema de video vigilancia, ya que este permite la existencia de los **video sensores**, que no es más que una herramienta de análisis de video digital que ofrece información significativa proveniente de una secuencia de video (Albiol Colomer, 2003).

La Universidad de las Ciencias Informáticas (UCI) se ha tomado la tarea de informatizar el país y llevarlo a un mayor desarrollo dándole solución a problemas existentes en todas sus esferas, no dejando detrás los sistemas de seguridad para la protección de los medios materiales y las vidas de los ciudadanos. En la UCI se ha creado el proyecto Video Vigilancia SURIA, llevado a cabo por el Centro Geoinformática y Señales Digitales (GEYSED), perteneciente a la Facultad 6 de dicha institución. Una de las tareas del proyecto Video Vigilancia es el procesamiento de imágenes y video digital.

El proyecto Video Vigilancia Suria cuenta con cinco módulos: Recuperador, Grabador, Visor, Análisis y el módulo central llamado Gestor. Este sistema brinda funcionalidades de gestión de la información, recuperación, grabación y visualización en tiempo real de los flujos de videos obtenidos desde cámaras IP. El módulo de análisis posee los videos sensores de Detección de movimiento, Objetos abandonados, Perimetrado virtual y Conteo de objetos.

Video Sensor para la detección de manipulación de cámaras de video vigilancia.

El comportamiento de las cámaras puede ser establecido por los responsables de la vigilancia remotamente, pero estas al estar accesibles en el entorno pueden ser manipuladas accidental o intencionalmente. El hecho de que las cámaras se puedan manipular implica que podría ser cambiado el ángulo de visibilidad, también el lente puede ser tapado por algún objeto o *spray* de color y al tener algunos dispositivos los lentes expuestos se pueden dar los casos de desenfoque, lo cual impediría ver la imagen con nitidez. La manipulación de la cámara es una situación que en estos momentos no detecta el sistema Suria.

A partir de la situación descrita anteriormente se identifica el problema a resolver: ¿Cómo detectar automáticamente la manipulación de la cámara a partir del flujo de video obtenido de las cámaras IP? Se plantea como objeto de estudio: El procesamiento inteligente de flujos de videos. Se enmarca en el campo de acción: El procesamiento inteligente de flujos de videos para detectar la manipulación de la cámara en el sistema de Video Vigilancia Suria.

Para solucionar el problema planteado se traza como objetivo general: Desarrollar un video sensor para la detección de manipulación de la cámara que procese los flujos de video obtenidos de cámaras IP. Planteando como idea a defender: La construcción y puesta en funcionamiento de un video sensor en el sistema Suria, proveerá al mismo de la capacidad de un procesamiento inteligente de los flujos de videos para determinar cuando una cámara ha sido tapada, movida o desenfocada, permitiendo un mejor control de los entornos que se monitorean.

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

1- Analizar los sistemas de video vigilancia existentes en la actualidad que detecten la manipulación de la cámara.

2- Identificar algoritmos y bibliotecas que permitan el procesamiento de los videos e imágenes digitales que puedan ser utilizados en el desarrollo del video sensor.

3- Realizar la documentación asociada al proceso de desarrollo.

4- Implementar un video sensor para la detección de tapado visual, desenfoque y movimiento de la cámara.

5- Crear una aplicación de prueba con el objetivo de validar el resultado esperado.

Como posible resultado de esta investigación se obtendría un componente video sensor que se le integraría al Sistema de Video Vigilancia Suria, aportándole al mismo una herramienta que procese la información para detectar la manipulación.

La realización de la investigación se apoya en métodos científicos que se fueron aplicando durante su desarrollo, tales como:

Métodos teóricos:

Histórico-Lógico, permite realizar un estudio acerca de los antecedentes y tendencias actuales de los videos sensores para la detección de manipulación de cámara, así como los conceptos y metodologías de desarrollo existentes para llevar a cabo su construcción.

Analítico-Sintético, facilita la búsqueda de los aspectos fundamentales en la bibliografía consultada para poder efectuar una investigación sobre los elementos que se relacionan con el procesamiento de imágenes digitales y bajo que conceptos se relacionan la manipulación en los distintos sistemas de vigilancia que se existen.

La tesis está compuesta por la presente introducción, cuatro capítulos, conclusiones, recomendaciones, bibliografía, anexos y glosario de términos que complementan la información comprendida en ella.

En el documento se aborda en el capítulo uno sobre la definición de los elementos teóricos que sustentan la investigación como lo son definiciones importantes para el entendimiento de la investigación y se analizan soluciones similares existentes en el mundo. También son expuestas las herramientas y tecnologías a utilizadas.

En el capítulo dos se analiza la solución para desarrollar un video sensor. En este capítulo se expone el modelo de dominio, se identifican los requisitos funcionales y no funcionales y los casos de uso que se

derivan de estos requisitos. Además se defiende la arquitectura escogida para el desarrollo del video sensor y los patrones de diseño utilizados.

El capítulo tres expone los resultados del proceso de diseño del sistema así como los diagramas que fueron necesarios para un mejor entendimiento de la solución propuesta.

En el capítulo cuatro se analizan los algoritmos que se implementaron en la solución además se exponen los diagramas de despliegue y componente como parte del modelo de implementación. También se muestran las pruebas de rendimiento realizadas al sistema y los resultados obtenidos de las mismas, comprobándose así con que eficiencia se detecta la manipulación.

Completa esta investigación un conjunto de conclusiones y recomendaciones, así como los anexos, el glosario de términos que aportan información valiosa al trabajo realizado y las fuentes bibliográficas consultadas se encuentran asentadas al final de la tesis.

CAPÍTULO 1: FUNDAMENTOS BASICOS PARA EL DESARROLLO DEL PROCESAMIENTO DE IMAGEN Y VIDEO DIGITAL

1.1 Introducción.

A continuación se presentan conceptos y aspectos teóricos asociados al dominio del problema planteado. Por otra parte contiene un análisis de las tendencias actuales de los sistemas de vigilancia y los videos sensores. Finalmente se exponen, la metodología y plataforma de desarrollo de software así como la herramienta de diseño y modelado a emplear.

1.2 Definiciones generales.

Imagen

Según el Instituto Nacional de Tecnologías Educativas y de formación de Profesorado la imagen se define como: ¨Una escultura, una pintura, un dibujo, una fotografía, la imagen televisiva o la imagen de un videojuego son ejemplos de diversos tipos de imágenes. ¿Qué es lo que tienen en común? Son una forma de comunicación, de transmitir información, de expresar sentimientos o ideas, de representar la realidad.¨ (Instituto Nacional de Tecnologías Educativas y de formación de Profesorado)

Imagen digital

Una imagen digital es una función bidimensional f(m,n), en la cual los valores m, n, y f(m,n) son discretos donde m, n son coordenadas espaciales y f(m,n) es la intensidad de color del punto. Como se muestra en la Ilustración 1 una imagen digital se representa mediante una matriz donde cada punto es un “píxel”. (C. Gonzalez, et al., 2002)

	Col1	Col2	Col3
Fil1	f(1,1)	f(1,2)	f(1,3)
Fil2	f(2,1)	f(2,2)	f(2,3)
Fil3	f(3,1)	f(3,2)	f(3,3)
Fil4	f(4,1)	f(4,2)	f(4,3)

Ilustración 1: Imagen Digital

Píxel

El píxel (del inglés *picture element*, o sea, elemento de la imagen) es la menor unidad en la que se descompone una imagen digital, ya sea una fotografía, un fotograma de video o un gráfico como se aprecia en la Ilustración 2. (GSMspain, 1996)

Al ampliar fuertemente una imagen digital (*zoom*), por ejemplo en la pantalla de un ordenador, pueden observarse los píxeles que componen la imagen. Los píxeles aparecen como pequeños cuadrados en color. Las imágenes se forman como una matriz rectangular de píxeles, donde cada píxel forma un punto diminuto en la imagen total.

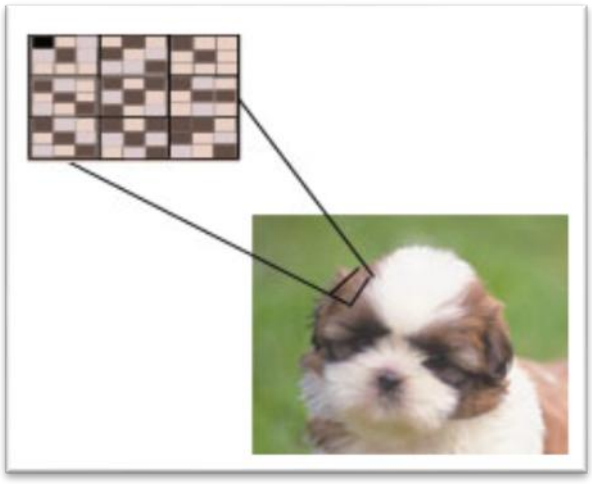


Ilustración 2: Píxel

Video

El vídeo no es más que la reproducción en forma secuencial de imágenes, que al verse con una determinada velocidad y continuidad dan la sensación al ojo humano de apreciar el movimiento natural, por ejemplo, lo mostrado en la Ilustración 3. Junto con la imagen, el otro componente es el sonido. (GSMspain, 1996)

El video, es la tecnología de la captación, grabación, procesamiento, almacenamiento, transmisión y reconstrucción por medios electrónicos o analógicos de una secuencia de imágenes que representan escenas en movimiento. Etimológicamente la palabra video proviene del verbo latino videre, y significa "yo veo". Se suele aplicar este término a la "señal de vídeo" y muchas veces se le denomina a la misma como "el vídeo" o "la vídeo" en reducción del nombre completo de la misma. (ABC, 2012)

Ilustración 3: Video

Video inteligente

El procesamiento inteligente de video 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 visión por computadora, con el objetivo de tomar decisiones automatizadas que contemplen ciertas capacidades. Ejemplo de los objetivos visuales de varios procesamientos inteligente de video son las Ilustraciones 4, 5 y 6. (La Inteligencia en aplicaciones de video, 2010)

•Detección inteligente de movimientos.
• Detección y clasificación de objetos, como la identificación automática de siluetas humanas.

•Aprendizaje de escenas y eventos rutinarios
•Detección de eventos inesperados
•Comportamiento humano y vehicular
•Reconocimiento de patrones. (La Inteligencia en aplicaciones de video, 2010)

Ilustración 4: Procesamiento inteligente de video para la detección de objetos perdidos

Ilustración 5: Procesamiento inteligente para la detección de robo

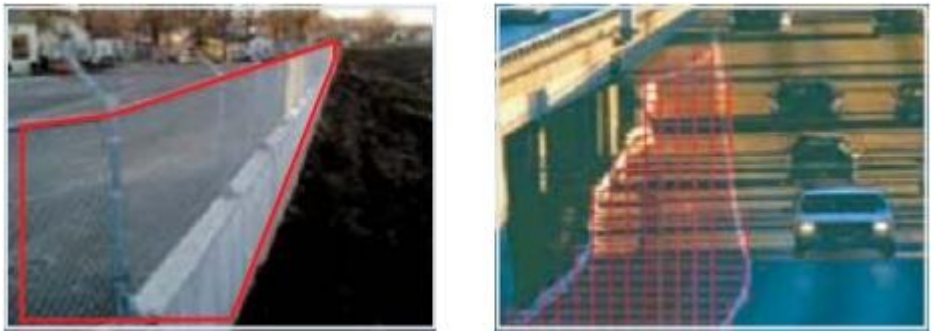


Ilustración 6: Procesamiento inteligente para la detección perimetral

1.3 Objeto de estudio.

El objeto de estudio de esta investigación está centrado en el procesamiento de imagen y video digital para optimizar la toma inteligente de decisiones del sistema de video vigilancia Suria.

1.3.1 Descripción general.

Los elementos usuales de un sistema de procesamiento de imágenes son los siguientes: Captación, Almacenamiento, Procesamiento, Comunicaciones y Visualización. (Molina, 1998)

El término “Procesamiento de imágenes digitales” (PID) abarca a los procesos que:

•Reciben como entrada una imagen y retornan otra imagen.
•Reciben como entrada una imagen y retornan rasgos de objetos presentes en la imagen.
•Reciben como entrada un conjunto de rasgos y le asocian una categoría predefinida. (C. Gonzalez, y otros, 2002)

En el procesamiento inteligente del flujo de video, como se muestra en la Ilustración 7, 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 minimizar 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 identificar situaciones y ayudar al sistema a tomar decisiones automáticas, agilizando el proceso de seguridad en los sistemas.

Pasos a seguir en el procesamiento de imágenes.

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

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

El paso siguiente es la segmentación, su objetivo es dividir la imagen en regiones de objetos que la forman y retorna un conjunto de puntos por cada región encontrada.

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

Se realiza la descripción después de haber representado las regiones de interés, con el objetivo de extraer los datos que las caracterizan.

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

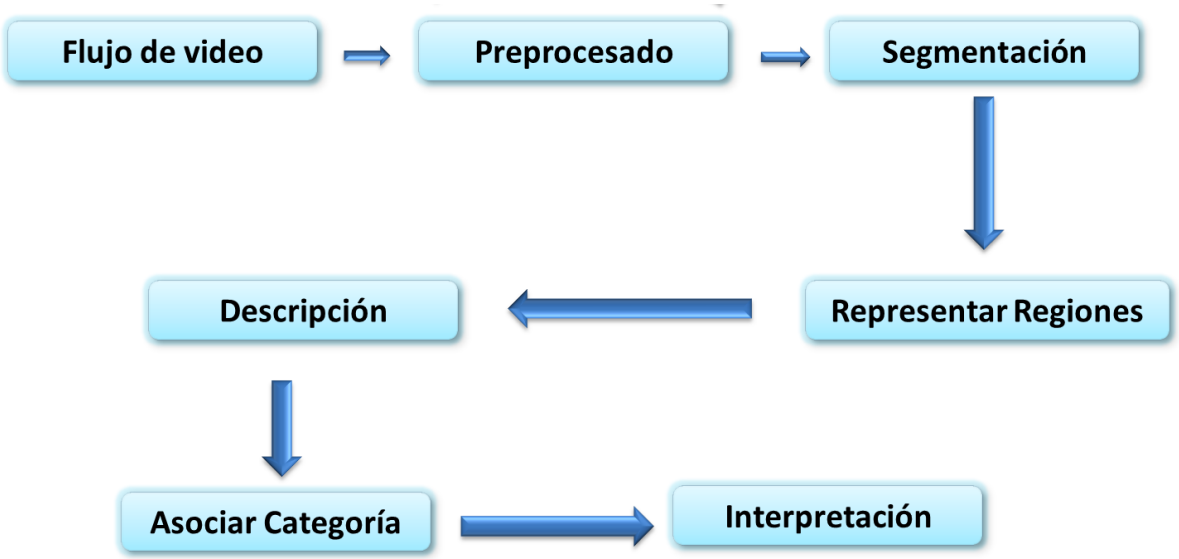


Ilustración 7: Análisis del procesamiento.

1.4 Análisis de soluciones existentes.

Luego de un estudio bibliográfico se han caracterizado varios sistemas existentes en el mercado mundial que tienen implementada la funcionalidad de detección de manipulación de la cámara. En este estudio los sistemas investigados permiten definir los principales elementos que se incluyen dentro de la manipulación. A continuación se describen las principales características de varios de estos sistemas.

Intellivision Inc.

Esta compañía ha desarrollado una suite completa de análisis de video inteligente que incluye detección de movimiento, detección de intrusiones y detección de manipulación de la cámara (Ilustración 8). La detección de movimiento puede diferenciar si la alarma ha sido activada por una persona o un animal, mandando la alerta solo cuando sea necesario. La detección de intrusos manda alertas cuando alguien abre una puerta o entra a un espacio restringido. Con la detección de manipulación, se avisará si la cámara ha sido movida, bloqueada o apagada. (EE Times University, 2012)

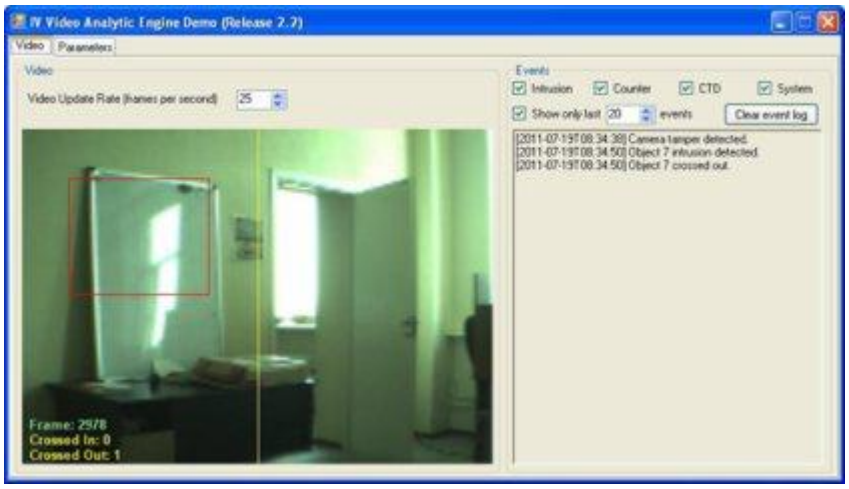


Ilustración 8: Captura de pantalla del sistema de análisis de video de Intellivision

Vivotek IP7160

El análisis inteligente del video puede estar embebido en la cámara, un ejemplo de esto es la cámara Vivotek IP7160 (Ilustración 9) que cuenta con una resolución de 2 megapíxel y la inclusión de funciones avanzadas como la detección de movimiento y manipulación.

Con la introducción de la detección de manipulación, la cámara tiene la habilidad de detectar la pérdida de información en tiempo real. Es capaz de reconocer actos como, pintado o bloqueo por espray y redireccionamiento del lente. Se puede configurar para mandar alertas automáticas cada vez que ocurra algún evento. (Video analytics and preemptive surveillance, 2010)

Ilustración 9: Cámara Vivotek IP7160

Cisco® Video Analytics

Este software ofrece maneras innovadoras de realizar el análisis de video. Provee una interfaz intuitiva, y poderosas herramientas para permitir a las organizaciones hacer el mejor uso de los sistemas de video vigilancia (Ilustración 10). El procesamiento ocurre en dispositivos periféricos, proveyendo un método efectivo en cuanto a costo para desplegar el análisis de video en la red.

Este software se divide en dos paquetes de análisis, *Security* y *Counting*. Básicamente el paquete *Security* provee funcionalidades de seguimiento, clasificación de objetos, detección de manipulación, merodeo y robo. El paquete básico de *Counting* provee seguimiento multihilo y clasificación de objetos. (Cisco, 2011)

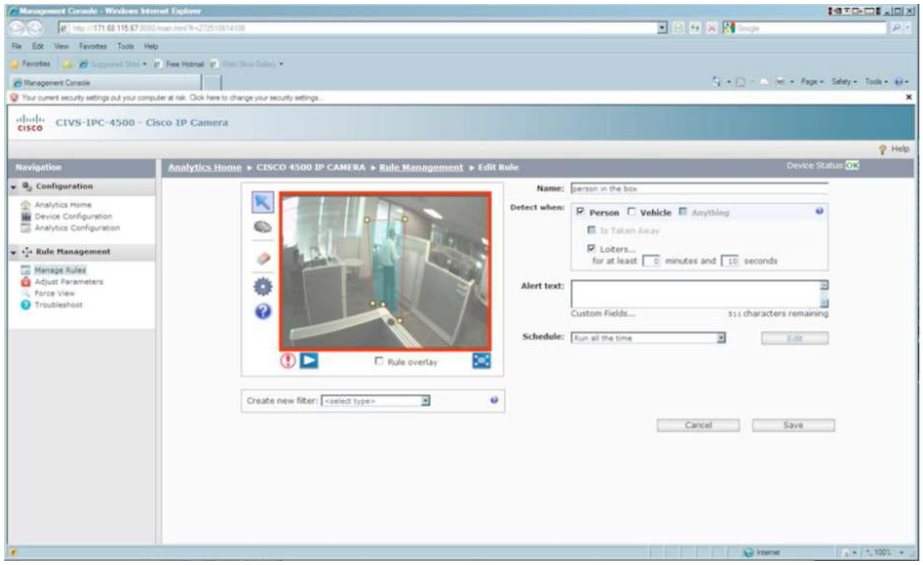


Ilustración 10: Interfaz de Cisco® Video Analytics

Los sistemas analizados son privativos y no se pueden usar sus algoritmos en esta solución, pero permitieron identificar el concepto de manipulación de la cámara. Estos softwares no exponen como implementan sus algoritmos de detectar manipulación pero algunos abordan las vías que utilizaron, como la detección de borde, chequeo de histogramas, y modelado de fondo permitiendo guiar la profundización del estudio hacia este tipo de técnicas. Cuba es un país en desarrollo que está abogando por la soberanía tecnológica y el uso de software libre, forma profesionales altamente calificados que tienen la capacidad y los recursos para desarrollar software, utilizando generalmente herramientas libres que permitan realizar las funcionalidades requeridas para detectar la manipulación de las cámaras de video vigilancia.

1.5 Herramientas y tecnologías a utilizar.

1.5.1 Metodología de desarrollo.

Para la producción de un software es de vital importancia la organización del trabajo. Debe existir un proceso que estructure e integre las múltiples etapas del desarrollo de la aplicación. Es necesario contar con una guía para organizar sus actividades. Es preciso definir los artefactos que deben ser producidos para lograr los objetivos propuestos. Las metodologías de desarrollo de software proveen las guías para poder conocer el camino a recorrer desde las primeras fases de la construcción del software, asegurando la calidad del producto final, así como también el cumplimiento en la entrega del mismo en un tiempo

estipulado. Actualmente dos de las vertientes en el mundo de las metodologías de desarrollo de software más utilizadas son, las metodologías pesadas o tradicionales y las metodologías ágiles.

Proceso Unificado de Desarrollo de Software.

El Proceso Unificado de Desarrollo de Software (RUP por sus siglas en inglés) proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de alta calidad que resuelva las necesidades de los usuarios dentro de presupuestos y tiempos establecidos.

RUP es un marco de procesos altamente configurables para satisfacer necesidades específicas. Implementa las mejores prácticas del desarrollo de software. Es orientada a objetos y utiliza el Lenguaje Unificado de Modelado como lenguaje de representación visual. A su vez define Quién, Cómo, Cuándo y Qué debe de hacerse en el proyecto. Presentando como principales características:

• Dirigido por casos de uso: utiliza los casos de uso para el desarrollo de las disciplinas con los artefactos, roles y actividades necesarias. Además estos son la base para la implementación de las fases y disciplinas de RUP.
• Iterativo e Incremental: plantea la implementación del proyecto a realizar en iteraciones, por lo que se pueden definir objetivos por cumplir en cada iteración y así completar el proyecto iteración por iteración. Ejecuta una evaluación satisfactoria permitiendo que el proyecto se mueva a la próxima fase y realiza una revisión del ciclo de vida al finalizar la misma.
• Centrado en la Arquitectura: define la estructura de las partes más relevantes de un sistema y una arquitectura ejecutable construida como un prototipo evolutivo.
• RUP es uno de los procesos más generales de los existentes en la actualidad, debido a que está pensado para adaptarse a cualquier proyecto. (Chacón, 2006)

En la Ilustración 11 se muestra el ciclo de vida de esta metodología. RUP posee 4 fases en su ciclo de vida: Inicio, Elaboración, Construcción y Transición. Estas fases de trabajo poseen actividades las cuales son agrupadas en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

La metodología de desarrollo antes expuesta ha sido seleccionada para guiar todo el proceso del desarrollo de esta investigación. La metodología RUP es de gran importancia como apoyo en el desarrollo del software, debido a varias prestaciones que brinda como son: provee alta calidad del resultado de los sistemas, aumenta la productividad. Presenta herramientas de apoyo en cada fase del proyecto y tiene un alto potencial de organización.

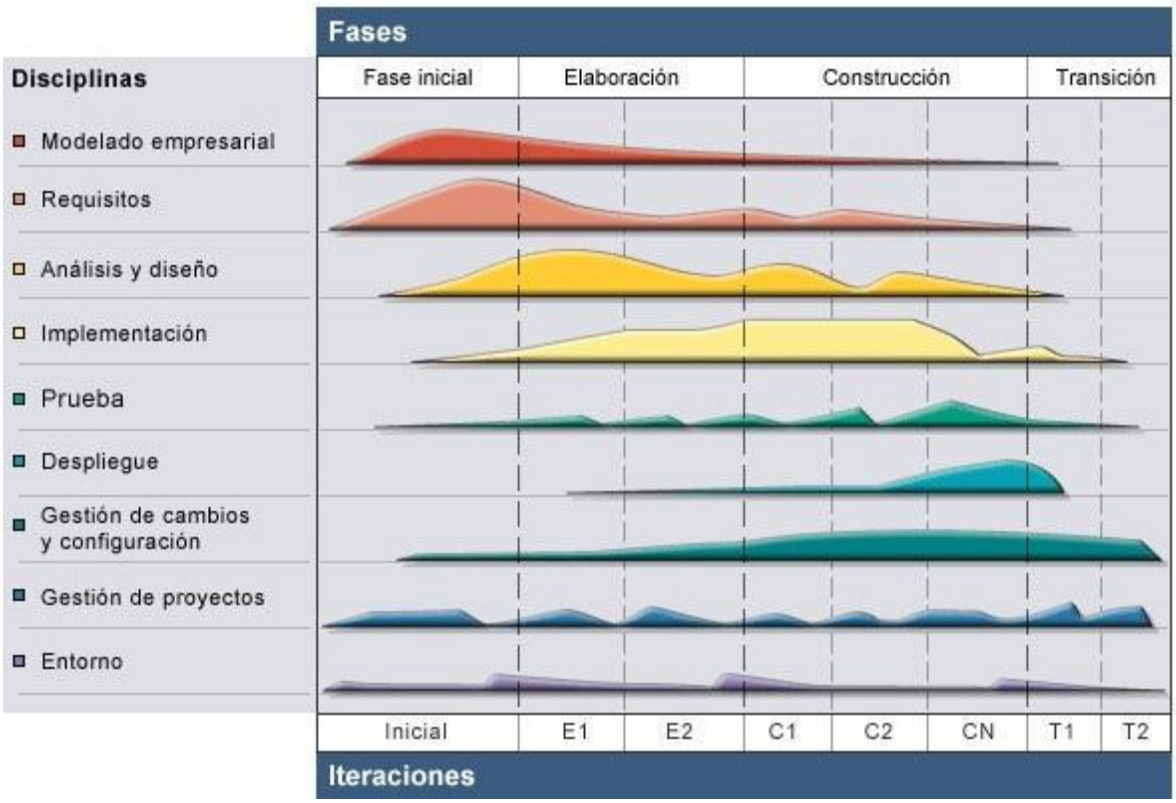


Ilustración 11: Flujos de trabajo RUP (Chacón, 2006)

1.5.2 Lenguaje de modelado.

UML (Unified Modeling Language), es un lenguaje de modelado respaldado por Object Management Group (OMG). Es utilizado para representar gráficamente, construir y documentar un sistema de software, además para detallar los artefactos y tener un modelo entendible por los trabajadores. Facilita la planeación y el control del proyecto, brinda también una mayor reutilización en proyectos similares y minimiza los costos. (Giraldo, y otros, 2005) (Orallo, 2010)

1.5.3 Herramienta para el modelado

Visual Paradigm.

Visual Paradigm, es una herramienta profesional que posibilita un ahorro considerable de tiempo y una calidad óptima en el proceso. Posibilita la captura de requisitos, análisis, diseño e implementación para una aplicación en desarrollo.

Entre las ventajas que ofrece están:

- Soporte para UML versión 2.1: para la metodología RUP esta característica es muy provechosa ya que para el modelado de los diagramas se utiliza UML.
- Interoperabilidad entre diagramas: permite a partir de un diagrama obtener otro que guarde relación con el mismo.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Disponibilidad en múltiples plataformas: *Microsoft Windows* (98, 2000, XP, Vista o 7), *Linux*, *Mac OS X, Solaris o Java.*
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas. (Sierra, 2006)

Por lo antes expuesto se decide que *Visual* Paradigm es una alternativa de herramienta CASE para un desarrollo exitoso del video sensor ya que agilizará el proceso de desarrollo y generará los artefactos necesarios con la estructura y relaciones deseada.

1.5.4 Lenguaje de Programación

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. (Stroustrup, 1997)

Para llevar a cabo la creación del Video Sensor para la detección de manipulación de cámara se ha seleccionado a C++ como lenguaje de programación por su rapidez, portabilidad y documentación en el procesamiento de imágenes.

1.5.5 Biblioteca a utilizar.

Las bibliotecas 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.

OpenCV

OpenCV (Open Source Computer Vision Library) es la biblioteca seleccionada para la creación del Video Sensor para la detección de manipulación de cámara, es de código abierto ha sido optimizada y proyectada para la construcción de aplicaciones en tiempo real. Es independiente del hardware y del sistema operativo. Lee, escribe y adquiere genéricamente las imágenes y videos. Provee interfaces de optimización para procesadores Intel y además permite el procesamiento básico de imágenes así como el análisis de movimiento y reconocimiento de objetos.

La biblioteca 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 (Motion History Images 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). (Adrian Kaehler, 2008)

FFmpeg

FFmpeg es una colección de software libre que puede grabar, convertir (transcodificar) y hacer streaming de audio y vídeo. Incluye libavcodec, una biblioteca de códecs. FFmpeg está desarrollado en GNU/Linux, pero puede ser compilado en la mayoría de los sistemas operativos, incluyendo Windows. El proyecto

comenzó por Gerard Lantau, un seudónimo de Fabrice Bellard, y ahora es mantenido por Michael Niedermayer. Es destacable que la mayoría de los desarrolladores de FFmpeg lo sean también del proyecto MPlayer (más un miembro del proyecto Xine), y que FFmpeg esté hospedado en el servidor del proyecto MPlayer.

FFmpeg está liberado bajo una licencia GNU Lesser General Public License 2.1+ o GNU General Public License 2+ (dependiendo de cuáles bibliotecas estén incluidas). Los desarrolladores recomiendan utilizar el último *snapshot* de Subversion ya que mantienen constantemente una versión estable.

FFmpeg es un programa sencillo de usar, orientado tanto a personas con conocimientos avanzados como usuarios novatos. Es capaz de elegir el códec con sólo escribir la extensión. Por ejemplo, mpeg4 si se usa .avi, VP8 si se usa .webm.

FFmpeg es usado en proyectos libres y propietarios, incluyendo ffmpeg2theora, VLC, MPlayer, HandBrake, Blender, Google Chrome y otros. También hay varios *frameworks* multimedia que hacen uso de FFmpeg como DirectShow/VFW (ffdshow), QuickTime (perian), GStreamer, OpenMAX, xine. (Duck, 2010)

La biblioteca Ffmpeg es escogida ya que brinda funcionalidades necesarias para la captura de los flujos de video y decodificación en tiempo real, agilizando el proceso de captura y extracción de fotogramas en tiempo real.

1.5.6 Entorno de desarrollo

Qt Creator

Qt Creator. Es un Entorno Integrado de Desarrollo (IDE), multiplataforma, diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil.

Qt Creator no es un reemplazo de Eclipse ni Visual Studio, sino un IDE ligero pensado especialmente para el desarrollo en múltiples plataformas: Windows XP, Vista y Windows 7, Linux (desde la versión 2.6) y Mac OSX (desde 10.4 en adelante).

Principales características de QtCreator:

•Posee un avanzado editor de código C++.
•Posee también una GUI integrada y diseñador de formularios.
•Herramienta para proyectos y administración.
•Ayuda sensible al contexto integrado.
•Depurador visual.
•Resaltado y auto-completado de código.
•Soporte para refactorización de código. (Pixelco, 2012)

QtCreator es distribuido bajo tres tipos de licencias: QtCommercial Developer License, Qt GNU LGPL v. 2.1, Qt GNU GPL v. 3.0 y está disponible para las plataformas: Linux, Mac OSX; Windows, Windows CE, Symbian y Maemo. (Pixelco, 2012)

El sistema SURIA es desarrollado utilizando el *framework* QT y por lo que es escogido el mismo *framework* para una compatibilidad óptima entre los sistemas y así incluir esta funcionalidad en el sistema SURIA.

1.6 Conclusiones

El estudio del procesamiento inteligente de video e imágenes digitales ha permitido encontrar técnicas que posibilitan cumplir con el objetivo general planteado. El estudio de las soluciones existentes muestra que éstas son privativas y en la información que brindan los sitios oficiales de las empresas no muestran para el uso público sus soluciones. Este componente al decidirse que sea desarrollado bajo la metodología RUP y construido con el lenguaje C++ se considera que puede derivar en una aplicación robusta, con una alta calidad y distribuible que no necesita pagar licencia lo cual es una de sus mayores ventajas.

CAPÍTULO 2: ANÁLISIS DE LAS CARACTERISTICAS DEL SISTEMA

2.1 Introducción.

El presente capitulo presenta el análisis de la solución para desarrollar un video sensor para la detección de manipulación de las cámaras IP. Para describir la solución se realiza un modelo de dominio y la especificación de los requisitos funcionales y no funcionales que debe cumplir el video sensor. Además se determinan los casos de uso y los actores que interactúan en el sistema, realizándose una descripción de cada uno de los mismos. Se selecciona además la arquitectura más adecuada para la llevar a cabo el desarrollo exitoso y con calidad, del sistema a implementar.

2.2 Sistema propuesto.

Se propone el desarrollo de un video sensor que procese los flujos de video proveniente de varias cámaras IP y sea capaz de detectar la manipulación de dichas cámaras. Este componente debe ser sencillo de integrar a sistemas de vigilancia en desarrollo como Suria para así dotarlos de una herramienta que les permita hacer este tipo de procesamientos. La solución se centra en el procesamiento de imágenes digitales, para mediante algoritmos de comparación de imágenes detectar si las cámaras han sido tapadas, movidas o desenfocadas. El procesamiento empieza desde que es capturado el flujo de video y extraído de este los fotogramas. Posteriormente se realizarían los algoritmos para detectar la manipulación.

2.3 Modelo de dominio.

Un Modelo de dominio es una representación de las clases conceptuales del mundo real que contiene el producto a desarrollar. En este se exponen los principales conceptos asociados al desarrollo del componente. El Modelo de dominio se realiza ya que al no existir clientes y socios no se expone el modelo de negocio en todo su contexto por lo que se realiza el Modelo de dominio.

2.3.1 Conceptos fundamentales.

Con el objetivo de un mejor entendimiento del Modelo de dominio se proporcionan definiciones de las distintas clases identificadas.

Video sensor: Herramienta que permite la detección automática de la manipulación de la cámara.

Cámara IP: Dispositivo de captura de imágenes y videos.

Video: Secuencia de imágenes transmitidas por las cámaras y procesadas por el sistema.

Manipulación: Manipulación de la cámara vista como: tapada, movida o desenfocada.

Tapado: Obstrucción de la visibilidad de la cámara.

Movimiento: Se ha cambiado el campo de observación de la cámara.

Desenfoque: Perdida de nitidez en la imagen que no permite observar con claridad los objetos en la escena.

2.3.2 Diagrama del Modelo de dominio.

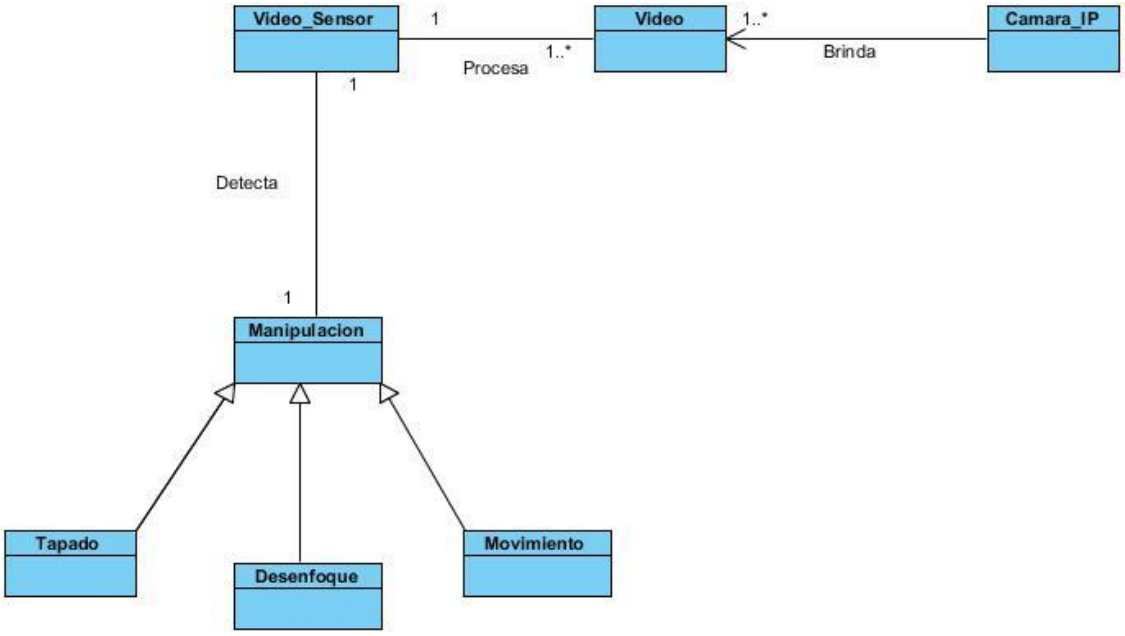


Diagrama 1: Diagrama conceptual del Modelo de dominio

Descripción del modelo de dominio:

El video sensor para la detección de manipulación procesa los flujos de video que brinda una o varias cámaras IP, para detectar automáticamente si una o varias cámaras han sido tapadas, desenfocadas o movidas. Diagrama 1.

2.4 Especificación de los requisitos del Software.

2.4.1 Requisitos Funcionales

Los requisitos funcionales son condiciones o capacidades que debe cumplir el software, estas necesidades las plantean los clientes sobre lo que debe tener y hacer el sistema.

RF#1: Capturar el flujo de video proveniente de la cámara IP y extraer fotogramas del flujo de video.

Este requisito funcional especifica que el sistema debe permitir en tiempo real capturar los flujos de video que brindan las cámaras IP. Además debe permitir extraer los fotogramas de flujos de videos para realizarle el procesamiento necesario.

RF#2: Procesar fotograma.

Este requisito especifica que debe ser identificado en la secuencia de fotogramas los objetos que están en movimiento, permitiendo localizar las regiones que son estáticas, estas serían el fondo de la imagen. El sistema debe eliminar los objetos que no sean del fondo cuyo tamaño sea irrelevante en la escena.

RF#3: Detectar el movimiento de la cámara.

Se debe detectar si el ángulo de observación de la cámara ha sido cambiado y la escena vigilada en ese momento no sea la que esta previamente determinada.

RF#4: Detectar el desenfoque de la cámara.

Se debe detectar que la poca nitidez de las imágenes no permita identificar los objetos que se encuentran en la escena a monitorear.

RF#5: Detectar el tapado de la cámara.

Se debe detectar cuándo la cámara transmita una secuencia que no tenga contenido en la imagen; que no se distingan formas u objetos en la escena o que la visibilidad sea obstruida total o parcialmente.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que debe tener el software para ser un producto atractivo, usable, rápido o confiable. Los requisitos se separan por categoría según lo que el software tiene que satisfacer.

RNF1. Rendimiento

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

RNF2. Usabilidad

RNF2.1: El algoritmo debe ser configurable y parametrizable.

El sistema debe permitir cuál de los tipos de detección se desea realizar y si se debe introducir algún parámetro en específico se verifica la fiabilidad de los datos para evitar que se produzcan errores.

RNF3. Fiabilidad

RNF3.1: El video sensor debe funcionar a tiempo completo.

El sistema debe permitir correr el algoritmo en todo momento, para así no correr riesgos de que ocurra algún hecho de este tipo en el tiempo en que no esté disponible el video sensor.

RNF4. Portabilidad

RNF4.1: El video sensor debe tener la posibilidad de utilizarse sobre Windows y Linux.

El video sensor debe estar implementado de tal forma que pueda ser utilizado en las plataformas de Windows y Linux indistintamente para así lograr un mayor despliegue.

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

La arquitectura debe ser de 32 bits porque las bibliotecas utilizadas para el procesamiento de imagen son compiladas en un sistema operativo con esta arquitectura.

RNF5 Hardware

RNF5.1: Se debe tener como mínimo 1Gb de RAM y procesador Core2 Duo a 2.2 GHz.

RNF6 Restricciones del diseño

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

RNF6.2: La biblioteca que se utilizará para el procesamiento de imágenes será OpenCV.

2.5 Definición de Casos de Uso.

El diagrama de Casos de Uso del Sistema muestra cómo es que van a interactuar el actor del sistema y los elementos que componen al mismo y en el orden en que lo hacen.

2.5.1 Definición de actores

Actores	**Justificación**
Operador del Sistema de Video Vigilancia	El sistema es el que interactúa con el video sensor, es el que puede escoger la opción de detectar la manipulación.

2.5.2 Listado de los casos de uso.

CU#1: Tratar video.

CU#2: Capturar flujo de video y extraer fotogramas.

CU#3: Procesar fotograma.

CU#4: Detectar manipulación.

2.6 Diagrama de Casos de Uso.

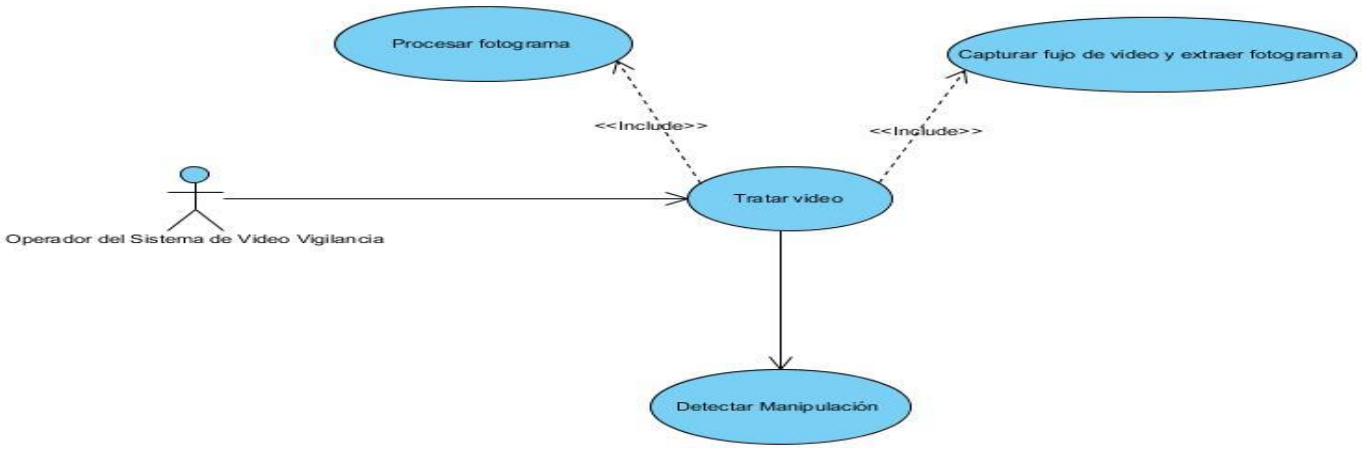


Diagrama 2: Diagrama de Casos de Uso del Sistema

2.6.1 Descripción de los Casos de Uso del Sistema

Descripción del caso de uso Tratamiento del video.

CU#1	Tratar video
Actores:	Operador del Sistema Video Vigilancia
Resumen:	El caso de uso inicia cuando el operador del Sistema Video Vigilante escoge la opción de detectar la manipulación de la cámara, el sistema capturará los flujos de videos obtenidos desde la cámara IP. Se le hace un procesado de imagen. Por último se ejecutan los algoritmos de detección y analiza los datos para identificar si ha sido manipulada la cámara.
Referencias	RF#1, RF#2, RF#3, RF#4, RF#5
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
Escenario Detectar manipulación	
1- El operador le hace la petición al módulo Analytics de detectar la manipulación.	2- Si existen cámaras en la lista del sistema se escoge la que se desea. 3- Si el video proveniente de la cámara no contiene error, se le extraen los fotogramas al flujo de video. Ver descripción del caso de uso Captura de flujo y extracción de fotogramas. 4- Se modela el fondo. Se eliminan los ruidos. Ver descripción del caso de uso Procesamiento del fotograma.
Flujo Alternativo de Eventos Paso 2	
Acción del Actor	**Respuesta del Sistema**
	2- Si no existen cámaras disponibles en el sistema se muestra un mensaje informándoselo al usuario.

Flujo Alternativo de Eventos Paso 3	
Acción del Actor	**Respuesta del Sistema**
	3- Si el flujo proveniente de la cámara contiene errores se muestra un mensaje informándoselo al usuario.

El resto de las descripciones de los Casos de Uso se encuentran en el **Anexo 1**

2.7 Arquitectura a utilizar y patrones de diseño seleccionados.

2.7.1 Arquitectura.

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

Los videos sensores son los que realizan todo el procesamiento inteligente de video, poseen un estilo arquitectónico llamado *Pipes & Filters* (Tuberías y Filtros), donde cada componente tiene un conjunto de entradas y un conjunto de salidas. Un componente lee un flujo de datos en la entrada y produce un flujo de datos diferente en su salida. Esto es logrado aplicando una transformación local al flujo de entrada mientras este se lee, de tal forma que el flujo de salida empieza antes que se consume todo el flujo de entrada. Este patrón divide la tarea de un sistema en varios pasos de procesamiento secuenciales. Estos pasos están conectados por el flujo de datos a través del sistema, cada paso del procesamiento está encapsulado en un componente de filtro. Los datos pasan a través de las tuberías que son los conectores que sirven como conductos para transmitir las salidas de un filtro a las entradas de otro. Ilustración 12.

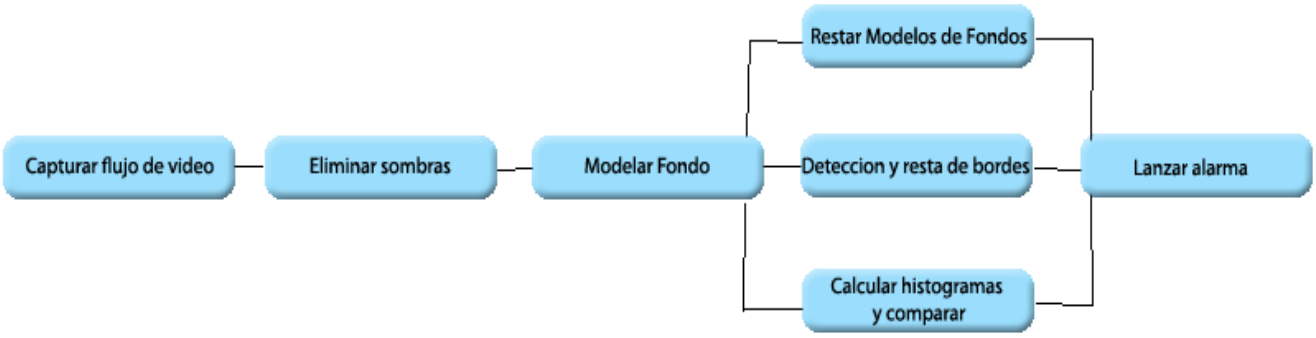


Ilustración 12: Representación gráfica de la Arquitectura de diseño del Video Sensor.

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

2.7.2 Patrones de Diseño.

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, “Consiste en asignar la responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir dicha responsabilidad” (González, 2010). Se vería reflejado en las clases DefocusedDetection, CoveredDetection, MoveDetection, que son expertos en la responsabilidad de detectar el desenfoque, tapado y movimiento respectivamente ya que son las que poseen los elementos necesarios para lograr sus objetivos y se apoyan en otros expertos como: ShadowRemoval y BGFG_Gauss_Segm para cumplir sus responsabilidades.

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.” (González, 2010) Este patrón es utilizado para mantener la clase VideoProssesor con pocas responsabilidades ya que se

cohesiona con las clases de detección logrando realizar solo las responsabilidades suficientes y necesarias para el funcionamiento de la aplicación.

El patrón Bajo Acoplamiento “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) Bajo acoplamiento se ve en las relaciones que hay entre las clases BGFG_Gauss_Segm y ShadowRemoval con las clases que hacen cada tipo de detección, tienen bajo acoplamiento ya que estas son utilizadas por las de detección pero ellas no utilizan nada de las de detección, permitiendo que si hay algún cambio en una de estas no afecten a las de detección por estar con acoplamiento bajo.

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) Este patrón se evidencia en la clase VideoProssesor, que es la encargada de invocar los métodos para el funcionamiento del video sensor. Las otras clases también manejan los eventos relacionados con sus responsabilidades.

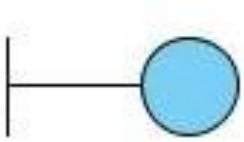
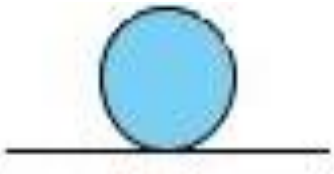
2.8 Conclusiones

Al reconocer los requisitos funcionales y no funcionales que componen la solución son obtenidos los casos de uso del sistema logrando un mayor entendimiento del procesamiento interno del sistema y satisfaciendo las necesidades de los clientes. Con la realización del Modelo de Dominio se tienen todos los conocimientos necesarios para un mejor entendimiento, de lo que compone al Video Sensor. La selección de una arquitectura adecuada permite tener una correcta estructura para los elementos de software y al escogerse los patrones de diseños a utilizar se facilita la implementación de la solución, brindando estos una mayor organización en los códigos y en el tránsito de información entre los distintos filtro.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Modelo de Análisis.

El análisis es una abstracción, es un bosquejo del diseño de la aplicación el cual define una estructura para modelar el sistema y permite dar forma a dicho sistema en el diseño y la implementación.

 Clase Interfaz	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.
 Clase Controladora	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
 Clase Entidad	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.

3.1.1 Diagrama de clases de Análisis.

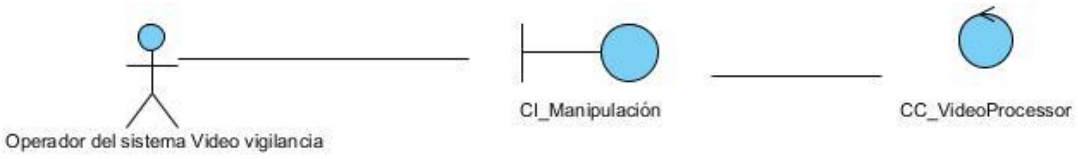


Diagrama 3: Diagrama de clases de Análisis CU Tratar video.

En el Diagrama 3 se muestra el Diagrama de clases de Análisis del caso de uso: Tratamiento de video. En el cual interviene la clase CC_VideoProcessor que se encarga de abrir la conexión del video para su posterior procesamiento.

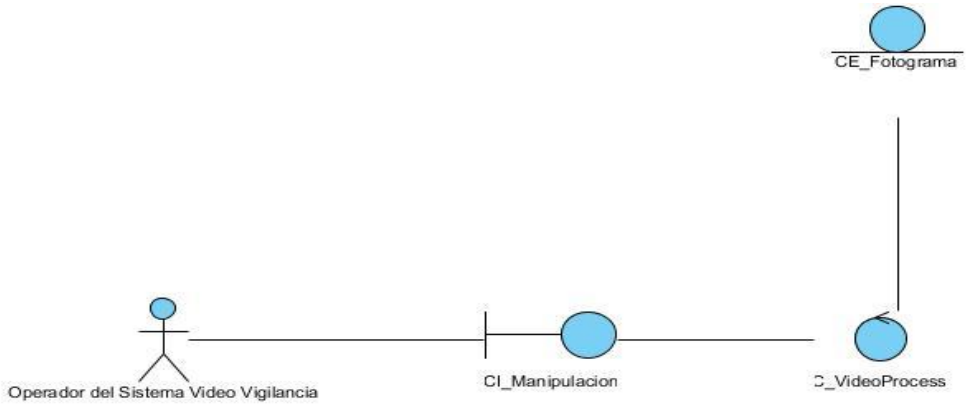


Diagrama 4: Diagrama de clases de Análisis CU Capturar flujo de video y extraer fotograma.

En el Diagrama 4 se muestra el Diagrama de clases de Análisis del caso de uso: Captura de flujo y extracción de fotograma. En el que interviene la clase CC_VideoProcessor que se encarga de manipular el objeto creado en esta, de la clase, CE_Fotograma, con el objetivo de obtener los fotogramas, desde el flujo de video capturado en la clase CC_VideoProcessor.

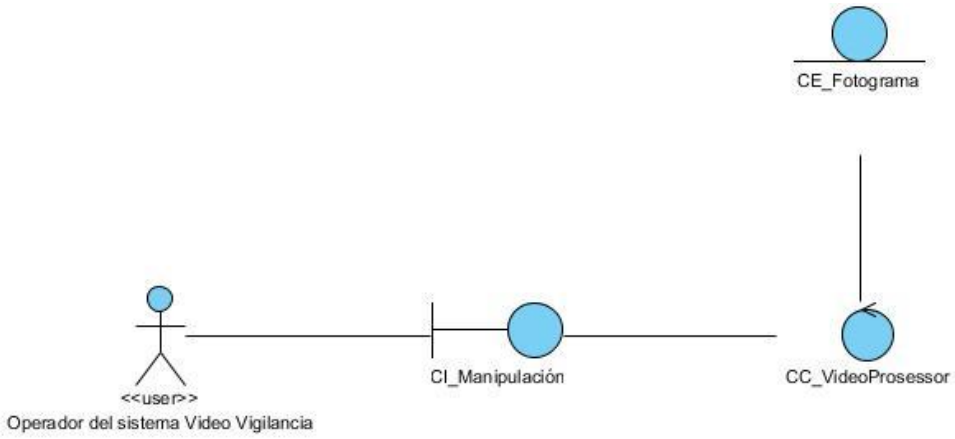


Diagrama 5: Diagrama de clases de Análisis CU Procesar fotograma.

En el Diagrama 5 se muestra el Diagrama de clases de Análisis del caso de uso: Procesamiento de fotograma. En el cual interviene la clase CC_VideoProcessor que se encarga de manipular los objetos creados en esta, de las clases, CE_Fondo y CE_Fotograma, con el objetivo de obtener los fotogramas para modelarle el fondo y eliminar las sombras.

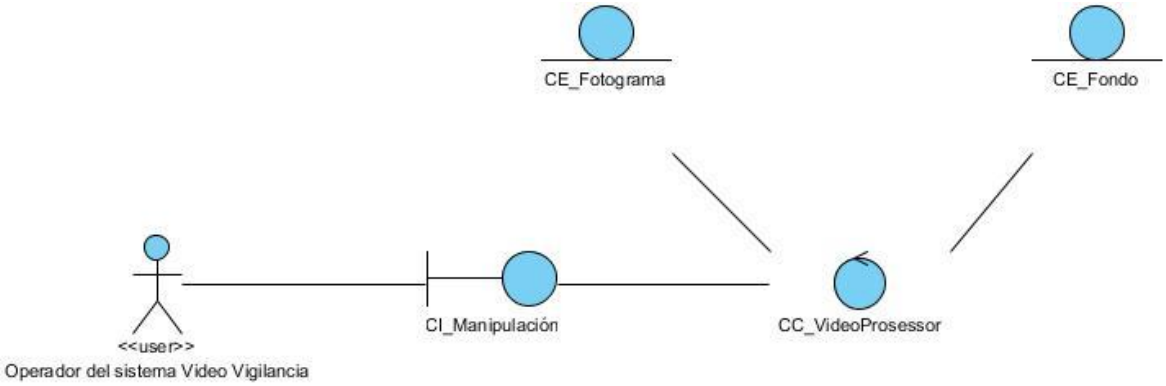


Diagrama 6: Diagrama de Clases de Análisis CU Detectar Manipulación.

En el Diagrama 6 se muestra el Diagrama de clases de Análisis del caso de uso: Detectar Manipulación. En el cual interviene la clase CC_VideoProcessor que se encarga de manipular los objetos creados en esta, de las clases, CE_Fondo, CE_Fotograma y CE_Detección, con el objetivo de detectar la manipulación.

3.2 Diagramas de Interacción.

Estos diagramas le muestran a los desarrolladores la manera en que las clases interactúan unas con otras y los mensajes que se envían. Según UML existen dos tipos de diagramas de interacción, estos son:

Diagramas de colaboración o comunicación:

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

Diagramas de secuencia:

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

3.2.1 Diagramas de Comunicación de los Caso de Uso.

En los diagramas 7, 8, 9, 10, 11 y 12 se muestran los diagramas de comunicación de los Casos de Uso: Captura de flujo y extracción de fotograma, Procesamiento de fotograma y Detección de manipulación en sus 3 secciones: Detectar desenfoque de la cámara, Detectar movimiento de la cámara y Detectar tapado de la cámara respectivamente. En estos diagramas se observan los principales mensajes intercambiados entre las clases para realizar los casos de usos en cuestión.

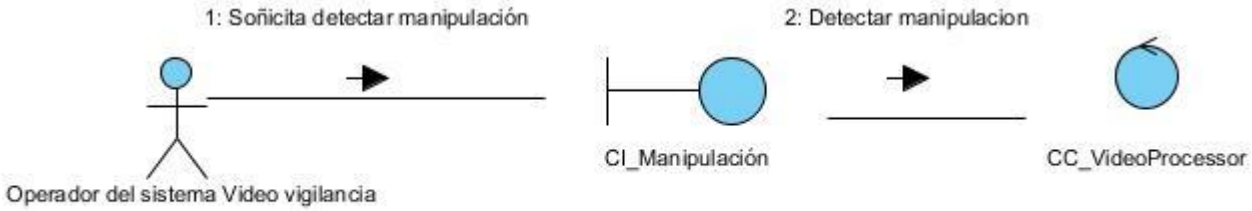


Diagrama 7: Diagrama de Comunicación CU Tratamiento de video.

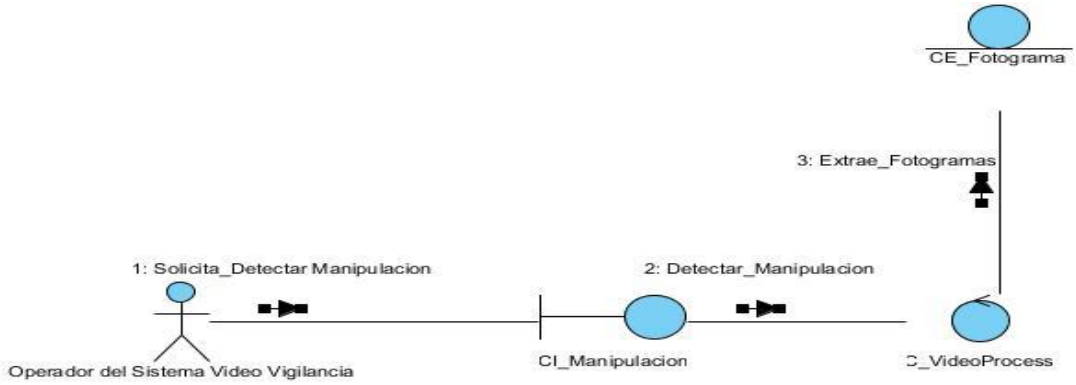


Diagrama 8: Diagrama de Comunicación CU Captura de flujo y extracción de fotograma.

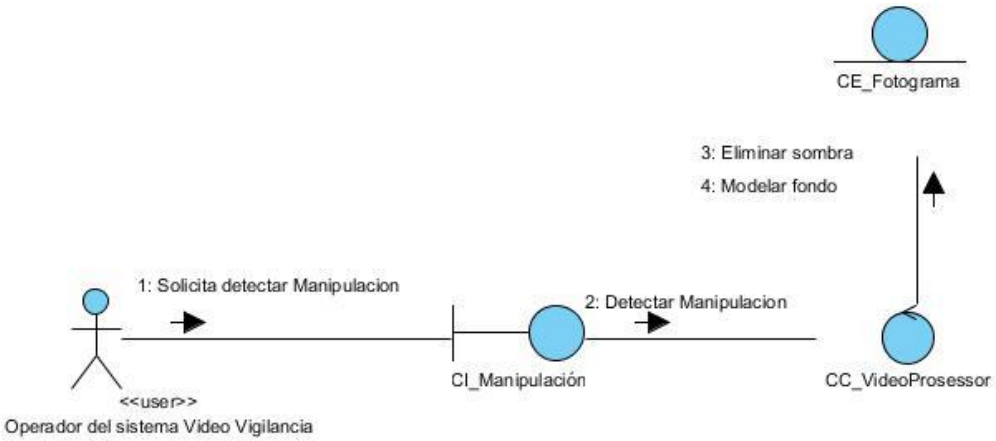


Diagrama 9: Diagrama de Comunicación CU Procesamiento de fotograma.

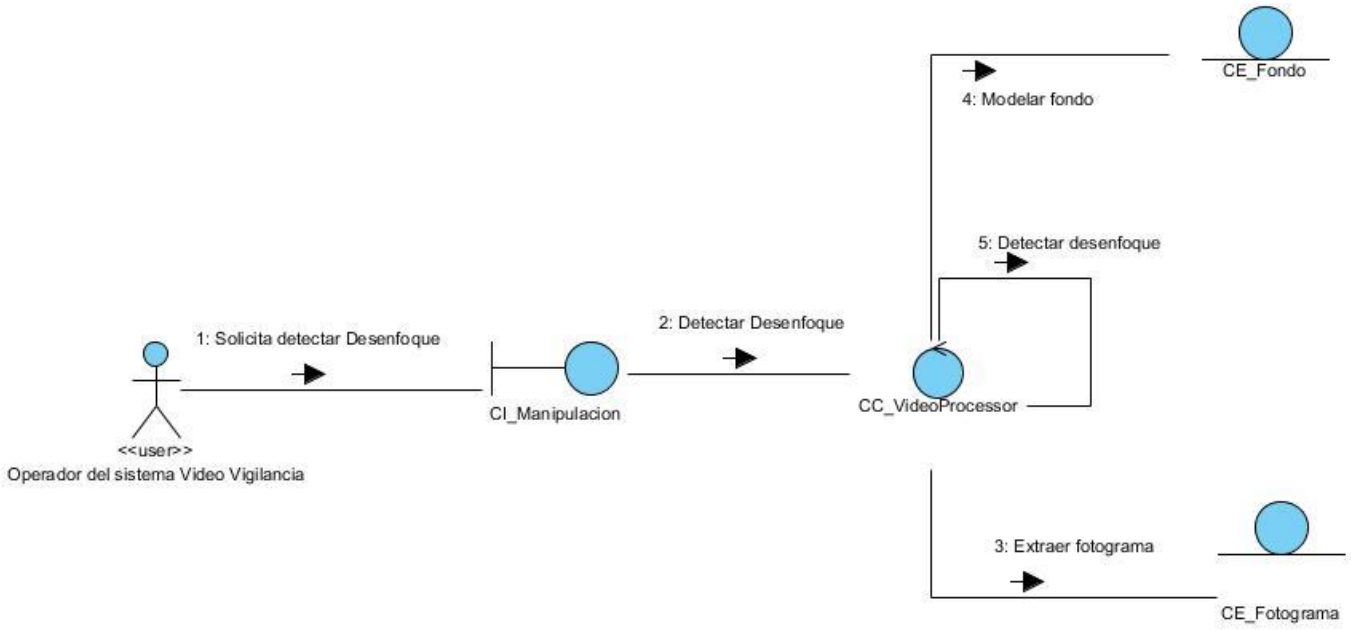


Diagrama 10: Diagrama de Comunicación CU Detectar Manipulación sección Detectar desenfoque de la cámara.

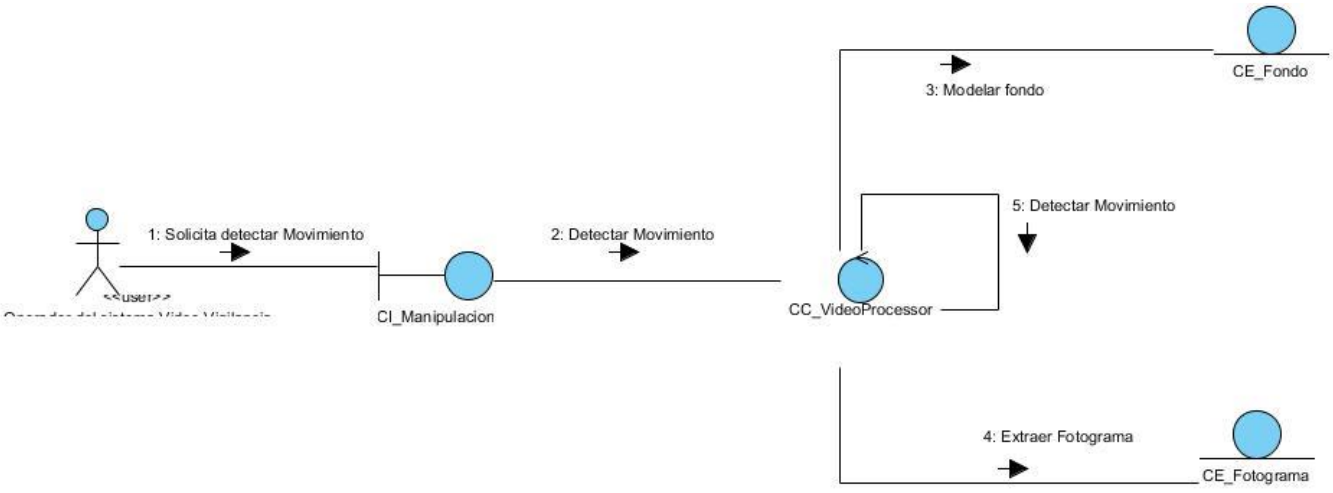


Diagrama 11: Diagrama de Comunicación CU Detectar Manipulación sección Detectar movimiento de la cámara.

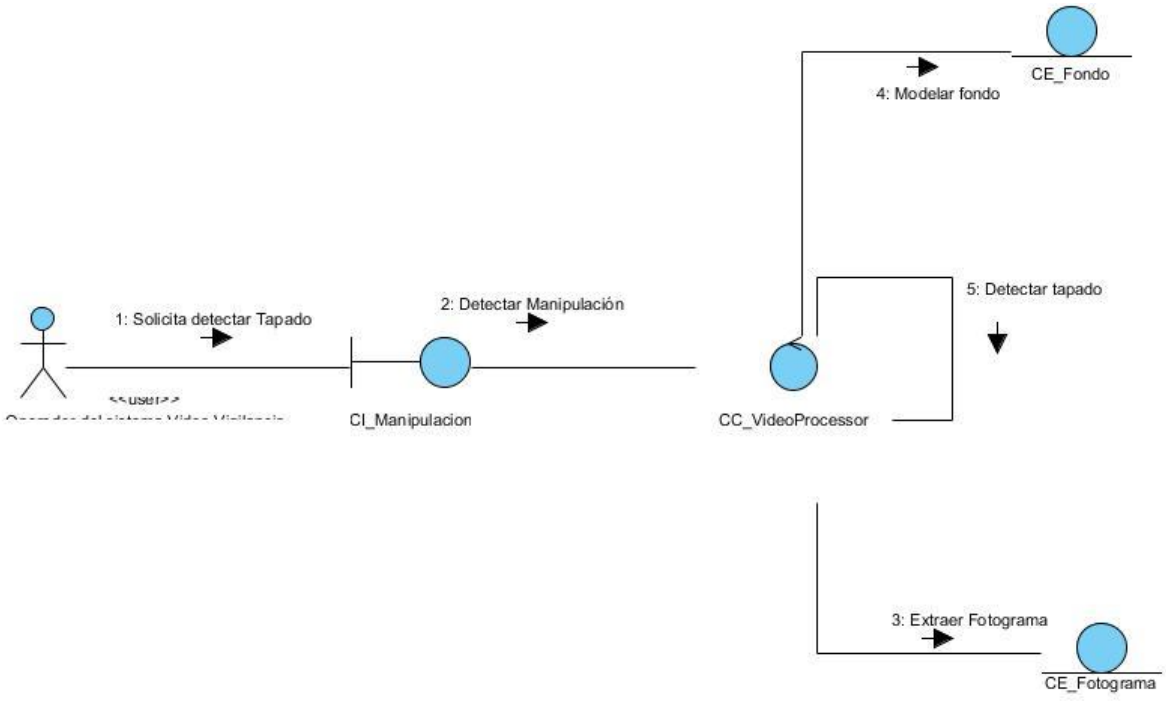


Diagrama 12: Diagrama de Comunicación CU Detectar Manipulación sección Detectar tapado de la cámara.

3.3 Modelo de Diseño.

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

3.3.1 Diagrama de clases del diseño.

El diagrama de clases del diseño es en esencia como va a quedar constituida la implementación del producto. Contiene las clases y en cada una de ellas los atributos y métodos que las componen. Del diagrama de clases del diseño al código final no hay muchos cambios por lo cual este es un artefacto importante en la ingeniería del Video Sensor.

Las clases que se observan en el Diagrama 13 son: VideoProcesor, clase controladora que es la que se encarga de delegar responsabilidades a las otras clases, coordina el funcionamiento, se le inserta el tipo

de sensor e inicia el procesamiento del video, entre otras acciones; FFread, esta clase es la que permite el trabajo con los fotogramas, abriendo o cerrando el flujo de video que captura la cámara; ShadowRemoval, esta clase se encarga de eliminar la sombra; BGFG_Gauss_Segm, mediante esta clase se extrae el frente y fondo de las imágenes; CoveredDetection, en esta clase se implementa un algoritmo que identifica si en la imagen la distribución de colores es distinta en dos imágenes diferentes, se hace uso de histogramas para comparar las intensidades de color y así detectar si está tapada la cámara; MoveDetection, en esta clase se implementa un algoritmo que compara dos modelos de fondo uno de un instante de tiempo anterior y el otro del instante actual para detectar si la cámara ha sido movida; DefocusedDetection, en esta clase se implementa un algoritmo que utiliza la comparación de los bordes de la imagen y su modelo de fondo y detectar si la cámara esta desenfocada.

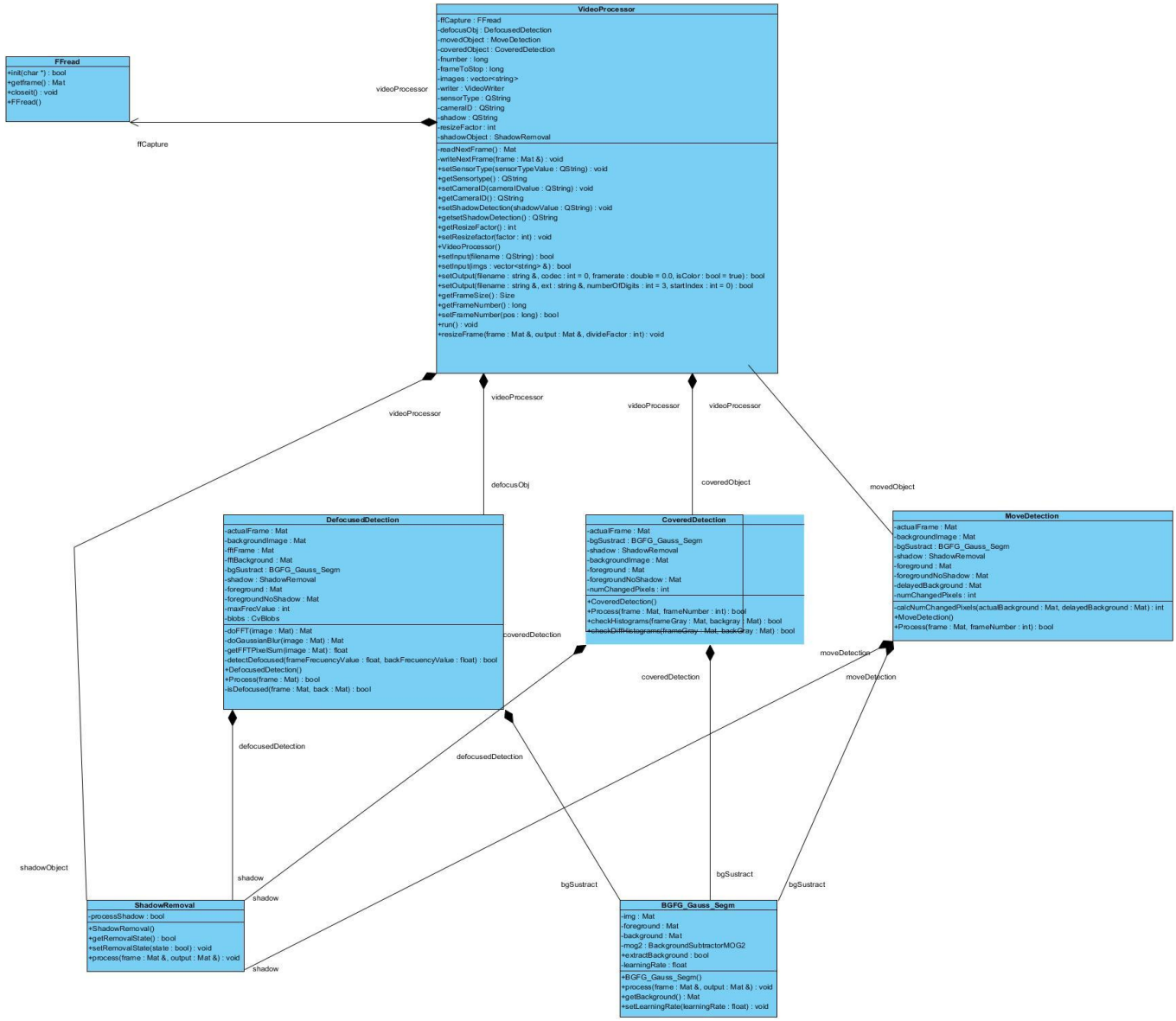


Diagrama 13: Diagrama de Clases del Diseño

3.3.2 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.

En el Diagrama 14 se representa el diagrama de secuencia para el Caso de Uso Captura de flujo de video y extracción de fotograma, en el que intervienen las clases videoProcessor y FFread además las acciones de comunicación entre estas.

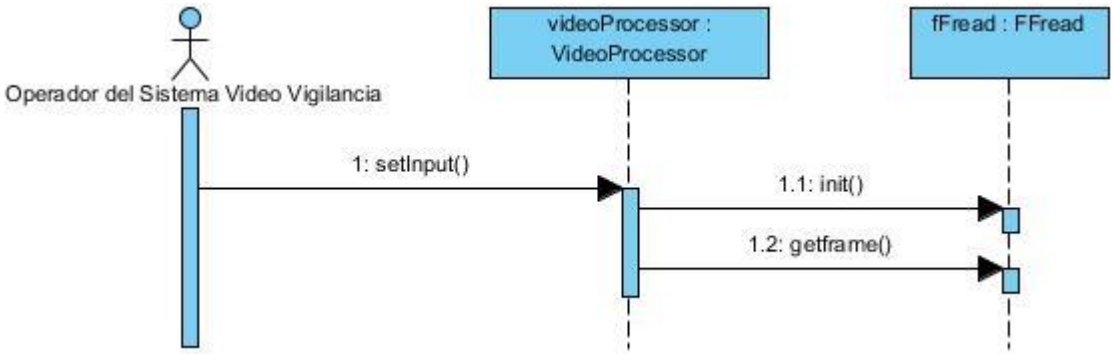


Diagrama 14: Diagrama de secuencia CU Capturar de flujo de video y extraer fotograma

En el Diagrama 15 se representa el diagrama de secuencia para el Caso de Uso Procesamiento de fotograma, en el que intervienen las clases videoProcessor y shadowRemoval además las acciones de comunicación entre estas.

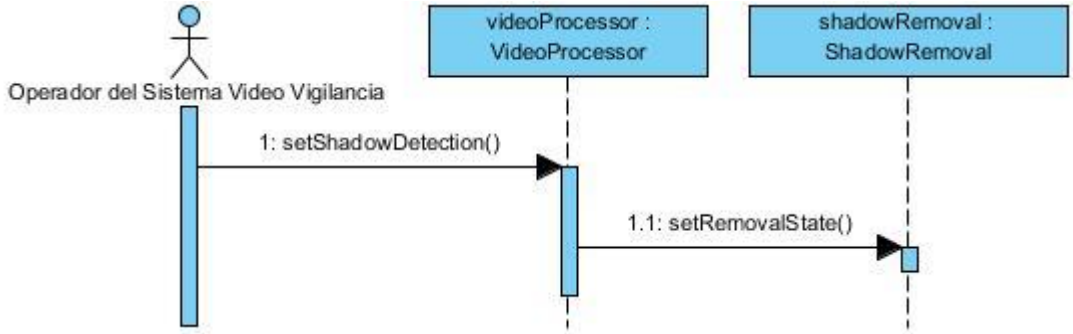


Diagrama 15: Diagrama de secuencia CU Procesar fotograma

En el Diagrama 16 se representa el diagrama de secuencia para el Caso de Uso Detección de manipulación sección Detectar desenfoque de cámara, en el que intervienen las clases videoProcessor, defocusedDetection, shadowRemovel, FFread, BGFG_Gauss_Segm y además las acciones de comunicación entre estas.

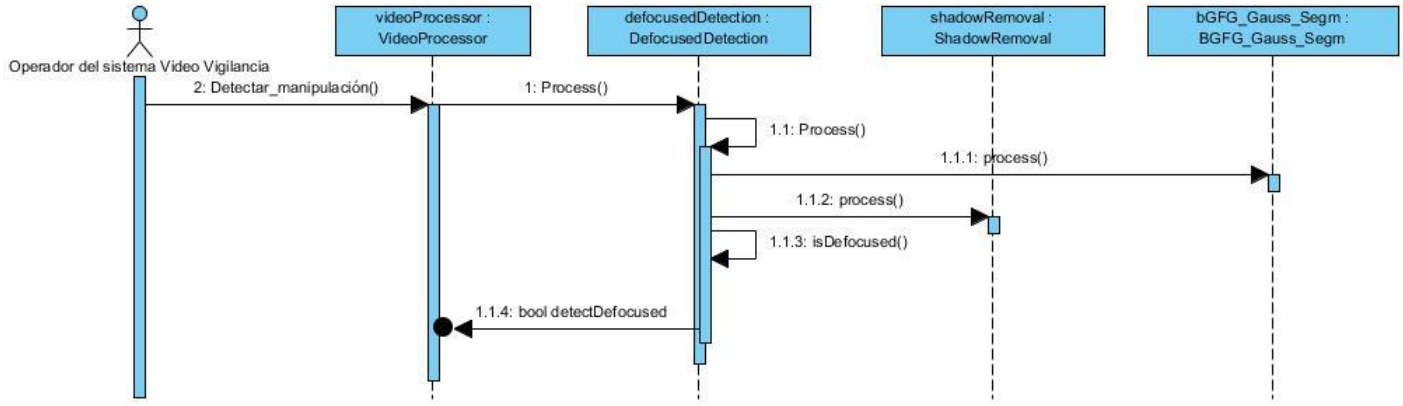


Diagrama 16: Diagrama de secuencia CU Detectar manipulación sección Detectar desenfoque de cámara.

En el Diagrama 17 se representa el diagrama de secuencia para el Caso de Uso Detección de manipulación sección Detectar tapado de cámara, en el que intervienen las clases videoProcessor, coveredDetection, shadowRemovel, FFread, BGFG_Gauss_Segm y además las acciones de comunicación entre estas.

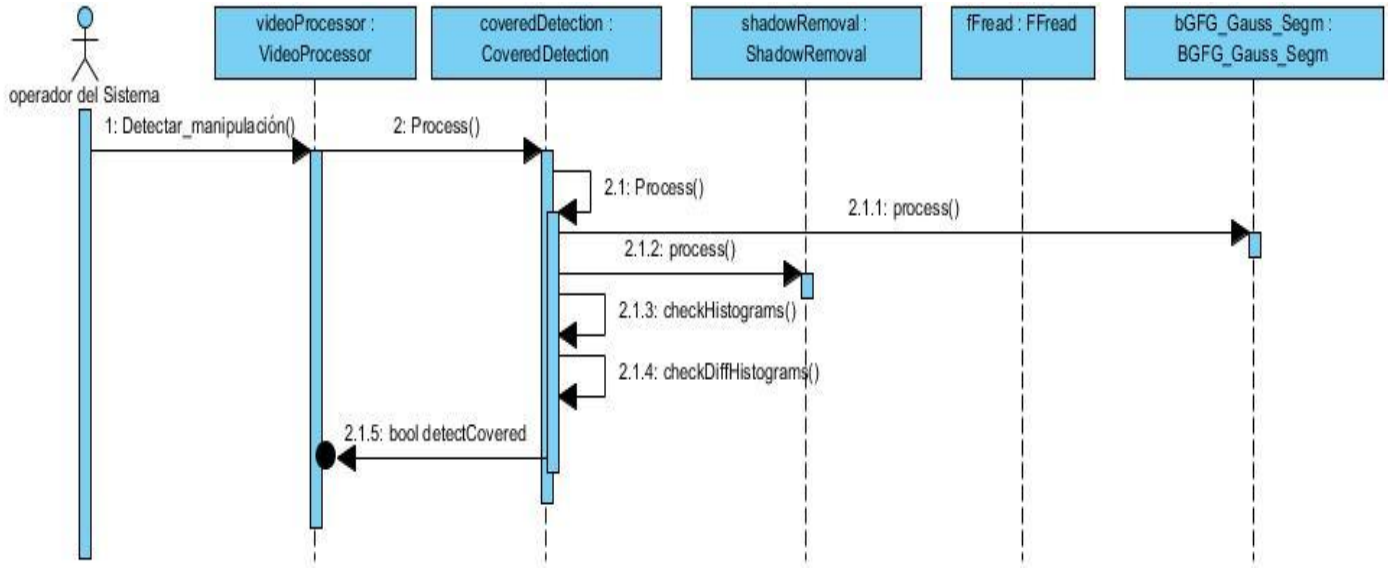


Diagrama 17: Diagrama de secuencia CU Detectar manipulación sección Detectar tapado de cámara.

En el Diagrama 18 se representa el diagrama de secuencia para el Caso de Uso Detección de manipulación sección Detectar el movimiento de la cámara, en el que intervienen las clases videoProcessor, moveDetection, shadowRemovel, FFread, BGFG_Gauss_Segm y además las acciones de comunicación entre estas.

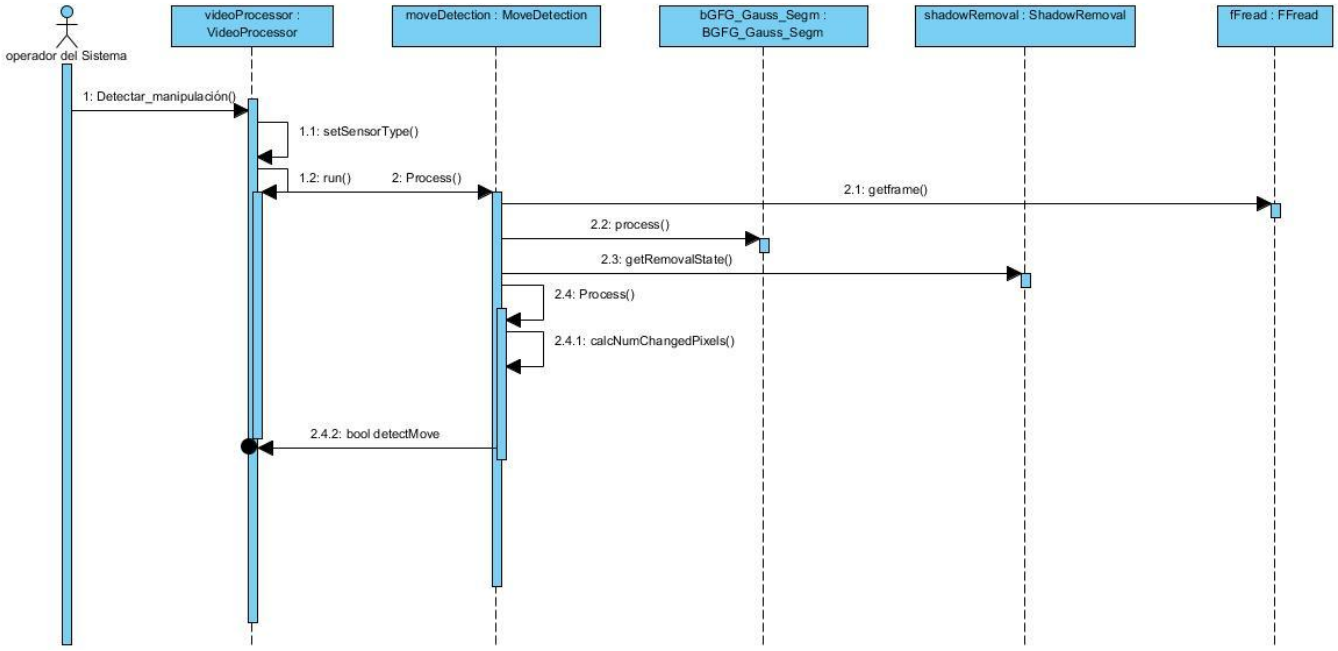


Diagrama 18: Diagrama de secuencia CU Detectar manipulación sección Detectar el movimiento de la cámara.

3.4 Conclusiones.

El Modelo de Análisis y Modelo de Diseño han sido pasos en la construcción del video sensor que permiten refinar y estructurar los requisitos funcionales y no funcionales identificados en capítulos anteriores. Gracias al análisis se llega a tener un mayor dominio del problema y permite modelar fácilmente una estructura básica para el funcionamiento del mismo. Con el Modelo de diseño se refinan las ideas modeladas en el análisis y este llega a generar un artefacto (Diagrama de clases del diseño), que puede utilizarse como diseño final en la implementación y sin presentar ambigüedades.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En este capítulo se presenta una muestra de los principales algoritmos para desarrollar el video sensor como son: modelado de fondo, detectar movimiento de cámara, detectar desenfoque del lente, detectar tapado del lente. Se exponen los diagramas de componente y de despliegue. Son expuestos los resultados de las pruebas de rendimiento para comprobar el correcto funcionamiento de la aplicación.

4.2 Análisis de los algoritmos.

El procesamiento digital de imágenes permite identificar características matemáticas que permitirán procesar las imágenes y así poder identificar las situaciones que se pueden estar dando en el ambiente vigilado.

En las técnicas expuestas en este epígrafe se utilizan en el procesamiento de imágenes para llevar a cabo la captura de fotogramas, el modelado de fondo, extracción de bordes de los objetos en las imágenes, se calculan los histogramas, para en conjunto facilitar la detección de: movimiento de cámara, desenfoque del lente y tapado del lente. Las funciones antes dichas son utilizadas gracias a la biblioteca OpenCV la cual trae implementadas disímiles funciones como estas.

4.2.1 Modelado de fondo.

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

El primer paso para aplicar MoG es inicializar los parámetros: media, desviación y factor de peso, los cuales representan el fondo de la imagen. Luego se inicializa la media de una de las Gaussianas en 1 a la primera imagen (por Ejemplo: la Gaussiana i=1), el resto de medias tendrá valores aleatorios. La primera Gaussiana modelará el pixel en la primera imagen, en la cual el peso w para k=1 debe ser un valor próximo a 1, y para el resto será un valor próximo a 0, debido a que la suma de los pesos de las Gaussianas es igual a 1.

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

Ecuación 1

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

↓

Ecuación 2

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

El factor $\rho = \alpha \Pr(\frac{I_t(x,y)}{\mu_{i,t-1}(x,y)}, \sigma^2_{i,t-1}(x,y))$, α es un parámetro predefinido al inicio de la ecuación, $\mu_{i,t}(x,y)$ es el valor de la media del píxel en el instante t e $I_t(x,y)$ es el valor del pixel en el instante t. Los píxeles que no sean marcados como *background* (fondo) serán marcados como *foreground*. (Richard Bowden, 2011)

Ilustración 13: Fondo del fotograma.

Ilustración 14: Fotograma.

4.2.2 Detectar desenfoque del lente.

Cuando una cámara es desenfocada en la imagen que se visualiza es difícil de distinguir las formas y los objetos dado a que se afectan sus bordes.

Para la detección de desenfoque, se extraen los bordes presentes en la imagen actual I_n, y su respectivo modelo de fondo B_n. La cantidad de pixeles que pertenecen al borde (M) son contadas en cada imagen:

Si el pixel representa un borde, se incrementa el valor de M que denota la cantidad de pixeles que son de borde, en este trabajo un borde representa un pixel blanco.

Ecuación 3: Suma de los pixeles que representan bordes en una imagen.

$$M = M + 1$$

Con el valor Mde la imagen actual y del modelo de fondo calculados, se resta $M_{B_n} - M_{I_n}$. El resultado obtenido tiene que ser positivo, esto denota una reducción de los bordes en la imagen actual, esto evidencia el desenfoque. El valor de la resta se acota que sea mayor que un número N, que no es más que el umbral de sensibilidad, a menor N, mayor sensibilidad.

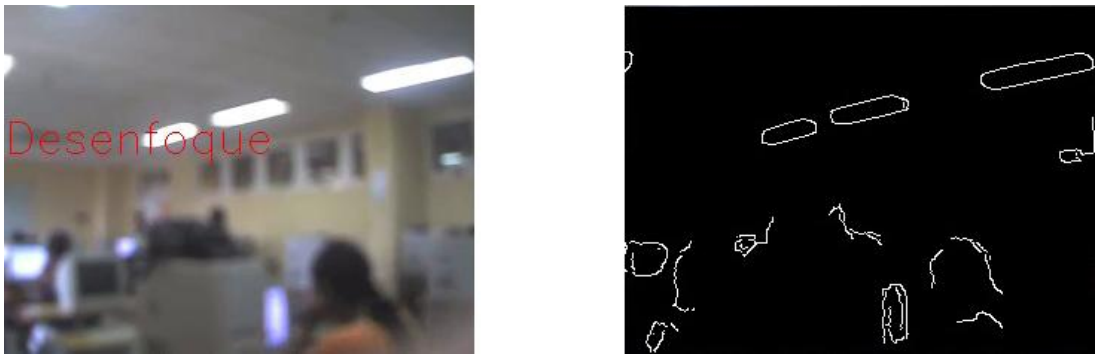


Ilustración 15: Imagen actual y bordes extraídos.

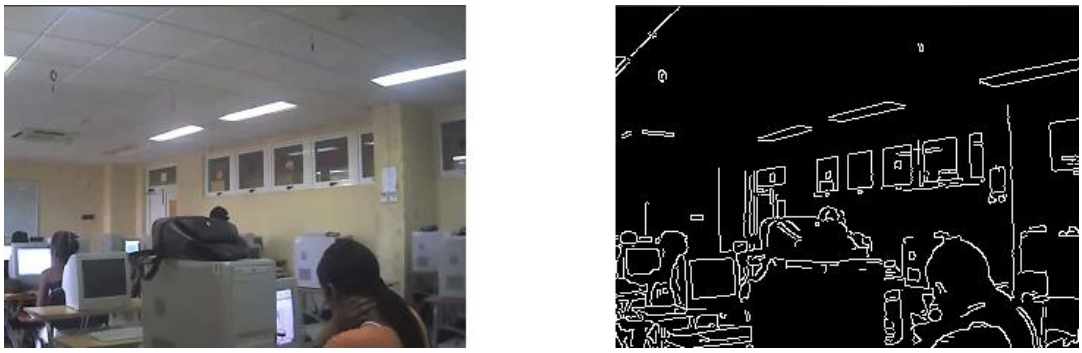


Ilustración 16: Modelo de fondo y bordes extraídos.

4.2.3 Detectar movimiento de la cámara.

Cuando una cámara se mueve a una dirección diferente, la imagen de fondo B_n se actualiza para reflejar los cambios ocurridos. En este algoritmo se propone utilizar otra imagen de fondo anterior que se representa como B_{n-k}, donde $k \in Z^+$. B_{n-k} es comparada con B_n para encontrar si una cámara apunta una dirección diferente. Ilustraciones 28 y 29. Un valor de proporción, denotado P, se calcula mediante la comparación de cada pixel en B_n con el correspondiente en B_{n-k}. Ecuación 4.

Ilustración 17: Modelo de fondo anterior.

Ilustración 18: Modelo de fondo actual.

Ecuación 4

$$P = \begin{cases} P+1, if\ B_n(x,y) \neq B_{n-k}(x,y) \\ P, if\ B_n(x,y) = B_{n-k}(x,y) \end{cases}$$

La cámara esta movida a una diferente dirección si $P > Th_2 k$, donde $0 < Th_2 < 1$ y es un valor de umbral que aumenta la sensibilidad cuando está más cerca de 0 y k es el número total de píxeles.

En este algoritmo Th_2 es el porciento de pixeles que tienen un cambio significativo de un modelo de fondo a otro, entre más cercano a 0, más sensible es. (SAĞLAM, 2009)

4.2.4 Detectar tapado del lente de la cámara.

Cuando el lente de la cámara es tapado por algún objeto, el histograma adquirido de la imagen I_n se espera que tenga valores más altos en un rango específico comparándolo con el histograma del modelo de fondo B_n. Se espera que con la mayor parte de la escena tapada por el objeto una parte significativa de la I_n tenga el mismo color. Ilustración 19 y 20.

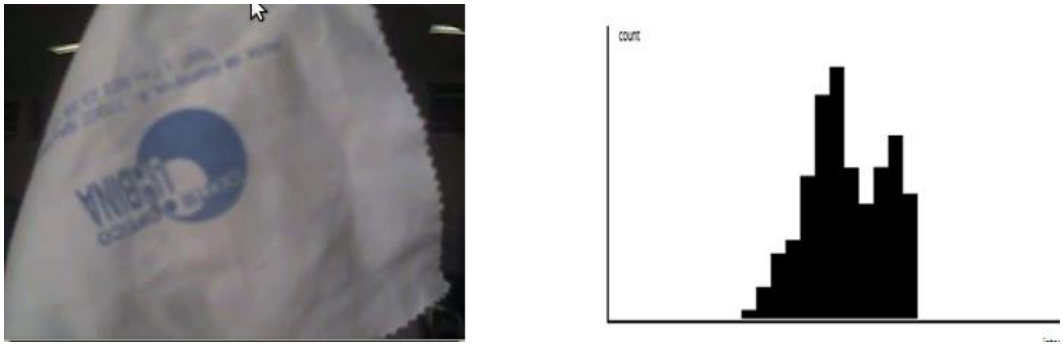


Ilustración 19: Imagen actual y su histograma.

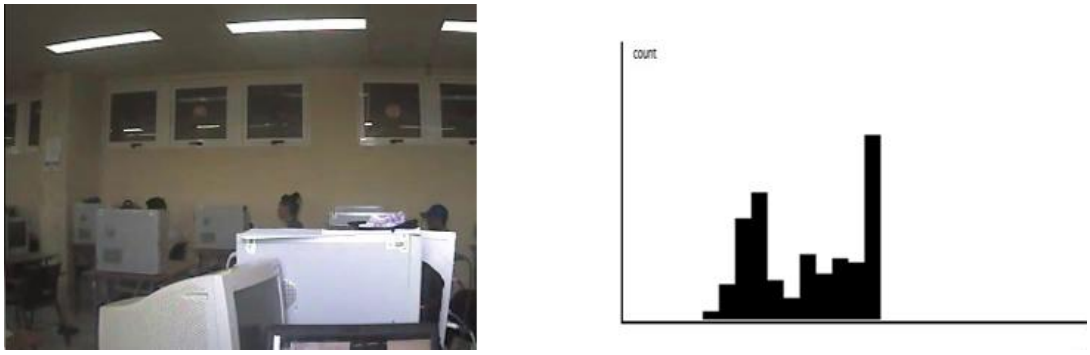


Ilustración 20: Modelo de fondo y su histograma.

En este algoritmo se calculan los histogramas de I_n y B_n. Sea $\max(H(A))$ el mayor valor del histograma en una imagen A. El primer paso a realizar es, comparar los números máximos de I_n y B_n para ver cual tiene el pico más alto. Para el segundo paso, el histograma de la diferencia absoluta $|I_n - B_n|$ se comprueba para ver si la mayoría de los valores se acumulan al final, cerca del color negro.

El histograma a utilizar será de 32 dimensiones de una imagen denominada $H_i()$ donde $1 \leq i \leq 32$. Con las ecuaciones 5 y 6 se chequea si la cámara esta tapada o no.

Ecuación 5

$$\left(H_{\max(H(I_n))-1}(I_n) + H_{\max(H(I_n))}(I_n) + H_{\max(H(I_n))+1}(I_n)\right) > Th_3\left(H_{\max(H(I_n))-1}(B_n) + H_{\max(H(I_n))}(B_n) + H_{\max(H(I_n))+1}(B_n)\right)$$

En la Ecuación 5 toma la suma del valor máximo del histograma con sus dos vecinos más cercanos y lo compara con esta misma suma multiplicada por el Th_3.

En la Ecuación 5 Th_3 es el umbral que se puede aumentar para una mayor sensibilidad del algoritmo. En la ecuación anterior si el valor de i (posición del color en el histograma) es menor que 1 o mayor que 32 entonces se toma como 1 o 32, respectivamente.

En el segundo paso del algoritmo se utiliza la diferencia entre imágenes. Cuando la cámara no es tapada, la diferencia de imágenes consta de pixeles mayormente negros. En este caso, todos los valores del histograma de las diferencias de imágenes, se encuentran en la primera dimensión. Si la cámara está tapada, I_n y B_n son diferentes el uno del otro, en el histograma de la diferencia de imágenes, los valores se expanden por el histograma. El segundo paso del algoritmo se calcula de la siguiente manera:

Ecuación 6

$$\sum_{i=1}^{32} H_i(|I_n - B_n|) > Th_4 \sum_{i=1}^{k} H_i(|I_n - B_n|)$$

El valor del umbral Th_4 se puede aumentar para una mejor sensibilidad del algoritmo. Cuando $0 \leq k \leq 32$

y cuando más cerca este del límite inferior, la sensibilidad será mayor, para obtener resultados satisfactorios se le da un valor de $k = 3$. (SAĞLAM, 2009)

4.3 Modelo de implementación.

4.3.1 Diagrama de Componentes.

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

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

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

En el diagrama 19 se muestran los componentes del Video Sensor para la detección de manipulación de cámara IP.

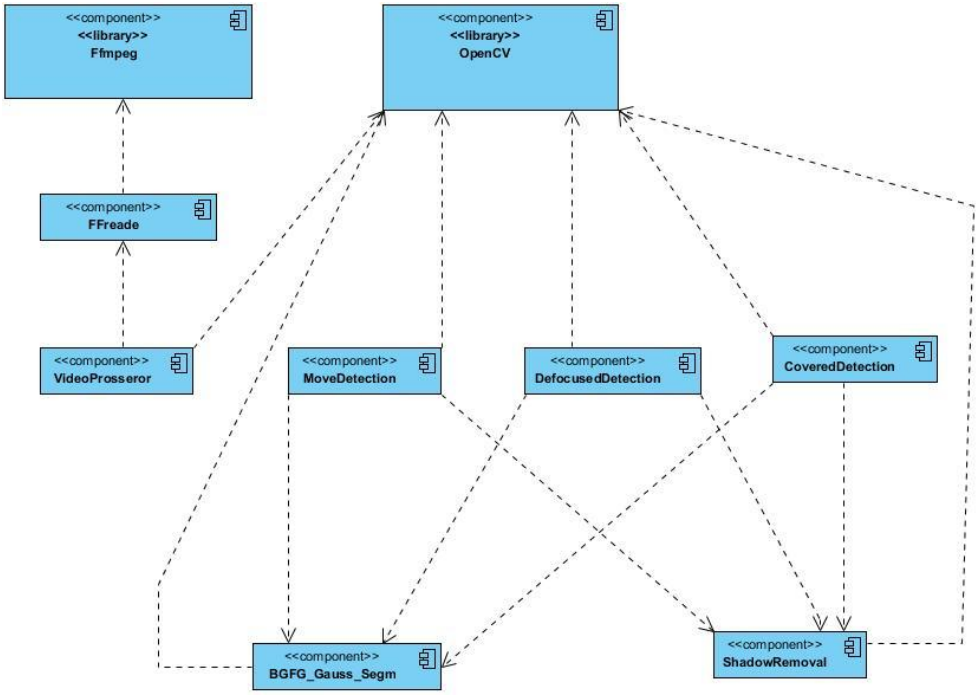


Diagrama 19: Diagrama de componentes del Video Sensor para la detección de manipulación.

4.3.2 Diagrama de despliegue.

En el diagrama de despliegue se muestran físicamente los dispositivos que interactúan en la composición del sistema, estos dispositivos se muestran como nodos, además se encuentran los protocolos con los que se comunican dichos nodos y también los programas ejecutables sobre los nodos.

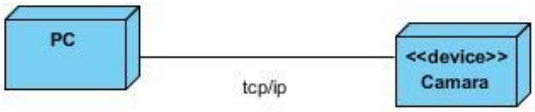


Diagrama 20: Diagrama de despliegue del Video Sensor para la detección de manipulación.

4.4 Pruebas de Software.

Cuando el código ya ha sido generado, el software debe ser probado para descubrir y corregir el máximo de errores posibles antes de su liberación. El buen diseño de prueba permite, comprobar la lógica interna de los componentes de software y se verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad el comportamiento y el rendimiento.

El software debe probarse desde dos perspectivas diferentes: la lógica interna del programa se comprueba utilizando técnicas de diseño de casos de prueba de Caja Blanca. Los requisitos del software se comprueban utilizando técnicas de diseño de casos de prueba de Caja Negra. Lo que se intenta hacer en ambos casos es encontrar la mayor cantidad de errores con la menor cantidad de esfuerzo y tiempo.

4.4.1 Prueba de Falsos Positivos y Falsos Negativos.

Al terminar la implementación del Video Sensor es necesario realizarle pruebas de rendimiento, la cual consiste en detectar Falsos Positivos y Falsos Negativos para obtener un valor aproximado de posible error en la detección de la manipulación. Las pruebas realizadas fueron cerca de 200 casos a cada una de las secciones: tapado, desenfoque y movimiento de la cámara.

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

$$Precision\,rate = \frac{CC}{CC+FP} \quad Recall\,rate = \frac{CC}{CC+FN}$$

Donde CC es el número de casos correctos que el sistema detectó correctamente, FP son los Falsos Positivos que detectó el sistema como manipulado y FN son los Falsos Negativos, la manipulación que debería haber detectado el sistema y no lo hizo.

Una medida que combina $Precision$ y $Recall$ es F que mientras mayor sea su valor, mejor se considera el rendimiento del sistema.

$$F = 2 * \frac{Precision\,rate * Recall\,rate}{Precision\,rate + Recall\,rate}$$

Una vez obtenida F se puede calcular el porciento de error E que tiene el sistema. (Makhoul, y otros)

$$E = 1 - F$$

4.4.2 Resultados experimentales.

La prueba es realizada a cada algoritmo implementado por separado para así determinar la eficiencia de cada uno de ellos. En cada caso se han realizado acciones que el sistema debería haber reconocido, llegando de esa manera a los resultados seguidamente mostrados:

Tapado.

En este algoritmo se reconocen falsos positivos cuando se mueve la cámara entre otros ejemplos, los falsos negativos que se detectaron son entre otros: por la obstrucción del lente con una hoja de papel blanco la cual refleja la claridad con más intensidad que otro objeto por lo que no se llega a detectar el tapado. Los resultados de la prueba se muestran en la siguiente tabla:

Parámetro	Valor
Conteos Correctos (CC)	150
Falsos Positivos (Fp)	6
Falsos Negativos (Fn)	2
$Presicion\ rate$	0.96
$Recall\ rate$	0.98
F	0.96
Error (E)	0.04

Movimiento

Para el algoritmo de detección de movimiento los resultados fueron los siguientes:

Parámetro	Valor
Conteos Correctos (CC)	180

Falsos Positivos (Fp)	10
Falsos Negativos (Fn)	2
$Presicion\ rate$	0.94
$Recall\ rate$	0.98
F	0.94
Error (E)	0.06

Desenfoque.

En este algoritmos se reconocen falsos positivos cuando se obstruye la cámara, ya que al taparse se reducen los bordes de los objetos en la escena y eso hace disparar la alarma. Los resultados obtenidos se muestran a continuación:

Parámetro	**Valor**
Conteos Correctos (CC)	180
Falsos Positivos (Fp)	10
Falsos Negativos (Fn)	18
$Presicion\ rate$	0.94
$Recall\ rate$	0.90
F	0.92
Error (E)	0.08

4.5 Conclusiones.

La implementación de los algoritmos expuestos utilizando las bibliotecas OpenCV y Ffmpeg demuestra que al reconocer características específicas en las imágenes procesadas se pueden identificar situaciones dadas en un entorno. Se evidencia que la sustracción de imágenes, modelado de fondo, cálculo de histograma y extracción de bordes son pasos fundamentales para obtener una solución funcional. A través de las pruebas de rendimiento se llega a la conclusión de que el algoritmo que mejor detecta es el de tapado con una eficiencia del 96% y se deben tener en cuenta situaciones que pueden afectar el correcto funcionamiento del video sensor como los cambios bruscos de iluminación. Se definieron los umbrales de sensibilidad de los algoritmos, permitiendo un mejor funcionamiento ante los diferentes tipos de situaciones a detectar.

CONCLUSIONES

Los sistemas de video vigilancia existentes que contienen Videos sensores para la detección de manipulación los cuales son propietarios. Con la fusión de algoritmos y técnicas de procesamiento de imágenes y videos digitales, se implementó un video sensor que detecta la manipulación de una cámara a partir de flujos de videos brindados por cámaras IP. La importancia de este componente está dada en que se pueda detectar la manipulación de la cámara ya que en un ambiente monitoreado se pueden dar brechas en la seguridad al ocurrir alguna de las formas de manipulación.

Este Video Sensor será integrado en el sistema de Video Vigilancia SURIA en el módulo de Análisis. Se utilizaron algoritmos específicos para cada tipo de manipulación que se identificó: tapado, desenfoque y movimiento de la cámara. Se utilizaron bibliotecas para el procesamiento de imagen como OpenCV y Ffmpeg las cuales tienen integradas funcionalidades para la Visión por Computadora y captura de flujos de video respectivamente.

Se obtuvo un componente confiable y eficiente que cumple con los requisitos identificados, después de las pruebas realizadas mostraron los siguientes índices de rendimiento: en el caso de Tapado un 96%, un 94% en la detección del Movimiento de la cámara y en el caso de Desenfoque un 92%. Estas pruebas demuestran que el sistema es fiable y ayuda a mejorar los procesos de vigilancia y control de los Sistemas de Video Vigilancia.

RECOMENDACIONES

La autora del trabajo de diploma recomienda:

- Que en la integración del Video Sensor en el sistema, cada tipo de detección de manipulación sea utilizado como procesos independientes para lograr estabilidad e independencia en caso de que uno de los procesos no trabaje correctamente.
- Que se realicen pruebas más extensas en diferentes ambientes, dígase exteriores e interiores, para ajustar bien los parámetros y mejorar si es posible los resultados de las pruebas realizadas.
- Que se estudie la posibilidad de extender el alcance de los sensores a detección de desastres, como por ejemplo incendios.
- Mantener actualizado el Video Sensor con las últimas versiones de las bibliotecas utilizadas, para mejorar el tiempo de captura y procesamiento del video.

TRABAJOS CITADOS

GSMspain. 1996. GSMspain. [En línea] GSMspain S.A, 1996. [Citado el: 4 de Diciembre de 2012.] http://www.gsmspain.com/glosario/?palabra=P%CDXEL.
ABC, Definicion. 2012. Definicion ABC. [En línea] 2012. [Citado el: 28 de Mayo de 2013.] www.definicionabc.com/tecnologia/video.
Adrian Kaehler, Gary Bradski. 2008. *Leraning OpenCV.* 2008.
Albiol Colomer, Antonio. 2003. *Seguimiento de objetos en secuencias de video.* Valencia : s.n., 2003.
Aventura Technologies, Inc. Aventura. *Aventura. Soluciones de Seguridad de Diseño.* [En línea] Aventura Technologies, Inc. [Citado el: 21 de 12 de 2012.] http://www.aventuracctv.com/es/.
C. Gonzalez, Rafael y E. Woods, Richard. 2002. *Digital Image Processing.* Upper Saddle River, New Jersey : Prentice-Hall, Inc, 2002. 0-201-18075-8.
Chacón, Julio Cesar Rueda. 2006. Aplicación de la Metodología RUP para el desarrollo. Guatemala : s.n., 2006.
Cisco. 2011. Cisco.com. [En línea] 2011. [Citado el: 21 de Diciembre de 2012.] www.cisco.com .
Duck, Black. 2010. Ohloh. *Ohloh BY BLACK DUCK.* [En línea] Black Duck Software, Inc., 2010. [Citado el: 26 de 1 de 2013.] http://www.ohloh.net/p/ffmpeg.
EE Times University. 2012. EE Times. *EE Times Design.* [En línea] EE Times University, 10 de Octubre de 2012. [Citado el: 21 de 12 de 2012.] http://www.eetimes.com/.
FFMPEG. *FFMPEG.* [En línea] [Citado el: 26 de Enero de 3013.] http://www.ffmpeg.org/about.html.
G4S Technology Ltd. 2012. G4S Securing Your World. *G4S Securing Your World. Video Content Analytics.* [En línea] G4S Technology Ltd, 2012. [Citado el: 21 de Diciembre de 2012.] http://www.g4stechnology.co.uk/.
Giraldo, Luis y Zapata, Yuliana. 2005. *Herramientas de Desarrollo de Ingenieria de SW para Linux.* 2005.
González, Danays Rodríguez Martínez, Marilys Valiente. 2010. *Módulo para la gestión de configuración del equipamiento tecnológico en la Universidad de las Ciencias Informáticas.* Habana : s.n., 2010.
GSMspain. GSMspain. [En línea] GSMspain. [Citado el: 19 de diciembre de 2012.] http://www.gsmspain.com/glosario/?palabra=P%CDXEL.
Instituto Nacional de Tecnologías Educativas y de formación de Profesorado. Cultura audiovisual: INTEF. *Sitio Web de INTEF.* [En línea] [Citado el: 12 de noviembre de 2012.] http://recursostic.educacion.es/artes/plastic/web/cms/index.php?id=579.
Jacobson,G. Booch, J. Rumbaugh. 2000. *El lenguaje Unificado de Modelado, Manual de referencia.* 2000.
La Inteligencia en aplicaciones de video. **Revista de Negocios de Seguridad. 2010.** s.l. : Revista de Negocios de Seguridad, 2010.
Laganière, Robert. 2011. *OpenCV 2 Computer Vision Application Programming* . s.l. : Packt Publishing Ltd., 2011. 978-1-849513-24-1.
Larman, Craig. *UML y Patrones.*
Loy, Chen Change. 2010. *Activity Understanding and Unusual Event.* Londres : s.n., 2010.

Makhoul, John, y otros. *PERFORMANCE MEASURES FOR INFORMATION EXTRACTION.* Cambridge : BBN Technologies, GTE Corp.
Molina, R. 1998. *Introducción al Procesamiento y Análisis de Imágenes Digitales.* Granada : s.n., 1998. 18071.
Orallo, Enrique Hernández. 2010. *El Lenguaje Unificado de Modelado (UML).* 2010.
Pixelco. 2012. Pixelco Blog. *Pixelco Media.* [En línea] 2012. [Citado el: 12 de diciembre de 2012.] http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/.
Pressman, Roger S. 2001. *Ingenieria del Software. Un enfoque práctico.* Madrid : s.n., 2001.
Richard Bowden, Pakorn KaewTraKulPong. 2011. *An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection.* Brunel : Kluwer Academic Publishers, 2011.
Robert Laganière. 2011. *OpenCV 2 Computer Vision .* Olton : Packt Publishing Ltd, 2011. 1180511.
SAĞLAM, ALĠ. 2009. *ADAPTIVE CAMERA TAMPER DETECTION FOR VIDEO SURVEILLANCE.* 2009.
Sierra, María. 2006. *Trabajando con Visual Paradigm for UML.* Cantabria : s.n., 2006.
2012. Sobre Concepto. [En línea] Octubre de 2012. [Citado el: 28 de Mayo de 2013.] www.sobreconceptos.com/video.
Stroustrup, Bjarne. 1997. *The C++ ProgramingLanguage (Third Edition).* s.l. : Addison-Wesley, 1997.
THRIVE Intelligence, LLC. 2012. THRIVE Intelligence. *THRIVE Intelligence.* [En línea] 2012. [Citado el: 21 de Diciembre de 2012.] www.THRIVEintelligence.com .
Video analytics and preemptive surveillance. **VideoSurveillance.com. 2010.** Portland : VideoSurveillance.com, 2010.

BIBLIOGRAFÍA

GSMspain. 1996. GSMspain. [En línea] GSMspain S.A, 1996. [Citado el: 4 de Diciembre de 2012.] http://www.gsmspain.com/glosario/?palabra=P%CDXEL.
ABC, Definicion. 2012. Definicion ABC. [En línea] 2012. [Citado el: 28 de Mayo de 2013.] www.definicionabc.com/tecnologia/video.
Adrian Kaehler, Gary Bradski. 2008. *Leraning OpenCV.* 2008.
Albiol Colomer, Antonio. 2003. *Seguimiento de objetos en secuencias de video.* Valencia : s.n., 2003.
Aventura Technologies, Inc. Aventura. *Aventura. Soluciones de Seguridad de Diseño.* [En línea] Aventura Technologies, Inc. [Citado el: 21 de 12 de 2012.] http://www.aventuracctv.com/es/.
C. Gonzalez, Rafael y E. Woods, Richard. 2002. *Digital Image Processing.* Upper Saddle River, New Jersey : Prentice-Hall, Inc, 2002. 0-201-18075-8.
Chacón, Julio Cesar Rueda. 2006. Aplicación de la Metodología RUP para el desarrollo. Guatemala : s.n., 2006.
Cisco. 2011. Cisco.com. [En línea] 2011. [Citado el: 21 de Diciembre de 2012.] www.cisco.com .
Duck, Black. 2010. Ohloh. *Ohloh BY BLACK DUCK.* [En línea] Black Duck Software, Inc., 2010. [Citado el: 26 de 1 de 2013.] http://www.ohloh.net/p/ffmpeg.
EE Times University. 2012. EE Times. *EE Times Design.* [En línea] EE Times University, 10 de Octubre de 2012. [Citado el: 21 de 12 de 2012.] http://www.eetimes.com/.
FFMPEG. *FFMPEG.* [En línea] [Citado el: 26 de Enero de 3013.] http://www.ffmpeg.org/about.html.
G4S Technology Ltd. 2012. G4S Securing Your World. *G4S Securing Your World. Video Content Analytics.* [En línea] G4S Technology Ltd, 2012. [Citado el: 21 de Diciembre de 2012.] http://www.g4stechnology.co.uk/.
Giraldo, Luis y Zapata, Yuliana. 2005. *Herramientas de Desarrollo de Ingenieria de SW para Linux.* 2005.
González, Danays Rodríguez Martínez, Marilys Valiente. 2010. *Módulo para la gestión de configuración del equipamiento tecnológico en la Universidad de las Ciencias Informáticas.* Habana : s.n., 2010.
GSMspain. GSMspain. [En línea] GSMspain. [Citado el: 19 de diciembre de 2012.] http://www.gsmspain.com/glosario/?palabra=P%CDXEL.
Instituto Nacional de Tecnologías Educativas y de formación de Profesorado. Cultura audiovisual: INTEF. *Sitio Web de INTEF.* [En línea] [Citado el: 12 de noviembre de 2012.] http://recursostic.educacion.es/artes/plastic/web/cms/index.php?id=579.
Jacobson,G. Booch, J. Rumbaugh. 2000. *El lenguaje Unificado de Modelado, Manual de referencia.* 2000.
La Inteligencia en aplicaciones de video. **Revista de Negocios de Seguridad. 2010.** s.l. : Revista de Negocios de Seguridad, 2010.
Laganière, Robert. 2011. *OpenCV 2 Computer Vision Application Programming* . s.l. : Packt Publishing Ltd., 2011. 978-1-849513-24-1.
Larman, Craig. *UML y Patrones.*
Loy, Chen Change. 2010. *Activity Understanding and Unusual Event.* Londres : s.n., 2010.

Makhoul, John, y otros. *PERFORMANCE MEASURES FOR INFORMATION EXTRACTION.* Cambridge : BBN Technologies, GTE Corp.
Molina, R. 1998. *Introducción al Procesamiento y Análisis de Imágenes Digitales.* Granada : s.n., 1998. 18071.
Orallo, Enrique Hernández. 2010. *El Lenguaje Unificado de Modelado (UML).* 2010.
Pixelco. 2012. Pixelco Blog. *Pixelco Media.* [En línea] 2012. [Citado el: 12 de diciembre de 2012.] http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollo-multiplataforma/.
Pressman, Roger S. 2001. *Ingenieria del Software. Un enfoque práctico.* Madrid : s.n., 2001.
Richard Bowden, Pakorn KaewTraKulPong. 2011. *An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection.* Brunel : Kluwer Academic Publishers, 2011.
Robert Laganière. 2011. *OpenCV 2 Computer Vision .* Olton : Packt Publishing Ltd, 2011. 1180511.
SAĞLAM, ALĠ. 2009. *ADAPTIVE CAMERA TAMPER DETECTION FOR VIDEO SURVEILLANCE.* 2009.
Sierra, María. 2006. *Trabajando con Visual Paradigm for UML.* Cantabria : s.n., 2006.
2012. Sobre Concepto. [En línea] Octubre de 2012. [Citado el: 28 de Mayo de 2013.] www.sobreconceptos.com/video.
Stroustrup, Bjarne. 1997. *The C++ ProgramingLanguage (Third Edition).* s.l. : Addison-Wesley, 1997.
THRIVE Intelligence, LLC. 2012. THRIVE Intelligence. *THRIVE Intelligence.* [En línea] 2012. [Citado el: 21 de Diciembre de 2012.] www.THRIVEintelligence.com .
Video analytics and preemptive surveillance. **VideoSurveillance.com. 2010.** Portland : VideoSurveillance.com, 2010.

ANEXOS

Anexo 1: Descripciones de los casos de Uso

Descripción del caso de uso Capturar flujo de video y extraer fotogramas.

CU#2	Capturar flujo de video y extraer fotogramas.
Actores:	
Resumen	El flujo ve video proveniente de la cámara IP escogida es descompuesto en las imágenes que lo forman.
Referencias	RF#1
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
Escenario Captura de flujo y extracción de fotogramas.	
	1- Se obtiene la dirección del video. 2- Con la dirección del video se hace una petición del flujo de video a la cámara. 3- Si la cámara brinda flujo se crea una estructura que referencia el flujo de video. 4- Se extraen los fotogramas del flujo de video.
Flujo Alternativo de Eventos Paso 3	
Acción del Actor	**Respuesta del Sistema**
	3- Si la cámara no brinda flujo de video se muestra un mensaje indicándole al usuario.

Descripción del caso de uso Procesar fotograma.

CU#3	Procesar fotograma.

Actores:	
Resumen	Se sustrae el fondo para obtener los objetos en movimiento que aparecen en la escena y se elimina el ruido.
Referencias	RF#2
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
Escenario Procesamiento del fotograma.	
	1- Se crea un modelo de fondo gaussiano. 2- Se actualiza el modelo de fondo. 3- Se obtiene la máscara de frente. 4- Es segmentada la imagen para obtener los objetos. 5- Los objetos muy pequeños son eliminados.

Descripción del caso de uso Detectar manipulación.

CU#2	Detectar manipulación.
Actores:	
Resumen	Con el frente y el fondo obtenidos se aplican algoritmos que detectan si la cámara ha sido movida, tapada o desenfocada.
Referencias	RF#3, RF#4, RF#5
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
Escenario Detectar manipulación.	
	1- El sistema realiza las siguientes secciones. • Detectar movimiento de la cámara. • Detectar desenfoque de la cámara.

	• Detectar tapado de la cámara.
Sección Detectar movimiento de la cámara.	
Acción del Actor	**Respuesta del Sistema**
	1- Se obtiene una imagen del modelo del fondo anterior al modelo actual. 2- Se compara el modelo anterior con el actual. 3- Si hay cambios significativos en la cantidad de puntos del modelo anterior y el actual se detecta movimiento de la cámara. 4- Se envía una alerta de que hubo movimiento.
Flujo alterno Sección: Detectar movimiento de la cámara Paso 3	
Acción del Actor	**Respuesta del Sistema**
	3- Si no hay cambios significativos se realizan los pasos de nuevo.
Sección Detectar desenfoque de la cámara.	
Acción del Actor	**Respuesta del Sistema**
	1- Se obtiene una imagen actual y su modelo del fondo. 2- Se extraen los bordes a la imagen y al modelo de fondo. 3- Se comparan si existe una cantidad alta de pixeles que representan bordes en el modelo de fondo y una cantidad baja de pixeles que representan bordes en la imagen. 4- Si la diferencia de los pixeles es alta se envía una alerta de que esta

	desenfocada la imagen.
Flujo alterno Sección: Detectar desenfoque de la cámara Paso 4	
Acción del Actor	**Respuesta del Sistema**
	4- Si no hay diferencia en la cantidad de bordes se realizan los pasos de nuevo.
Sección Detectar tapado de la cámara.	
Acción del Actor	**Respuesta del Sistema**
	1- Se obtiene los histogramas de la imagen actual y su modelo de fondo. 2- Se comparan los histogramas para ver los cambios de la distribución de colores. 3- Si existen grandes cambios en la distribución de colores se envía una alerta de que la cámara ha sido tapada.
Flujo alterno Sección: Detectar tapado de la cámara Paso 3	
Acción del Actor	**Respuesta del Sistema**
	3- Si no hay diferencia en la distribución de colores se realizan los pasos de nuevo.

GLOSARIO

Biblioteca: Colección de clases y funciones.

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

Desenfoque del lente: La imagen que se visualiza es difícil de distinguir, con poca nitidez. Los bordes de los objetos se verán afectados.

Ffmpeg: Biblioteca de código abierto que se utiliza para grabar, convertir, y hacer Streaming de audio y video.

Fotogramas: Imágenes contenidas en un video.

***Frame*:** Imágenes que componen un video.

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

Histograma: Gráfica que muestra la frecuencia de los datos, en la que el eje horizontal representa unidades discretas, en tanto que el eje vertical representa la frecuencia.

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

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

Modelo de fondo: La imagen discrimina los objetos en movimiento y los objetos estáticos en un video. Los objetos estáticos van a ser el fondo de la imagen.

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

Video Sensor: Herramienta de procesamiento inteligente de video, que contiene algoritmos implementados, el video sensor apoya a los guardias de seguridad durante la vigilancia.

Sistema de Vigilancia: Un sistema informático que, controla un grupo de cámaras conectadas en un circuito cerrado, controlados por un vigilante.