Universidad de las Ciencias Informáticas

“Facultad 6”

“Video sensor para el seguimiento y estimación de velocidad de vehículos.”

Trabajo de Diploma para optar por el título de Ingeniero Informático.

Autor: Ernesto Antonio Herrera Collazo.

Tutor: Ing. Fernando Echemendia Tour.

Co-tutor: Ing. Reinier Pupo Ruiz.

PENSAMIENTO

"Codifica siempre como si la persona que finalmente mantendrá tu código fuera un psicópata violento que sabe dónde vives"

Martin Golding

DEDICATORIA

A mis padres por su apoyo incondicional, guiarme en la vida y sobre todo enseñarme a tomar mis propias decisiones.

A mis hermanas Lissanca y Miryelky por su comprensión y apoyo.

A mi sobrino Ángel David que siempre ha creído en mí y ve un ejemplo a seguir cada día.

A Quiñones que es otro padre para mí.

A mis amigos de siempre Adriel, Yunio, JJ, Miguel y Frank. Y sus familias que también son la mía, además de estar siempre presentes en mí vida.

A Damay mí amiga y esposa, en las buenas y en las no tan buenas durante estos cinco años. Gracias por existir y ser parte de mi vida.

AGRADECIMIENTOS

A mis padres por su educación y amor.

A mi familia, por su amor.

A mis nuevos amigos Reinel, Mariemy, Ruby, Dailin, Lieter, Yasmany, Ada, Walfrido, Miriam, Tailén, Yosvany y Ulises por su amistad y tantos gratos momentos inolvidables que hemos pasado juntos.

A mis tutores Fernando y Pupo, por su apoyo.

A mis primos Loreto, Luis Raúl, Adolfo y Roberto.

A Caridad por sus consejos.

A mis tíos Herminia, Nancy, Elida, Kinka y Adonis por creer en mí.

DECLARACIÓN DE LA AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Geoinformática y Señales Digitales de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

“Ernesto A. Herrera Collazo” Ing. Fernando Echemendia Tour

_____ _____

DATOS DE CONTACTO

Ing. Fernando EchemendiaTourt

Correo electrónico: fechemendia@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en el año 2008. Se desempeña como desarrollador e investigador en el proyecto Video Vigilancia (SURIA) del departamento de Señales Digitales del Centro Geoinformática y Señales Digitales (GEySED).

Ing. Reinier Pupo Ruiz.

Correo electrónico: rpupo@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en el año 2008. Se desempeña como desarrollador y el responsable del módulo Video Sensores, en el proyecto Video Vigilancia (SURIA) del departamento de Señales Digitales del Centro Geoinformática y Señales Digitales (GEySED).

RESUMEN

Con el progresivo aumento de la tecnología digital en cuanto a almacenamiento y transmisión por red de la información visual, ha aumentado considerablemente el progreso y ejecución de sistemas de seguridad basados en cámaras digítales.

El Centro de Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI) se encuentra el sistema de Video Vigilancia Suria, que en la actualidad se le está incorporando un nuevo módulo de video sensores, el cual permitirá disponer de funcionalidades como detección de objetos abandonados, conteo de personas, protección perimetral o alambrado virtual y el seguimiento y estimación de velocidad de vehículos.

Este documento recoge un estudio sobre sistemas de video sensores que realizan el seguimiento y estimación de velocidad de vehículos. En el mismo quedan plasmadas las características de las herramientas usadas, como la metodología de desarrollo, la herramienta CASE, el lenguaje de programación, la librería y el IDE (Entorno de Desarrollo Integrado) utilizado.

Asimismo se realizó una investigación que arrojó como resultado los algoritmos más factibles para la implementación del video sensor. Además se recoge la validación de la solución mediante los casos de prueba. Se obtuvo como resultado un video sensor para el seguimiento y estimación de velocidad de vehículos, perfeccionando así el sistema de Video Vigilancia Suria.

PALABRAS CLAVES:

Cámaras IP, Mezcla de Gaussianas, Video sensor, Video Vigilancia.

ABSTRACT

With the progressive increase of digital technology in storage and network transmission of visual information, has greatly increased the progress and implementation of security systems based on digital cameras.

The Centre for Geoinformatics and Digital Signal (GEYSED) at the University of Informatics Sciences (UCI) is the Suria Video Surveillance System, which is currently being incorporated into a new module of video sensors, which will provide functionality as abandoned object detection, people counting, virtual fencing or perimeter protection and monitoring and vehicle speed estimation.

This document presents a study on video systems that track sensors and vehicle speed estimation. At the same are embodied the characteristics of the tools used, such as the development methodology, CASE tool, programming language, library and IDE (Integrated Development Environment) used.

It also conducted an investigation that resulted in the algorithms more feasible to implement the video sensor. It also includes the validation of the solution through the test cases. The result was a video sensor for monitoring and estimating vehicle speeds, thereby improving the Suria Video Surveillance system.

KEY WORDS:

IP cameras, video sensor, Video Surveillance, Mixture of Gaussians.

Índice

Índice de figuras

Índice de tablas

INTRODUCCIÓN

En la actualidad, con el creciente desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), la humanidad ha presenciado unas de las mayores revoluciones tecnológicas en la historia. El sector audiovisual no se encuentra ajeno a ella, siendo una de las mayores tendencias el desarrollo de soluciones informáticas para el sector audiovisual. Existen diversos factores que propician este avance, pero sin duda alguna, el movimiento a la tecnología digital, el avance en cuanto a codificación, almacenamiento, disponibilidad y transmisión a través de la red de información visual, ha aumentado considerablemente el auge de sistemas de seguridad basados en cámaras digítales.

Los sistemas de video vigilancia han evolucionado gradualmente, cruzando por diferentes generaciones en las últimas décadas y aprovechando el creciente desarrollo de la tecnología digital.

La primera generación de sistemas de vigilancia basada en video empleaba señales y transmisiones analógicas. En estos sistemas el factor humano era el encargado de realizar el análisis de las secuencias del video presentadas en uno o varios monitores situados en una sala de control remoto, donde las escenas monitorizadas por diversas cámaras son multiplexadas y se presentan en un orden periódico predefinido. Adicionalmente, la vigilancia por video tradicional precisa gran cantidad de espacio de almacenamiento. Todo lo que captura una cámara de seguridad, o se guarda en un archivo de video o se sobrescribe periódicamente. Este procedimiento limita la duración de video que puede guardarse y hace que el tiempo necesario para su revisión, sea elevado. (1)

La segunda generación de sistemas de vigilancia se basa, principalmente, en métodos de procesamiento y comunicación híbridos analógico-digitales, o completamente digitales. Aprovechando la flexibilidad ofrecida por los primeros algoritmos de procesado de video que permiten centrar la atención del operador humano en un grupo de situaciones de interés y además, las facilidades proporcionadas por los primeros métodos de compresión digital para aprovechar el ancho de banda de transmisión.(1)

Actualmente, se está produciendo una migración de los sistemas de video vigilancia clásicos a los sistemas de tercera generación, estos aprovechan el progreso de las

redes de ordenadores de bajo coste y alto rendimiento, y las comunicaciones multimedia fijas y móviles. La investigación en este campo trabaja en técnicas distribuidas de procesamiento de video. Esta tercera generación de soluciones de video vigilancia utiliza recursos existentes para transformarlos en un sistema inteligente con la introducción de los videos sensores, permitiendo que cada cámara tenga su propio procesamiento de la información, asegurando de esta manera la calidad y confiabilidad de la información brindada.

Los sistemas de video vigilancia se pueden encontrar en varios entornos urbanos, exteriores e interiores de centros comerciales, hoteles, bibliotecas, museos y control de tráfico en las autopistas, proporcionando diversos usos. Estos sistemas en el mundo son de un alto costo adquisitivo, por lo que Cuba aunque los necesita solo los posee en algunas entidades. En el panorama nacional, se tiene la empresa **¡Error! No se encuentra el origen de la referencia.** que comercializa una solución de video vigilancia llamada Xyma Safe Vision, pero al ser socios de la compañía **¡Error! No se encuentra el origen de la referencia.**, su precio aumenta demasiado, llegando a ser casi tan cara como las que brindan otras compañías a nivel mundial.

En el Centro de Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI) se desarrolla un sistema de Video Vigilancia, que en la actualidad cuenta con los módulos siguientes: Visor, Grabador, Recuperador, Gestor y Módulos de Diseños Web y se le está incorporando un nuevo módulo de video sensores, el cual permitirá disponer de funcionalidades como detección de objetos abandonados, conteo de personas, protección perimetral o alambrado virtual y el seguimiento y estimación de velocidad de vehículos.

Se quiere hacer énfasis en este último aspecto por la necesidad de perfeccionar el sistema de Video Vigilancia Suria, para que este pueda desempeñarse exitosamente en el monitoreo de tráfico en autopistas, ya que no cuenta con herramientas para realizar este monitoreo. En este último aspecto es válido aclarar que se disminuirán recursos, puesto que todo este proceso de monitoreo de tráfico actualmente se realiza de forma manual en las autopistas nacionales.

La situación problemática anteriormente expuesta permite plantear el siguiente **problema a resolver:** ¿Cómo mejorar el control de los entornos de monitorización de vehículos, en flujos de video obtenidos de cámaras IP?

Teniendo como **objeto de estudio:** las técnicas de procesamiento de flujos de video digital. Se define como **campo de acción:** los videos sensores para el seguimiento y estimación de velocidad de vehículos en flujos de videos obtenidos de cámaras IP.

Teniendo en cuenta como **objetivo general** de la investigación: desarrollar un video sensor que procese los flujos de video obtenidos de cámaras IP, siendo capaz de realizar el seguimiento y estimación de la velocidad de vehículos.

Para el cumplimiento del objetivo propuesto se plantean las siguientes **tareas de la investigación**:

- Analizar el estado del arte asociado a la Video Vigilancia y los videos sensores que realice la detección, seguimiento y estimación de la velocidad de vehículo, a través de un flujo de video obtenido de cámaras IP.
- Definición de los procesos relacionados con el video sensor que realice el seguimiento y estimación de la velocidad de vehículo, a través de un flujo de video.
- Definir las tecnologías, algoritmos y librerías a utilizar para el desarrollo del componente seguimiento y estimación de velocidad en vehículos, en flujos de video obtenidos de cámaras IP.
- Generar toda la documentación asociada a la investigación.
- Implementación el video sensor para el seguimiento y estimación de velocidad de vehículos.
- Implementar un demo con el objetivo de validar el funcionamiento del video sensor.

Teniendo como **idea a defender** que:

Si se desarrolla un video sensor, que permita el seguimiento y estimación de velocidad de vehículos, se mejorará el control de los entornos de monitorización, permitiendo perfeccionar el Sistema de Video Vigilancia Suria.

Métodos de Investigación

Métodos teóricos.

- Analítico–sintético: Se utilizará para estudiar el problema con mayor profundidad para dar la solución adecuada al problema científico, comprender

sobre la evolución y desarrollo que han tenido los sistemas de grabación de video, lo que facilitará el entendimiento en el tema de los videos sensores para el seguimiento y estimación de velocidad de vehículos y así extraer las características generales de cada uno de los estudiados.

Métodos Empíricos.

- Observación: permitirá realizar valoraciones y obtener información a partir de la observación. Esto se manifiesta cuando se realizan observaciones a las funcionalidades de sistemas similares al que se está desarrollando, lo que da una visión de cómo debe ser el sistema.

El trabajo de diploma está estructurado de la siguiente forma:

- **Capítulo 1:** Fundamentación teórica. Se realizará la fundamentación teórica que justifica la investigación, se analizará el estado actual del tema a tratar a nivel nacional e internacional, así como las nuevas tendencias, las tecnologías y metodologías que se usarán en la solución.
- **Capítulo 2:** Características del sistema. Es este capítulo se definirán las características del sistema. Al no ser identificados claramente los procesos de negocio se plantea la conceptualización del entorno mediante un modelo de dominio. Se analizan y relacionan los conceptos y entidades que están presentes donde funcionará el sistema. Se identifican los requisitos funcionales y no funcionales con los que contará el sistema. Se analizan y detallan los casos de uso que tendrá el sistema y se describen los actores.
- **Capítulo 3:** Análisis y diseño del sistema. En este capítulo se definen las clases de análisis y diseño de los casos de usos; así como los diagramas de secuencias cumpliendo con las descripciones de los casos de uso. Se presenta la arquitectura del sistema, describiendo, además, las clases entidades y las controladoras del flujo de trabajo análisis y diseño.
- **Capítulo 4:** Algoritmos. En este capítulo se explica en detalles los diferentes algoritmos que se emplean en la realización del “Video sensor para el seguimiento y estimación de velocidad de vehículos.”
- **Capítulo 5:** Implementación y pruebas. En este capítulo los elementos del modelo del diseño se implementan en términos de componentes como son los ficheros de código fuente, ficheros de código binario y ejecutable. Además se realizan pruebas al sistema para lograr erradicar errores que puedan ser

introducidos en la implementación del video sensor y para comprobar que el producto final cumple con los requisitos establecidos en la primera fase de la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

El presente capítulo se realizará un breve análisis del surgimiento de los videos sensores para el seguimiento de vehículos y su definición, así como los conceptos asociados al tema, además de una resumida descripción de las principales herramientas y tecnologías imprescindibles para el desarrollar un buen video sensor para el seguimiento de vehículos.

1.1 Conceptos asociados al dominio del problema

Para mejor comprensión y una visión general de la investigación de hace necesario estudiar varios conceptos y características que crean los cimientos de la presente investigación. Todas estas unidades cognitivas son utilizadas directa o indirectamente durante el desarrollo del trabajo de diploma.

1.1.1 Video Vigilancia

Se denomina Video Vigilancia a la vigilancia a través de un sistema de cámaras, fijas o móviles. (2)

La Video Vigilancia es el tratamiento de una o varias cámaras, que realicen operaciones y procedimientos automatizados o no, y que permita además la elaboración, almacenamiento, modificación, cancelación y transferencias, de un flujo de imágenes obtenidas de dichas cámara.

Figura 1: Esquema de un Sistema de Video Vigilancia

1.1.2 Cámara IP

Una cámara IP (o cámara de red) puede describirse como una cámara y un ordenador combinados para formar una única unidad inteligente. Captura y envía vídeo en directo directamente a través de una red IP, como una LAN, Intranet o Internet, permite a los usuarios ver o gestionar la cámara con un navegador Web estándar o con software de gestión de vídeo en cualquier equipo local o remoto conectado a una red. Permite a usuarios autorizados de distintas ubicaciones acceder simultáneamente a las imágenes captadas por la misma cámara IP de red. (3)

Figura 2: Cámara IP.

1.1.3 Video sensor

Es una herramienta de análisis de video digital que ofrece información significativa proveniente de una secuencia de video. El desarrollo de video sensores tiene una base en el procesamiento digital de la imagen. (1)

1.1.4 Tecnología IP

Emplea el protocolo IP para la comunicación entre las unidades que componen el sistema.

1.1.5 Protocolo de Internet

Protocolo de Internet o IP es un protocolo no orientado a conexión, usado tanto por el origen como por el destino para la comunicación de datos, a través de una red de paquetes transformados, no fiable y de mejor entrega posible sin garantías.

1.1.6 Protocolo TCP / IP

TCP/IP es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos. El conjunto TCP/IP está diseñado para enrutar y tiene un grado muy elevado de fiabilidad, es adecuado para redes grandes y medianas, así como en redes empresariales. Se utiliza a nivel mundial para conectarse a Internet y a los servidores web. Es compatible con las herramientas estándar para analizar el funcionamiento de la red.

1.2 Técnicas, algoritmos y tendencias actuales

Actualmente los sistemas de conteo de vehículos son muy caros y de alta complejidad para su implementación, como pueden ser los de sensores ubicados debajo de la carretera como los Lazos Inductivos, los Sensores Piezoeléctricos, sensores en base a cables de fibra óptica, entre otros. Por lo que el mantenimiento de estos sistemas es más caro que el de una simple cámara de video.

En el panorama internacional existen proyectos comerciales y cuantiosos artículos científicos en diferentes publicaciones que han determinado una base importante en el proceso de investigación y desarrollo de este trabajo de diploma. Los sistemas basados en visión sobre los sensores comerciales presentan estos tipos de ventajas: mantenimiento menos costoso, ofrecen mayor variedad de datos de tráfico, solución más económica, acceso a las grabaciones de videos, entre otras.

1.2.1 Sistemas Comerciales.

XYMA SAFE VISION

Es un software de video vigilancia profesional basado en tecnología IP, adaptable a una gran cantidad de entornos, flexible, escalable y que admite también el uso de tecnologías analógicas. Monitorea y controla en tiempo real y de forma histórica cada

uno de los movimientos que ocurren en las áreas sensibles que se identifiquen. Las posibilidades del sistema están distribuidas para darle flexibilidad y su arquitectura modular lo convierte en un potente sistema que permite administrar, monitorear, grabar, revisar, configurar alertas y alarmas y obtener reportes, para cualquier solución de video vigilancia que se requiera implementar. (4)

Figura 3: DATYS

Traficon

Traficon es la empresa de referencia en la detección de tráfico basada en el procesamiento de imágenes de video. Propone una alta gama única de aplicaciones, lo cual lo avalan sus años de experiencia en el tema, además de la gran diversidad de entornos a los que se adapta.

El factor clave en un sistema de detección de Traficon es el procesador de imagen de video (VIP), un tablero de detector de serie en el que varios tipos de software de detección se puede ejecutar. La señal de video desde la cámara de vigilancia del tráfico se utiliza como entrada para la unidad de detección.

Los detectores que emplea brindan datos de tráfico (velocidad, volumen, entre otros) que pueden ser utilizados para otros fines como: detección automática de incidentes (detección rápida de vehículos detenidos o conductores suicidas) y control de flujo (control exacto del promedio de la velocidad de flujo ayuda a distinguir diferentes niveles de servicio). (5)

Figura 4: Vídeo detección de tráfico

1.2.2 Técnicas de seguimiento de automóviles

Sistema de monitorización y control del tráfico en carretera.

(Jorge E. Faytong, 2009) propone una solución de detección de vehículos a través de visión por computador. Las principales etapas son:

Frame de Entrada: En esta fase se obtuvo un frame del video .avi bajo estudio. La aplicación trabaja con frames en escala de grises reduciendo de esta forma el costo computacional.

Resta de Fondo: En esta fase se realiza una segmentación de la imagen de entrada. Por otra parte se clasifica qué pixeles pertenecen a un primer plano y cuáles corresponden al fondo. Este análisis se realiza mediante la obtención de parámetros estadísticos de los pixeles.

Umbralización: Se umbraliza la imagen obtenida luego de aplicar la Resta de Fondo y se aplican operadores morfológicos. Aquellos pixeles que pertenezcan al fondo se les dará un valor 0, y los que sean de primer plano un valor 1 (255).

Clustering: Se agrupan los distintos pixeles según condiciones de proximidad y de vecindad entre pixeles. Lo que se pretende es obtener de entre todos los pixeles correspondientes a un primer plano, cuáles de ellos pertenecen a un mismo cluster y de ese modo a partir del número de cluster que se tengan poder obtener una idea aproximada del número de vehículos en la imagen. También se realizan operadores de limpieza de cluster.

Tracking: Se realiza un seguimiento a cada uno de los vehículos detectados. Se realiza un filtrado de la posición en 2D sobre la que se construye el modelo 3D utilizado el Filtro de Kalman para conseguir suavizar los resultados finales.

Modelado 3D: Para cada uno de los clusters obtenidos, se realiza un modelado 3D. Se consideran conocidas las dimensiones de los vehículos en 3D, así como la posición y la calibración estimada de la cámara en escena. Por lo tanto, a partir de un punto de cada uno de los clusters en 2D, se proyectará ese en 3D y se construirá el modelo en 3D a partir de ese punto, para posteriormente volverlo a proyectar en 2D.

Visualización: Finalmente se muestran los resultados de todo el proceso anterior en una nueva imagen.

Esquema del algoritmo:

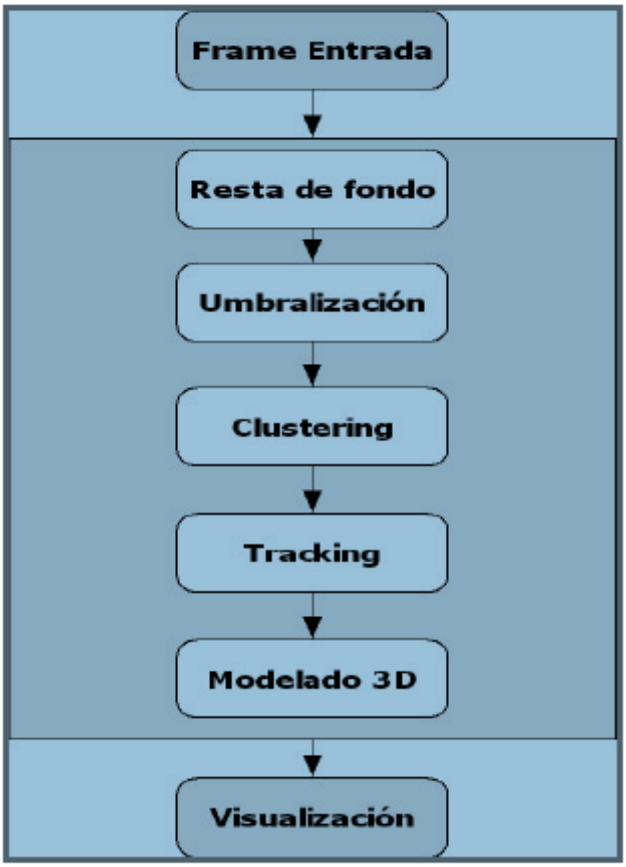

Figura 5: Esquema del algoritmo.

Conteo automático de vehículos

En este artículo se tuvo en cuenta, en primer lugar, el estudio de los siguientes algoritmos:

Mezcla de Gaussianas (MoG): En este algoritmo, el fondo no se modela por los valores de una imagen, sino a través de un modelo paramétrico cuyo objetivo es aproximar una función de distribución a los últimos valores de cada píxel. Cada píxel localizado en una posición (x, y) de la imagen de fondo en el instante t, se representa por un número K de distribuciones Gaussianas cuya combinación constituye una función de densidad de probabilidad. La Mezcla de Gaussianas es uno de los métodos complejos más utilizado en la literatura. Entre sus ventajas destaca la robustez, ya que

permite modelar fondos multimodales, es decir, manejar múltiples modos de distribución (o tipos de movimiento). (6)(7)

Extracción automática de características de vehículos en movimiento a partir de videos basada en la red neuronal de Kohonen.

En este artículo se emplea el procesamiento de imágenes y redes neuronales. La estructura del sistema está formada por un detector de objetos en movimiento y de un extractor de características de vehículos. Para la detección de objetos en movimiento se utilizó una técnica de substracción de fondo. Para la extracción de características se implementó un filtro Sobel y aprendizaje a través de una red neuronal de Kohonen. Este sistema cuanta con un subsistema diseñado para detectar vehículos en movimiento y de un subsistema para la extracción de características.

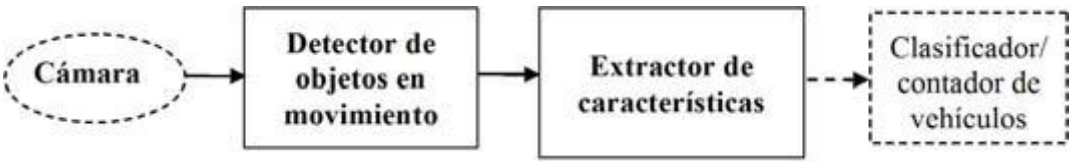

Figura 6: Subsistemas desarrollados.

Consideraron la necesidad de adicionar filtros para disminuir el ruido causado por el dispositivo de captura, por eso los siguientes bloques que conforman el sistema superan los inconvenientes de la técnica de la substracción de fondo, y los falsos movimientos detectados por las variaciones de iluminación, movimientos de ramas y sombras. La siguiente figura muestra el procesamiento del detector de movimiento:

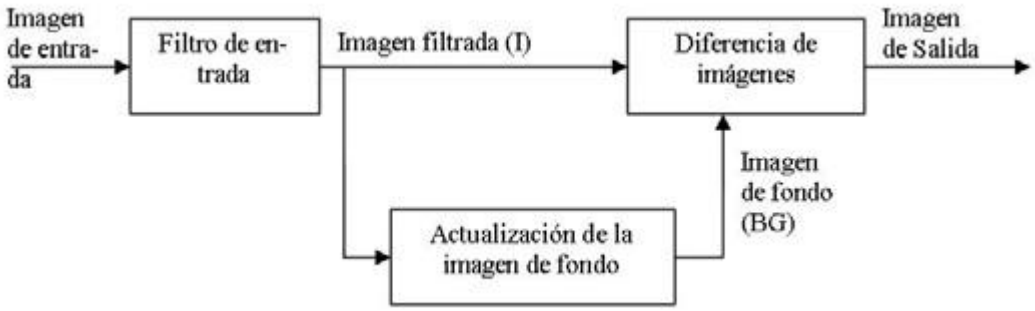

Figura 7: Procesamiento del detector de movimiento.

A continuación se muestra un diagrama del algoritmo empleado para cada pixel de la imagen del bloque para actualizar la imagen de fondo:

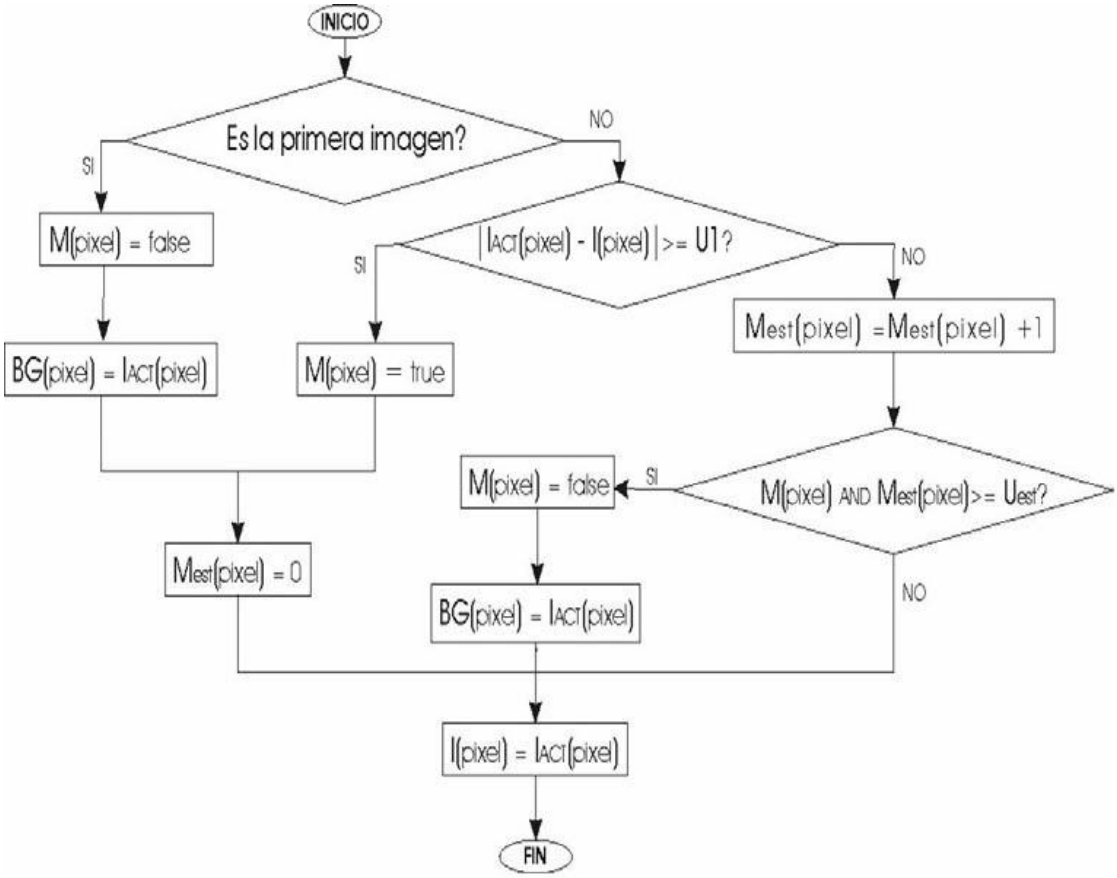

Figura 8: Algoritmo empleado para cada pixel de la imagen.

Luego se pasa a la extracción de características, que para esto le fue adicionado un identificador de regiones entre el filtro de Sobel y la red de Kohonen.

Los sistemas que se mencionaron anteriormente cuentan con disímiles funcionalidades para la video vigilancia, utilizando los videos sensores. En todo el mundo se está abogando por el software libre y esta es la estrategia que se está llevando en la UCI, en el proyecto productivo Sistema de Video Vigilancia Suria para que el país alcance una mayor soberanía tecnológica y potenciar un mayor desarrollo autóctono en el campo de la informática. Aunque es factible comentar que el estudio de todas estas soluciones antes mencionadas forma una base sólida para el proceso de desarrollo que se está llevando a delante en la universidad. Se tomaron de cada uno de estos algoritmos estudiados las características más relevantes para llevar a cabo la solución del problema

1.3 Metodologías, herramientas y tecnologías a utilizar en el desarrollo de la solución.

Todo desarrollo de software necesita de una metodología, un conjunto de pasos o procedimientos para guiar su realización. Todo esto contribuye a mantener el proceso de realización del producto ordenado desde su inicio hasta su fin. Hoy, seguir una metodología de desarrollo garantiza mantenerse en competencia, pues el mercado de software es muy amplio variable y competitivo.

1.3.1 El Proceso Unificado de Desarrollo de Software (RUP).

Racional Unified Process (RUP) o Proceso Unificado de Software es una metodología tradicional o pesada. Es utilizada principalmente en proyectos grandes y con requisitos de poca variación. Tiene como objetivo lograr un producto de máxima calidad que cumpla con las necesidades planteadas por el usuario en tiempo y con un presupuesto acordado con anterioridad. Realiza un modelado visual de software y permite gestionar una potente documentación y control de cambios.

RUP es una metodología adaptable a las necesidades de la empresa. Los roles están bien definidos y en caso de ser un proyecto muy grande participan varios equipos con una comunicación bien fluida. El proceso de desarrollo RUP lo divide en cuatro fases, dentro de las cuales se van realizando varias iteraciones, según el proyecto y la importancia de cada actividad que se realiza.

RUP posee tres características esenciales que lo distinguen de las restantes metodologías y lo hacen único:

Dirigido por casos de uso: Los casos de uso son el claro reflejo de lo que el cliente planteó que deseaba en la modelación del negocio a través de los requerimientos. A partir de que aparecen los casos de uso, estos pasan a guiar el proceso de desarrollo, ya que los modelos que se obtienen como resultado de la realización de los flujos de trabajo están presentados en los casos de uso.

Iterativo e incremental: Todas las fases se desarrollan mediante iteraciones. Una iteración incluye actividades de todos los flujos de trabajo, las cuales va refinando en cada iteración.

Centrado en la arquitectura: La arquitectura muestra una visión común del sistema completo. RUP se desarrolla mediante iteraciones, dando prioridad a los casos de uso arquitectónicamente significativos.

1.3.2 Extreme Programing (XP)

Extreme Programing (XP) o Programación Extrema, se encuentra dentro de las metodologías ágiles o ligeras. Está centrada principalmente en las relaciones interpersonales como clave para el éxito del software. Se enfoca en el trabajo en equipo, donde clientes y desarrolladores trabajan de la mano persiguiendo un único objetivo: el desarrollo de un software con calidad. Además, existe una retroalimentación continua entre ellos.

XP se utiliza en proyectos cortos donde los requisitos son inestables e inesperados y las soluciones son poco complejas. El equipo de trabajo suele ser pequeño y existen pocos roles. Está enfocado en tres características principales:

Comunicación: Los desarrolladores se comunican continuamente con los clientes y con el resto de los desarrolladores.

Retroalimentación: Se realizan pruebas desde los primeros días de desarrollo y el producto es entregado a sus clientes tan pronto sea posible.

Simplicidad: El diseño del software es tan simple como sea posible. No se realiza mucho énfasis en la arquitectura.

1.3.3 Fundamentación de la metodología seleccionada

El pasar de los años ha demostrado la necesidad de la utilización de una metodología de software para que el producto tenga una mayor calidad, exista más organización en el trabajo y el sistema sea puesto en práctica en el tiempo acordado. Después de la realización de un análisis detallado de las metodologías RUP y XP se determinó que la más apropiada para modelar el sistema es RUP, por las siguientes razones:

- RUP es una metodología que tiene la particularidad de que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

- RUP no necesita que el cliente forme parte del equipo de desarrollo al contrario de XP y en este caso es muy beneficioso ya que no hay un cliente específico para el sistema.
- RUP realiza las pruebas al final del producto al contrario de XP que está constantemente realizando pruebas.
- La gran cantidad de artefactos que se generan en RUP contribuyen a un mejor entendimiento del problema.
- Es una metodología con alta adaptabilidad a las condiciones reales del desarrollo del sistema. Es decir que se puede hacer más ágil según se necesite.
- RUP solamente implementa lo se encuentra en los casos de uso.

1.3.4 Herramientas CASE

➢ Enterprise Architect 7.0

Es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambio. Entre las principales características que este brinda se pueden mencionar:

- Crea elementos del modelo UML para un amplio alcance de objetos.
- Ubica dichos elementos en diagramas y paquetes.
- Documenta los elementos que ha creado.
- Genera código para el software que está construyendo.
- Realiza ingeniería directa e inversa de código en lenguajes como Action Script, C++, C#, Java, PHP, entre otros.

Soporta todos los diagramas y modelos del UML. Puede modelar procesos de negocio, sitios web, interfaces de usuario, redes, configuraciones de hardware, mensajes y más. Estima el tamaño del proyecto en esfuerzo de trabajo en horas. Captura y traza requisitos, recursos, planes de prueba, solicitudes de cambio y efectos.

➢ Rational Rose Enterprise Edition

DsdsdRational Rose Enterprise Edition es el producto más completo de la familia Rational Rose, incluye, al igual que el resto de los productos de esta familia, soporte UML. Es una herramienta propietaria de diseño de software, que es encargada de llevar a cabo la automatización de los sistemas para la posterior generación de código, o sea, realiza la modelación visual y construcción de componentes de dichos sistemas.

Permite el modelado en UML para diseñar BD, que integra los requisitos de datos y aplicaciones mediante diseños lógicos y analíticos, proporciona al equipo de desarrollo un lenguaje común de modelado, que facilita la rápida creación de un software de calidad, incluye funciones de visualización, modelado y herramientas para desarrollar aplicaciones Web, posibilita la integración con otras herramientas de desarrollo de IBM Rational, permite la generación de código a partir de modelos Ada, ANSI C++, C++, CORBA y Visual Basic

Es reconocida como líder en su campo, está compuesta por varias herramientas que cubren todos los aspectos de la ingeniería y documentación de cualquier tipo de proyecto utilizando UML. Es bastante cara y poco intuitiva de trabajar.

➢ Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Presenta la desventaja de que requiere bastante RAM debido a estar desarrollado en JAVA y tiene problemas de integración con otras herramientas de desarrollo.

➢ Enterprise Architect 7.0 como herramienta propuesta para la modelación del sistema.

Luego de una investigación sobre algunas de las herramientas CASE se decidió que la más factible para realizar el sistema es el Enterprise Architect (EA) ya que es una

herramienta completa de análisis y diseño UML, que cubre el desarrollo de software desde la concepción de las exigencias, a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Permite gestionar todos los aspectos importantes del diagrama, sus clases, métodos, atributos, estereotipos y modo de acceso. EA está diseñada para construir software robusto y conservable. Además despliega documentación de salida flexible y de alta calidad. Puede modelar procesos de negocio, sitios web, interfaces de usuario, redes, configuraciones de hardware y mensajes. Estima el tamaño del proyecto en esfuerzo de trabajo en horas. Captura y traza requisitos, recursos, planes de prueba, solicitudes de cambio y defectos.

1.3.5 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta

El lenguaje unificado de modelación (UML por sus siglas en inglés) es un lenguaje que permite especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema de software. El lenguaje ha ganado un significante soporte de la industria de varias organizaciones vía al consorcio de socios de UML. Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje con el que está descrito el modelo.

1.4 Lenguajes de programación.

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático. El lenguaje de programación permite a los programadores especificar de forma precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de situaciones. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje léxico. En la actualidad existen disímiles lenguajes de programación cada uno de ellos con características que lo distinguen, a continuación se exponen algunos de ellos como posible lenguajes a utilizar.

1.4.1 Lenguaje C++

En la actualidad, C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de cualquier tipo de aplicación. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original. La evolución de C++ ha continuado con la aparición de Java, un lenguaje creado simplificando algunas cosas de C++ y añadiendo otras, que se utiliza para realizar aplicaciones en Internet.

Hay que señalar que C++ ha influido en algunos puntos muy importantes del ANSI C, como por ejemplo, en la forma de declarar las funciones, en los punteros a void, etc. En efecto, aunque el C++ es posterior al C, sus primeras versiones son anteriores al ANSI C, y algunas de las mejoras de este fueron tomadas del C++.

El C++ es a la vez un lenguaje procedural (orientado a algoritmos) y orientado a objetos. Como lenguaje procedural se asemeja al C y es compatible con él, aunque ya se ha dicho que presenta ciertas ventajas (las modificaciones menores, que se verán a continuación). Como lenguaje orientado a objetos se basa en una filosofía completamente diferente, que exige del programador un completo cambio de mentalidad. Las características propias de la Programación Orientada a Objetos (Object Oriented Programming, u OOP) de C++ son modificaciones mayores que sí que cambian radicalmente su naturaleza. (8)

1.4.2 Lenguaje C#

Este lenguaje es moderno y altamente expresivo. Cuenta con una extensa librería base para el desarrollo de todo tipo de aplicaciones.

Algunas de las principales características de C# son:

Sencillez de uso: Elimina muchos elementos añadidos por otros lenguajes y que facilitan su uso y comprensión. El código en escritorio es auto contenido, lo que significa que no necesita de ficheros adicionales a la propia fuente tales como ficheros de cabecera o ficheros IDL. El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes es compatible (no como e C++), lo que facilita la portabilidad del código.

Modernidad: Al ser un lenguaje de última generación incorpora elementos que se ha demostrado a lo largo del tiempo que son muy útiles para el programador, como tipos

decimales o booleanos, un tipo básico string, así como una instrucción que permita recorrer colecciones con facilidad (instrucción foreach).

Orientado a objetos: Es orientado a objetos. No permite las inclusión de funciones ni variables globales que no estén incluidos en una definición de tipos, por lo que la orientación de objetos es más pura y clara que en otros lenguajes como C++. Soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo.

Orientado a componentes: La propia sintaxis incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular. Las sintaxis de C# incluye por ejemplo, formas de definir propiedades, eventos o atributos.

Las desventajas que se derivan del uso de este lenguaje de programación son que en primer lugar se tiene que tener algunos requerimientos mínimos del sistema para poder trabajar adecuadamente, tales como contar con Windows NT 4 o superior, tener alrededor de 4 Gb de espacio libre para la pura instalación, etc.

1.4.3 Fundamentación del Lenguaje Seleccionado

Luego de realizar un análisis detallado de los lenguajes de programación antes mencionado, se llegó a la conclusión de que el lenguaje seleccionado para el desarrollo del sistema será C++, por ser un lenguaje estandarizado y un mismo código fuente se puede compilar en diversas plataformas, además es muy rápido en cuanto a ejecución, por lo que es de mucha importancia para el trabajo con flujo de videos, y también se integra muy bien con diversos IDE.

1.5 Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado o en inglés Integrated Developer Environment (IDE), es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios.

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. Estas pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

➢ QtCreator

Qt Creator es un entorno de desarrollo (IDE) multiplataforma, para el desarrollo de aplicaciones con las bibliotecas Qt.

Principales características de QtCreator:

- Posee un avanzado editor de código C++.
- Además, soporta los lenguajes: C#/.NET Languages (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- 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.

QtCreator es distribuido bajo tres tipos de licencias: Qt Comercial 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.

➢ KDevelop

El Proyecto KDevelop surgió en 1998 con el fin de desarrollar un IDE fácil de usar en entornos de escritorio (KDE). Desde entonces, el IDE KDevelop está públicamente disponible bajo la licencia GPL y soporta varios lenguajes de programación. A diferencia de muchas otras interfaces de desarrollo, KDevelop, no cuenta con un compilador propio, por lo que depende de gcc para producir código binario. Cuenta con editor de código fuente con indentado automático (Kate) y la gestión de diferentes tipos de proyectos, como CMake, Automake, qmake (para proyectos basados en la biblioteca Qt y Ant (para proyectos basados en Java). También proporciona completado automático del código en C y C++ y asistentes para generar y actualizar las definiciones de las clases y el framework de la aplicación.

➢ QT Creator como IDE propuesto para la solución del sistema.

Para la implementación del sistema se decidió usar como IDE el QtCreator debido a las múltiples ventajas que ofrece el mismo para la programación con C++, las cuales

se exponen a continuación: Cuenta con una interfaz muy cómoda, que es funcional y estéticamente agradable. Tiene plugins para varios sistemas de control de versiones como Perforce y Subversion. Tiene una herramienta de búsqueda eficaz donde se puede buscar fácilmente las clases, métodos y archivos. El QtCreator cuenta con el QtDesigner que ayuda a diseñar formas de interfaz de usuario e incluye también la gestión y finalización del proyecto de código.

1.6 Librerías a utilizar en el desarrollo del sistema

1.6.1 OpenCV2.1.0

Opencv es una librería de procesamiento de imágenes, de código libre para visión por computador, destinada principalmente a aplicaciones de visión por computador en tiempo real. Es una biblioteca abierta (opensource) desarrollada por Intel, la cual proporciona un alto nivel de funciones para el procesado de imágenes, permitiendo a los programadores crear aplicaciones poderosas en el dominio de la visión digital.

Dentro de la gama de funciones que ofrece, permite operaciones básicas, procesado de imágenes y análisis estructural y de movimiento, reconocimiento de modelo, reconstrucción 3D, detección de rasgos, para rastrear (Flujo Óptico), análisis de la forma (Geometría, Contorno que Procesa), segmentación de objetos, reconocimiento (Histograma), calibración de la cámara, interfaz gráfica y adquisición. Es compatible con Intel Image Preocessing Library (IPL) que implementa algunas operaciones en imágenes digitales.

Los algoritmos están basados en estructuras de datos muy flexibles, acoplados con estructuras IPL; más de la mitad de las funciones han sido optimizadas aprovechándose de la Arquitectura de Intel. En cuanto a análisis de movimiento y seguimiento de objetos, ofrece una funcionalidad interesante. Incorpora funciones básicas para modelar el fondo para su posterior sustracción, generar imágenes de movimiento MHI (Motion History Image) para determinar dónde hubo movimiento y en qué dirección, lo cual es de vital importancia para la realización de un “Video sensor para el seguimiento y estimación de velocidad de vehículos”.

1.7 Conclusiones Parciales

En el presente capítulo se cumplieron los objetivos esperados, dando cumplimiento a las tareas que complementaban el mismo. Se pudo apreciar que existen varios métodos para el procesamiento de videos obtenidos de cámara IP, así como algoritmos que permiten resolver la problemática planteada.

Considerando las tendencias actuales y las políticas de migración hacia el software libre del país, se puede concluir que las herramientas seleccionadas son las óptimas para la realización de la solución que se propone en el trabajo. Además, una vez realizado el capítulo ya están creadas las bases para comenzar el desarrollo del video sensor.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

Es este capítulo se definirán las características del sistema. Al no ser identificados claramente los procesos de negocio se plantea la conceptualización del entorno mediante un modelo de dominio. Se analizan y relacionan los conceptos y entidades que están presentes donde funcionará el sistema. Se identifican los requisitos funcionales y no funcionales con los que contará el sistema. Se analizan y detallan los casos de uso que tendrá el sistema y se describen los actores. Además, se realiza el análisis y el diseño del sistema y de la base de datos.

2.1 Modelo de dominio

RUP considera el modelo de dominio como un subconjunto del modelo de negocio. El modelo de dominio se realiza cuando los procesos del negocio no están claros debido a que no se conocen sus orígenes o simplemente son sucesos o eventos. Además, no es posible identificar los trabajadores del negocio debido a que existe una sobrecarga de responsabilidades y es muy difícil establecer las reglas de funcionamiento del sistema a implementar.

El modelo de dominio no es más que la representación visual de los objetos y conceptos de la entidad donde se realizará el sistema. Cuando se usa este modelo se capturan los objetos y eventos más importantes y no importa quién es el responsable de realizar las actividades debido a que los procesos no están bien definidos.

El modelo de Dominio se representa, usando el del lenguaje de modelado UML, a través de un Diagrama de Clases donde se muestran conceptos u objetos del dominio del problema (clases conceptuales), sus relaciones y atributos.

2.1.1 Conceptos y eventos principales asociados al entorno.

Vehículo: Medio de transporte de personas u objetos.

2.1.2 Diagrama de Clases del Modelo de Dominio

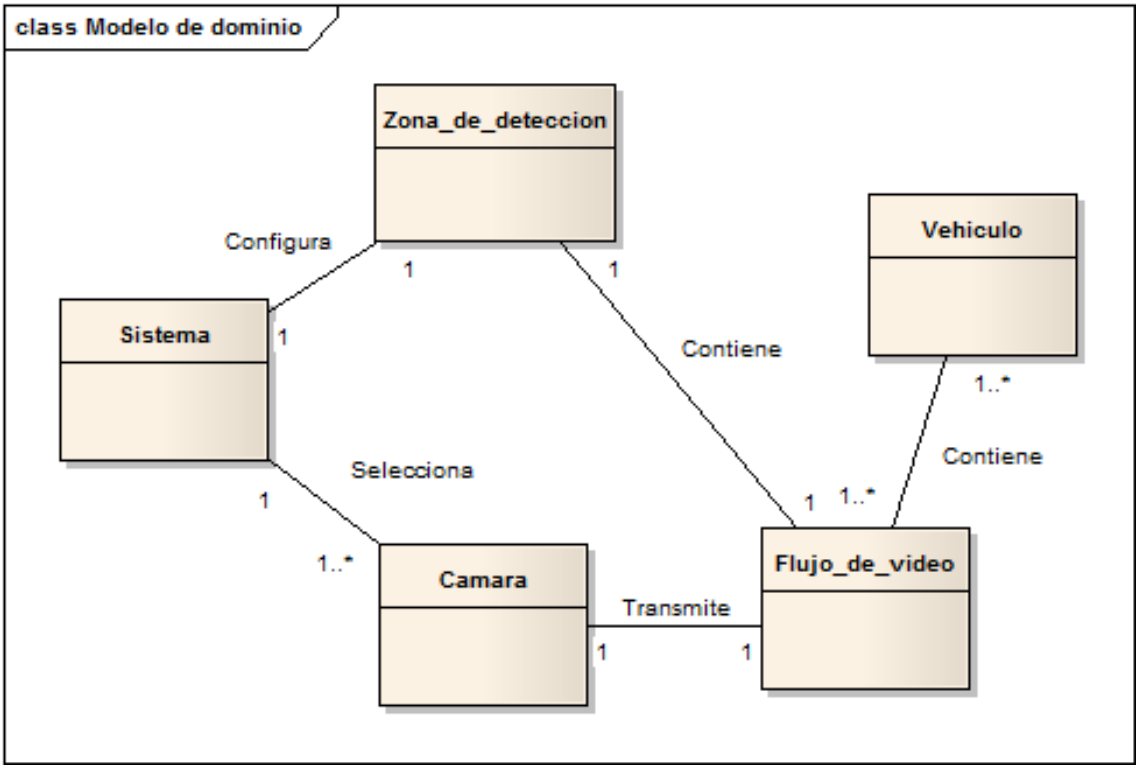

Figura 9: Diagrama de Clases del Modelo de dominio.

En la figura anterior el sistema selecciona una cámara, de la cual se obtiene un flujo de video, luego el sistema crea una zona de detección en el flujo de video. En esta zona de detección se encuentran los objetos de interés (vehículos).

2.2 Requisitos funcionales del sistema

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. De acuerdo con los objetivos propuestos en el trabajo, el sistema debe ser capaz de:

RF1 Analizar un flujo de video.

RF2 Definir zona donde se va a realizar el cálculo de velocidad.

RF3 Seguir Vehículos.

RF4 Calcular la velocidad de los vehículos.

RF5 Notificar si la velocidad del vehículo está por encima de lo debido.

2.3 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son características que hacen al producto atractivo, usable, rápido o confiable. Estos requisitos se agrupan en varias categorías.

2.3.1 Usabilidad

Nota: Se usa el prefijo **RNU** en su nomenclatura.

- **RNU 1** Se debe hacer uso de botones, imágenes que indiquen de modo intuitivo la función que realiza.

2.3.2 Fiabilidad

Nota: Se usa el prefijo **RNF** en su nomenclatura.

- **RNF 2** se tendrá en cuenta la recuperación ante fallos y errores.

2.3.3 Eficiencia

Nota: Se usa el prefijo **RNE** en su nomenclatura.

- **RNE 3** La velocidad de procesamiento de la información, la actualización y la recuperación dependerá de la cantidad de información que tenga que procesar.

2.3.4 Soporte

Nota: Se usa el prefijo **RNSO** en su nomenclatura.

- **RNSO 4** Se le debe suministrar al cliente un plugin de video.

2.3.5 Interfaz de Usuario

Nota: se usa el prefijo **RNIU** en su nomenclatura.

- **RNIU 5** El sistema debe tener una apariencia profesional y un diseño gráfico sencillo.
- **RNIU 6** El sistema debe ser intuitivo.

2.3.6 Software

Nota: Se usa el prefijo **RNFO** en su nomenclatura.

- **RNFO 7** Usar como sistema operativo Windows XP SP2 o superior.

2.3.7 Requisitos Legales, de Derecho de Autor y otros.

- El sistema debe ajustarse y registrarse por la ley, decretos leyes, resoluciones y manuales (órdenes) establecidos, que norman los procesos que serán automatizados.
- La mayoría de las herramientas de desarrollo son libres y del reto las licencias están avaladas.
- Como producto se distribuye amparado bajo las normativas legales establecidas en el registro comercial emitido por las entidades jurídicas de la Universidad de las Ciencias Informáticas

2.3.8 Hardware

Nota: Se usa el prefijo **RNH** en su nomenclatura.

- **RFH 8** Se requiere que las PC tengan tarjeta de red.
- **RFH 9** Que la memoria REM sea de al menos 256 MB.
- **RFH 10** Se requiere al menos 1 GB de disco duro.
- **RFH 11** Se requiere de un procesador de 512 MHz como mínimo.

2.4 Descripción de la Solución Propuesta

Una vez realizado el levantamiento de requisitos en la entidad se identifican las condiciones que el sistema debe tener y debe cumplir. A partir de los requisitos funcionales (condiciones que el sistema debe tener) se realizó el Diagrama de Casos de Uso del Sistema (Ver Figura 11) el cual muestra la relación que existe entre el usuario final de la aplicación y las funcionalidades que esta tendrá.

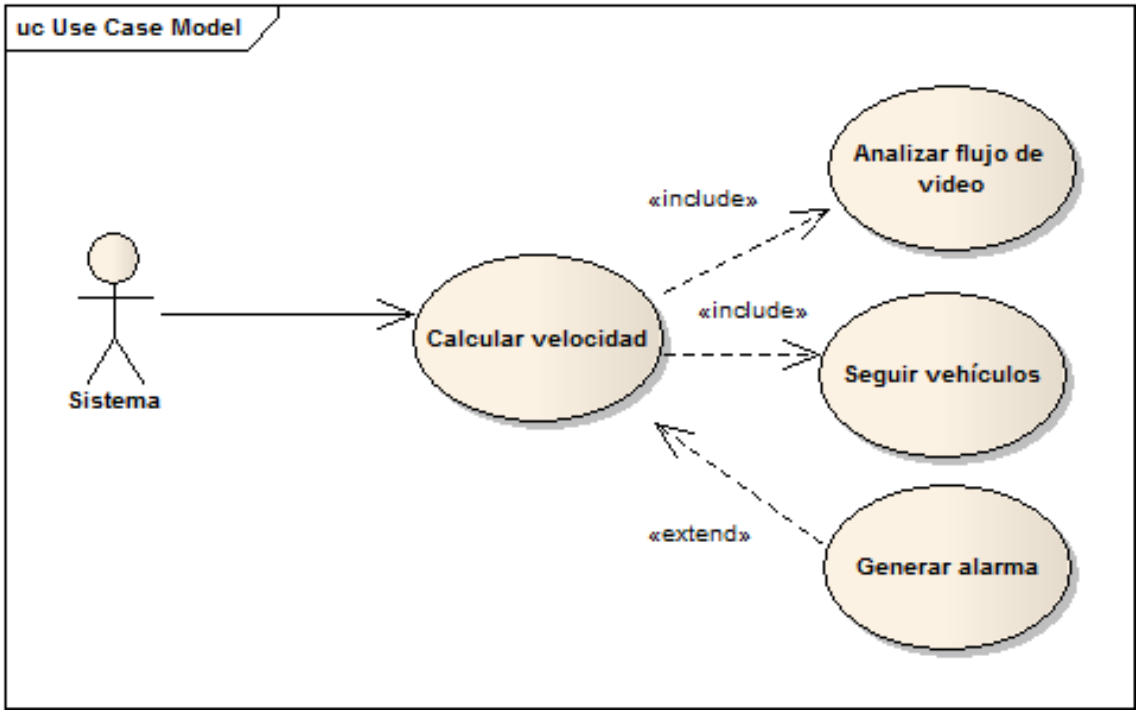

Figura 10: Diagrama de Casos de Uso del Sistema.

2.4.1 Definición de los actores

Actor	Descripción
Sistema	Es el que se ocupa de realizar la obtención del flujo de video de una cámara IP, de analizarlo, realizar petición de calcular la velocidad y luego lanzar una alarma en caso de ser necesario.

Tabla 1: Definición de los actores.

2.4.2 Descripción de los Casos de Uso del Sistema

Caso de Uso Analizar el flujo de video.

Caso de Uso	Analizar el flujo de video
Actores	Sistema
Propósito	Este caso de uso tiene como objetivo mostrar el flujo de video obtenido y luego analizar los objetos que aparezcan en la imagen y encerrar los

	vehículos en un rectángulo.
Resumen	El caso de uso inicia cuando el Sistema de Video Vigilancia hace la petición de que se seguimiento y cálculo de velocidad de vehículos, por lo que el sistema captura un video y luego extrae de él los fotogramas, modela el fondo y pasa a detectar los objetos que se encuentran en el video. Luego determina si el objeto un vehículo, y lo muestra.
Precondiciones	
Referencias	RF 1
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El caso de uso inicia cuando el sistema solicita el seguimiento y cálculo de velocidad de vehículos en un flujo de video.	2. El sistema obtiene el flujo de video proveniente de una cámara IP.
	3. Son extraídos los fotogramas del flujo de video.
	4. Se modela el frente y el fondo del video.
	5. Es eliminado el ruido de la imagen.
	6. Se muestran los vehículos que aparecen en el flujo de video enmarcados en un rectángulo rojo.

Flujos Alternos	
Acción del Actor	Respuesta del Sistema

Pos condiciones	

Tabla 2: Caso de Uso Analizar el flujo de video.

Caso de Uso Seguir vehículo

Caso de Uso	Seguir vehículo	
Actores	Sistema	
Propósito	Este caso de uso tiene como objetivo el seguimiento del vehículo dentro de la región seleccionada, donde se realizara el cálculo de la velocidad de los vehículos.	
Resumen	El caso de uso inicia con el análisis del flujo de video, extrayendo el movimiento de los vehículos y realizando un seguimiento de estos.	
Precondiciones		
Referencias	RF 2	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor		Respuesta del Sistema
1. El caso de uso inicia cuando el operario realiza la selección de la región a partir de donde se va a comenzar el cálculo de velocidad de vehículos en un flujo de video.		2. El sistema pinta una línea horizontal en la región seleccionada.
Flujos Alternos		
Acción del Actor		Respuesta del Sistema
Pos condiciones		

Tabla 3: Caso de Uso Seguir vehículo.

Caso de Uso Calcular Velocidad.

Caso de Uso	Calcular Velocidad
Actores	Sistema
Propósito	Este caso de uso tiene como objetivo realizar el cálculo de la velocidad de vehículos, determinando si esta velocidad no está como es debido.
Resumen	El caso de uso inicia cuando el operario selecciona el contorno a partir de donde se desea que comience el cálculo de la velocidad de los vehículos, luego realiza el seguimiento a los vehículos haciendo un cálculo de la velocidad de los mismos.
Precondiciones	
Referencias	RF 3
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El caso de uso inicia cuando el sistema dibuja la zona de detección	1. El sistema comienza el cálculo de la velocidad de vehículos.
	2. El sistema verifica si la velocidad esta como es debido.

Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Pos condiciones	

Tabla 4: Caso de Uso Calcular Velocidad.

Caso de Uso Generar alarma

Caso de Uso	Generar alarma
Actores	Sistema
Propósito	Este caso de uso tiene como objetivo lanzar un cartel de alarma si la velocidad no está como es debido.
Resumen	El caso de uso inicia cuando el sistema haya realizado el cálculo de la velocidad de los vehículos y comprueba si está por encima de lo debido, si es así lanza un cartel de alarma "*La velocidad está por encima de lo debido*".
Precondiciones	
Referencias	RF 4
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El caso de uso inicia cuando el sistema está calculando la velocidad de los vehículos.	**1.1** El sistema verifica que esas velocidades estén como es debido.
	2.1 Cuando haya una velocidad fuera de lo debido la aplicación lanza una alarma, en este caso un cartel que diga "*La velocidad está por encima de lo debido*".

Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Pos condiciones	

Tabla 5: Caso de Uso Generar alarma.

2.5 Conclusiones parciales

En este capítulo se definió el modelo de dominio para representar de forma visual el entorno real del proyecto, se especificaron los requisitos de software llegando así a un entendimiento de lo que hay que hacer. Se definieron los casos de uso del sistema, diagrama de caso de uso del sistema y la descripción de los mismos para llegar a una comprensión del sistema propuesto.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Diseño del Sistema

La etapa de diseño de un sistema es de vital importancia ya que es aquí donde se modela dicho sistema y encuentra su forma (la arquitectura), lo que permite dar soporte a los requisitos del mismo, tanto funcional como no funcional, así como a las restricciones que se le suponen. Es en esta etapa donde de crea una entrada apropiada y un punto de partida para la implementación de la aplicación.

3.1.1 Descripción de la arquitectura.

La arquitectura representa la estructura de los componentes de un programa o sistema, sus interrelaciones, los principios y reglas que gobiernan su diseño y evolución en el tiempo. (9)

La misma es la representación de alto nivel de la estructura de un sistema o aplicación, que describe los componentes que lo integran, interacciones entre ellos, patrones que supervisan su composición y restricciones para aplicar dichos patrones.

Un patrón es una solución probada que se puede aplicar con éxito a un determinado tipo de problemas que aparecen repetidamente en el desarrollo de software. (10)

Este permite dar solución a un problema en un contexto y reutiliza soluciones a problemas comunes. Son un esqueleto básico que cada diseñador adapta a las características de su aplicación.

El video sensor que se está desarrollando, presenta una arquitectura centrada en los fulos de datos, fundamentada en el estilo tubería-filtros se considera como una serie de transformaciones sucesivas sobre datos de entrada. Una tubería es una arquitectura que conecta componentes, que en este caso son los filtros, a través de conectores. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas.

La elección de este estilo de arquitectura, se basa en que el sistema recibe como entrada un flujo de video, este fluye a través de los componentes, donde estos

componentes son programas independientes, que es la estructura seguida por el estilo Tubería-filtros.

Este estilo posee disimiles ventajas como la eficiencia en la representación y acceso a datos, la solución es atractivamente intuitiva, mantiene el flujo de procesamiento automático, los filtros son altamente reusables, pueden ser agregadas nuevas funciones, por ser los filtro independientes puede ser modificada. Se descompone en cuatro filtros: entrada, shift, ordenar y salida. Cada filtro procesa datos y los pasa al siguiente. Cada filtro puede ejecutar siempre que tenga datos con los que trabajar, o sea posee control distribuido. Los datos que se comparten y se limitan estrictamente a los que viajan por el filtro. (11)

3.1.2 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones, a problemas comunes en el desarrollo de software.

En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que se encuentran sujetos a contextos similares.

En la implementación de un software existen problemas que se repiten o que son análogos, respondiendo así a un cierto patrón. La utilización de estos patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software, mejoran a flexibilidad, modularidad y extensibilidad, factores internos e íntimamente relacionados con la calidad percibida por el usuario. Existen patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

El grupo de patrones **GoF** clasificaron los patrones en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

Creacionales: Tratan con las formas de crear instancias de objetos. Su objetivo es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

Estructurales: Describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.

Comportamiento: Ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Los patrones **GRASP** describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Estos constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.

Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento, es por esto que los siguientes patrones se evidencian en la implementación del video sensor:

Patrón Experto: Se utiliza más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen.

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

Patrón Bajo acoplamiento: Acoplamiento bajo significa que una clase no depende de muchas clases, ya que uno de los principios para protegerse frente a los cambios es mantener bajo el acoplamiento entre variedades. Resulta evidente que, cuanto menor sea el acoplamiento entre clases, menor influencia tendrán los cambios.

Patrón Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones y seguridad). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

Patrón Alta Cohesión: Expresa que se debe asignar una responsabilidad de modo que la cohesión siga siendo alta, la cohesión no es más que la medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. El patrón de Alta Cohesión, más que un diseño directamente implementable en código, se trata de un principio director que guiará el diseño. Una clase estará más cohesionada cuanto más enfocado sea su comportamiento. Es decir, al asignar responsabilidades en el diseño,

se buscará soluciones que asignen los métodos a las clases de forma coherente, completa y relacionada. De esta forma, se obtendrá clases cohesionadas.

3.2 Modelo de Análisis

El análisis consiste en obtener una visión del sistema que se preocupa de ver que es lo que hace el mismo. El modelo de análisis va a ser la entrada fundamental para el modelo del diseño. Con el objetivo de comprender mejor los requisitos y a su vez que ayude a estructurar todo el sistema, incluyendo su arquitectura. Se puede considerar como una primera aproximación al modelo de diseño y es por tanto una entrada fundamental cuando se da forma al sistema en el diseño y la implementación.

3.2.1 Diagrama de clases de análisis.

Un diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa el funcionamiento del mundo real, no de la implementación automatizada del mismo.

Diagrama de clase de analisis del CU (Caso de Uso) Seguir vehículos.

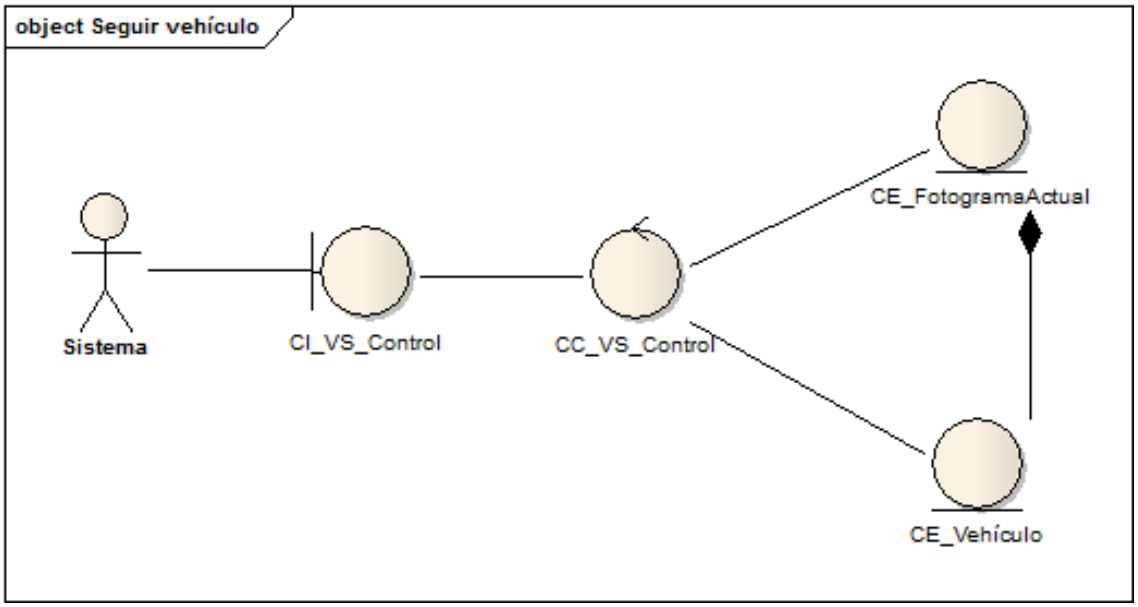

Figura 11: Diagrama de clases de análisis CU Seguir vehículos.

En la figura 12 se muestra el diagrama de clases del análisis para el caso de uso Seguir vehículos, se pueden apreciar las clases entidades CE_FotogramaActual, CE_Vehiculo y la clase control CC_VS_Control.

Diagrama de clase de analisis del CU (Caso de Uso) Calcular Velocidad.

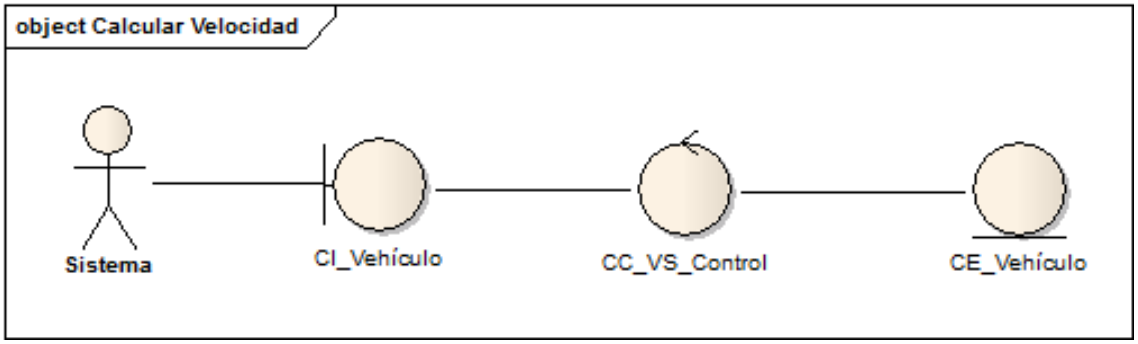

Figura 12: Diagrama de clases de análisis CU Calcular Velocidad.

En la figura 13 se muestra el diagrama de clases del análisis para el caso de uso Calcular Velocidad, se pueden apreciar la clase entidad CE_Vehiculo y la clase control CC_VS_Control.

Diagrama de clase de analisis del CU (Caso de Uso) Generar Alarma

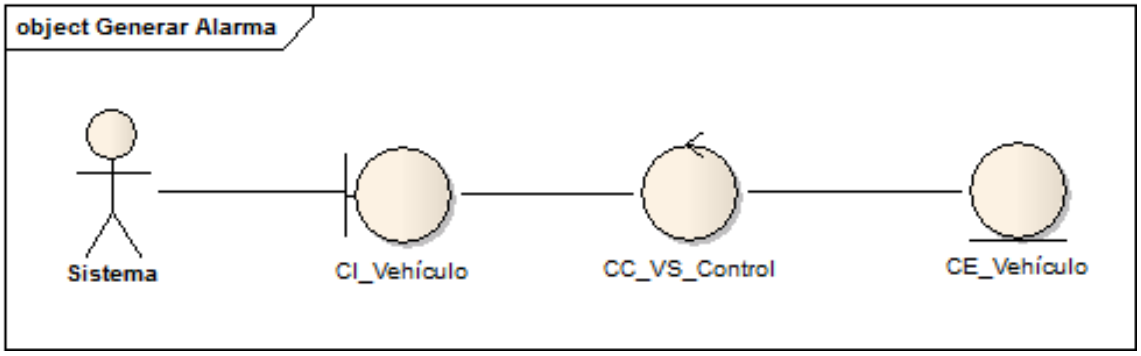

Figura 13: Diagrama de clases de análisis CU Generar Alarma.

En la figura 14 se muestra el diagrama de clases del análisis para el caso de uso Generar Alarma, se pueden apreciar la clase entidad CE_Vehiculo y la clase control CC_VS_Control.

Diagrama de clase de análisis del CU (Caso de Uso) Analizar el flujo de video

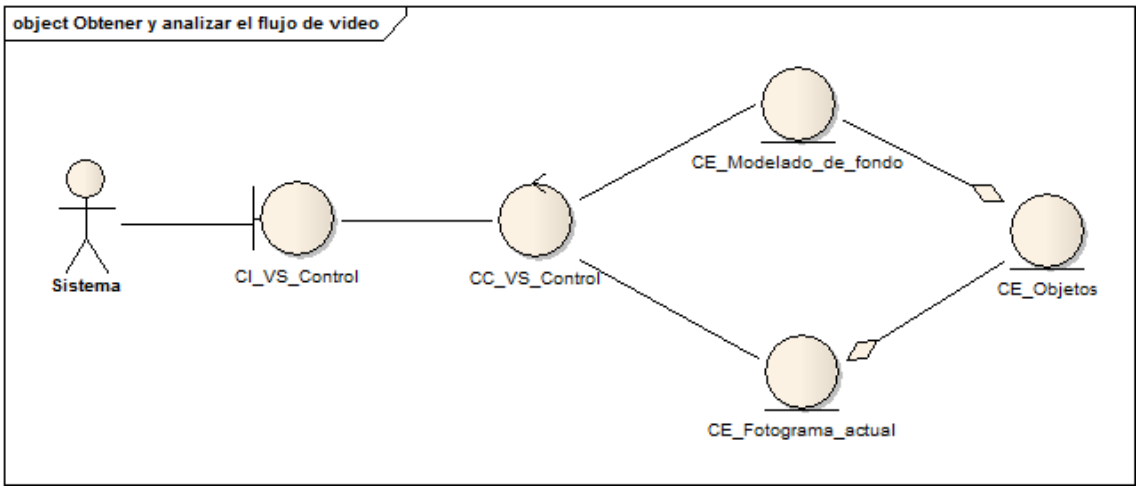

Figura 14: Diagrama de clases de análisis CU Analizar el flujo de video.

En la figura 15 se muestra el diagrama de clases del análisis para el caso de uso Analizar el flujo de video, se pueden apreciar las clases entidades CE_FotogramaActual, CE_Modelado_de_fondo, CE_Objetos y la clase control CC_VS_Control.

3.2.2 Diagramas de Interacción.

Los diagramas de interacción muestran las interacciones entre objetos mediante transferencias de mensajes entre objetos y subsistemas, por lo que son empleados para modelar aspectos dinámicos del sistema. Los diagramas de colaboración y secuencia son dos tipos de diagramas de interacción que son semánticamente equivalentes pero sin embargo los diagramas de colaboración destacan el orden estructural de los objetos que interactúan y los de secuencia destacan el orden temporal de los mensajes. Para la realización de los casos de uso del diseño es más factible el empleo de diagramas de secuencia ya que representan con más claridad el flujo de las acciones que debe realizar el sistema.

Diagrama de colaboración (Caso de Uso) CU Seguir Vehículo.

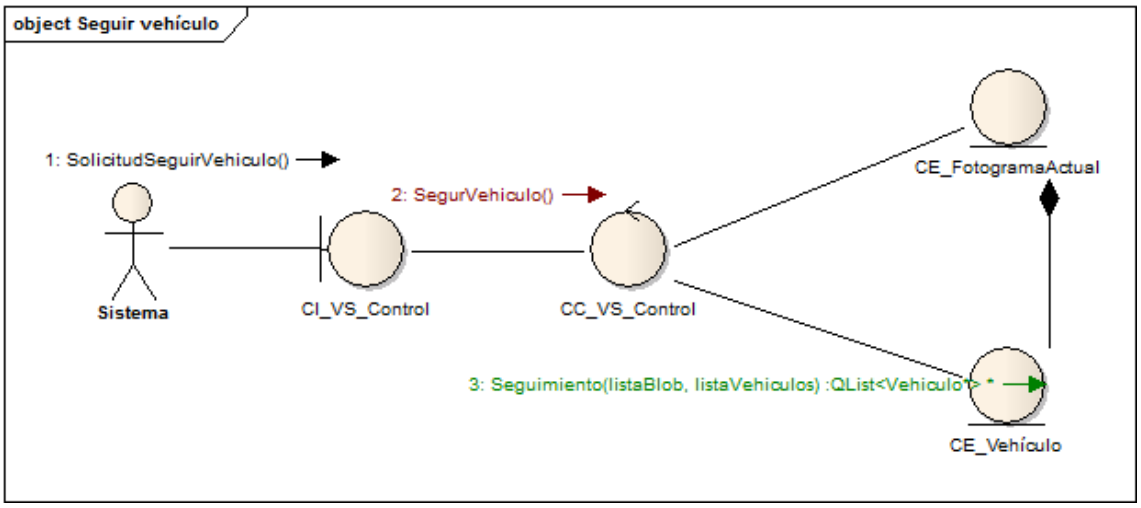

Figura 15: Diagrama de colaboración CU Seguir Vehículo.

En la figura 16 se muestra el diagrama de colaboración para el caso de uso Seguir Vehículo. Se observan los principales mensajes intercambiados entre las clases para realizar el caso de uso en cuestión.

Diagrama de Colaboración del (Caso de Uso) CU Calcular Velocidad.

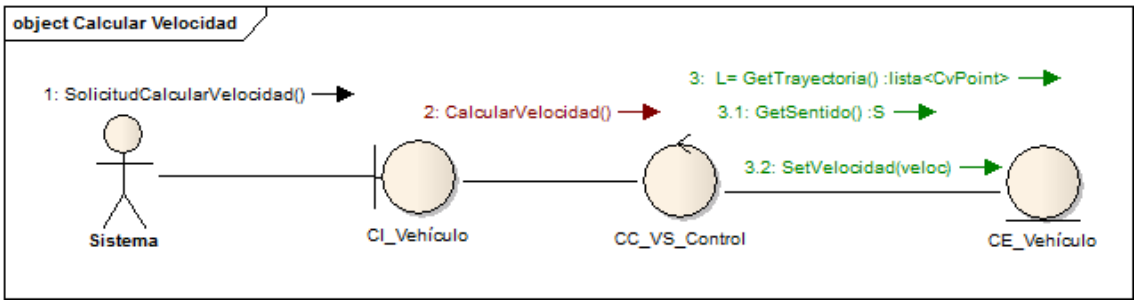

Figura 16: Diagrama de colaboración CU Calcular velocidad.

En la figura 17 se muestra el diagrama de colaboración para el caso de uso Calcular velocidad. Se observan los principales mensajes intercambiados entre las clases para realizar el caso de uso en cuestión.

Diagrama de Colaboración del (Caso de Uso) CU Generar Alarma.

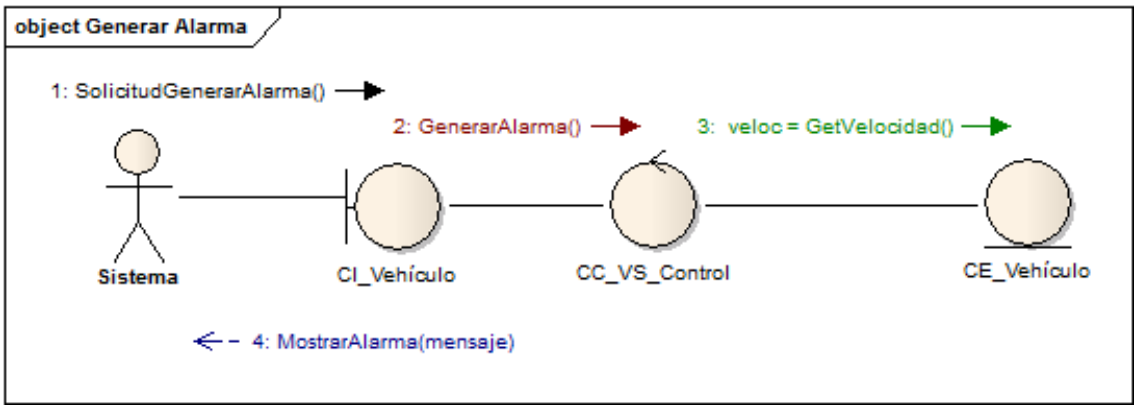

Figura 17: Diagrama de colaboración CU Generar Alarma.

En la figura 18 se muestra el diagrama de colaboración para el caso de uso Generar Alarma. Se observan los principales mensajes intercambiados entre las clases para realizar el caso de uso en cuestión.

Diagrama de Colaboración del (Caso de Uso) Analizar el Flujo de video.

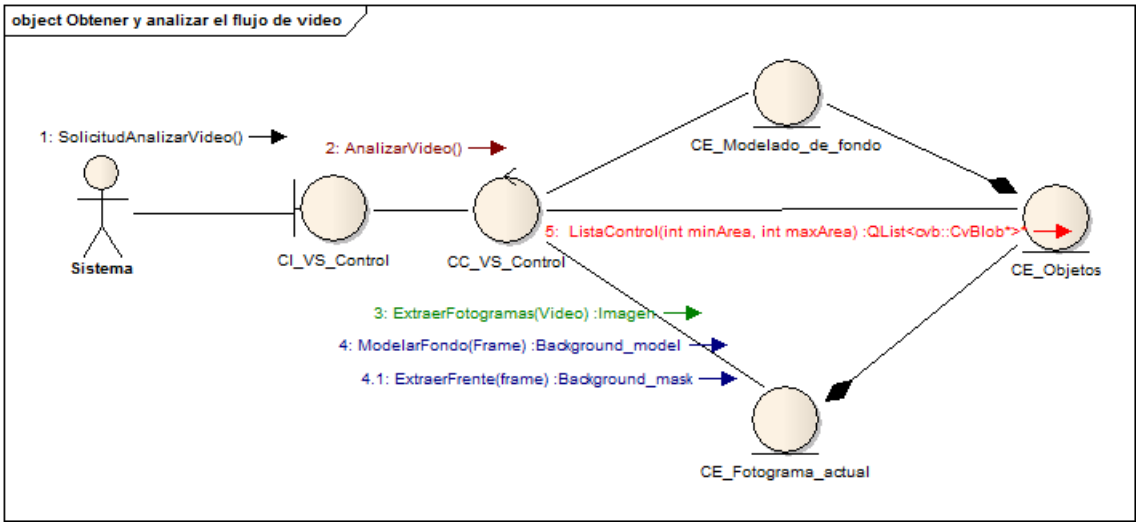

Figura 18: Diagrama de colaboración CU Obtener y analizar el Flujo de video.

En la figura 19 se muestra el diagrama de colaboración para el caso de uso Analizar el Flujo de video. Se observan los principales mensajes intercambiados entre las clases para realizar el caso de uso en cuestión.

3.3 Modelo de Diseño

Un modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto

con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema. Este artefacto constituye la entrada fundamental utilizada para el correcto desarrollo de las entradas de implementación.

A continuación de presentan los diagramas de clases de algunos de los casos de uso fundamentales y un diagrama de secuencia o colaboración para cada uno de ellos y las descripciones de las principales clases involucradas en ellos.

3.3.1 Diagrama de Clases del Diseño

Los diagramas de clases que se muestran a continuación brindan un mayor acercamiento a la forma y al contenido de la solución propuesta. De este modo se han elaborado los diagramas de acuerdo con la descripción del sistema brindada en el capítulo anterior.

Diagrama de clases del diseño del (Caso de Uso) CU Seguir Vehículo.

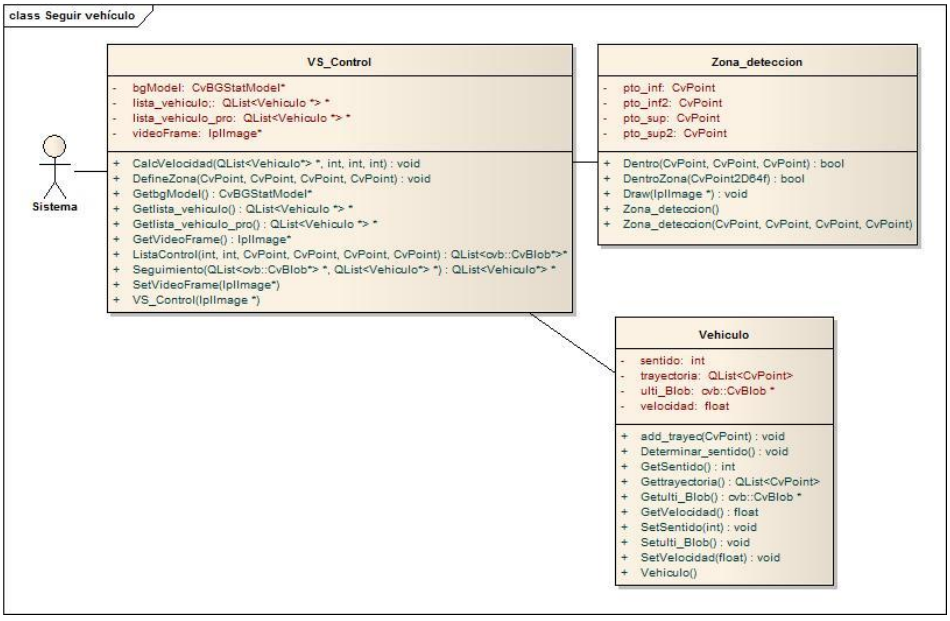

Figura 19: Diagrama clases del diseño del CU Seguir Vehículo.

En la figura 20 se muestra el diagrama de clases del diseñodel CU Seguir Vehículo. La clase VS_Control es la encargada de interactuar con la clase Zona_deteccion y Vehículo que permiten la realización de las funcionalidades del componente. Posee los métodos necesarios para cumplir con los requisitos funcionales.

Diagrama de clases del diseño del (Caso de Uso) CU Calcular Velocidad.

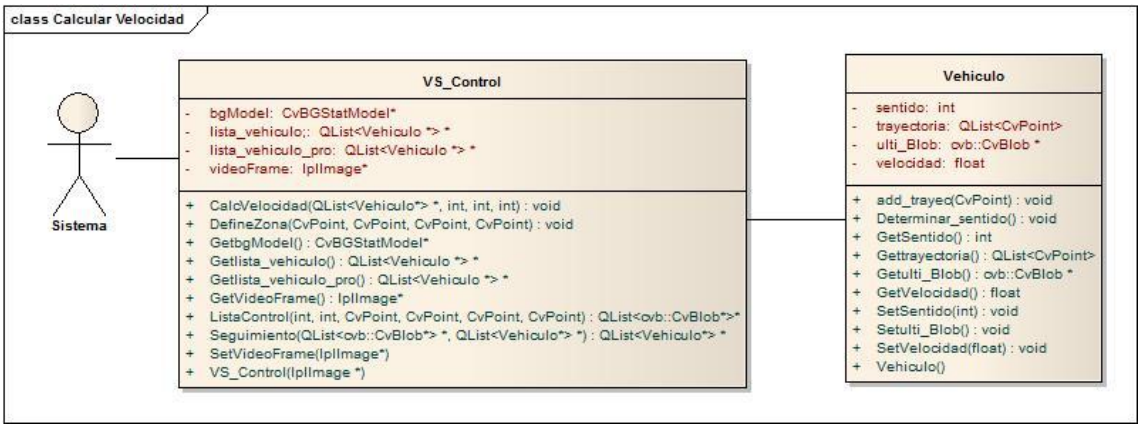

Figura 20: Diagrama clases del diseño del CU Calcular Velocidad.

En la figura 21 se muestra el diagrama de clases del diseñodel CU Calcular Velocidad. La clase VS_Controles la encargada de interactuar con la clase Vehículoque permiten la realización de las funcionalidades del componente. Posee los métodos necesarios para cumplir con los requisitos funcionales.

Diagrama de clases del diseño del (Caso de Uso) CU Generar Alarma.

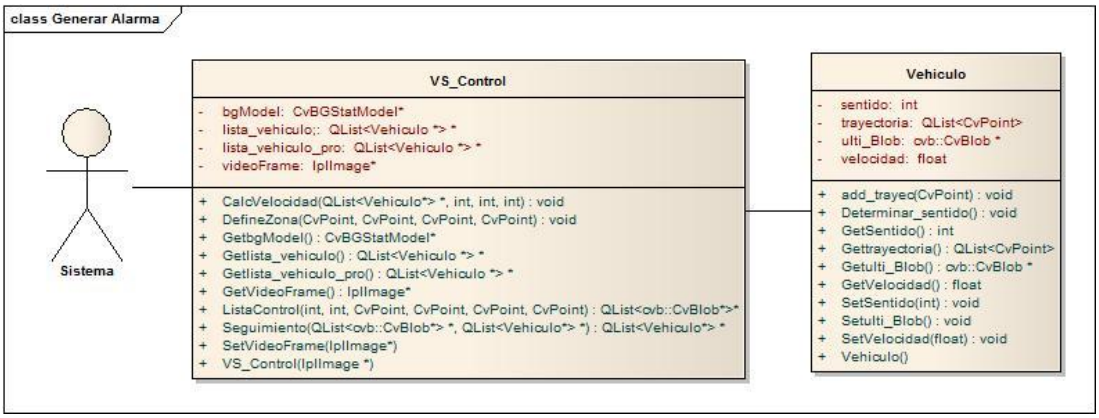

Figura 21: Diagrama clases del diseño del CU Generar Alarma.

En la figura 22 se muestra el diagrama de clases del diseño del CU Generar Alarma.La clase VS_Control es la encargada de interactuar con la clase Vehículoque permiten la realización de las funcionalidades del componente. Posee los métodos necesarios para cumplir con los requisitos funcionales.

Diagrama de clases del diseño del (Caso de Uso) CU Obtener y analizar el Flujo de video.

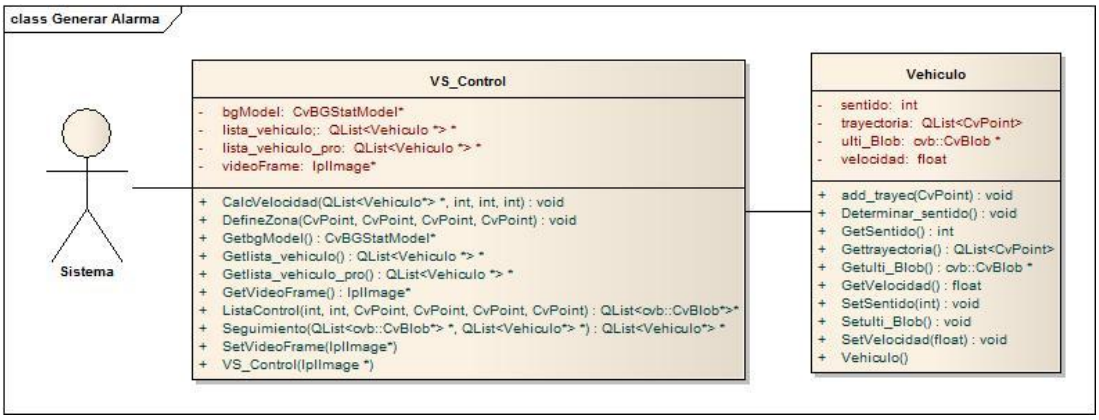

Figura 22: Diagrama clases del diseño del CU Obtener y analizar el Flujo de video.

En la figura 23 se muestra el diagrama de clases del diseño del CU Obtener y analizar el Flujo de video.La claseVS_Control es la encargada de analisar el flujo de videoy brindarle a esta las funcionalidades para e. Posee los atributos y métodos necesarios para cumplir con los requisitos funcionales.

3.3.2 Diagramas de Interacción del Diseño.

Diagrama de secuencia del CU (Caso de Uso) Analizar el flujo de video

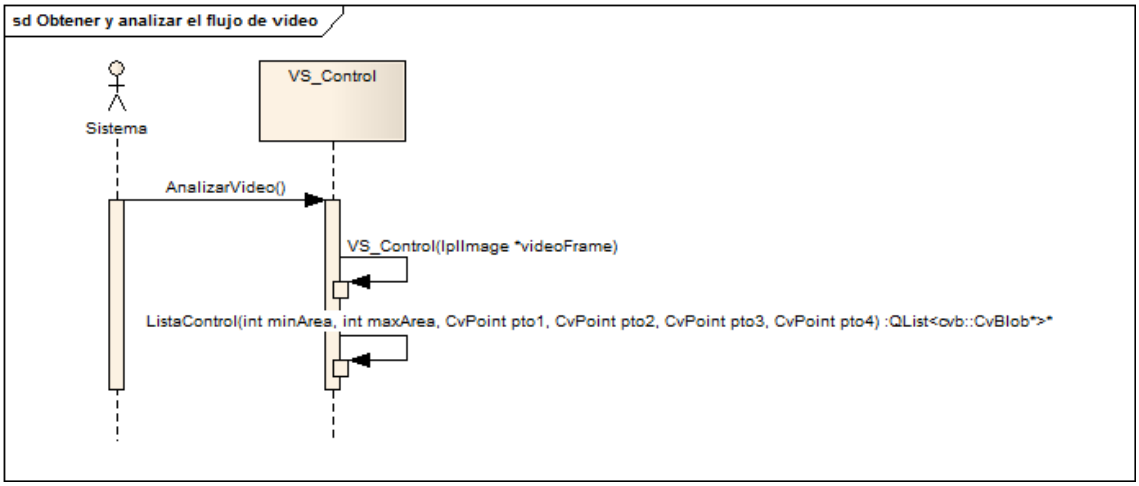

Figura 23: Diagrama de secuencia del CU Analizar el flujo de video.

En la figura 24 se muestra el diagrama de secuencia para el caso de uso Analizar el flujo de video. Se aprecia la interacción en el tiempo entre los objetos y los mensajes enviados entre estos. El sistema interactúa con la clase VS_Control la cual le ofrece las funcionalidades del video sensor.

Diagrama de secuencia del (Caso de Uso) Seguir vehículo.

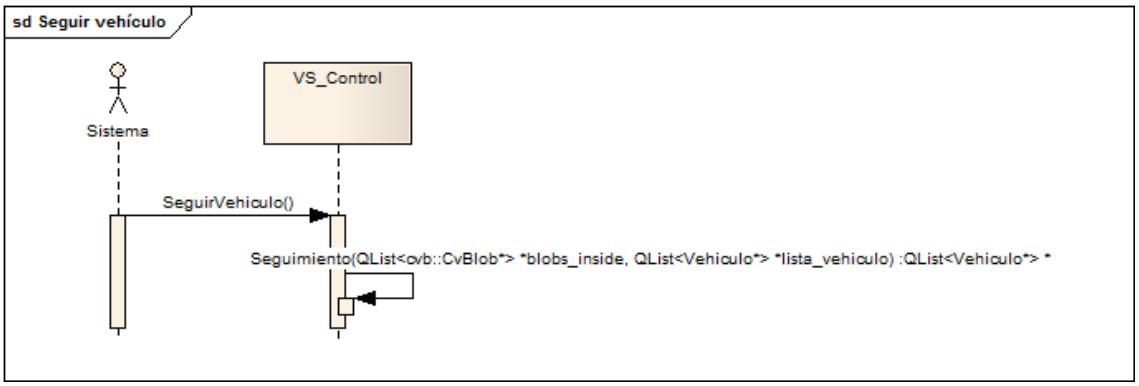

Figura 24: Diagrama de secuencia del CU Seguir vehículo.

En la figura 25 se muestra el diagrama de secuencia para el caso de uso Seguir vehículo. Se aprecia la interacción en el tiempo entre los objetos y los mensajes enviados entre estos. El sistema interactúa con la clase VS_Control la cual le ofrece las funcionalidades del video sensor.

Diagrama de secuencia del (Caso de Uso) Mostar Alarma.

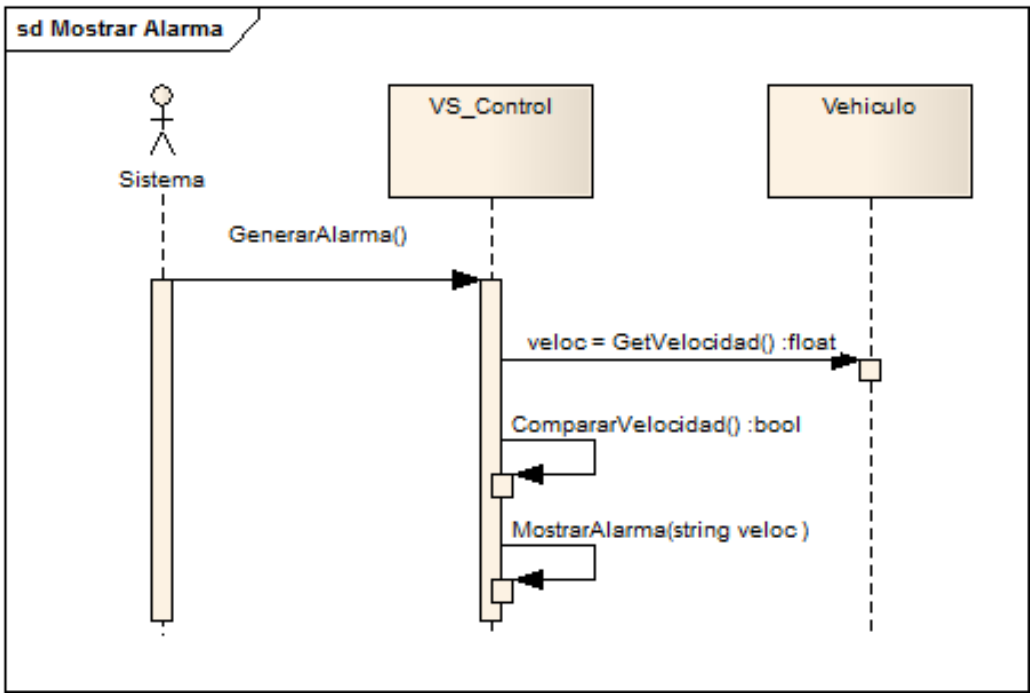

Figura 25: Diagrama de secuencia del CU Mostar Alarma.

En la figura 26 se muestra el diagrama de secuencia para el caso de uso Móstar Alarma. Se aprecia la interacción en el tiempo entre los objetos y los mensajes enviados entre estos. El sistema interactúa con la clase VS_Control la cual le ofrece las funcionalidades del video sensor y esta a su vez con la clase Vehículo.

Diagrama de secuencia del (Caso de Uso) Calcular Velocidad.

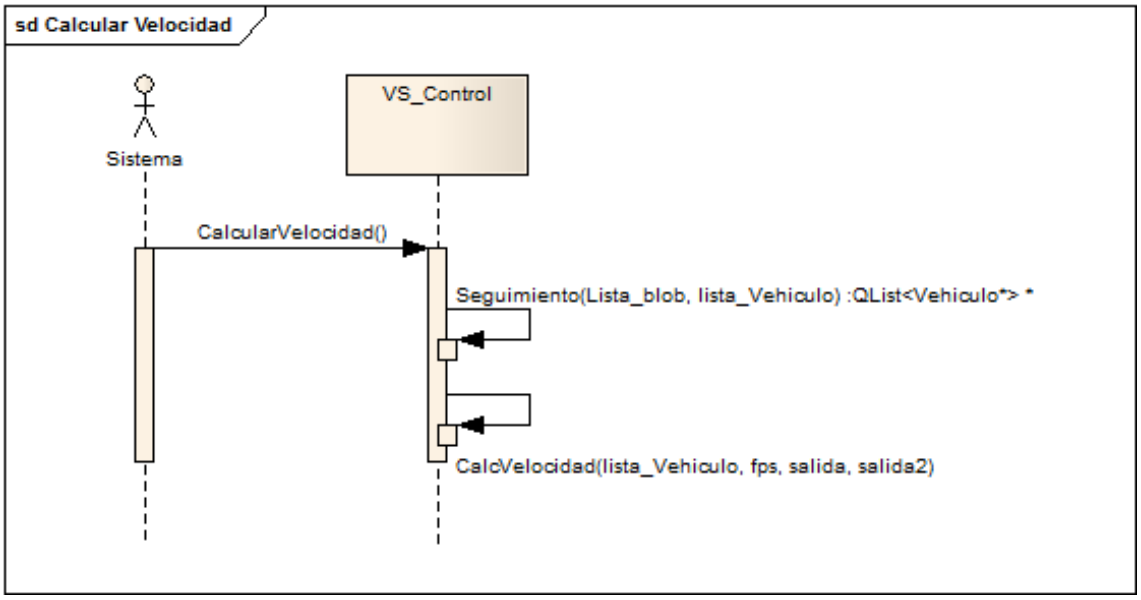

Figura 26: Diagrama de secuencia del CU Calcular Velocidad.

En la figura 27 se muestra el diagrama de secuencia para el caso de uso Móstar Alarma. Se aprecia la interacción en el tiempo entre los objetos y los mensajes enviados entre estos. El sistema interactúa con la clase VS_Control la cual le ofrece las funcionalidades del video sensor.

3.4 Descripción de las Clases.

Descripción de las clases de Video sensor Control (VS_Control).

Nombre	**VS_Control**
Tipo de clase	Controladora

Atributo	**Tipo**
frame	IplImage
bgModel	CvBGStatModel
Para cada responsabilidad	
Nombre	VS_Control(IplImage *videoFrame)
Descripción	Constructor de la clase, inicializa los parámetros de la clase VS_Control.
Nombre	Control (int minArea, int maxArea) : QList<cvb::CvBlob*>*
Descripción	Lista de lista, la cual contiene los blob que se procesan para realizar el cálculo de velocidad.
Nombre	DefineZona ():void
Descripción	Define el perímetro donde se realizara el cálculo de velocidad.
Nombre	SetvideoFrame (IplImage* _frame): void
Descripción	Cambia el valor del frame que se está procesando
Nombre	GetvideoFrame (): IplImage
Descripción	Devuelve el valor del atributo videoFrame
Nombre	GetbgModel():CvBGStatModel
Descripción	Devuelve el valor del atributo bgModel
Nombre	Procesamiento(QList<cvb::CvBlob*>* listaB, QList<Vehiculo*> *listaV): QList<Vehiculo*>
Descripción	Realiza el procesamiento de los blob para determinar a qué vehículo corresponden los frames procesados.
Nombre	CalcVelocidad(QList<Vehiculo*> *listaV, int FPS):void

Descripción	Realiza el cálculo de la velocidad del vehículo.

Tabla 6: Descripción de las clases de Video sensor Control.

Descripción de las clases Vehículo.

Nombre	**Vehiculo**
Tipo de clase	Auxiliar
Atributo	**Tipo**
velocidad	float
sentido	int
ulti_Blob	cvb::CvBlob
trayectoria	QList<CvPoint>
Para cada responsabilidad	
Nombre	Vehiculo()
Descripción	Constructor por defecto de la clase.
Nombre	GetVelocidad():float
Descripción	Devuelve el valor del atributo velocidad
Nombre	SetVelocidad(float val) :void
Descripción	Se utiliza para modificar el valor del atributo velocidad.
Nombre	GetSentido():int
Descripción	Devuelve el valor del atributo sentido
Nombre	SetSentido(int val):viod

Descripción	Se utiliza para modificar el valor del atributo sentido
Nombre	Getulti_Blob():cvb::CvBlob
Descripción	Devuelve el valor del atributo ulti_Blob
Nombre	Setulti_Blob(cvb::CvBlob *val):void
Descripción	Se utiliza para modificar el valor del atributo ulti_Blob
Nombre	Gettrayectoria():QList<CvPoint>
Descripción	Devuelve el valor del atributo trayectoria
Nombre	add_trayec(CvPoint p): void
Descripción	Adiciona un punto a la lista de trayectoria
Nombre	Determinar_sentido():
Descripción	Determina el sentido de un objeto determinado para realizar el cálculo de velocidad

Tabla 7: Descripción de las clases Vehículo.

Descripción de las clases del Zona de detección (Zona_deteccion)

Nombre	**Zona_deteccion**
Tipo de clase	Auxiliar
Atributo	**Tipo**
CvPoint	pto_sup
CvPoint	pto_sup2
CvPoint	pto_inf
CvPoint	pto_inf2

Para cada responsabilidad	
Nombre	Zona_deteccion ()
Descripción	Constructor por defecto de la clase.
Nombre	Zona_deteccion(CvPointpto_sup, CvPoint pto_sup2, CvPointpto_inf, CvPoint pto_inf2)
Descripción	Constructor de la clase, inicializa los parámetros de la clase Zona_deteccion.
Nombre	DentroZona(CvPointpoint):bool
Descripción	Determina si un punto se encuentra dentro de un área determinada.
Nombre	Draw(IplImage *videoFrame):void
Descripción	Dibuja sobre la imagen (frame) un área determinada.

Tabla 8: Descripción de las clases del Zona de detección.

3.5 Conclusiones parciales

En este capítulo se mostró en términos de componentes, el análisis y el diseño, se especificaron los requisitos de software, este flujo de trabajo permite profundizar en los casos de usos detallándolos de manera que se refleje una vista interna del sistema descrita con el lenguaje de los desarrolladores y se mostró la arquitectura escogida para la realización de la aplicación.

CAPÍTULO 4: ALGORITMOS

Introducción

En este capítulo se explica en detalles los diferentes algoritmos que se emplean en la realización del “Video sensor para el seguimiento y estimación de velocidad de vehículos.”

4.1 Algoritmo de estimación y sustracción de fondo

La detección de objetos se puede obtener, mediante la construcción de una representación de la escena llamada modelo de fondo y después encontrando las desviaciones del modelo para cada fotograma entrante. Cualquier cambio significativo en una región de la imagen del modelo de fondo representa un objeto en movimiento. Los píxeles que constituyen las regiones en proceso de cambio se marcan para su posterior procesamiento. En general, un algoritmo de componentes conectados se aplica para obtener regiones conectadas que corresponden a los objetos. Este proceso se conoce como la sustracción de fondo. Para el desarrollo de este video sensor se utilizará el algoritmo mezcla de Gaussianas (MoG).

4.1.1 Mezcla de Gaussianas (MoG)

Es uno de los métodos complejos más utilizado (6), entre sus ventajas destaca la robustez, ya que permite modelar fondos multimodales, es decir, manejar múltiples modos de distribución (o tipos de movimiento). Por ejemplo, una hoja agitándose bajo un cielo azul presenta dos modos o tipos de movimiento: el de las hojas y el cielo.

En este algoritmo, el fondo no se modela por los valores de una imagen sino a través de un modelo paramétrico cuyo objetivo es aproximar una función de distribución a los últimos valores de cada píxel. Cada píxel $p_t$localizado en la posición (x, y) de la imagen de fondo en el instante t se representa por un número K de distribuciones Gaussianas cuya combinación constituye una función de densidad de probabilidad $F(P_t)$.

$$F(p_t) = \sum_{i=1} w_{i,t} \cdot \eta(p_t; \mu_{i,t}, \sigma_{i,t})$$

$$\sum_{i=1}^{i=K} w_{i,t} = 1$$

Figura 27: Función de densidad de probabilidad.

Donde$W_{i,t}$ es el parámetro que indica el peso de la componente Gaussiana i-ésima en el instante t y $\eta(P_t, \mu_{i,t}, \sigma_{i,t})$ la componente o Gaussiana i-ésima del pixelP_t, que posee una media $\mu_{i,t}$ y una desviación $\sigma_{i,t}$, representada por:

$$\eta(p_t; \mu_{i,t}, \sigma_{i,t}) = \frac{1}{(2\pi)^{\frac{1}{2}} \sigma_{i,t}} e^{\frac{1}{2}\left(\frac{p_t - \mu_{i,t}}{\sigma_{i,t}}\right)^2}$$

Figura 28: Función de Densidad de probabilidad.

Las medias de la Gaussianas (o componentes) son los valores medios que toman los píxeles a lo largo de la secuencia ponderados por un peso. El número de Gaussianas K (componentes por píxel) son típicamente 3-5 y se escoge generalmente en función de las limitaciones de memoria.

A continuación se explicarán paso a paso las operaciones realizadas para modelar el fondo con MoG y extraer el frente.

4.1.2 Inicialización de Parámetros

En primer lugar, se inicializan los parámetros que representan el fondo: media, desviación y peso. Para ello, es necesario tomar una serie de decisiones iniciales; en nuestro caso, hemos inicializado la media de una de las Gaussianas (por ejemplo, la Gaussiana i =1) a la primera imagen y el resto de medias a valores aleatorios. Con ello consideramos que la primera Gaussiana modelará el píxel en la primera imagen. Por este motivo, el peso w o porcentaje de distribución para k = 1 debe ser un valor alto (próximo a 1) y para el resto será un valor muy pequeño (próximo a 0), ya que la suma de los pesos de las Gaussianas debe ser 1. En cuanto a la desviación inicial dependerá del tipo de secuencia, generalmente, se da un valor elevado a las tres componentes.

$$B_1(x,y) = \{\mu_i(x,y),\ \sigma_i(x,y),\ i = 1 \ldots K\} \quad /$$

$$\begin{cases} i = 1 \Rightarrow \mu_{i,1}(x,y) = I_1(x,y) \quad \sigma_{i,1}(x,y) = P \quad w_{i,1} = 1 \\ 1 < i \le K \Rightarrow \mu_{i,1}(x,y) = aleatorio \quad \sigma_{i,1}(x,y) = P \quad w_{i,1} = 0 \end{cases} , \ \sum_{i=1}^{K} w_{i,t} = 1$$

Figura 29: Función de Desviación inicial.

4.1.3 Estimación del BG y actualización de parámetros

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

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

$$\Downarrow$$

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

Figura 30: Función de Píxeles pertenecientes al fondo.

Donde el factor $\rho = \alpha \Pr(I_t(x,y) / \mu_{i,t-1}(x,y), \sigma_{i,t-1}(x,y))$, α es un parámetro predefinido al inicio de la ejecución $\mu_{i,t}(x,y)$ es el valor de la media del pixel en el instante t e $I_t(x,y)$ es el valor del pixel en el instante t.

Las componentes que no han sido marcadas como píxeles de fondo se actualizan de la siguiente forma; la media y varianza queda con el mismo valor y el peso desciende lo cual significa que la probabilidad de modelar el píxel con esta Gaussiana disminuye:

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

$$\Downarrow$$

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

Figura 31: Función de Píxeles pertenecientes al fondo.

Para que la suma de pesos siga valiendo uno, se normalizan los pesos:

$$w_{i,t+1}(x, y) = \frac{w_{i,t+1}(x, y)}{\sum_{j=1}^{j=K} w_{j,t+1}(x, y)}$$

Figura 32: Función de Normalizan de los pesos.

Si no se ha encontrado parecido con ninguna de las Gaussianas que modelaban el píxel, se reemplaza la de menor peso por una nueva Gaussiana de peso muy pequeño y de media el valor del píxel; con ello, se consigue introducir un modo en la distribución del píxel.

Una vez actualizados los parámetros, para obtener una imagen de fondo se calcula para cada píxel la media ponderada de las K distribuciones:

$$B_t(x, y) = \sum_{i=1}^{K} w_{i,t}(x, y) \cdot \mu_{i,t}(x, y)$$

Figura 33: Función Media ponderada de las K distribuciones.

4.1.4 Obtención del FG

En el proceso de detección de píxeles de frente y fondo, en primer lugar, se ordenan los pesos de mayor a menor y se escogen como modelo de fondo de cada píxel las B Gaussianas de mayor peso, es decir, aquellas cuyo peso acumulado supere un umbral (por ejemplo, un 0.85). Si para la Gaussiana b que supere dicho umbral la diferencia

entre la media y el píxel de la imagen actual es inferior a c veces su desviación, se considerará píxel de fondo. En caso contrario, será objeto en movimiento $F_t(FG)$.

$$B = \arg\min_b \sum_{i=1}^{b} w_{i,t}(x,y) > \tau$$
$$\exists b \in [1,B] / \left| I_t(x,y) - \mu_{b,t}(x,y) \right| \leq c\sigma_{b,t}(x,y) \Rightarrow F_t(x,y) = 0$$
$$\forall b \in [1,B], \left| I_t(x,y) - \mu_{b,t}(x,y) \right| > c\sigma_{b,t}(x,y) \Rightarrow F_t(x,y) = 1$$

Figura 34: Función de Píxeles de frente y fondo.

La mezcla de Gaussianas es un algoritmo complejo algorítmicamente pero produce buenos resultados. No obstante, presenta un problema de ajuste de parámetros (el porcentaje de la componente de fondo τ , la desviación estándar σ inicial, el número de veces que se toma la desviación de las gaussianas *c* para decidir entre fondo y frente, el factor de aprendizaje ρ, el número total de componentes de Gaussianas K y el número máximo de componentes B en el modelo de fondo) todos ellos con un impacto significativo sobre el desempeño del algoritmo.

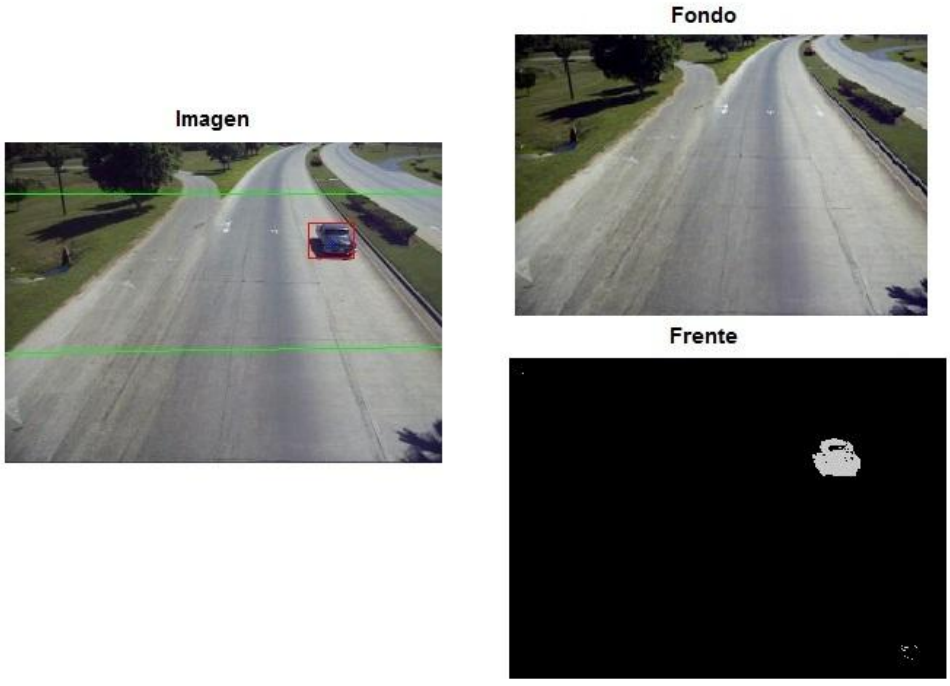

Figura 35: Imagen, modelo MoG del Fondo y Frente de la secuencia.

En la figura se muestra la imagen, el fondo y frente de la secuencia en el instante 63, la imagen muestra un vehículo enmarcado en un rectángulo rojo, en el frente el vehículo en los píxeles de color blanco.

4.2 Algoritmo de seguimiento de objetos

El seguimiento de objetos es una tarea muy importante dentro del campo del procesado de video. El objetivo principal de las técnicas de seguimiento de objetos es generar la trayectoria de un objeto a través del tiempo, posicionando éste dentro de la imagen determinada.

En el caso especial del video sensor para el seguimiento y estimación de velocidad de vehículos, que se está desarrollando, es necesario realizar un seguimiento a los vehículos que se encuentran en movimiento en la escena, para saber dónde se encuentran ubicados en cada instante de tiempo y de esta manera determinar cuando entran la zona de detección.

Para realizar el seguimiento es necesario determinar cuáles son los objetos de interés en la escena, en este caso se toman los objetos en movimiento por el área de los mismos.
Cada objeto tiene un centroide, área y si se le está realizando seguimiento o no, estos valores se actualizan y se guardan en una lista. Los mismos son analizados fotograma a fotograma comparando la distancia euclidiana entre ellos. Si la distancia está en un rango mínimo determinado, entonces el objeto del fotograma anterior es el mismo que el fotograma que se está analizando. También se analiza la posibilidad que puedan entrar en la escena nuevos objetos, estos son comparados con los que se le están realizando el seguimiento y si no coinciden en el identificador, características o posición se agregan a la lista de objetos a seguir y se les da un identificador nuevo. Si algún objeto deja de ser seguido por un tiempo (número de frames) predeterminado, se elimina de la lista de seguimiento.

Figura 36: Seguimiento de objetos.

4.3 Cálculo de velocidad

La velocidad es la magnitud física de carácter vectorial que determina el recorrido de un objeto en una distancia en determinado espacio de tiempo. Su unidad en el Sistema Internacional es el m/s.

En el desarrollo de este video sensor se realiza el cálculo de la velocidad a partir del seguimiento realizado a los vehículos, teniendo en cuenta que la distancia ya está determinada y el tiempo se determina por la división entre la cantidad de fotogramas que se siguió el objeto y la cantidad de fps a la que la cámara realiza la captura, estableciendo así el período de tiempo necesario para esta magnitud física.

4.4 Conclusiones Parciales

En este capítulo se explicaron en detalle los algoritmos empleados para la realización del “Video sensor para el seguimiento y estimación de velocidad de vehículos”. Se determinó que el algoritmo de seguimiento de vehículo para saber en cualquier instante de tiempo si el objeto está dentro de la zona de detección, además se describió el algoritmo de estimación y sustracción de fondo basado en mezcla de Gaussianas, los cuales permitieron obtener el producto desarrollado.

CAPÍTULO 5: IMPLEMENTACIÓN Y PRUEBA

Introducción

En este capítulo los elementos del modelo del diseño se implementan en términos de componentes como son los ficheros de código fuente, ficheros de código binario y ejecutable. Además se realizan pruebas al sistema para lograr erradicar errores que puedan ser introducidos en la implementación del video sensor y para comprobar que el producto final cumple con los requisitos establecidos en la primera fase de la investigación.

5.1 Descripción de los componentes

Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación, un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos. Son las piezas reutilizables de alto nivel a partir de las cuales se pueden construir los sistemas. El sistema cuenta con 3 componentes: cvBlob.dll, OpenCV.dll y VS_Control que se encargan de distribuir las funcionalidades necesarias para la realización del video sensor.

- **OpenCV:** Esta librería se utiliza para el tratamiento de imágenes.
- **cvBlob:** Esta librería se utiliza para trabajar con los Blobs, la misma ofrece diversas funcionalidades que facilitan el trabajo con los objetos que se encuentran en la escena.
- **VS_Control:** Tiene como objetivo brindar todas las funcionalidades del componente.

5.1.1 Diagrama de Componentes

Un diagrama de componentes se utiliza para modelar los componentes de un sistema y mostrar las dependencias entre ellos. Un componente representa un módulo de software con un interface bien definido. Ejemplos de componentes son el código fuente, código binario, ejecutable, DLL, fichero de inicialización (.ini), fichero de registro (.log), tablas, documentos, mapas de bits de los iconos de una aplicación.

Los componentes pueden existir en tiempo de compilación, de enlace o de ejecución o incluso en varios o todos esos tiempos. Estos son equivalentes a tipos, siendo sus instancias las que aparecerán en el diagrama de despliegue. Hay una gran diferencia entre clases y componentes. Las clases son “componentes lógicos”, mientras que los componentes tienen significado físico. Antes de poder generar código, las clases deben ser mapeadas a componentes (esto es, las clases residen o son implementadas en componentes), y estos componentes serán los que contengan su código fuente.

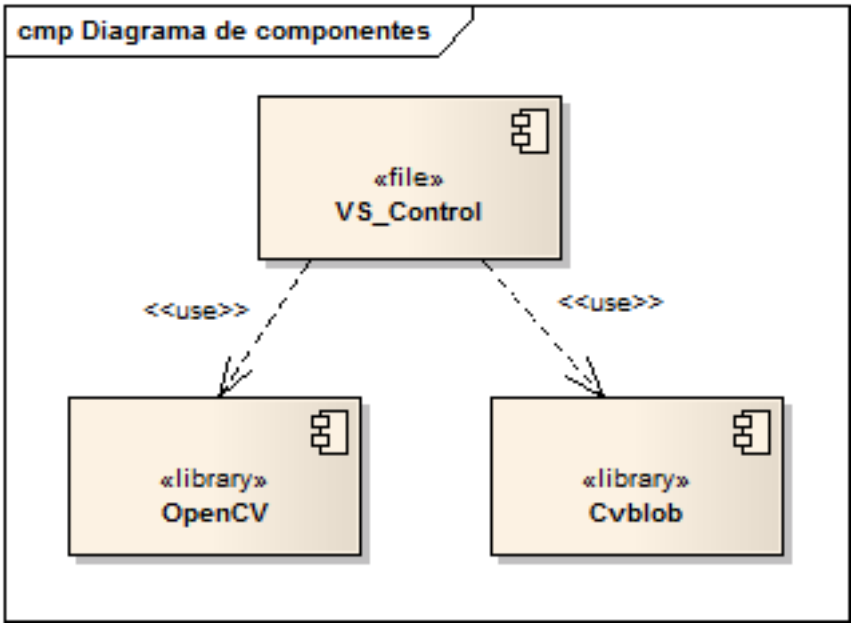

Figura 37: Diagrama de Componentes.

5.2 Pruebas de software

La calidad de un sistema está determinada, entre otras cosas, por la coincidencia entre lo que se programó y los requisitos establecidos en la primera fase. Para comprobar el grado de cumplimiento de estos requisitos se usan las pruebas del sistema. Estas definen un conjunto amplio de acciones de comprobación que abarcan todas las características que determinan la calidad de un software. Se comprueban las funcionalidades diseñando casos de prueba que definen cómo proceder. Estos casos de prueba incluyen los juegos de datos a usar que son los válidos o esperados y los no válidos o no esperados por el programa. Además establecen los resultados a alcanzar en correspondencia de la lógica del programa y los datos ingresados. Describen las condiciones generales en las que se debe aplicar las pruebas para obtener los objetivos propuestos. Es beneficioso que los desarrolladores prueben su

producto pero que no falte la mano de terceras personas que no intervinieron en el proyecto directamente ya que así se detecta mayor cantidad de fallas. (12)

5.2.1 Pruebas de eficiencia

La realización de las pruebas de eficiencia requirió el diseño de los casos de prueba en correspondencia con las descripciones de casos de uso. Los casos de prueba permiten comprobar todos los flujos de información del sistema y validar que este cumple con los requisitos del cliente. A continuación se presentan los casos de prueba utilizados para los casos de uso más importantes del sistema:

Procesamiento en tiempo real.

Caso de prueba					
Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo central**	**Resultados Esperados**	**Resultados Obtenidos**
Calcular Velocidad	Analizar un flujo de video	El objetivo es mostrar el flujo de video obtenido, realizar el seguimiento de los vehículos y calcular la velocidad de los mismos.	Obtener el flujo de video. Extraer fotogramas. Modelado de fondo. Mostrar los vehículos Calcular la velocidad de los vehículos.	El sistema procese entre 25 y 30 fotogramas por segundo (para cada fotograma se necesitaran 0.033 s).	El sistema es capaz de procesar 100 fotogramas en 0.54 segundo (0.00054 s)

Tabla 9: Caso de prueba para el procesamiento en tiempo real.

5.3 Conclusiones Parciales

En este capítulo se realizó el diagrama de componente logrando así establecer una dependencia lógica entre componentes de software VS_Control, cvblob y OpenCV. También se validó el diseño a través de las pruebas de eficiencia, al caso de uso, Calcular Velocidad. En general se obtuvieron resultados favorables porque se pudo probar que los procesos implementados realizan las funcionalidades requeridas con la

calidad y eficiencia necesaria para que la aplicación cumpla con los requisitos especificados y realice el procesamiento en tiempo real.

Conclusiones generales

En el proceso de desarrollo del presente trabajo se llevaron a cabo una serie de fases que permitieron dar cumplimiento al objetivo general planteado y se culminaron todas las tareas investigativas propuestas.

Por consiguiente se concluye que:

- El estudio realizado de temas referentes a la evolución de los sistemas de video sensores, demostró que la principal deficiencia de los sistemas existentes es que se encuentran implementados bajo software privativo. Además posibilitó una mayor comprensión en la realización del “Video sensor para el seguimiento y estimación de velocidad de vehículos”.

- Se logró obtener un modelo diseño a partir de la descripción de los requisitos y se modeló la solución propuesta.

- Se definió una arquitectura centrada en el flujo de datos, implementando el patrón “Filtros y Tuberías”, acorde a las necesidades operacionales, posibilitando la implementación de un componente con alto grado de reusabilidad del código.

- Se implementó un componente que permite calcular la velocidad de los vehículos, en un flujo de video obtenido de una cámara IP para el sistema de Video Vigilancia Suria. El cual cumple todos los requisitos especificados y se puede integrar a otros componentes y módulos del proyecto.

- La validación del componente a través de la prueba de eficiencia, demuestra que el componente desarrollado posee la calidad requerida y cumple con los requisitos especificados, demostrando la eficiencia del mismo.

Recomendaciones

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

- Que el sistema de video vigilancia Suria permita modificar los parámetros de la mezcla gaussiana, permitiendo así mayor robustez del video sensor.

- Implementar el algoritmo filtro de Kalman, que permitiría identificar objetos según un grupo de características determinada.

- Optimizar el algoritmo de seguimiento, para lograr mayor eficiencia a la hora de seguir un vehículo de gran dimensión.

Glosario de términos

Frame: fotograma o cuadro, es una imagen particular dentro de una sucesión de imágenes que componen una animación.

Píxel: abreviatura de Picture Element, es la menor unidad posible con la que se compone cualquier imagen digital en una computadora.

Gaussianas: función matemática en honor a Carl Friedrich Gauss.

VIP: Procesador de imagen de video

Framework: estructura conceptual y tecnológica de soporte definida, normalmente con artefactos de software concretos, mediante la cual otro proyecto de software puede ser organizado y desarrollado

IDE: Entorno de Desarrollo Integrado.

Herramientas CASE: Ingeniería de Software Asistida por Computadora.

Plugin: Pequeño programa que se adhiere a otro para poder ejecutar cierto tipo de archivos o para aportarle una función nueva, generalmente muy específica.

TIC: tecnologías de la información y la comunicación

GEYSED: Geoinformática y Señales Digitales

Referencias bibliográficas

1. **Colomer, Antonio Albiol.***Seguimiento de objeto en secuencia de video.* España : s.n, 2003.

2. **DICCIONARIO DE LA LENGUA ESPAÑOLA.** Real Acamdemia Española. *RAE.* [En línea] 2001. [Citado el: 14 de noviembre de 2011.] http://buscon.rae.es/draeI/..

3. Electronic Dreams. [En línea] [Citado el: 2011 de octubre de 15.] http://www.camara-ip.es/.

4. **Datys.** Datys. [En línea] 2010. [Citado el: 29 de octubre de 2011.] http://www.datys.cu/wpinfnoticias..

5. **Alcantarilla, Pablo Fernández.** 2006.

6. **Stauffer C, Grimson W.***Learning Patts of activity using real time Tracking.* 2000. 22:747-57.

7. **Javed O., Shafique K., Shah M.***A hierarchical approach to robustbackground subtraction using color and gradient information. motion.Workshop.*

8. Historia del lenguaje C++. [En línea] [Citado el: noviembre de 20 de 2011.] http://informatica-full2.blogspot.com/2009/06/historia-del-lenguaje-c.html.

9. **Matias Simarro, Juan.***Aplicación del análisis de dependencias a la arquitectura software.* España : s.n, 2004.

10. **Lleonart Martín, Eva García-Menacho, Asunción.***Patrones.* Valencia, España : Universidad Politécnica de Valencia., 2007.

11. **Mary Shaw, David Garlan.***Software Architecture: Perspectives on an emerging discipline.* Upper Saddle River : Prentice Hall, 2009.

12. **Pressman, Roger S.***Software engineering : a practitioner's approach.* United States : p . c m. ISBN 978-0-07-337597-.

Bibliografía

1. **Jorge E. Faytong, Giancarlo J. Moggia, Boris X. Vintimilla Ph. D.** *Monitoreo Automático de Carreteras Mediante el Uso de un Sistema de Detección, Seguimiento y Extracción de Características Básicas de Vehículos con Técnicas de Visión por Computador.* Ecuador : s.n., 2009.

2. **Thanarat Horprasert, David Harwood, Larry S. Davis.** *A Robust Background Subtraction and Shadow Detection.* Maryland : s.n., 2002.

3. **David Armando Insuasti, Julián Quiroga, Alejandro Forero.** *Detección y Seguimiento de Vehículos Automotores en Video.* Bogota Colombia : s.n., 2008.

4. **Andrew Senior, Arun Hampapur, Ying-Li Tian.** *Appearance Models for Occlusion Handling.* New York : s.n., 2004.

5. **David Mora, Andrés Páez y Julián Quiroga Sepúlveda.** *Detección de Objetos Móviles en una Escena.* Bogota, Colombia : s.n., 2009.

6. **Martín, Sonsoles Herrero.** *Análisis comparativo de técnicas de segmentación de secuencias de video basadas en el modelado del fondo.* Madrid, España : s.n., 2009.

7. **Valle, Antonio Gutiérrez del.** *Videovigilancia y privacidad. La ocultación del rostro en vídeo para el cumplimiento del derecho a la intimidad.* Sevilla, España : s.n., 2011.

8. **Moya, Álvaro Rodríguez.** *Estudio del filtro de partículas aplicado al seguimiento de objetos en secuencias de imágenes.* Madrid, España : s.n., 2009.

9. **Gómez, Álvaro Bayona.** *Detección de objetos abandonados/robados en secuencias de vídeo-seguridad.* Madrid, España : s.n., 2009.

10. **Mosqueda, Rolando Payán.** *Video sensor de seguimiento de objetos para el sistema de vigilancia por cámaras.* Ciudad de La Habana, Cuba : s.n., 2009.

11. **Edmis Deivis Semanat Aldana, Fernando Echemendia Tourt, Reinier Pupo Rúiz.** *Sustracción de fondo en tiempo real utilizando GPUs.* Ciudad de La Habana, Cuba : s.n., 2011.

12. **Reinier Pupo Ruiz, Fernando EchemendiaTourt.** *Video sensor para el conteo de personas.* 2012.

13. **Jordi Sánchez, Ginés Benet, José E. Simó.** *Video Sensor Architecture for Surveillance Applications.* Valencia, España : s.n., 2012. 1424-8220.