Universidad de las Ciencias Informáticas

FACULTAD 6

Título: 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.

La Habana, Cuba.

"Diciembre 2012"

PENSAMIENTO.

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

Martin Golding

DECLARACIÓN DE AUTORÍA.

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

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

Ernesto A. Herrera Collazo.

_____

Firma del Autor

Ing. Fernando Echemendia Tour

_____

Firma del Tutor

DATOS DE CONTACTO.

Ing. Fernando Echemendia Tourt

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 2009. 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).

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.

RESUMEN

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) ha desarrollado 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, Video sensor, Video Vigilancia.

ABSTRACT

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.

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 (7).

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 (7).

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 lo tiene 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, a pesar de ser un gran avance en cuanto a producción de software nacional respecta, no llega a satisfacer las necesidades de empresas menores y pequeños comercios ya que este tiene un alto precio de adquisición.

En el Centro de Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI) se desarrolla un sistema de Video Vigilancia Suria, 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 la estimación de velocidad de vehículos.

Precisamente en este último aspecto está centrada la presente investigación, debido a que el sistema desplegado por Datys con respecto al control de tráfico vehicular, solamente cuenta con la identificación de matrículas. Xyma Safe Vision carece de una herramienta capaz de estimar la velocidad de vehículos y desaprovecha las ventajas que pudiese brindar el sistema en la automatización de multas, detección y prevención de infracciones del tránsito, a través del flujo de video que trasmiten las cámaras IP instaladas en las autopistas nacionales. Suria cuenta con varios videos sensores, pero ninguno está dedicado a estimar la velocidad de

vehículos. De instalarse dicho sistema, este no sería capaz de realizar esta función, y no se desempeñaría correctamente en el control de tráfico vehicular.

La situación problemática anteriormente expuesta permite plantear el siguiente **problema a resolver:** ¿Cómo controlar el tráfico vehicular a partir de flujos de video obtenidos de cámara 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 la estimación de velocidad de vehículos, se podrá controlar el tráfico vehicular con el Sistema de Video Vigilancia Suria.

Métodos de Investigación

Métodos teóricos.

- Analítico–sintético: Se utilizará para estudiar el problema con mayor profundidad y dar una solución adecuada al problema científico, comprender la evolución y desarrollo que han tenido los sistemas de video vigilancia, facilitará comprensión en el tema de los videos sensores para la estimación de velocidad de vehículos y permitirá extraer las características generales de cada uno de los sistemas estudiados.

Métodos Empíricos.

- Observación: Permitirá realizar valoraciones y obtener información a partir de la observación. Esto se manifiesta cuando se realiza el análisis del tráfico en las autopistas y el comportamiento de los automóviles en la vía, con el objetivo de inferir modelos matemáticos de movimiento, cambios de iluminación y objetos sin interés en las escenas, brindando una visión de cómo se debe desempeñar el video sensor.

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 desarrollo de un video sensor, capaz de realizar la estimación de velocidad de vehículos.

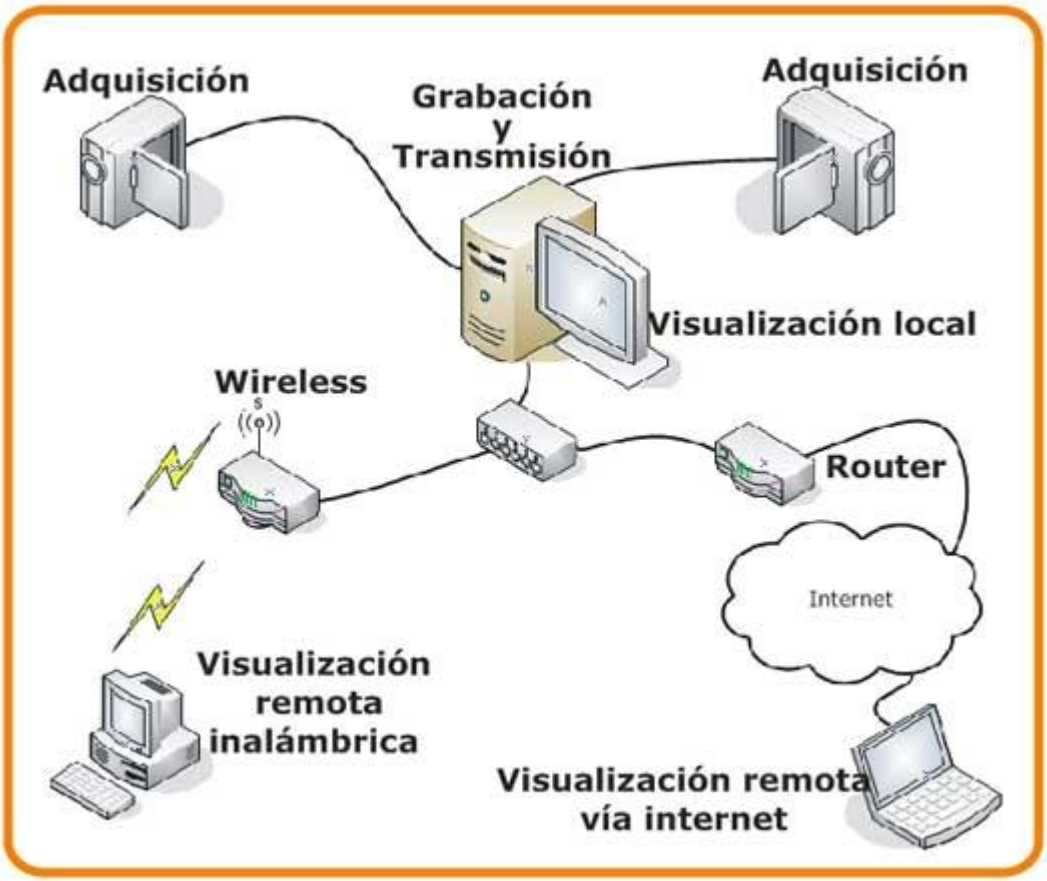
1.1 Conceptos asociados al dominio del problema.

Para mejor comprensión y una visión general de la investigación se 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 (10).

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, así como el procesamiento digital de las imágenes, para aplicaciones como el reconocimiento de matrículas, detección de objetos abandonados, estimación de velocidad de vehículos entre otras.



(15)

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 (14).

Figura 2: Cámara IP.

1.1.3 Sensor.

Dispositivo que detecta una determinada acción externa, temperatura, presión, movimiento, entre otros, y la transmite adecuadamente (8).

1.1.4 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 (7).

1.2.1 Flujos de Video.

“Un flujos de video o streaming es un vídeo que se reproduce al mismo tiempo que los datos llegan al ordenador a través de Internet. A diferencia del video por Internet para descargar, el video streaming comienza a reproducirse una vez que los datos comprimidos se reciben de una determinada aplicación, y elimina la preocupación de virus que pueden acompañar a las descargas” (5).

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

1.2 Técnicas, algoritmos y tendencias actuales.

Actualmente los sistemas de conteo de vehículos son muy caros y de alta complejidad para su implementación, así como los de sensores ubicados debajo de la carretera como Lazos

Inductivos, sensores Piezo-eléctricos, entre otros, además del alto costo del mantenimiento de estos sistemas el cual es mayor que el de una simple cámara de video.

En el panorama internacional existen sistemas 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 video sensores presentan las siguientes ventajas: menor costo en el mantenimiento, ofrecen mayor variedad de datos de tráfico, acceso a las grabaciones de videos, seguimiento de objetos, entre otras.

1.2.1 Sistemas Comerciales.

XYMA SAFE VISION

Es un sistema de video vigilancia basado en la tecnología IP, mediante cámaras de videos digitales conectadas a redes de datos. Admite además, cámaras analógicas mediante servidores de videos. Su objetivo es el monitoreo en tiempo real y la vigilancia de instalaciones y exteriores (9).

Está compuesto por varios módulos interconectados, que se comunican y comparten la base de datos donde se almacena el sistema. Entre sus prestaciones tiene la video vigilancia, desde los más diversos escenarios: monitoreo en autopistas, grabación de video con audio, revisión y manejo de grabaciones de videos, grabaciones de forma manual, programada, continua y por detección de movimiento, sistema de alertas, análisis inteligente de video y de reportes, con posibilidades de configuración (9).

En el escenario de monitoreo en las autopistas este sistema solo cuenta con un video sensor capaz de identificar las matrículas de los autos, esto se realiza a partir del procesamiento inteligente de los flujos de video obtenidos de las cámara IP conectadas al sistema. XYMA SAFE VISION no cuenta con video sensores capaces de detectar objetos abandonados y estimación de velocidad de vehículos, por lo que no realiza un control total sobre el tráfico en las autopistas.

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) (6).

Figura 4: Vídeo detección de tráfico

1.2.2 Técnicas de seguimiento de automóviles.

Filtro de Kalman

El filtro de Kalman consiste en un conjunto de ecuaciones matemáticas que proveen una solución recursiva óptima al problema de filtrado lineal de datos discretos, mediante el método de mínimos cuadrados. Se trata de una solución óptima en el sentido de que minimiza la covarianza estimada del error. La meta de esta solución consiste en calcular un estimador lineal y óptimo, del estado de un sistema en t con base en la información disponible en t-1, y actualizar, con la información adicional disponible en t, dichas estimaciones (4).

El filtro de Kalman es el principal algoritmo para estimar sistemas dinámicos representados en la forma de espacio de estados. En esta representación el sistema es descrito por un conjunto de variables denominadas de estado. El estado contiene toda la información relativa al sistema en un cierto instante de tiempo. Esta información debe permitir la inferencia del comportamiento pasado del sistema, con el objetivo de predecir su comportamiento futuro, aun cuando la naturaleza precisa del sistema modelado es desconocida (4).

Sistema de monitorización y control del tráfico en carretera.

Pablo F. Alcantarilla, propone una solución de detección de vehículos a través de visión por computador. Las principales etapas son:

Frame de Entrada: En esta fase se obtiene 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 o 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 (6).

Esquema del algoritmo:

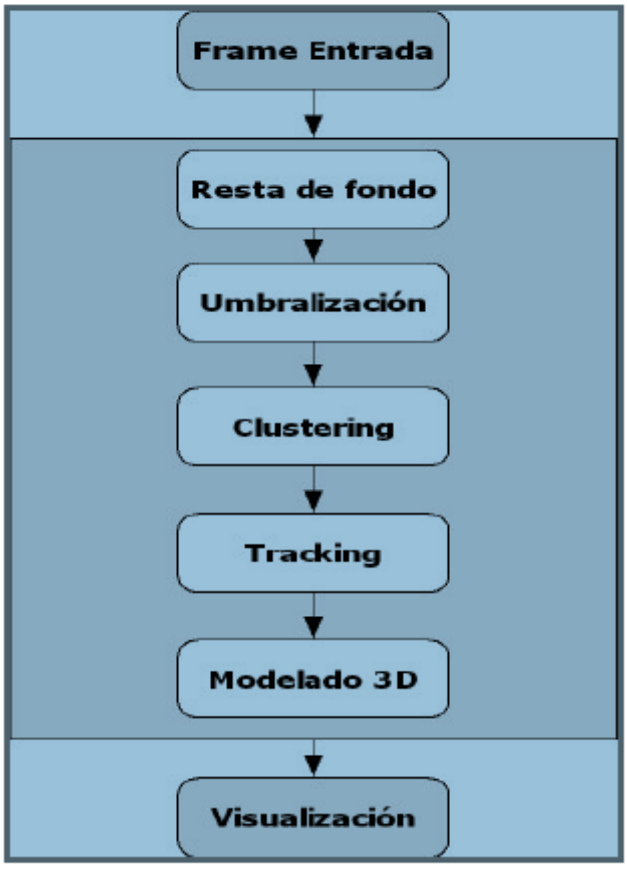


Figura 5: Esquema del algoritmo.

Este sistema utiliza en la fase resta de fondo dos algoritmos, con el estadístico se obtiene una imagen ruidosa y necesita que no existan vehículos en los primeros frame de entrada. Con el Suavizado Gaussiano se introducen mejoras en la imagen obtenida, permitiendo una umbralización óptima. En el proceso de Clustering se agrupan todos los píxeles que pertenezcan a un mismo vehículo y se realiza una clasificación según sus características a través del algoritmo Limpieza de Clusters. En la fase tracking se realiza el Filtro de Kalman, permitiendo emparejar correctamente cluster y tracking.

1.2.3 Algoritmos de modelado de fondo

Modelado de fondo.

El modelado de fondo, también denominado actualización de fondo, es la etapa fundamental en los algoritmos de segmentación de objetos en movimiento. Su función es la inicialización, actualización y representación de un modelo de fondo robusto de la secuencia de video analizada. El fondo se describe mediante un modelo matemático, para cada píxel de la imagen en cada instante de tiempo. El objetivo de esta etapa no es encontrar la información relativa a los objetos en movimiento, ya que esta tarea pertenece a una etapa posterior llamada extracción de frente (3).

Una primera aproximación para modelar el fondo de una escena sería suponer que los píxeles de fondo no cambian de valor a lo largo de la secuencia, es decir, sólo los objetos en movimiento se modificarán en los correspondientes píxeles. Esta suposición se suele realizar en entornos muy controlados (por ejemplo, interiores con una imagen de fondo conocida y estática y sin variaciones de iluminación) y requiere únicamente modelar factores externos como el ruido de la cámara. Este tipo de fondos se conoce como fondos unimodales. En cambio, en entornos menos controlados, los píxeles del fondo pueden cambiar su valor debido a cambios de iluminación (como sucede, por ejemplo, en escenas capturadas al aire libre y a distintas horas del día) o debido al movimiento de los objetos que pertenecen al fondo (como ocurre en escenas con árboles agitándose u olas de mar). Este tipo de fondos se conoce con el nombre de multimodales.

Modelos Básicos.

Los métodos básicos de sustracción de fondo utilizan modelos matemáticos sencillos tales como, diferencias entre imágenes, promedios, máximos y mínimos, entre otros, que permiten modelar simplificadamente los píxeles de la imagen de fondo (3).

- Diferencia de imágenes (*Frame differencing*).

  El método de diferencia de imágenes, también llamado diferencia temporal, es posiblemente la forma más sencilla de sustracción de fondo. Utiliza como modelo de fondo B_t para la imagen de la secuencia en el instante I_t, la imagen anterior, es decir, la imagen del instante de tiempo t −1. Como ventajas, esta técnica posee una baja carga computacional, y el frente se adapta bien a los cambios rápidos (3).

  Esta técnica es sensible al ruido y a las variaciones de iluminación. Además, debido a que sólo utiliza una única imagen anterior como modelo de fondo, la diferencia con el cuadro anterior no es capaz de identificar el interior de los objetos en movimiento, puesto que entre imágenes consecutivas, esos píxeles se mantienen invariantes.

- Filtro promedio temporal (*Average filter*).

  En el filtro de promedio, el modelo de fondo B_t se calcula como una imagen estática hasta que se producen movimientos, en cuyo caso, el B_t corresponde al promedio de un conjunto de imágenes consecutivas en el tiempo (3).

  Este modelo no es robusto en secuencias con muchos objetos en movimiento sobre todo si se mueven lentamente. Además, no puede manejar fondos multimodales, adapta lentamente las variaciones del fondo y posee un único umbral para toda la imagen.

Modelos paramétricos.

Los algoritmos basados en modelos paramétricos definen modelos de fondo más complejos, que permiten cierta tolerancia al ruido y a pequeñas fluctuaciones (hojas en movimiento, parpadeo de luces, pequeñas vibraciones de las cámaras y cambios bruscos de luz). Describen la imagen de fondo en base a parámetros de una distribución de probabilidad estándar usualmente Gaussiana. Algunos de estos ejemplos son los que se muestran a continuación:

- Gaussiana simple (*Simple Gaussian*).

El método de la Gaussiana Simple (SG) modela los pequeños cambios que ocurren en la imagen de fondo B_t representando cada píxel con una distribución unimodal Gaussiana definida por dos parámetros: media μ_t y varianza σ_t^2 (3). Este método no es capaz de adaptarse a fondos multimodales, en los que cada píxel de fondo puede tomar valores muy diferentes, sin por ello, dejar de ser un píxel de fondo.

- Mezcla de Gaussianas (*Mixture of Gaussian*).

En fondos multimodales, que contienen objetos no estáticos, tales como hojas de árboles en movimiento, olas, entre otras, hay píxeles cuyos valores de intensidad varían entorno a un conjunto finito de valores característicos. Por este motivo, un píxel no puede modelarse por medio de un valor (una media) y un conjunto en torno a éste (la varianza) utilizando una distribución Gaussiana. La Mezcla de Gaussianas (MoG) propone una solución a este problema que consiste en modelar la intensidad de los píxeles con una mezcla de k distribuciones Gaussianas (donde k es un número pequeño, frecuentemente se utiliza de 3 a 5) definidas por los siguientes parámetros: media $\mu_{k,t}$, varianza $\sigma_{k,t}^2$ y peso $w_{k,t}$. No obstante, la MoG también posee inconvenientes. En primer lugar, conlleva una alta carga computacional. Por otro lado, es muy poco robusta a cambios repentinos de iluminación. Además, los fondos multimodales requieren un número de distribuciones k elevado para modelar cada píxel, lo cual implica un incremento en la carga computacional. En este método de representación del fondo es muy importante la forma de actualizar las medias y las varianzas para adaptarse a los cambios del fondo (3).

A pesar de estos inconvenientes, la MoG es capaz de manejar una distribución multimodal de fondo ya que mantiene una función de densidad de probabilidad para cada píxel. Al ser un método paramétrico, puede adaptarse a los cambios de fondo sin necesidad de actualizar un gran buffer de almacenamiento de imágenes como, según se verá, requieren los métodos no paramétricos.

Para la inicialización de los parámetros es necesario tomar una serie de decisiones iniciales; en este caso, se ha 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 se considera 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.

La estimación del fondo y actualización de parámetros se realiza 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.

Modelos no paramétricos

Los modelos no paramétricos también son métodos complejos en los que no se asumen distribuciones estándar de probabilidad para modelar a los píxeles de fondo, sino técnicas más generales, como pueden ser almacenamiento de los últimos valores del píxel, cálculo de rangos de valores del píxel o ajustes de funciones de predicción. Algunos modelos no paramétricos son los siguientes:

- Densidad de Núcleo (*Kernel Density Estimat*).

El método de representación del fondo Densidad de Núcleo (KDE) estima la función de densidad de probabilidad de cada píxel de la imagen de fondo en cada instante de tiempo. Esta operación se realiza gracias a la información de la historia reciente de dicho píxel que se haya almacenada en un buffer. El objetivo es obtener mayor sensibilidad de detección que utilizando un método de representación de fondo con una distribución de probabilidad fija (3).

El modelo KDE soporta parpadeo de fondo, ruido en la imagen y es capaz de adaptarse a los cambios rápidos y progresivos del fondo. No obstante, posee un requisito muy importante que es la alta carga computacional.

- Modelos ocultos de Markov (*HMM*).

Los modelos estudiados en los apartados anteriores pueden adaptarse a cambios graduales en la iluminación, no obstante, un cambio súbito puede representar un gran problema.

Otro enfoque para modelar una amplia gama de variaciones en la intensidad de los píxeles sería utilizando los modelos ocultos de Markov (HMM) que representan estas variaciones como un conjunto de estados discretos correspondientes a los distintos modos de iluminación que pueden presentarse en la escena: luces de encendido, apagado, nublado o soleado. Por ejemplo, en el seguimiento de tráfico la intensidad de un píxel se puede modelar con tres estados de distribución Gaussiana: fondo, frente y sombra (3).

Los modelos de representación del fondo basado en HMMs requieren un tiempo de cálculo elevado ya que suponen la evaluación de un conjunto de estados, y la topología del sistema puede ser muy compleja según las características de la escena.

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 (16).

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 (16).

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 (16).

1.3.2 Fundamentación de la metodología.

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. La metodología de desarrollo antes expuesta ha sido seleccionada para guiar todo el proceso de desarrollo de software del producto SURIA por las siguientes razones:

- El soporte que se brinda a los expedientes de proyectos desarrollados con esta metodología por parte del equipo de calidad, el cual se encarga de mantener actualizados dichos expedientes proporcionándole a los desarrolladores las plantillas de los distintos documentos a desarrollar.
- Presenta herramientas de apoyo en cada fase del proyecto. Se centra en la producción y tiene un alto potencial de organización.
- Realiza la evaluación en cada fase, permitiendo cambios de objetivos, entre más temprano se detecten los cambios menos costoso serán para el proyecto.
- RUP solamente implementa lo se encuentra en los casos de uso, permitiendo concentrar el desarrollo y aumenta la productividad.
- RUP realiza las pruebas al final del producto al contrario de XP que está constantemente realizando pruebas.

1.3.3 El Lenguaje Unificado de Modelado (UML).

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 (17).

1.3.4 Herramienta de modelado.

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 (18).

1.4.1 Fundamentación de la herramienta de modelación.

Para modelar todos los artefactos de RUP se ha seleccionado Enterprise Architect 7.0 como herramienta de modelado, por ser una plataforma de diseño, administración y colaborativa, basada en UML 2.1 y estándares relacionados. También por ser una herramienta ágil, intuitiva

y extensible, con poderosas características para dominios específicos totalmente integrados. Un aspecto de vital importancia que se tuvo en cuenta para dicha selección es que ofrece salida de documentación flexible y de alta calidad. Además posee todas las funcionalidades del Visual Paradigm. Desde los inicios el sistema de Video Vigilancia SURIA utiliza esta herramienta de modelado ya que fue seleccionada por el arquitecto del proyecto por sus características.

1.4 Lenguaje 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.

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 (10).

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++ (10).

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 (10).

1.4.2 Fundamentación del Lenguaje Seleccionado.

Para llevar a cabo el desarrollo del video sensor para la estimación de velocidad de vehículos, se ha seleccionado a C++ como lenguaje de programación, es potente, compila a través de punteros a memoria que permiten mayor rapidez, es el lenguaje por excelencia para el desarrollo de aplicaciones de procesamiento de imágenes, está estandarizado y un mismo código fuente se puede compilar en diversas plataformas, presenta gran versatilidad es decir, es un lenguaje de propósito general, por lo que se puede emplear para resolver cualquier tipo de problema.

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 (20).

1.5.1 Qt Creator.

Qt Creator es un IDE (Entorno de desarrollo integrado) creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt. Soporta sistemas operativos como GNU/Linux, Mac OS X, Windows XP y Vista, por lo que es multiplataforma y software libre. Permite construir interfaces de usuario complejas de una forma visual y rápida ya que incluye un editor de texto con autocompletado, diseñador de interfaces gráficas, gestión de proyectos, sistema de depuración e integración con sistemas de control de versiones (19).

Qt Creator se integra con C++ usando las librerías de Qt, abastece no sólo a los desarrolladores que están acostumbrados a utilizar el ratón, sino también a los que se sienten

más cómodos con el teclado, con una amplia gama de métodos abreviados de teclado y navegación, que están disponibles para ayudar a acelerar el proceso de desarrollo de una aplicación (19).

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

Para la implementación del sistema se decidió usar como IDE el Qt Creator debido a las múltiples ventajas para la programación con C++, las cuales se exponen a continuación:

Tiene plugins para varios sistemas de control de versiones.

Posee una herramienta de búsqueda eficaz donde se puede buscar fácilmente las clases, métodos y archivos.

- Depurador visual para C++, es consciente de la estructura de muchas clases de Qt, lo que aumenta la capacidad de mostrar los datos de Qt con claridad.
- Soporte para refactorización de código.
- Se integra perfectamente con el lenguaje C++.

1.5 Biblioteca de procesamiento de imágenes.

OpenCV 2.1.0.

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, 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 (1).

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”.

1.6 Conclusiones Parciales.

Las soluciones existentes en la actualidad referente a los sistemas de video vigilancia que incorporan la estimación de velocidad de vehículos son eficientes pero limitadas, además de ser propietarias son altamente costosas y no todas las personas pueden tener acceso a esta herramienta. El empleo de la metodología de desarrollo RUP, la selección de la herramienta Case, el lenguaje de programación C++, con el entorno de desarrollo integrado escogido, la selección de la biblioteca a utilizar y el estudio de los algoritmos de modelado de fondo, permiten la creación de un video sensor para la estimación de velocidad de vehículos con la garantía de ser altamente robusto y seguro. Una vez realizado el capítulo quedan consolidadas 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 (16).

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 (14).

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 (14).

2.1.1 Conceptos y eventos principales asociados al entorno.

Vehículo: Medio de transporte de personas u objetos.
Cámara IP: cámara que emite las imágenes directamente a la red.

Flujo de video: Es una secuencia de video que se reproduce al mismo tiempo que los datos llegan al ordenador a través de Internet.

1.1.2 Diagrama de clases del modelo de dominio

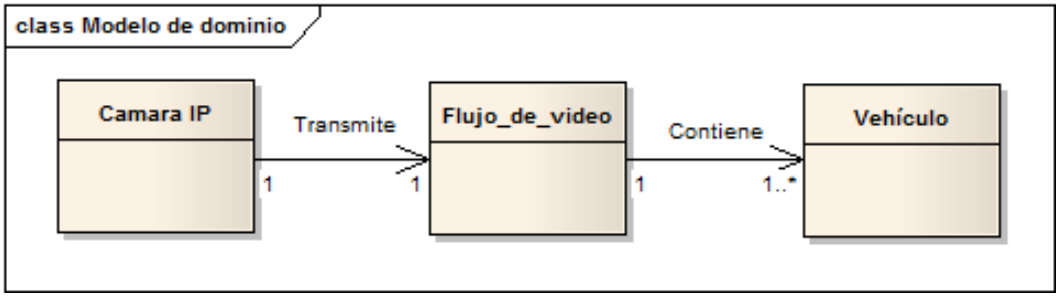


Figura 6: Diagrama de Clases del Modelo de dominio.

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:

RF 1 Analizar un flujo de video.

RF 2 Definir zona donde se va a realizar el cálculo de velocidad.

RF 3 Seguir Vehículos.

RF 4 Calcular la velocidad de los vehículos.

RF 5 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 Eficiencia.

- **RNF 1** El algoritmo debe procesar entre 25 y 30 fotogramas por segundo para realizar procesamiento en tiempo real.

2.3.2 Software.

- **RNF 2** Usar como sistema operativo Ubuntu 10.04 o superior.

2.3.3 Hardware.

- **RNF 3** Se requiere que las PC tengan tarjeta de red.
- **RNF 4** Memoria RAM de 1 GB o más.
- **RNF 5** Se requiere al menos 80 GB de disco duro.
- **RNF 6** Se requiere de un procesador de Dual Core 2.6 GHz, 800 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.

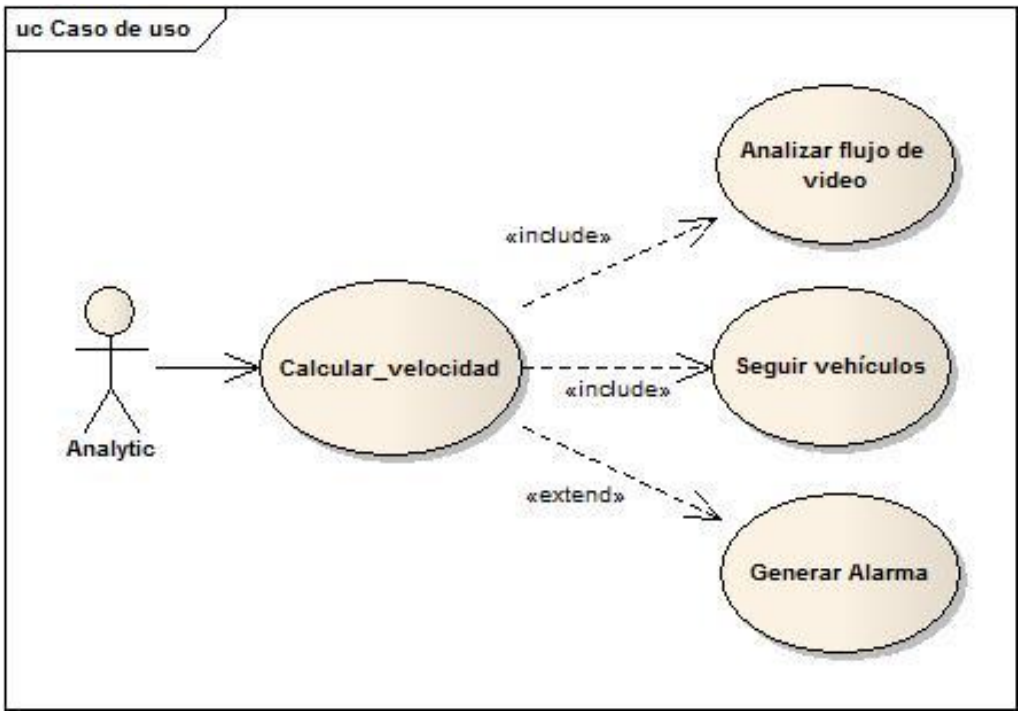


Figura 7: Diagrama de Casos de Uso del Sistema.

2.4.1 Definición de los actores

Actor	Descripción

Analytic	Es un actor ficticio que se ocupa de gestionar el procesamiento inteligente de video, a través de los videos sensores, realiza la obtención del flujo de video de una cámara IP y realizar petición de calcular la velocidad.

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	Analytic
Propósito	Este caso de uso tiene como objetivo analizar el flujo de video obtenido, detectar los objetos que aparezcan en la imagen y encerrar los vehículos en un rectángulo.
Resumen	El caso de uso inicia cuando el módulo Analytic del Sistema de Video Vigilancia hace la petición de cálculo de velocidad de vehículos, por lo que el sistema captura un video y luego extrae de él los fotogramas, define una zona, 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	Que exista un flujo de video.
Referencias	RF 1, RF2
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 y concluye el caso de uso.
Pos condiciones	Se obtienen todos los objetos del fotograma

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

Caso de Uso Seguir vehículo

Caso de Uso	Seguir vehículo
Actores	Analytic
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	Que existan vehículos en la escena.

Referencias	RF 2	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor		Respuesta del Sistema
1. El caso de uso inicia cuando el sistema 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 dos líneas horizontales en la región seleccionada.
		3. El sistema extrae el movimiento de los vehículos detectados en la escena.
		4. El sistema realiza el seguimiento de los vehículos detectados en la escena y concluye el caso de uso.
Pos condiciones	Se obtienen todos los vehículos y datos necesarios para el cálculo de velocidad.	

Tabla 3: Caso de Uso Seguir vehículo.

Caso de Uso Calcular Velocidad.

Caso de Uso	Calcular Velocidad
Actores	Analytic
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 sistema define 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	Que se esté realizando el seguimiento a vehículos que contenga la escena.
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 y concluye el caso de uso.
Pos condiciones	Se obtienen las velocidades de los vehículos analizados.

Tabla 4: Caso de Uso Calcular Velocidad.

Caso de Uso Generar alarma

Caso de Uso	Generar alarma
Actores	Video sensor
Propósito	Este caso de uso tiene como objetivo lanzar una 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 una alarma "*La velocidad está por encima de lo*

	debido”.	
Precondiciones	Que la velocidad del vehículo que se encuentre en la escena, este por encima del límite de velocidad establecido.	
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.		2. El sistema verifica que esas velocidades estén según lo establecido.
		3. Cuando exista 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*” y concluye el caso de uso.
Pos condiciones	Se genera una alarma en caso de que la velocidad estimada del vehículo este por encima del límite establecido.	

Tabla 5: Caso de Uso Generar alarma.

2.5 Conclusiones parciales.

Con el apoyo de los requisitos funcionales y la obtención de los casos de usos se logra una mejor comprensión de que debe hacer el sistema, Se definieron los casos de uso del sistema y la descripción de los mismos para lograr una mejor comprensión del sistema propuesto. Una vez realizado el capítulo quedan consolidadas las bases para comenzar el desarrollo del video sensor.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

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 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. 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 (12).

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. 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 (12).

El video sensor a desarrollar, presenta una arquitectura centrada en los flujos de datos, fundamentada en el estilo tubería-filtros que se considera como una serie de transformaciones sucesivas sobre datos de entrada. Los datos se transportan a través de las tuberías entre los filtros, estos transforman gradualmente las entradas en salidas para hacer cumplir requerimientos funcionales y no funcionales.

La elección de este estilo de arquitectura, se basa en que un filtro recibe como entrada un flujo de video y produce un flujo de datos transformados a su salida. Esto se aplica cuando el filtro Analizar flujo de video, procesa el flujo de video obtenido de la cámara IP, enviando los fotogramas al siguiente filtro, el cual es encargado de extraer los vehículos para su posterior seguimiento. El filtro Seguimiento de vehículo después de realizar esta operación envía los vehículos para realizar la estimación de velocidad.

3.1.1 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 (14).

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 al 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 (14).

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.

Esto se ve en la clase VS_Control que es la encargada de asignar las responsabilidades dentro del video sensor, es decir, posee los atributos necesarios para la selección de instancias de acuerdo con el algoritmo.

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.

Esto se aprecia en la clase VS_Control que contiene objetos de la clase Vehículo, por lo que es capaz de asumir la responsabilidad de crear instancias de la clase Vehículo.

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.

Esto se aprecia en la clase VS_Control la cual es encargada de ejecutar el algoritmo utilizando funcionalidades de las clases Vehículo y Zona_detección.

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.

Esto se manifiesta al no asignarle todas las responsabilidades a la clase VS_Control, sino que los métodos y atributos necesarios para darle la funcionalidad a la clase VS_Control, lo tienen las clases Vehículo y Zona_detección.

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 Diagramas de clases del 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 análisis del CU (Caso de Uso) Seguir vehículos.

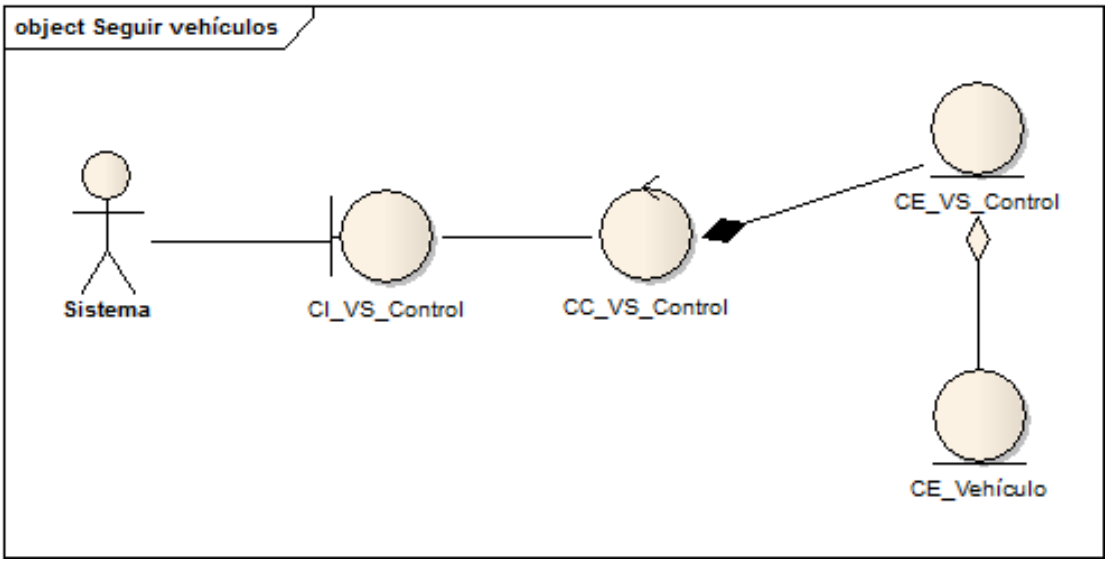


Figura 8: Diagrama de clases de análisis CU Seguir vehículos.

En la figura 8 se muestra el diagrama de clases del análisis para el caso de uso Seguir vehículos, se pueden apreciar las clases entidades CE_VS_Control, CE_Vehiculo y la clase control CC_VS_Control.

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

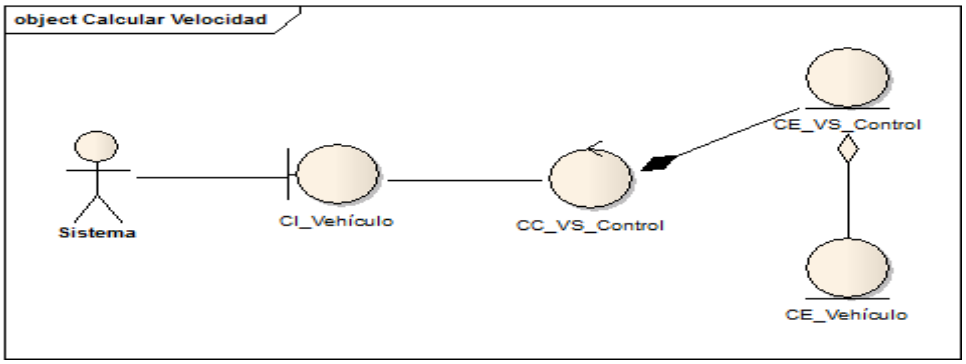


Figura 9: Diagrama de clases de análisis CU Calcular Velocidad.

En la figura 9 se muestra el diagrama de clases del análisis para el caso de uso Calcular Velocidad, se pueden apreciar las clases entidades CE_VS_Control, CE_Vehiculo y la clase control CC_VS_Control.

Diagrama de clase de análisis del CU (Caso de Uso) Generar Alarma.

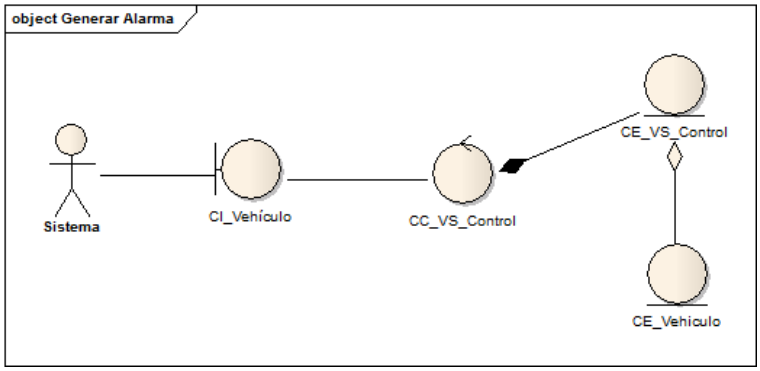


Figura 10: Diagrama de clases de análisis CU Generar Alarma.

En la figura 10 se muestra el diagrama de clases del análisis para el caso de uso Generar Alarma, se pueden apreciar las clases entidades CE_VS_Control, CE_Vehiculo y la clase control CC_VS_Control.

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

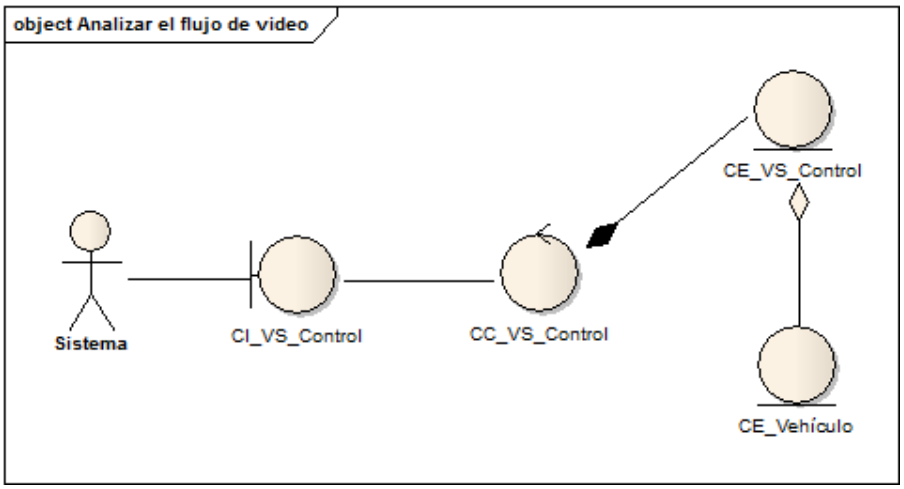


Figura 11: Diagrama de clases de análisis CU Analizar el flujo de video.

En la figura 11 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_ VS_Control, CE_Vehículo 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.

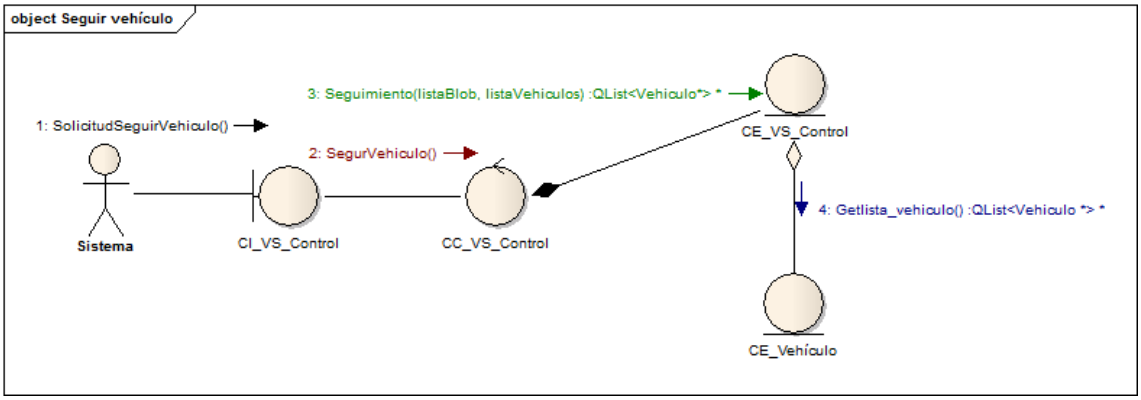


Figura 12: Diagrama de colaboración CU Seguir Vehículo.

En la figura 12 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.

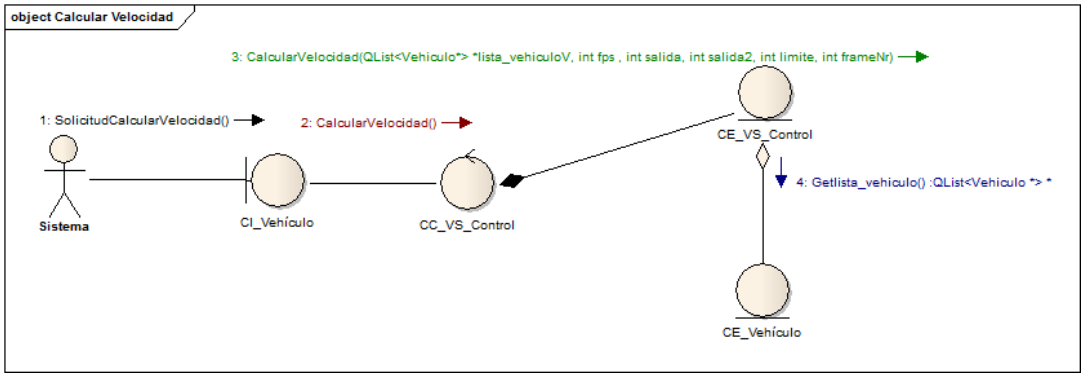


Figura 13: Diagrama de colaboración CU Calcular velocidad.

En la figura 13 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.

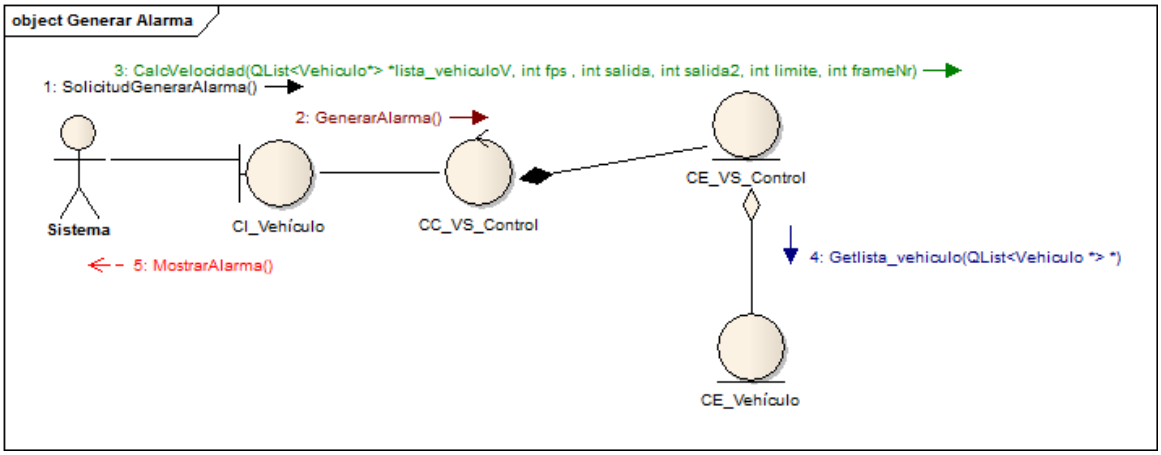


Figura 14: Diagrama de colaboración CU Generar Alarma.

En la figura 14 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.

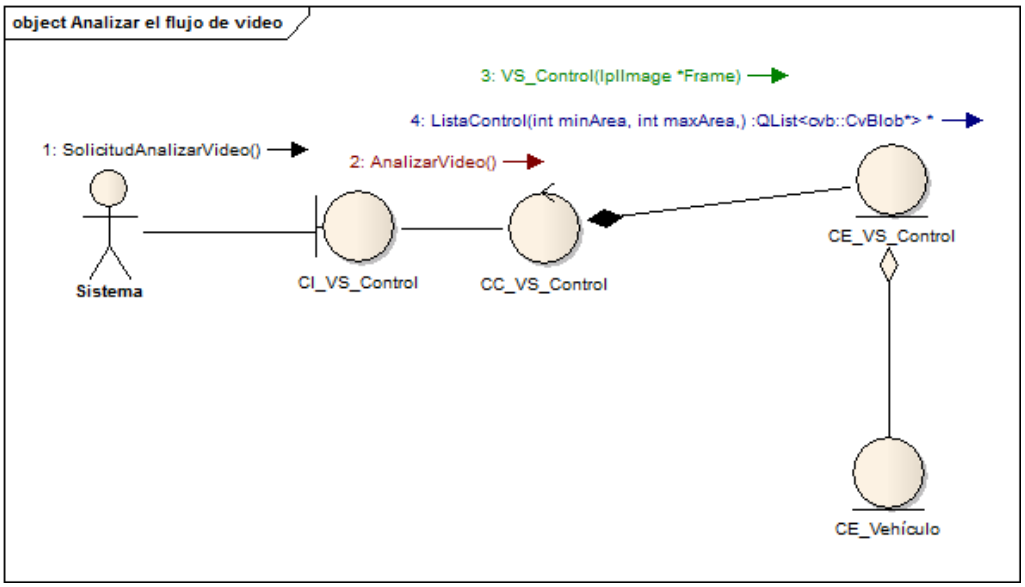


Figura 15: Diagrama de colaboración CU Obtener y analizar el Flujo de video.

En la figura 15 se muestra el diagrama de colaboración para el caso de uso Aanalizar 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 Diagramas 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.

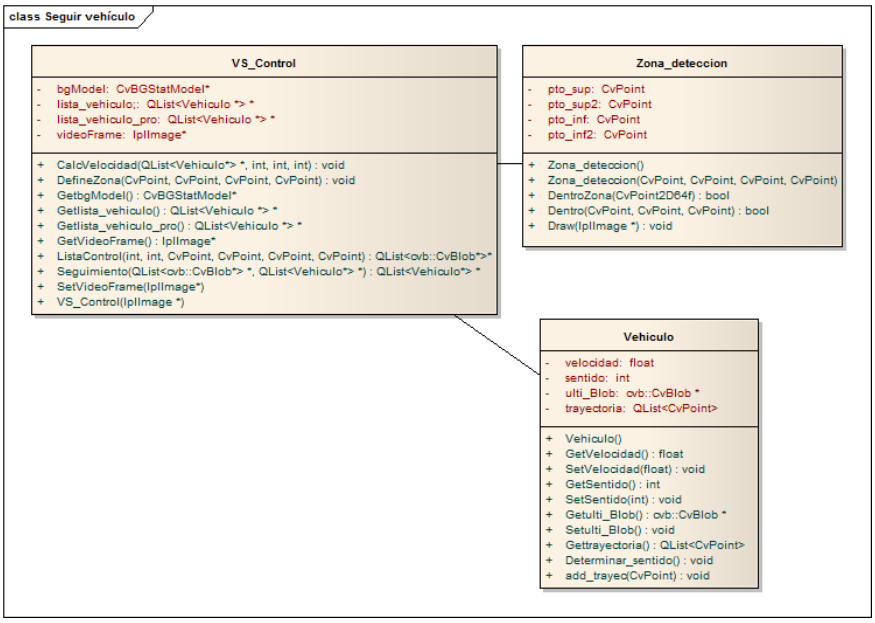


Figura 16: Diagrama clases del diseño del CU Seguir Vehículo.

En la figura 16 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.

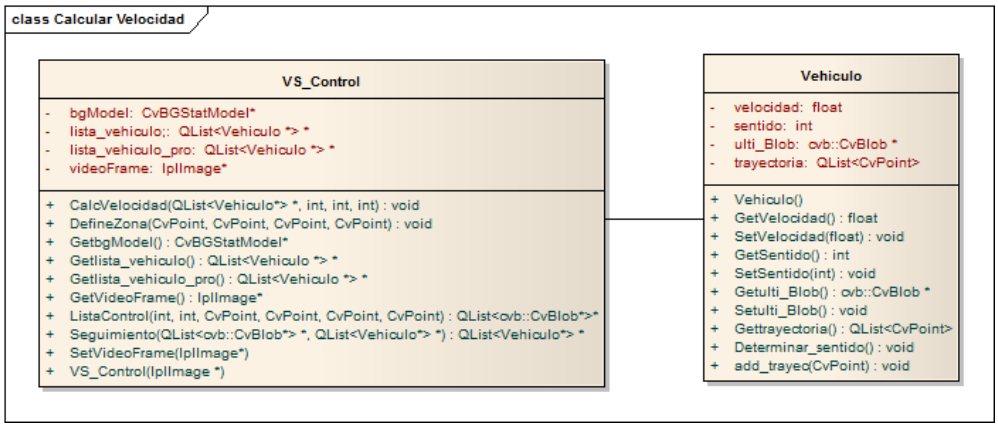


Figura 17: Diagrama clases del diseño del CU Calcular Velocidad.

En la figura 17 se muestra el diagrama de clases del diseñodel CU Calcular Velocidad. La clase VS_Controles la encargada de interactuar con la clase 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 Generar Alarma.

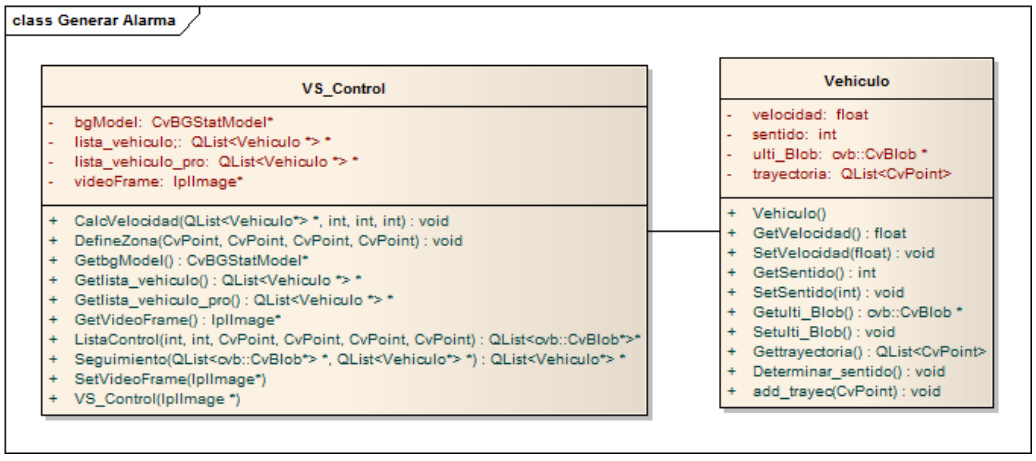


Figura 18: Diagrama clases del diseño del CU Generar Alarma.

En la figura 18 se muestra el diagrama de clases del diseño del CU Generar Alarma.La clase VS_Control es la encargada de interactuar con la clase 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 Analizar el flujo de video.

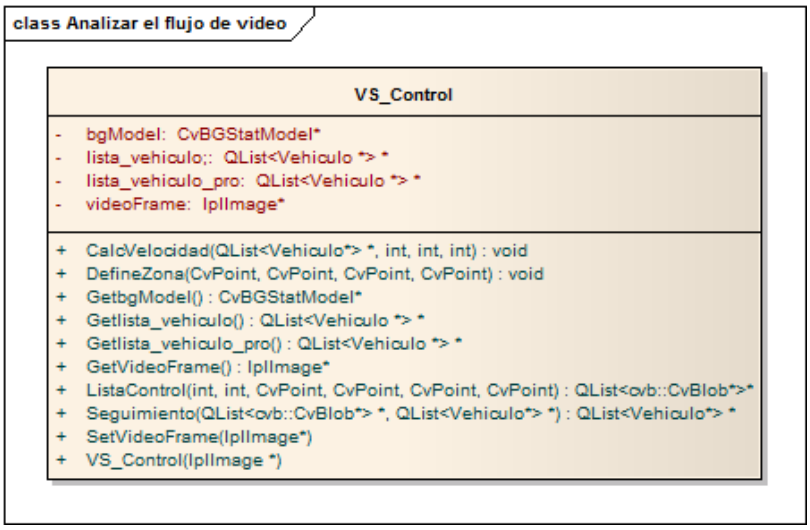


Figura 19: Diagrama clases del diseño del CU Obtener y analizar el Flujo de video.

En la figura 19 se muestra el diagrama de clases del diseño del CU Obtener y analizar el Flujo de video. La clase VS_Control es la encargada de analisar el flujo de video y brindarle a esta las funcionalidades para del componente. 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

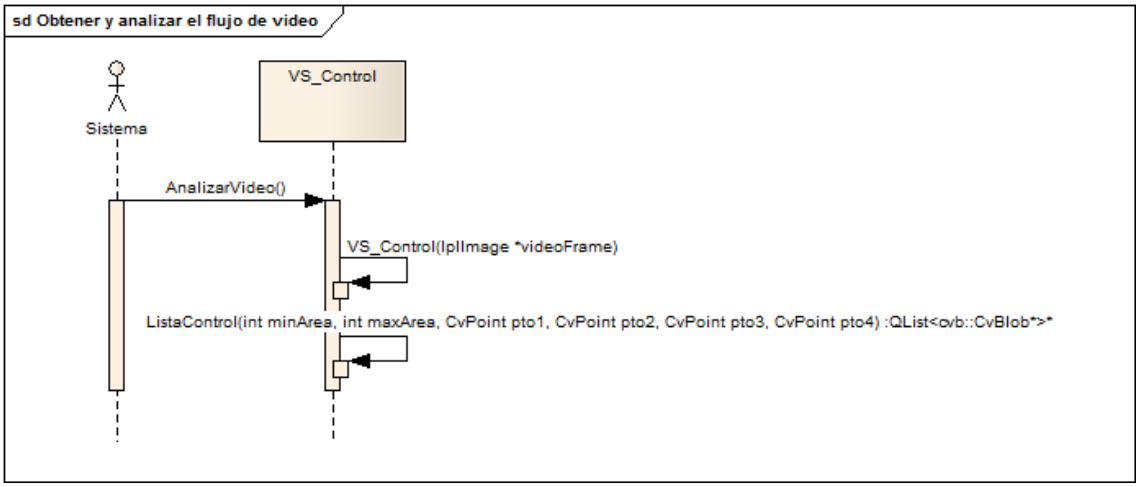


Figura 20: Diagrama de secuencia del CU Analizar el flujo de video.

En la figura 20 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.

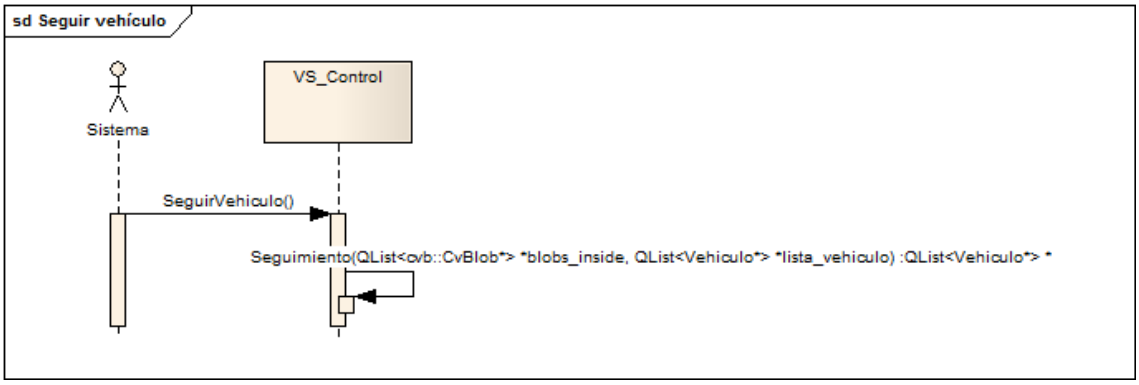


Figura 21: Diagrama de secuencia del CU Seguir vehículo.

En la figura 21 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.

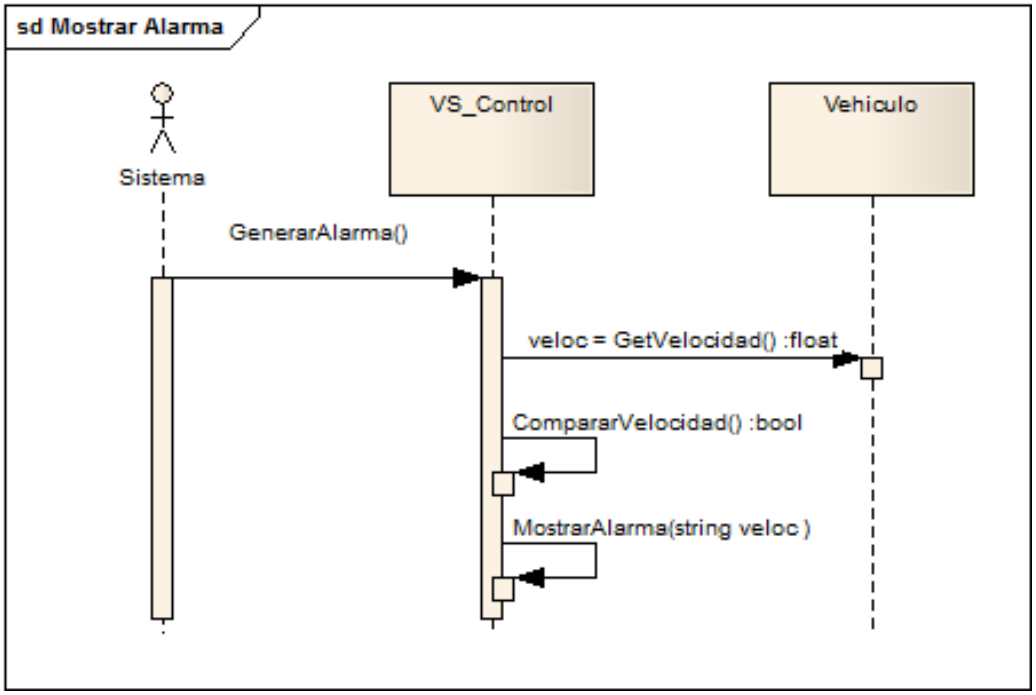


Figura 22: Diagrama de secuencia del CU Mostar Alarma.

En la figura 22 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.

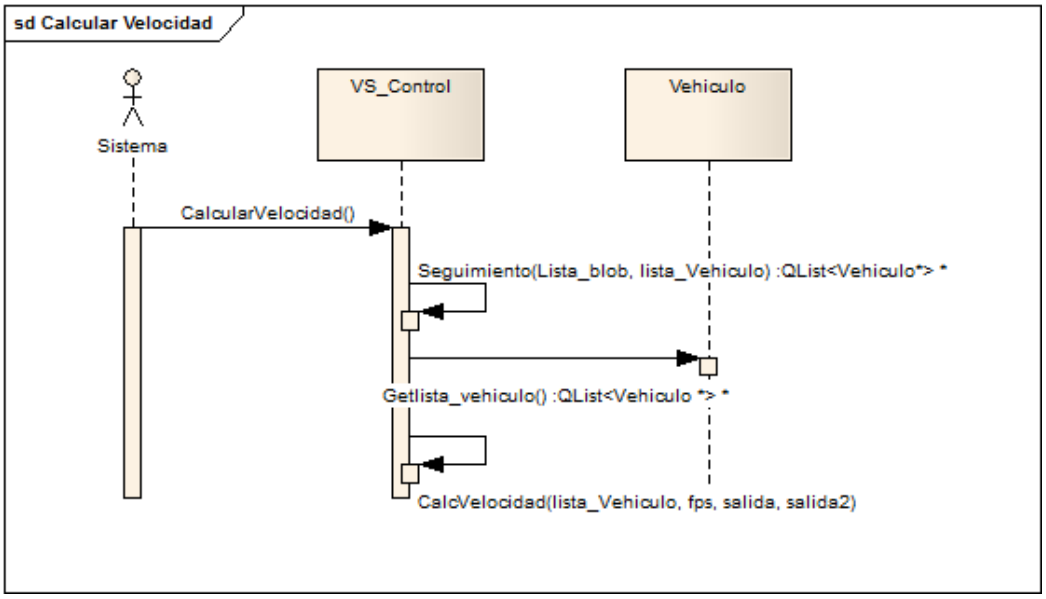


Figura 23: Diagrama de secuencia del CU Calcular Velocidad.

En la figura 23 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 la clase 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<Vehículo*> *listaV): QList<Vehículo*>
Descripción	Realiza el procesamiento de los blob para determinar a qué vehículo corresponden los frames procesados.
Nombre	CalcVelocidad(QList<Vehículo*> *listaV, int FPS):void
Descripción	Realiza el cálculo de la velocidad del vehículo.

Tabla 6: Descripción de la clase de Video sensor Control.

Descripción de la clase 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 la clase Vehículo.

Descripción de la clase 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 la clase del Zona de detección.

3.5 Conclusiones parciales.

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

CAPÍTULO 4: ALGORITMOS

Introducción.

En este capítulo se explicarán cada una de las fases más importantes de las que consta el algoritmo desarrollado para la estimación de la 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), el cual es uno de los métodos complejos más utilizado (2), 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.

4.2 Algoritmo desarrollado.

Después del estudio realizado en el capítulo 1de las diferentes técnicas y algoritmos que permiten el seguimiento y estimación de velocidad, se presenta el algoritmo desarrollado. Podemos ver un esquema del algoritmo en la figura 24.

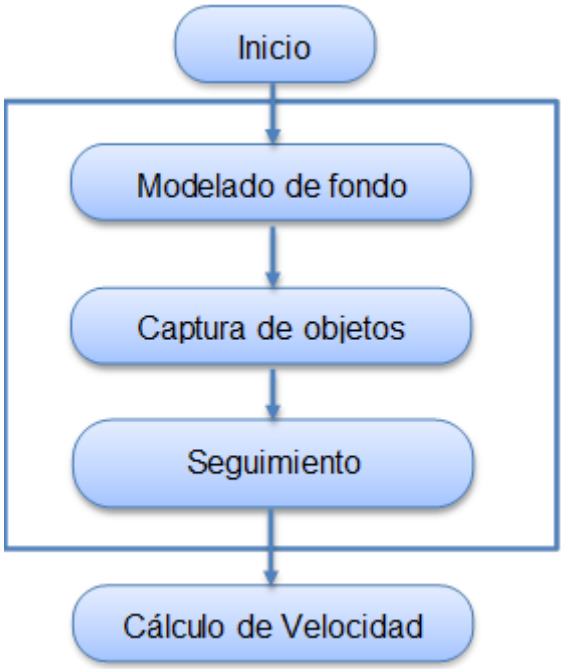


Figura 24: Algoritmo desarrollado.

Básicamente, cada una de las etapas anteriores consiste en lo siguiente:

Inicio: En esta fase se capturan los fotogramas que transmite la cámara IP, para ello se utiliza de la biblioteca de código OpenCV la funcionalidad cvCaptureFromCAM. Esta funcionalidad asigna a CvCapture una estructura que permite la unión a la cámara de vídeo.

Modelado de fondo: En esta fase se inicializan, actualizan y se representa un modelo de fondo robusto de la secuencia de video analizada, siendo la etapa fundamental en los algoritmos de segmentación de objetos en movimiento. Existen varias formas de realizar el modelado de fondo, en este caso se utiliza un modelo paramétrico que define modelos de fondo más complejos y permiten cierta tolerancia a pequeños movimientos. El algoritmo desarrollado utiliza la Mezcla de Gaussianas la cual modela la intensidad de los píxeles con una mezcla de 7 distribuciones Gaussianas definidas por los siguientes parámetros: media, varianza y peso. Con este proceso se determinan los píxeles correspondientes al fondo y frente del fotograma, los cuales se van actualizando en correspondencia con el flujo de video.

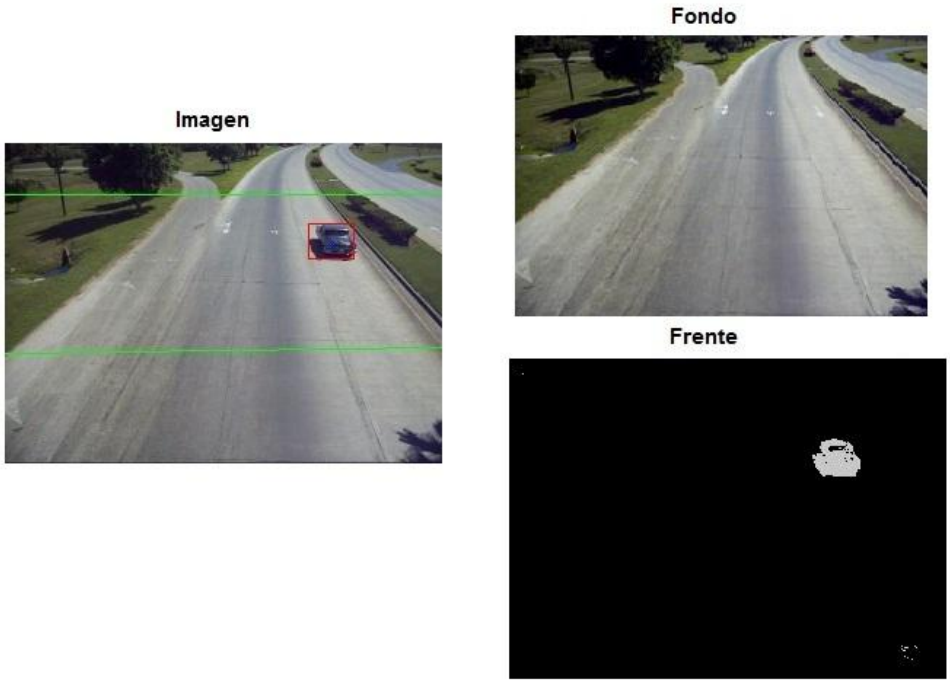


Figura 25: Imagen, modelo MoG del Fondo y Frente de la secuencia.

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.

Captura de objetos: El proceso de captura se realiza a partir del análisis de los píxeles del frente de la imagen, estos son previamente obtenidos del modelado del fondo. A estos píxeles se les realiza un análisis de las componentes conexas a través de la biblioteca de código cvblob, permitiendo detectar, capturar y enmarcar los vehículos en la secuencia de video. En esta fase se extraen características de los vehículos, como son: el centro de gravedad o centroide, el área que ocupan los píxeles en la imagen, entre otras.

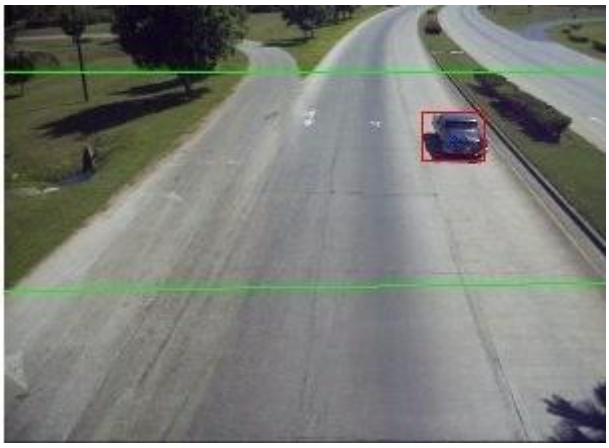


Figura 26: Captura de objetos

En la figura se muestra la imagen, el frente de la secuencia en el instante 63, la imagen muestra un vehículo enmarcado en un rectángulo rojo.

Seguimiento: 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 la 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 en la zona de detección y seguimiento.

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 cuenta con un centroide, área y si se le está realizando seguimiento, 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 y existe un solapamiento entre los blob, entonces el vehículo 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 vehículo, 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 27: Seguimiento de objetos.

Cálculo de Velocidad: La velocidad es la magnitud física de carácter vectorial que determina el recorrido de un objeto en una distancia y espacio de tiempo determinado, 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, el tiempo se determina por la división entre la cantidad de fotogramas que estuvo en seguimiento el vehículo y la cantidad de fps que la cámara realiza la captura, estableciendo así el período de tiempo necesario para determinar esta magnitud física. Una vez obtenido los datos necesarios, se calcula la velocidad, se actualizan las características del vehículo y se mueven estos a una lista de vehículos ya procesados. De esta lista se pueden obtener la trayectoria que describió un vehículo, la velocidad y el tiempo que se mantuvo en seguimiento.

4.3 Resultados del algoritmo.

Nombre de la sección	Descripción de la funcionalidad	Resultados Esperados	Resultados Obtenidos
Procesamiento en tiempo real.	El objetivo es realizar el procesamiento del video en tiempo real.	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 (para cada fotograma se necesitaran 0.0054 s).
Análisis de	El objetivo es	**Frames 1** 4 objetos 3 de	2 vehículos.

fotogramas.	analizar la cantidad de objetos que detecta el algoritmo, determinado de estos los vehículos.	**al 100**	ellos vehículos.	
		Frames 101 al 200	3 objetos 2 de ellos vehículos.	2 vehículos.
		Frames 201 al 300	3 objetos 1 de ellos vehículos.	1 vehículos.
		Frames 301 al 400	2 objetos 2 de ellos vehículos.	2 vehículos.
		Frames 401 al 500	6 objetos 4 de ellos vehículos.	3 vehículos.
		Frames 501 al 600	5 objetos 3 de ellos vehículos.	2 vehículos.
		Frames 601 al 700	2 objetos 2 de ellos vehículos.	2 vehículos.
		Frames 701 al 800	0 objetos 0 de ellos vehículos.	0 vehículos.
		Frames 801 al 900	1 objetos 1 de ellos vehículos.	1 vehículos.

Tabla 9: Resultados del algoritmo.

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 resultados obtenidos demuestran que el sistema realiza un procesamiento en tiempo real y la detección de los vehículos a partir de los objetos capturados en los fotogramas permite establecer que el 52% de los objetos son vehículos.

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, OpenCV y el empaquetamiento Video sensor 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.
- **Video sensor:** Es el empaquetamiento de las clases *VS_Control, Zona de detección* y *Vehículo.*

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.

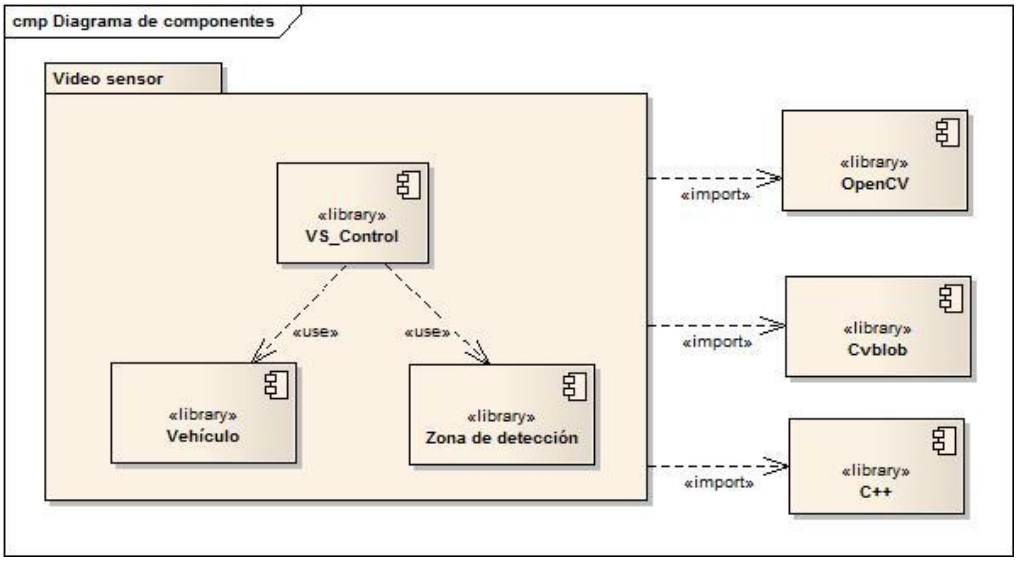


Figura 28: Diagrama de Componentes.

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 (16).

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.

Nombre de la sección	Descripción de la funcionalidad	Resultados Esperados	Resultados Obtenidos
Calcular Velocidad	El objetivo es mostrar el flujo de video obtenido, realizar el seguimiento de los vehículos y calcular la velocidad de los mismos.	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 (0.0054 s)

Tabla 10: Caso de prueba para el procesamiento en tiempo real.

Margen de error en el cálculo de velocidad.

Nombre de la sección	Descripción de la funcionalidad	Resultados Reales	Resultados Obtenidos	Cantidad de frames analizados
Calcular Velocidad.	El objetivo es mostrar el flujo de video obtenido, realizar el seguimiento de los	Vehículo 1 94.1 Km/h	Vehículo 1 92.3 Km/h	39
		Vehículo 2 62.7 Km/h	Vehículo 2 63.4 Km/h	56
		Vehículo 3 56.7 Km/h	Vehículo 3 52,5 Km/h	68
		Vehículo 4 63.4	Vehículo 4	55

	vehículos y calcular la velocidad de los mismos.	Km/h	64.8 Km/h	
		Vehículo 5 95.1 Km/h	Vehículo 5 106.8 Km/h	33
		Vehículo 6 62.2 Km/h	Vehículo 6 60.6 Km/h	59
		Vehículo 7 57.1 Km/h	Vehículo 7 53.42 Km/h	67
		Vehículo 8 97.5 Km/h	Vehículo 8 96.2 Km/h	37
		Vehículo 9 78.3 Km/h	Vehículo 9 73.6 Km/h	48
		Vehículo 10 88.6 Km/h	Vehículo 10 89.2 Km/h	40

Tabla 11: Caso de prueba para el caso de uso calcular velocidad.

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, se pudo probar que los procesos implementados realizan las funcionalidades requeridas con la calidad y eficiencia necesaria, permitiendo que la aplicación cumpla con los requisitos especificados y realice el procesamiento en tiempo real y determinando un margen de error en el cálculo de la velocidad de 2.84 Km/h.

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

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.

RF: Requisitos funcionales.

RNF: Requisitos no funcionales.

FG: Frente o *Foreground.*

BG: Fondo o *Background.*

TRABAJOS CITADOS

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

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

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

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

5. **Fernando Boronat Seguí, Miguel García Pineda.** *IPTV, la televisión por Internet.* Málaga, España : Vértice, 2010.

6. **Alcantarilla, Pablo Fernández.** *Sistema de monitorización y control de tráfico en carretera.* Madrid, España : s.n., 2006.

7. **Colemer, Antonio Albiol.** *Seguimiento de objeto en secuencias de video.* España : s.n, 2003.

8. **Real Acamdemia Española.** Diccionario de la lengua española. [En línea] [Citado el: 14 de noviembre de 2011.] http://buscon.rae.es/draeI.

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

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

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

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

13. **Electronic Dreams.** Tecnología IP. [En línea] [Citado el: 21 de octubre de 2011.] http://www.camara-ip.es.

14. **Pressman, Roger S.** *Software engineering: a practitioner's approach.* New York, United States. : c m, 2004. 978-0-07-337597.

15. **CCTV.** Video vigilancia. [En línea] Wer96. [Citado el: 15 de noviembre de 2012.] www.visualnetcam.com.

16. **Ivar Jacobson, Grady Booch,James Rumbaugh.** *El Proceso Unificado de Desarrollo.* Madrid : Pearson educación, 2007. 84-7829-036-2.

17. **J. Rumbaugh, I. Jacobson, G. Booch.** *El lenguaje Unificado de Modelado, Manual de Referencia.* España : Pearson educación, 2000. 8478290370.

18. **Sparx systems.** Sparx systems. [En línea] Sparx . [Citado el: 22 de Septiembre de 2012.] http://www.sparxsystems.com.ar/products/ea.html.

19. **Digia Plc.** Developer Tools - Digia Plc. [En línea] [Citado el: 7 de Marzo de 2012.] http://qt.digia.com/Product/Developer-Tools/.

20. **Blanco, Carlos.** Sistemas y Desarrollo de Aplicaciones. [En línea] [Citado el: 22 de Marzo de 2012.] http://carlosblanco.pro/.

BIBLIOGRAFÍA

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

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

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

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

5. **Feibel, Werner.** *The Encyclopedia of Networking, Second Edition. Alameda.* CA : Sybex, 1996. 0-7821-1829-1.

6. **Stallings, William.** *Comunicaciones y redes de computadoras.* Upper Saddle: NJ : Prentice Hall, 1998.

7. **Fernando Boronat Seguí, Miguel García Pineda.** *IPTV, la televisión por Internet.* Málaga, España : Vértice, 2010.

8. **Alcantarilla, Pablo Fernández.** *Sistema de monitorización y control de tráfico en carretera.* Madrid, España : s.n., 2006.

9. **Colemer, Antonio Albiol.** *Seguimiento de objeto en secuencias de video.* España : s.n, 2003.

10. **Real Acamdemia Española.** Diccionario de la lengua española. [Online] [Cited: noviembre 14, 2011.] http://buscon.rae.es/draeI.

11. **Datys.** Datys. [Online] [Cited: octubre 29, 2011.] http://www.datys.cu/wpinfnoticias.

12. **informática-full.** Historia del lenguaje C++. [Online] [Cited: noviembre 20, 2011.] http://informatica-full2.blogspot.com/2009/06/historia-del-lenguaje-c.html.

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

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

15. **Electronic Dreams.** Tecnología IP. [Online] [Cited: octubre 21, 2011.] http://www.camara-ip.es.

16. **Pressman, Roger S.** *Software engineering: a practitioner's approach.* New York, United States. : c m, 2004. 978-0-07-337597.

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

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

19. **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.

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

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

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

23. **Shafique K. Javed, Shah M.** *A hierarchical approach to robustbackground subtraction using color and gradient information.* s.l. : Workshop.

1