UNIVERSIDAD DE LA REPÚBLICA, URUGUAY

FACULTAD DE INGENIERÍA
INSTITUTO DE COMPUTACIÓN

PROYECTO DE GRADO

Detección de Patrones de Conducción no Prudente y Mediciones de Tránsito

Fernando Albano
fernaalbano@gmail.com

Diego Tolosa
d.tolosaf@gmail.com

Tutor: Sergio Nesmachnow

Cotutor: Santiago Iturriaga

11 de diciembre de 2015

Resumen

Este documento presenta el proyecto Detección de Patrones de Conducción no Prudente y Mediciones de Tránsito, desarrollado en el Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República.

La evaluación y fiscalización del tránsito a través de cámaras de video está ganando terreno en el mundo. En particular, desde hace algunos años en Uruguay, existe interés por parte de algunas intendencias en instalar sistemas que se encarguen de monitorear el tránsito. Para implementar un sistema de este tipo, es necesario recurrir a técnicas de visión por computadora y computación paralela.

Para este proyecto, se construyó un sistema sumamente flexible basado en configuración por escena, capaz de analizar escenas con diferentes infraestructuras viales, soportando cambios de luminosidad, sombras, superposiciones de vehículos y altas resoluciones de video. El sistema es capaz de realizar el seguimiento y conteo de vehiculos, clasificar vehículos por porte, medir sus velocidades y detectar infracciones como exceso de velocidad, semáforos no respetados y circulación en sentido contrario, entre otros. Permitiendo siempre la posibilidad de adaptarse a los requerimientos de la infraestructura, como cantidad de carriles, existencia de semáforos o carriles restringidos, etc. Los datos recolectados se almacenan en una base de datos, donde pueden ser consultados por una aplicación de escritorio que también permite visualizar los videos en tiempo real mientras son analizados.

A través del paradigma de computación en la nube, alojando el sistema en un ambiente de ato rendimiento, se podría escalar lo suficiente como para brindar servicio a una ciudad completa. Los resultados de las pruebas realizadas, sugieren que el sistema es apto para funcionar en un ambiente de este tipo, como Google Compute Engine.

Por otro lado, las pruebas efectuadas sobre varias escenas, con diferente luminosidad, puntos de vista e infraestructuras determinaron que el sistema es muy fiable, con un promedio de aciertos de alrededor de 93,61 % para la detección y el seguimiento de los vehículos. En cuanto a la clasificación de vehículos, se obtuvo un 92,21 % de aciertos. El resto de las características resultaron aún más fiables, la mayoría alcanzando el 100 % de aciertos, siempre que el vehículo haya sido detectado correctamente.

Índice general

Capítulo 1

Introducción

En las últimas dos décadas el parque automotor Uruguayo se ha duplicado [27]. Como consecuencia directa han aumentado casi en la misma proporción los accidentes de tránsito y la cantidad de personas que fallecen en ellos. El tránsito y la seguridad vial se han convertido en un tema de cabecera para los uruguayos a nivel departamental y nacional.

Los altos índices de accidentes y fatalidades, son sin dudas, uno de los más graves problemas que enfrenta la sociedad. Muchas veces se deben a falta de prudencia por parte de los conductores o directamente a la omisión de las normas de tránsito. La Unidad Nacional de Seguridad Vial (UNASEV) reveló cifras impactantes en el informe de siniestralidad vial en Uruguay del año 2014 [32], la cantidad de accidentes de tránsito ocurridos fue de casi 23.500, un 1,5 % menor que en el 2013, hubo más de 30.000 personas involucradas en estos siniestros que sufrieron lesiones de algún tipo (85 personas promedio por día) de las cuales casi 550 fueron fallecidos obteniéndose una tasa de mortalidad (fallecidos cada 100.000 habitantes) de 15,6 un 5 % más bajo que en el 2013 pero aún así hasta 5 veces más alto que la tasa de Suecia por ejemplo.

El control de las normas de tránsito es difícil de llevar a cabo por los inspectores en ciudades grandes o carreteras muy extensas. Escasos dispositivos se utilizan para realizar estos controles, uno de ellos es el radar que mide la velocidad. Este es muy limitado porque trabaja sobre una sola vía, ya que no es efectivo en casos de superposición de vehículos que transitan al mismo tiempo por distintos carriles. Además de ser un recurso costoso y de limitada disponibilidad, necesita ser trasladado y necesariamente debe ser manipulado por un operador en el lugar. Existen otros problemas por resolver en Uruguay como los grandes volúmenes de tránsito en horas pico en la capital o en determinadas fechas del año en algunas rutas que exceden la capacidad de la infraestructura vial disponible. La solución a estos problemas de vialidad radica en un correcto aprovechamiento de esta infraestructura. Los ingenieros y técnicos que estudian el tránsito, carecen de mecanismos

efectivos para extraer información que sea de utilidad para elaborar modelos y estadísticas que les permitan definir criterios y políticas de uso de la infraestructura de forma óptima.

Las cámaras de video se han popularizado mucho en los últimos años, han sido instaladas en la vía pública en muchos países con el fin de mejorar la seguridad y extraer información. En nuestro país, las Intendencias de Montevideo [23] y Maldonado [24] planean hacerlo en las respectivas capitales de cada departamento, cámaras de vídeo para extraer videos del tránsito. También lo ha hecho el Ministerio del Interior en algunos barrios de Montevideo pero con la finalidad de brindar más seguridad a la ciudadanía.

Por otro lado, otra tendencia que se destaca cada vez más en los últimos años, pero en el mundo de la informática, es la resolución de problemas utilizando visión por computadora [1]. Esta rama de la computación, ofrece soluciones a todo tipo de problemas a través del reconocimiento, clasificación y seguimiento de objetos en imágenes y videos. Los sistemas de monitoreo y vigilancia superan las limitaciones de la percepción humana, en cuanto a la velocidad de reacción, la cantidad de detalles que pueden ser apreciados y la cantidad de mediciones que puede ser realizadas. Es por eso que tiene sentido la aplicación de visión por computadora en este tipo de problemas. Su cometido es servir como una herramienta de procesamiento de información para permitir a sus usuarios extender su percepción y razonamiento.

Este proyecto pretende brindar una alternativa tecnológica a la extracción de datos y al ejercicio de los controles de tránsito a través de visión por computadora. Se construirá un software que sea capaz de extraer información que sea de utilidad para el estudio del tránsito a partir de videos tomados por cámaras ubicadas en la vía pública. Para lograr este cometido el software será capaz de detectar y hacer un seguimiento de cada vehículo durante su tránsito por la vía en un área específica del video, llamada región de interés. Clasificará cada vehículo en una categoría específica (vehículos ligeros, medianos o pesados) y calculará la velocidad a la que transita. El software también será capaz de detectar cuando un vehículo incumple una norma de tránsito como exceder la velocidad permitida, circular en sentido contrario, cruzar un semáforo en rojo o doblar cuando no sea permitido. Por último detectará maniobras poco prudentes o accidentes como dos vehículos que circulan a una distancia peligrosa, un vehículo que se detiene de golpe o dos o más vehículos que colisionan. Para sacar mayor provecho de una infraestructura de cámaras, el software podrá realizar la detección sobre varios videos en tiempo real, pudiendo dar alerta a un usuario capacitado en el momento en que se detecte una infracción o un accidente de tránsito y facilitando a las autoridades la toma de medidas que requiera el caso.

Se construyó un sistema de software, que cumple con los requerimientos especificados, y es capaz de realizar la detección y el seguimiento de vehículos, la clasificación de los mismos y la detección de los comportamientos inapropiados mencionados anteriormente. El sistema es sumamente

flexible, puede adaptarse a una amplia variedad de escenas, con diferentes luminosidades y diferentes infraestructuras viales. Soporta el procesamiento simultáneo de múltiples videos, solicitados por diferentes usuarios concurrentes. También permite la visualización de resultados en tiempo real, o luego de procesados, gracias a la persistencia de datos.

A la hora de implantar el sistema, sin dudas, la mejor alternativa es recurrir a infraestructuras de alto rendimiento y disponibilidad en la nube [9]. Las posibilidades de elasticidad en el uso de recursos [9] y la escalabilidad [9] que ofrecen estas infraestructuras las convierten en la mejor opción. Utilizando el modelo de software como servicio [9], es posible procesar cantidades significativas de videos y prestar servicio a grandes cantidades de usuarios, utilizando múltiples instancias del sistema. Considerando la aplicación en una ciudad como Montevideo, estas características se pueden aprovechar, por ejemplo, al agregar nuevas cámaras progresivamente, para cubrir más puntos de la ciudad. O también, pueden incrementarse temporalmente los recursos utilizados en los períodos en los que se intensifica el volumen de tránsito en algunas vías, y se necesita de más herramientas para monitorearlo adecuadamente, como por ejemplo, en las vías de salida hacia el este durante el verano. Considerando lo anterior, es que se decidió implantar el sistema en la nube de Google Compute Engine [26], donde se realizaron las pruebas de rendimiento. Se logró el análisis de 36 videos con resolución 360p, para ese número de usuarios concurrentes, manteniendo para todos una interacción en tiempo real.

Los peores resultados obtenidos en las pruebas, fueron para las escenas grabadas durante la noche, en las que los vehículos se dirigen hacia la cámara con sus luces encendidas proyectadas en el suelo, impidiendo cualquier tipo de detección. El resto de las pruebas realizadas arrojaron resultados sumamente positivos. En cuanto a las funcionalidades, se obtuvo un promedio de 93,61 % de aciertos para la detección y el seguimiento de vehículos. La característica que obtuvo peores resultados fue la clasificación de vehículos, con un 92,21 % de aciertos. Para el resto de las funcionalidades se obtuvo más de un 95 % de aciertos, llegando en algunas al 100 %.

En la página web del proyecto (https://www.fing.edu.uy/inco/grupos/cecal/hpc/DPT), puede encontrarse información acerca del trabajo realizado, junto con recursos de video que fueron utilizados en el proyecto.

El resto de este documento se divide en cuatro capítulos. El primero de ellos se concentra en el desarrollo de los conceptos teóricos esenciales para la comprensión del proyecto, el análisis de trabajos relacionados y la investigación de tecnologías disponibles. El segundo contiene todos los detalles del planteo del problema y su resolución. El tercero abarca la implantación y la evaluación de la solución planteada. Por último, el cuarto capítulo presenta las conclusiones del proyecto y las posibilidades que existen de continuar con el trabajo.

Capítulo 2

Estado del Arte

2.1. Marco Teórico

En esta sección se presenta una breve compilación de los conceptos que fueron necesarios para abordar el proyecto.

2.1.1. Conceptos Básicos

A continuación se tratan algunos conceptos básicas e imprescindibles para comprender este trabajo. El primero de ellos, visión por computadora [37], es un subcampo de la inteligencia artificial cuyo propósito es programar una computadora para que pueda adquirir y procesar imágenes digitales, para así clasificar o detectar características u objetos en ellas. Se emplea constantemente en robótica y en vigilancia. La visión por computadora es utilizada principalmente para generar modelos tridimensionales a partir de escenas u objetos en movimiento, hacer el seguimiento de objetos en una secuencia de imágenes, buscar imágenes digitales basándose en sus características, y para detectar, segmentar, localizar y reconocer ciertos objetos, como por ejemplo, caras humanas o vehículos como en este proyecto.

Es conveniente continuar definiendo qué es una imagen digital. Una imagen digital [8] es la representación bidimensional de una imagen real a partir de una matriz binaria. Vale aclarar que a partir de este punto, el término imagen hará referencia a imagen digital. La unidad mínima o elemento que compone una imagen es el Píxel [8]. Palabra que deriva del término picture element en inglés. Las imágenes pueden representarse de dos formas, en mapa de bits o en gráfico vectorial. En este proyecto solo se trabajará con imágenes en mapa de bits ya que es el formato que se suele emplear para tomar fotografías digitales y realizar capturas de vídeo a través de dispositivos de conversión analógica-digital, tales como escáneres y cámaras digitales.

Una imagen en mapa de bits [8] [14] se representa mediante una matriz de píxeles. Una de las características más importantes que la definen es su resolución [14]. Esta se representa con dos números enteros, donde el primero

es la cantidad de columnas y el segundo es la cantidad de filas de píxeles de la matriz, por ejemplo 600x480. La resolución es una propiedad estática de las imágenes de mapa de bits que, a diferencia de las que se representan mediante gráficos vectoriales, limita la calidad de la imagen y puede afectar el resultado al aplicar alguna transformación sobre ella.

Otra característica importante es la profundidad del color [8] [14], que es la cantidad de bits que ocupa la representación de un píxel. Esta característica determina el número de colores que se pueden almacenar en cada píxel y por lo tanto determina la calidad del color de la imagen. Esta información de color contenida en cada píxel se agrupa en canales [8] [14] separados que representan los componentes primarios del color que se pretende representar. A esta información se le puede añadir otro canal que representa la transparencia respecto al fondo de la imagen. En algunos formatos (GIF [3], por ejemplo) el canal de transparencia tiene un solo bit de información, lo que permite representar como totalmente opaco o totalmente transparente un píxel. En los formatos más avanzados (PNG, TIFF [3]) el canal de transparencia es un canal con la misma profundidad que el resto de los canales de color, lo que permite obtener niveles de transparencia distintos.

Existen varios modelos de representación de colores [8] [14] que se utilizan para representar imágenes digitales, algunos de ellos son CMYK, HSV y RGB. Este último será el utilizado en este proyecto. La sigla RGB proviene del inglés red, green and blue. Una imagen que utiliza este modelo tiene tres canales: rojo, verde y azul. Estos se derivan u obedecen a los receptores de color del ojo humano, y se usan en monitores y escáneres. Para las imágenes RGB de 24 bits cada canal tiene 8 bits para cada uno de los tres colores. Esta representación está compuesta de tres imágenes (una por cada canal), donde cada imagen puede almacenar píxeles con intensidades de brillo convencional entre 0 y 255. Otro modelo de color se usa en este proyecto, la escala de grises [8] [14]. Esta presenta en cada píxel el valor equivalente a una graduación de gris. En vez de utilizar tres canales, como en RGB, utiliza solamente uno, ya que solo es necesario representar la luminosidad. El último modelo de color utilizado, y el más simple y reducido de todos, es el binario [8] [14]. Las imágenes binarias permiten solo dos valores en cada uno de sus píxeles, generalmente se utilizan blanco y negro, aunque pueden ser otros colores. Estas imágenes resultan sumamente útiles para reducir la complejidad de los algoritmos y el uso de memoria cuando, por ejemplo, solo interesa destacar la ubicación de un objeto de otra imagen.

El uso de filtros [8] [14] [3] es crucial en la mayoría de las aplicaciones de visión por computadora. Un filtro es una función matemática que se aplica sobre una imagen digital con la finalidad de obtener como resultado una nueva imagen con características mejoradas que resulte más fácil de procesar. También se utilizan filtros para optimizar la representación de imágenes (por ejemplo almacenamiento versus calidad), enfatizar algunas características o lograr efectos especiales. Los filtros se utilizan para:

- Eliminar el ruido: transformar los píxeles con nivel de intensidad muy diferente al de sus vecinos de forma tal que coincida con estos. Estas distorsiones suelen aparecer al adquirir o transferir la imagen debido a las imperfecciones de los dispositivos utilizados o la existencia de interferencia en el medio.
- Suavizar la imagen: reducir la cantidad de variaciones de intensidad entre píxeles vecinos.
- Agudizar la imagen: amplificar las diferencias de intensidades entre píxeles vecinos, es lo opuesto a suavizar.
- Detectar bordes: detectar los píxeles en los que se produce un cambio brusco en la intensidad.
- Realzar bordes u objetos: destacar los bordes u objetos que se encuentran en la imagen.

Algunos ejemplos de Filtros son:

- Valor Umbral: o Thresholding (en inglés), son algoritmos que permiten distinguir entre píxeles de la imagen mediante un criterio en base a su intensidad. El filtro toma como entrada una imagen en escala de grises y la convierte en una imagen binaria. Este filtro es muy utilizado en la detección de objetos porque permite simplificar las características de la imagen, y aplicándolo apropiadamente podría lograr aislar los objetos del resto de la imagen. La Figura 2.1 muestra la aplicación de un filtro de valor umbral.

Figura 2.1: Aplicación de un filtro de valor umbral sobre una imagen.

- Gaussiano: este filtro se utiliza para bajar el nivel de detalle y el ruido de la imagen, para ello se aplica una función de Gauss sobre la matriz de representación de la imagen. La Figura 2.2 muestra la aplicación de un filtro Gaussiano sobre una imagen.
- Dilatación: la dilatación de imágenes binarias es muy utilizada en la detección de objetos ya que bien empleada permite, por ejemplo, corregir la representación de los objetos que se encuentran mal delimitados en la imagen o aumentar su tamaño. Se basa en un algoritmo muy simple para expandir las superficies blancas de la imagen sobre las negras,

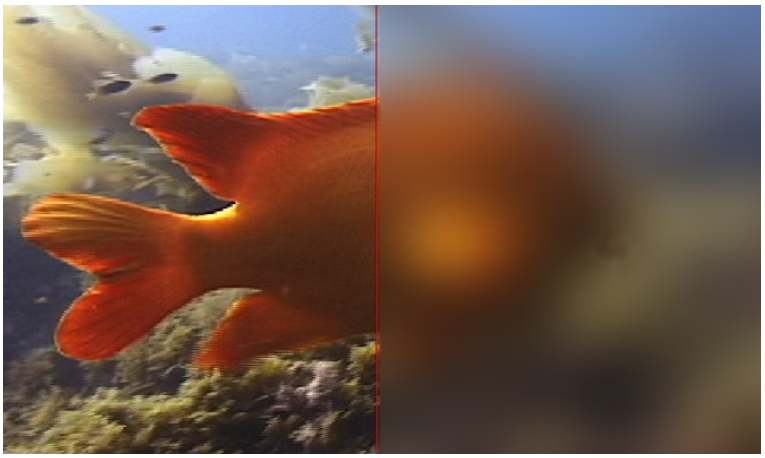

Figura 2.2: Aplicación de un filtro Gaussiano sobre una imagen.

simplemente recorre la matriz de representación de la imagen y para cada píxel en blanco asigna el color blanco a sus vecinos que están en color negro. El radio de píxeles vecinos que se transforma depende de la entrada del algoritmo. La Figura 2.3 ilustra el procedimiento.

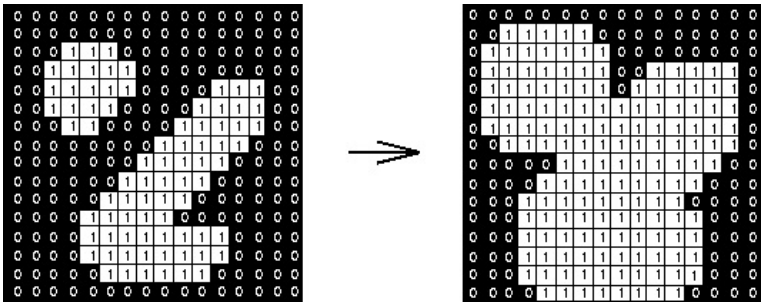

Figura 2.3: Aplicación de un filtro de dilatación sobre una imagen, representación.

- Erosión: la erosión es la operación opuesta a la dilatación, funciona de forma similar, convirtiendo los píxeles de blanco a negro. Suele utilizarse para disminuir el tamaño de los objetos que estorban en la detección o para disminuir el ruido de la imagen.

Habiendo sentado algunas bases teóricas sobre imágenes, conviene hacer lo mismo para los videos. Un clip de video digital [14] [10] consiste en una secuencia de imágenes digitales individuales llamadas cuadros, que se muestran en rápida sucesión, generando un efecto de aparente continuidad en el ojo humano. Para su representación se utiliza una matriz tridimensional con dos de sus dimensiones correspondientes al largo y ancho de los cuadros y la otra correspondiente a la secuencia de estas. Todos los cuadros de un mismo

video poseen las mismas características estáticas en cuanto a resolución, canales, representación y compresión. La cantidad de cuadros mostrados por segundo [10] (de la sigla FPS, frames per second en inglés) puede variar enormemente según el clip de video, desde diez hasta un millón, tomado por cámaras de alta velocidad. La calidad y el realismo del video dependerán de la cantidad de cuadros por segundo, de la resolución y de la calidad de colores que posean los cuadros.

Existen varios formatos de almacenamiento y varios algoritmos de compresión [10] para optimizar la calidad, el almacenamiento y la transmisión de los archivos de video ya que estos pueden llegar a ser muy grandes y por lo tanto poco prácticos para ser transferidos por red. Algunos de ellos funcionan comprimiendo solamente los cuadros de forma individual y similar a la forma en que se comprimen las imágenes. Otros añaden técnicas más complejas como mantener cierto porcentaje de cuadros originales del video y el resto calcularlos a partir de cuadros parciales que contienen solo las diferencias con los cuadros anteriores.

Los componentes de software que codifican y decodifican los videos en formatos y compresiones específicos se llaman codecs [10], estos son necesarios para poder reproducir o manipular un video en cualquier dispositivo. La falta de codecs puede ser una limitante en el procesamiento de videos, por eso es importante determinar los codecs que se encuentran disponibles para saber qué formatos de video son válidos para el software a producir.

Por último, el concepto de la Difusión en Flujo o Streaming [10], en inglés, refiere a la distribución de video o audio digital a través de una red de computadoras. El streaming permite la reproducción del medio en paralelo mientras se realiza su descarga. Esta tecnología se ha vuelto muy popular en los últimos años por ser la clave de varias aplicaciones en tiempo real como la difusión de radio y televisión o videoconferencias.

2.1.2. Algoritmos de Detección de Características

La detección de características [14] [4] (features, en inglés) es un área central del procesamiento de imágenes. Su principal objetivo es identificar las características distintivas de las imágenes que pueden resultar útiles en el contexto de un problema. Los siguientes son algunos algoritmos de detección de características que suelen utilizarse en la detección de objetos.

Detección de Bordes Los bordes [14] [4] son los puntos de una imagen donde hay un límite entre dos regiones de puntos con una diferencia de intensidad notable. Existen varios algoritmos de detección de bordes, algunos de ellos son [14] Gabor, Sobel y Canny. La Figura 2.4 muestra una imagen a la izquierda y el resultado de aplicar un filtro Sobel a la derecha.

Figura 2.4: Aplicación de un Filtro Sobel a una imagen.

Detección de Esquinas y Puntos de Interés Las esquinas [14] [4] son los puntos que resultan de la intersección entre dos bordes, se encuentran dentro del conjunto de puntos de interés que también abarcan puntos aislados en una región de puntos con diferente intensidad, el fin de un borde o el punto máximo en una curvatura de un borde. Dos de estos algoritmos son Shi y Tomasi [36] y SUSAN [38]. La Figura 2.5 muestra una imagen a la que se le aplicó un filtro de detección de esquinas.

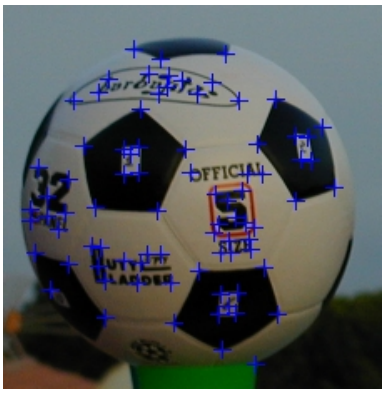

Figura 2.5: Aplicación del Filtro Morfológico sobre una imagen.

Detección de Gotas La detección de gotas [14] [4] (proveniente de blob, en inglés) se enfoca en identificar en una imagen las regiones que difieren en intensidad en comparación con las que la rodean según las especificaciones dadas. Los detectores de gotas son utilizados muy frecuentemente en la detección y el reconocimiento de objetos. Dos algoritmos de detección de gotas son Laplacian of Gaussian (LoG) [4] y Difference of Gaussians (DoG) [4]. La Figura 2.6 muestra una imagen a la que se le aplicó un filtro de detección de gotas.

Figura 2.6: Aplicación de un filtro de detección de gotas sobre una imagen.

2.1.3. Algoritmos de Reconocimiento de Objetos

Los algoritmos comentados a continuación tienen como finalidad la detección de objetos en imágenes digitales.

Redes Neuronales Artificiales Las redes de neuronas artificiales [14] [37] son un paradigma en el área de aprendizaje automático, utilizan los principios de organización de los cerebros biológicos para construir sistemas inteligentes. De esta forma, se intenta emular el funcionamiento del cerebro a bajo nivel. Así, se crean sistemas masivamente paralelos formados por un gran número de elementos de procesos simples interconectados. Estos están formados por un conjunto de neuronas unidas por conexiones unidireccionales que tienen un peso asociado y normalmente se organizan en capas. El procesamiento llevado a cabo en cada neurona es local y depende únicamente de las entradas, los pesos de las conexiones, y en algunos casos, el estado anterior de la neurona. Finalmente, la red se adapta mediante un aprendizaje que modifica los pesos de las conexiones.

Una de las primeras aplicaciones de las redes neuronales artificiales en detección de objetos en imágenes digitales en escala de grises fue implementada por Rowley, Baluja y Kanade en [35], este algoritmo podía detectar rostros erguidos y en posición frontal. El algoritmo obtuvo buenos resultados y se mostró a la par de sus competidores contemporáneos. Este y otros trabajos similares sentaron las bases para que varios algoritmos publicados posteriormente apliquen de igual forma este paradigma para resolver problemas más complejos, en particular se mencionará uno de ellos en la siguiente sección.

Clasificador en Cascada Los clasificadores [37] son una clase de algoritmos pertenecientes al área de aprendizaje automático cuyo cometido es la detección de patrones y regularidades en conjuntos de datos. Los clasificadores deben distinguir a partir de una entrada de datos predefinida a qué categoría pertenece un conjunto de datos, dado un conjunto de categorías. En pocas palabras, son utilizados para clasificar datos.

Los clasificadores en cascada son un tipo especial de clasificadores que nuclean varios clasificadores en un árbol de decisión. A cada clasificador ejecutado se le llama etapa, las etapas se ejecutan en forma secuencial. En visión por computadora su aplicación más conocida es el reconocimiento facial, disponible en muchos dispositivos hoy en día, aunque en verdad sirven para el reconocimiento de cualquier objeto. El primer clasificador en cascada fue el detector de caras de Viola y Jones [42], pensado justamente para ser implementado en sistemas con bajo poder de cómputo. Al poco tiempo, Lienhart and Maydt [17] implementaron un nuevo algoritmo llamado Haar-like Features introduciendo nuevos conceptos y mejorando la implementación anterior [16].

Estos algoritmos, como muchos algoritmos de aprendizaje automático, requieren de un conjunto de datos de prueba inicial para ser entrenados antes de ser ejecutados. En el caso de la detección de objetos en imágenes o video es necesario contar con varias centenas o hasta varios miles de imágenes de muestra, dependiendo de la complejidad del objeto o los objetos que se desean detectar. Por ejemplo, no es lo mismo intentar detectar siempre una misma lapicera en un escritorio que cualquier automóvil que transite por una avenida. Para realizar el entrenamiento se especifica un número de etapas para el algoritmo y se utilizan como entrada muestras positivas donde el objeto debe abarcar la mayor cantidad posible de la imagen y muestras negativas donde el objeto no debe aparecer en la imagen.

El resultado del entrenamiento generalmente es un archivo de texto que contiene, para cada etapa del clasificador, un conjunto de valores que define las características detectadas en las imagenes de muestra y que deberá buscar en las imágenes a evaluar al ser ejecutado. Dado este archivo de entrenamiento, durante su ejecución, el algoritmo comparará en cada etapa el resultado de la evaluación de la imagen a clasificar con los valores esperados y decidirá si esta cumple con el criterio. La Figura 2.7 muestra un ejemplo de detección facial utilizando el algoritmo Haar-like Features.

La efectividad del algoritmo depende, además de su implementación obviamente, de la cantidad de etapas, cuantas más sean es más probable que resulte más efectivo aunque esto hará que sea más larga su evaluación y por lo tanto su tiempo de procesamiento. Por otro lado, el tamaño y la calidad de la muestra utilizada en el entrenamiento también son clave para la efectividad en la detección.

2.1.4. Algoritmos de Detección de Movimiento

A continuación se describen un par de técnicas utilizadas para la detección de movimiento en video digital.

Diferencia de Cuadros La diferencia de cuadros es la más sencilla de las técnicas de detección de movimiento. Dados dos cuadros contiguos de

Figura 2.7: Detección facial utilizando el algoritmo Haar-like Features.

un video se aplica sobre ambos un filtro de cambio de modelo de color para convertirlos a escala de grises. Luego se calcula la diferencia absoluta entre ellos, esto es la resta entre ambas matrices de representación de cada uno. Como resultado, los píxeles que se mantienen iguales entre ambos cuadros pasarán a tener valor nulo (o sea, color negro) y los píxeles que varíen entre ambos cuadros tendrán diferentes intensidades, siempre dentro de la escala de grises. Este método sencillo resulta muy eficaz para videos de baja calidad, con iluminación constante y sombras tenues. En el próximo capítulo se estudiará la aplicación de este método mediante un prototipo que trabaja sobre un video de tránsito y se evaluará su efectividad en la detección de vehículos en movimiento.

Sustracción de Fondo Una técnica muy popular es la sustracción de fondo [34], ésta se basa en extraer los objetos en movimiento a través de la diferencia de un cuadro con el fondo de la escena. El fondo es un dato conocido y debe ser suministrado al algoritmo como parámetro de entrada al iniciar el algoritmo. Luego en cada invocación, a medida que se recorre la secuencia de video, se invoca al algoritmo por cada cuadro y este dará como resultado una imagen similar a la que se obtiene en la diferencia de cuadros, con los píxeles invariantes en negro y los píxeles que corresponden a objetos que no pertenecen al fondo en blanco.

Algunos algoritmos de sustracción de fondo tienen en cuenta los cambios en la iluminación y en el movimiento de la cámara o de los objetos de fondo (como puede ser el efecto producido por el viento sobre las hojas de un árbol) para realizar sus cálculos y mejorar la detección. Para poder llevar a cabo este seguimiento de los cambios de fondo de la escena es necesario que el algoritmo mantenga una representación del fondo que debe ser actualizada en cada invocación, lo cual agrega complejidad al algoritmo pero lo hace

más robusto y disminuye las posibilidades de incurrir en un error. Algunos algoritmos también son capaces de detectar sombras [7] en movimiento, estas se suelen representar con color gris para poder diferenciarlas de los objetos a la hora de procesar el resultado. La Figura 2.8 muestra la aplicación de la sustracción de fondo sobre un cuadro a partir del modelo del fondo de la escena y el resultado del algoritmo, al cual se le suele llamar máscara de fondo. En el siguiente capítulo se abordará la construcción de un prototipo que utilice este algoritmo para la detección de vehículos.

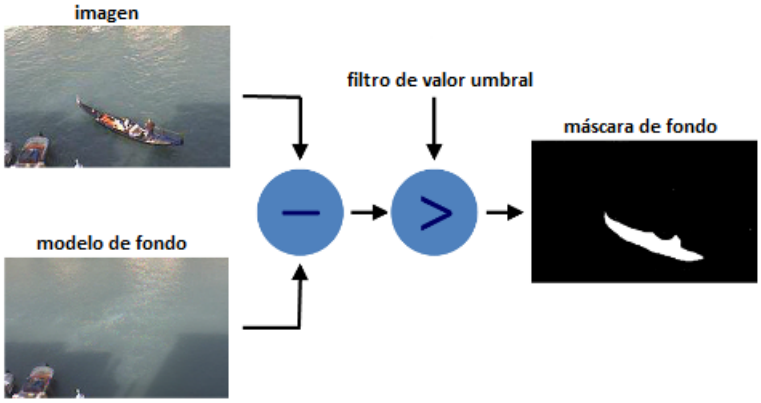

Figura 2.8: Aplicación de la sustracción de fondo sobre un cuadro de video.

2.2. Trabajos Relacionados

Existe una vasta variedad de trabajos en el área de visión por computadora aplicada al problema de conteo de vehículos en el tráfico. La mayoría de los trabajos mencionados a continuación fueron escritos por Ingenieros en Electricidad o Electrónica que se especializan en procesamiento digital de señales y coinciden en que están centrados en el desarrollo y la utilización de modelos matemáticos y algoritmos que sean útiles en la detección de objetos en movimiento. Si bien están enfocados a un nivel más bajo de desarrollo que el de este proyecto, el estudio de estos trabajos resulta fundamental para ampliar los conocimientos en esta área de investigación, además de aportar nuevas ideas al proyecto.

A continuación se hará mención de algunos de los trabajos estudiados previo a la implementación del proyecto que presentan posibles alternativas para el desarrollo del mismo. El análisis se hará según el orden cronológico de las publicaciones y se enfocará principalmente en métodos de detección de movimiento.

En el año 2007, Granados y Marin [11] plantearon el problema de la detección y clasificación de vehículos y peatones en la vía pública para detectar sus respectivos flujos. El proceso de detección puede dividirse en tres etapas:

- Detección de movimiento invariante a la iluminación. Primero se halla la reflectancia de los objetos, esto es la luz que reflejan, y se elimina la luminancia, que es la luz que incide en la escena. Luego se utiliza la diferencia entre cuadros consecutivos junto con la aplicación de un filtro de valor umbral para obtener una imagen binaria con los bordes de los objetos en movimiento en la escena.
- Extracción de características. Se utilizan Descriptores de Fourier para modelar las características de los objetos hallados anteriormente. Los Descriptores de Fourier [4] representan la forma del objeto utilizando Series de Fourier. Destacándose que los primeros descriptores indican la forma general del objeto y los últimos descriptores los más pequeños detalles, con lo que para una clasificación un pequeño conjunto de descriptores puede ser suficiente. Su gran ventaja es que son invariantes frente a la traslación, la rotación y la escalabilidad.
- Clasificación. La última etapa consta de la clasificación de los contornos detectados a través de una red neuronal de cuatro capas, que toma como entrada los primeros diez descriptores de cada objeto, o sea los más significativos, para reducir costos de procesamiento. Para lograr buenos resultados la red debió entrenarse con aproximadamente diez mil ciclos.

El algoritmo fue implementado en Matlab y probado con un procesador Pentium 4 de 2.3 GHz. Para probarlo se utilizaron imágenes pequeñas de

320x240 píxeles logrando procesar diez imágenes por segundo. La prueba fue realizada sobre diez horas de grabación obteniéndose un 87 % de aciertos sobre humanos y un 96 % sobre vehículos. La causa principal de los desaciertos fue la superposición de cuerpos y en el caso de los cuerpos humanos la diversidad de posiciones adoptadas por los cuerpos que no fue completamente prevista. Cabe destacar la importancia de las propiedades de los Descriptores de Fourier que hacen posible la correcta clasificación de los cuerpos sin importar su tamaño o rotación.

En el año 2007, Tsai y otros [40] plantearon el uso de un algoritmo de búsqueda de colores para identificar vehículos en una imagen. Para llevar a cabo su algoritmo, debieron realizar una investigación acerca del espectro de colores en diferentes condiciones climatológicas sobre la superficie de vehículos y sobre las vías de tránsito. A partir de este estudio lograron delimitar espacios de colores para los píxeles que corresponden a vehículos y para los que no. El algoritmo implementado consta de dos etapas, una de hipótesis y otra de verificación:

- Hipótesis. Primero se detectan los píxeles de la imagen cuyo color se encuentra dentro del espacio de colores de vehículos determinado previamente. Luego se aplican un clasificador en cascada y/o una red neuronal para hacer más fiable la selección de píxeles. Esta etapa finaliza con una lista de los píxeles que han sido encontrados y que pueden pertenecer a un vehículo.
- Verificación. A partir de la imagen y la lista de píxeles encontrados en la etapa anterior se verificará si existen vehículos en la escena. Esta etapa comienza con una dilatación de los píxeles candidatos en la imagen, esto permite considerar los píxeles vecinos que pueden no haber sido clasificados correctamente. Cada conglomerado de píxeles será considerado como una hipótesis de vehículo y tratado como una subimagen que será sujeta al proceso de verificación. El proceso de verificación utiliza un análisis de bordes y contornos junto con un clasificador en cascada que inspirado en el algoritmo de detección facial de Viola y Jones trabaja con escalados progresivos de la subimagen para reducir los costos de procesamiento.

El análisis experimental arrojó una media de 95 % de aciertos en 1197 vehículos evaluados. Las pruebas fueron realizadas con un procesador Pentium de 2.4 GHz. Se obtuvieron resultados entre 0,15 y 0,50 segundos por imagen, valores muy bajos en comparación con el resto de los métodos con los que fue comparado el algoritmo, pero no se especifica el tamaño de las imágenes. Este algoritmo resulta muy innovador al utilizar el color de los píxeles como una característica de búsqueda y discriminación de objetos, esta mecánica es su mayor fortaleza, ya que le significa un ahorro muy importante de cómputo en la etapa de verificación y le permite lidiar mejor con el problema de

la superposición de objetos.

En el año 2009, Urrego y otros [41] plantearon el uso de una estrategia de estimación de primer plano que consiste en distinguir los puntos que pertenecen a objetos móviles en la escena de los puntos que se mantienen estáticos porque pertenecen al fondo de la misma. La estrategia utiliza los algoritmos FGDStatmodel [2] modificado por los autores para suprimir el fondo, y Media Aumentada desarrollado por los autores para disminuir el margen de error y aumentar la velocidad de convergencia. Durante la ejecución del algoritmo el modelo del fondo es estimado periódicamente y comparado con cada cuadro del video. De esa comparación se obtienen los pixeles que no se mantienen estáticos y por ende pertenecen a objetos móviles. Los píxeles seleccionados son filtrados dependiendo de un cierto umbral de área mínima, si respetan este umbral son clasificados como vehículos u otro tipo de objeto móvil para luego hacer el seguimiento y la extracción de datos correspondiente. El algoritmo fue comparado contra otros algoritmos de detección, entre ellos dos basados en sustracción de fondo, a los cuales superó en efectividad. Las pruebas se llevaron a cabo con 15 videos y resultó en una efectividad del 99,85 % en 2106 vehículos. Sin embargo, no se especifican tiempos de ejecución, solo se garantiza que el algoritmo puede funcionar en tiempo real asegurando una ubicación de cámara que minimice la superposición de objetos.

En el año 2009, Mora y otros [33] plantearon el uso de un algoritmo de flujo óptico basado en una pirámide de imágenes. Luego en el año 2011, el mismo planteo fue realizado en la publicación de Zhan y Ji [43]. El flujo óptico es una medida de como los puntos se mueven en el espacio. Se basa en una hipótesis de conservación de la intensidad de cada píxel en el tiempo aunque éste cambie de posición. Esta hipótesis permite realizar el rastreo de píxeles en movimiento en un video al comparar dos cuadros consecutivos. Los algoritmos más utilizados para calcular el flujo óptico son Lukas-Kanade [18] y Horn y Shunk [12], datan de 1981 y 1980 respectivamente y se encuentran implementados en bibliotecas de visión por computadora. El procesamiento del flujo óptico no es sencillo y se vuelve más complejo en imágenes reales de mayor resolución y calidad. Por ello es que se plantea la idea de utilizar una pirámide de imágenes. Esta pirámide se construye a partir de las cuadros extraídos del video aplicando sobre ellas filtros que disminuyen su resolución intentando no perder los objetos de la imagen. Luego de llegar al último filtro y habiendo terminado la pirámide se tiene una versión reducida del cuadro original que resulta más sencilla de procesar por el algoritmo de flujo óptico empleado. Luego de aplicar otros filtros sobre la imagen resultado del algoritmo para mejorar la detección, los objetos en movimiento son identificados. En el caso Zhan y Ji, el primer filtro aplicado luego del el algoritmo de flujo óptico es un filtro de valor umbral para obtener los vehículos

delimitados en blanco sobre un fondo negro y luego se aplica un algoritmo de reconocimiento de figuras para encasillar cada vehículo en un rectángulo. Ninguno de los trabajos ofrece resultados de prueba. Sin embargo aseguran ser eficientes y efectivos. Cabe destacar que ambos algoritmos aparentan ser una alternativa útil ante los escenas con fuentes de iluminación variables debido a la naturaleza del algoritmo de flujo óptico.

En todas las publicaciones los algoritmos implementados obtienen buenos resultados en cuanto a la detección de vehículos, superando el 95 % de aciertos en la mayoría de las pruebas. Si bien todos prometen ser eficientes, algunos son más complejos que otros a nivel de fundamentos teóricos e implementación. Incurrir en el uso o implementación de alguno de ellos podría retrasar el proyecto si no se tienen claros dichos fundamentos.

Existen muchas técnicas o posibles combinaciones de ellas para atacar el problema en cuestión. Es importante tener en cuenta, al momento de elegir cuáles usar, qué tan efectivas pueden resultar al aplicarse a videos con diferentes características además de las ventajas y desventajas de cada una. En el siguiente capítulo se abordará la resolución al problema, pero antes es necesario enumerar aquellas características particulares de los videos que pueden agregar dificultad a la detección:

- Cambios de iluminación en la escena: los cambios de iluminación pueden afectar los píxeles de los vehículos y dificultar su correcta detección, resulta necesario entonces que el algoritmo elegido pueda responder frente a ellos. Los cambios de iluminación se pueden dar, por ejemplo, en la puesta de sol o en el amanecer, no solo porque la luz ambiental cambiaría sino porque también las luces de cada vehículo pasarían a tener o dejar de tener influencia en la escena.

- Sombras proyectadas: las sombras siempre resultan un estorbo al momento de detectar objetos en movimiento. Es necesario que el algoritmo utilizado pueda distinguir los objetos de sus sombras, para que no se produzcan errores de detección. Las sombras dependen de la iluminación de la escena, por ejemplo, en días nublados y sobre el mediodía las sombras no suelen interferir en la detección. En cambio, cuando el sol se encuentra próximo al horizonte, las sombras suelen ocupar mucho espacio y superponerse sobre los vehículos y por lo tanto hacer más difícil el reconocimiento.

- Superposición de vehículos: de todos los problemas que presentan los videos de tránsito, este parece ser el más complicado de resolver. Hasta el momento no existen algoritmos que puedan detectar correctamente a dos o más vehículos que se superponen y circulan a la misma velocidad. Si bien uno de los trabajos mencionados sí puede distinguir mediante su color a los vehículos y puede en determinados casos lograr distinguirlos al superponerse, está limitado a que el color de los

vehículos sea diferente, de otra forma, el algoritmo no tiene forma de hacer la distinción.

El objetivo del proyecto no es desarrollar un nuevo algoritmo, sino lograr adaptar de la mejor forma posible los que ya existan, según los requerimientos del proyecto. Luego de hecho esto, la atención debe centrarse en la extracción de datos en los videos, siendo de vital importancia la cantidad y la calidad de estos. Por último, es necesario concentrarse en la generación de nuevos datos, para el modelado del comportamiento de los vehículos, tomando como primitiva los datos extraídos anteriormente. En comparación con los trabajos mencionados, este proyecto se centra sobre un nivel superior de desarrollo, enfocado en la extracción y posterior utilización de los datos. Por esta razón, no tiene sentido realizar mayores comparaciones, ya que no es la esencia del proyecto.

A continuación, se detallan en la Tabla 2.1, las características tomadas de cada trabajo y a que parte del sistema desarrollado corresponden.

Autores	**Característica**	**Contemplada Por**
Granados y Marin	Detección invariante a la iluminación	Algoritmo de Sustracción de Fondo
Tsay y otros	Detección de superposición	Lógica de procesamiento de vehículos
Urrego y otros	Sustracción de Fondo	Algoritmo de Sustracción de Fondo
Mora y otros	Clasificación de objetos	Lógica de procesamiento de vehículos

Tabla 2.1: Utilización de trabajos ya existentes

Para cerrar esta sección, se detallan en la Tabla 2.2 un par de características extra que también forman parte del sistema.

Característica	**Contemplada Por**
Detección invariante a sombras	Algoritmo de Sustracción de Fondo
Detección de velocidad, sentido de circulación, etc.	Lógica de procesamiento de vehículos

Tabla 2.2: Características adicionales del sistema

2.3. Tecnologías Utilizadas

En esta sección se comentan las tecnologías utilizadas para llevar a cabo la implementación del proyecto. Como lenguaje de programación, se escogió Java en su versión 1.7, principalmente por su portabilidad, su amplio grado de compatibilidad, su adoptación a nivel global y por la gran cantidad de librerías disponibles. Para la manipulación de imágenes y videos se utiliza la librería JavaCV, que nuclea implementaciones o interfaces hecha en Java de varias bibliotecas de visión por computadora tales como OpenCV y FFmpeg. En las siguientes subsecciones, cada una de estas bibliotecas será brevemente descripta.

2.3.1. OpenCV

OpenCV [13] (Open Source Computer Vision) es una librería de código abierto de software de visión por computadora y aprendizaje automático. Fue construido para proporcionar a los desarrolladores de este tipo de aplicaciones una infraestructura común. Al momento, cuenta con más de 2500 algoritmos optimizados, tanto clásicos como de última generación. Estos algoritmos se pueden utilizar para detectar y reconocer rostros, identificar objetos, clasificar las acciones humanas en videos, movimientos de cámara, extraer modelos 3D de objetos, búsqueda de imágenes similares en una base de datos de imágenes, eliminar los ojos rojos de las imágenes tomadas con flash, seguir los movimientos de los ojos, etc. OpenCV cuenta con una comunidad con más de 47 mil usuarios y se estima que su número de descargas supera los 7 millones. Se utiliza ampliamente en empresas, grupos de investigación y organismos gubernamentales. OpenCV cuenta con interfaces de C, C++, Python, Java y MATLAB y es compatible con Windows, Linux, Android y MacOS. También cuenta con interfaces CUDA y OpenCL con todas las funcionalidades. OpenCV está escrita de forma nativa en C++.

Algunos de los algoritmos implementados en OpenCV que tienen posible aplicación en este proyecto son:

- Filtros de Imágenes: Valor Umbral, Erosión, Dilatación, Gaussiano.
- Algoritmos de Detección de Características: Detección de Bordes, Detección de Esquinas y Detección de Gotas.
- Algoritmos de Detección de Objetos: Clasificadores en Cascada para detección de objetos (genérico).
- Algoritmos de Detección de Movimiento: Diferencia de Cuadros, Sustracción de Fondo, Detección de Flujo Óptico.

2.3.2. FFmpeg

FFmpeg [39] es una librería de código libre que permite grabar, convertir y realizar streaming de audio y video. Además, incluye una bibliote-

ca con una vasta variedad de codecs implementados. Está desarrollado en GNU/Linux pero puede ser compilado en la mayoría de los sistemas operativos. El uso de FFmpeg para la manipulación de video permite abstraerse de las especificaciones de cada video, también permite leer archivos o consumir flujos de video de forma transparente y otorga amplia compatibilidad con los distintos formatos de video existentes, entre ellos mp4, mpeg y avi.

2.3.3. JavaCV

JavaCV [5] es una biblioteca para el lenguaje Java que nuclea interfaces o implementaciones de varias bibliotecas basadas en otros lenguajes. Su cometido es brindar facilidad en el uso de estas bibliotecas a los usuarios de la plataforma Java, incluyendo Android. JavaCV incluye las bibliotecas OpenCV, FFmpeg, libdc1394, PGR FlyCapture, OpenKinect, videoInput, ARToolKitPlus y flandmark. JavaCV mejora o simplifica algunas de estas bibliotecas para que se adapten mejor al lenguaje Java.

Si bien algunas de estas bibliotecas ya cuentan con una interfaz Java, como es el caso de OpenCV, JavaCV resulta más fácil y práctico de usar para el desarrollo de aplicaciones. De todas formas, la ventaja más importante de JavaCV viene de la mano de Java y es su portabilidad. Una aplicación escrita en Java utilizando JavaCV resulta más fácil de empaquetar y ejecutar en varios ambientes que una aplicación escrita en Java que utiliza interfaces sobre OpenCV, ya que esta última requiere de una instalación de OpenCV en el ambiente donde se ejecutará la aplicación. Este último factor resulta decisivo para la elección de esta tecnología.

2.4. Construcción de Prototipos

Para evaluar todos los conocimientos teóricos adquiridos, la experiencia de los trabajos relevados y las funcionalidades de las tecnologías disponibles es necesario construir prototipos de aplicaciones que aseguren las posibilidades de éxito. Los tres prototipos mencionados a continuación permitieron definir el algoritmo utilizado en la solución implementada. Se intentó construir un cuarto prototipo con un algoritmo de flujo óptico [37] pero no fue posible por problemas con el código en JavaCV.

Las versiones utilizadas para la implementación de los prototipos y de las aplicaciones finales son:

- Java: 1.7
- JavaCV: 0.8
- OpenCV: 2.4.9 (utilizado por JavaCV)
- FFmpeg: 2.2.1 (utilizado por JavaCV)

Los prototipos fueron implementados consultando las documentaciones disponibles de JavaCV [5] y OpenCV [13], además de bibliografía exter-

na [15] sobre OpenCV. Cabe destacar que si bien se utilizó JavaCV y no OpenCV para implementar, las implementaciones de las bibliotecas son muy parecidas, por lo tanto no se requiere de mucho esfuerzo para traducir un código de OpenCV a JavaCV. Además, la documentación de OpenCV es muy rica en contenido y ejemplos, a diferencia de la documentación de JavaCV que solo contiene ejemplos básicos de uso. También se acudió a los foros de JavaCV para encontrar respuestas a preguntas puntuales sobre código durante el desarrollo.

2.4.1. Prototipo 1: Diferencia de Cuadros

El primer prototipo construido fue el más simple de los tres. Se basa en la aplicación de la diferencia de cuadros consecutivos y la detección de contornos para hallar los objetos en movimiento en la escena. El pseudocódigo utilizado se muestra en el Algoritmo 1.

```
frameInicial = leerFrame();
mientras existan frames sin leer hacer
    frameSiguiente = leerFrame();
    frameInicialEG = convertirAEscalaGrises(frameInicial);
    frameSiguienteEG = convertirAEscalaGrises(frameSiguiente);
    aplicarFiltroValorUmbral(frameInicialEG);
    aplicarFiltroValorUmbral(frameSiguienteEG);
    diferencia = diferenciaFrames(frameInicialFil, frameSiguienteFil);
    contornos = buscarContornos(diferencia);
    dibujarContornos(contornos, frameInicial);
    mostrarCuadro(frameInicial);
    frameInicial = frameSiguiente;
fin
```

Algoritmo 1: Prototipo Diferencia de Cuadros.

La clave de este algoritmo es la correcta utilización de los filtros previo a calcular la diferencia entre los cuadros consecutivos, para aplicarlos es necesario convertir las imágenes a escala de grises. Se hicieron muchas pruebas con los videos disponibles utilizando diferentes combinaciones de filtros Gaussianos y de Valor Umbral. En algunas pruebas también se utilizaron filtros de erosión y dilatación para eliminar el ruido remanente en la diferencia obtenida entre ambos cuadros y mejorar así la posterior detección de contornos.

Si bien el prototipo fue capaz de detectar vehículos en todos los videos, se encontraron rápidamente los tres problemas mencionados al final de la sección de trabajos relacionados. El principal defecto del prototipo es que no puede distinguir entre las sombras y los vehículos que las generan debido a que ambos se mueven juntos, esto lleva a errores de precisión en las posiciones

de los vehículos y aumenta el número de objetos superpuestos detectados en la escena. Por esta razón el prototipo fue descartado.

2.4.2. Prototipo 2: Clasificador en Cascada

El segundo prototipo desarrollado se basa en la utilización de un clasificador en cascada genérico incluido en OpenCV y JavaCV para detectar vehículos. El pseudocódigo utilizado se encuentra en el Algoritmo 2.

```
inicializarClasificador();
frame = leerFrame();
mientras existan frames sin leer hacer
    frameEG = convertirAEscalaGrises(frame);
    contornos = buscarObjetosClasificador(frame);
    dibujarContornos(contornos, frame);
    mostrarCuadro(frame);
    frame = leerFrame();
fin
```

Algoritmo 2: Prototipo Clasificador en Cascada.

Para llevar a cabo esto es necesario contar con una base de datos de imágenes de vehículos para entrenar el clasificador. Como el cometido de este prototipo es evaluar las tecnologías disponibles se utilizó un video alternativo para simplificar la prueba y los conjuntos de entrenamiento. En el video una mano sostiene una lapicera y la desplaza alrededor de la escena, acercándola a la cámara y rotándola. Para crear el archivo de entrenamiento del clasificador se utilizaron alrededor de 10 fotos tomadas a la lapicera desde diferentes ángulos y en diferentes posiciones como los positivos y alrededor de 30 que no contenían la lapicera como los negativos.

El prototipo fue capaz de detectar la lapicera en dos tercios de los cuadros del video. Si bien el algoritmo acertó en la mayor parte del video, los resultados no fueron los esperados. Suponiendo que el algoritmo es correcto el problema se debe al pequeño conjunto de muestras para el entrenamiento. Considerando entonces la diversidad de vehículos y posiciones en una escena, el mínimo número de muestras para que el algoritmo funcione ascendería a 10.000 imágenes aproximadamente. Como no se contaba con bases de datos de imágenes de vehículos, el esfuerzo necesario para la toma de muestras sería muy grande y la efectividad del algoritmo no estaba probada se decidió descartar el prototipo.

2.4.3. Prototipo 3: Sustracción de Fondo

El tercer prototipo se basa en la utilización del algoritmo de sustracción de fondo en conjunto con la detección de gotas. El Algoritmo 3 contiene el

pseudocódigo utilizado.

```
inicializarSustractorFondo();
inicializarDetectorGotas();
frame = leerFrame();
mientras existan frames sin leer hacer
    frameSinFondo = sustraccionFondo(frame);
    aplicarFiltroValorUmbral(frameSinFondo);
    aplicarFiltroGaussiano(frameSinFondo);
    contornos = buscarGotas(frameSinFondo);
    dibujarContornos(contornos, frame);
    mostrarCuadro(frame);
    frame = leerFrame();
fin
```

Algoritmo 3: Prototipo sustracción de fondo y detección de gotas.

Durante las pruebas se utilizaron diferentes parámetros de configuración en ambos algoritmos en busca de mejorar la detección. En el caso del sustractor de fondo se probaron diferentes variaciones de parámetros como el coeficiente de aprendizaje o el largo del historial, y en el caso del detector de gotas se probaron diferentes condiciones de área mínima de rectángulos y se utilizaron otras propiedades para filtrar resultados no deseados. También se utilizaron varias combinaciones de filtros tanto para restringir el área de detección utilizando máscaras como para eliminar el ruido de las imágenes o remover los píxeles que no son de interés para la detección de gotas.

Los resultados de este prototipo fueron muy buenos. Un porcentaje muy alto de vehículos se detectó sin dificultades, sin embargo, en algunos casos partículares la detección no funcionaba correctamente, problema que fue resuelto posteriormente junto con la superposición de vehículos. En cuanto a los problemas de iluminación en la escena, el algoritmo de sustracción de fondo es capaz de lidiar con los cambios progresivos de iluminación debido a la constante actualización que hace del fondo de la escena, por eso, los eventos como la puesta de sol no deberían presentar problemas. Otra ventaja de la sustracción de fondo es la detección de sombras, gracias a esta característica las sombras no presentan un problema mayor porque si se utiliza una configuración adecuada los píxeles remanentes de las sombras son muy escasos.

Capítulo 3

Un Sistema de Detección y Seguimiento de Vehículos

3.1. Descripción del Problema

El proyecto se concibe ante la necesidad de extraer información útil sobre el tránsito y realizar controles de la normativa vial. Para ello se debe contar con grabaciones del tránsito tomadas por cámaras colocadas en la vía pública. Los videos entonces pueden ser transmitidos desde una cámara directamente al software o pueden ser leídos desde un archivo ya existente.

Figura 3.1: Vista frontal elevada de la M6 Motorway en Reino Unido.

Para que los videos sean válidos las cámaras deben cumplir ciertos requisitos como estar ubicadas a una altura y una distancia apropiada de la vía en forma estática, enfocando en un ángulo que permita captar los vehículos de modo que no se superpongan en el área en la que se detecten, y enfocar un

número limitado de carriles, entre otros. Además los archivos de video que serán grabados tendrán restricciones según su formato, tamaño, codificación, duración, etc.

Para ilustrar mejor el problema, la Figura 3.1 muestra una captura de ejemplo de un video. En este ejemplo se pueden distinguir dos vías en sentidos opuestos, cada una con tres carriles. Por cada carril transitan vehículos, posiblemente a diferentes velocidades. La vía se pierde en el horizonte bajo un puente donde no es posible distinguir entre los diferentes vehículos.

Dejando de lado el ejemplo anterior e intentando generalizar a otras infraestructuras viales y otros escenarios posibles, es necesario especificar al software, como es la infraestructura que se está estudiando para que pueda realizar las detecciones. Para esto es necesario especificar, a través de una configuración, los parámetros necesarios para cada video en particular:

- Las vías y los carriles con las que cuenta la infraestructura vial.
- La región de interés del video sobre la que debe trabajar el software, considerando el ejemplo de la Figura 3.1, desde la mitad de la imagen hacia abajo podría ser un área razonable para detectar los vehículos. La mitad superior de la imagen se vuelve más difusa y allí se superponen los vehículos pudiendo complicar la detección.
- La existencia de cruces de semáforos, las vías que se ven afectadas por estos y en que forma las afecta.
- La existencia de maniobras no permitidas en una vía o carril como por ejemplo un giro a la izquierda en un semáforo.

Habiendo extraído la información de un video existen dos alternativas para su presentación, ambas son complementarias. La primera es persistirla en un archivo por cada video analizado o en una base de datos común para todos los videos. Como una ejecución para un video de tamaño considerable podría llevar un buen tiempo de procesamiento esta opción de persistencia resulta necesaria para no perder los datos recabados. La segunda alternativa es mostrarlo al usuario en el video al mismo tiempo en que se van obteniendo los resultados, de esta forma se pueden analizar los datos en tiempo real, lo que puede resultar de gran utilidad ante situaciones puntuales.

Para que la interacción con el usuario en tiempo real sea exitosa, el usuariono debería tener que prestar atención a todos los detalles del video. Por lo que el software deberá alertar al usuario cada vez que detecte una situación peligrosa o prohibida como un accidente o una infracción. La generación de alertas en la pantalla le da más libertad para realizar otras tareas al usuario y más comodidad para trabajar.

Por último, para que el sistema sea capaz de procesar múltiples videos simultáneamente en tiempo real debe ejecutarse sobre una infraestructura con alto poder de cómputo. Para satisfacer esta necesidad es necesario que el desarrollo se base en un paradigma multihilo, esto implica no solo la

ejecución de instancias simultáneas de un mismo algoritmo por video sino también la paralelización de varias etapas de ese algoritmo con el fin de obtener el mejor desempeño posible en la plataforma disponible para la ejecución.

3.2. Aspectos Básicos de la Solución

En esta sección se describen las principales características de la solución implementada, como lo son los requisitos de video y la arquitectura del sistema.

3.2.1. Requisitos de Cámara y Video

Como cualquier pieza de software, este sistema posee restricciones de entrada. En este caso, las restricciones se centran principalmente en los videos que serán analizados. Las restricciones mínimas que debe cumplir un video para poder ser analizado son:

- Su formato y codificación deben ser soportados por la librería FFmpeg [39]. Algunos de los formatos más utilizados son MP4, AVI o FLV. Algunos de los codecs más utilizados son H-264, MPEG-1 o MPEG-2.
- La escena debe estar grabada desde una posición fija, o dicho de otra forma, no pueden existir variaciones en la posición y el ángulo de enfoque de la cámara. Aunque leves oscilaciones poco frecuentes, como puede ser el efecto del viento sobre la cámara, no tendrán efecto sobre la detección.
- La escena debe estar compuesta por una calle o una intersección de calles, con vehículos transitando sobre ella.
- La cámara debe ubicarse a una altura que permita enfocar claramente las vías que sean de interés, mínimo 5 metros aproximadamente, aunque puede variar según la infraestructura.
- El ángulo de enfoque de la cámara (respecto al suelo) debe ser lo suficientemente amplio como para que no se superpongan los vehículos. Dicho ángulo no debe ser inferior a 30 grados, tal como lo ilustra la Figura 3.2.

Si bien los requisitos mencionados anteriormente son los mínimos necesarios para un correcto funcionamiento del sistema, también deben tomarse en cuenta otros factores para obtener un mejor rendimiento:

- Es conveniente que la posición de la cámara y su ángulo minimicen la superposición de vehículos en el video, para simplificar el procesamiento.
- No es recomendable enfocar y tomar datos en más de 6 carriles en un mismo video, ya que la superposición de vehículos puede resultar en fallas en la detección.

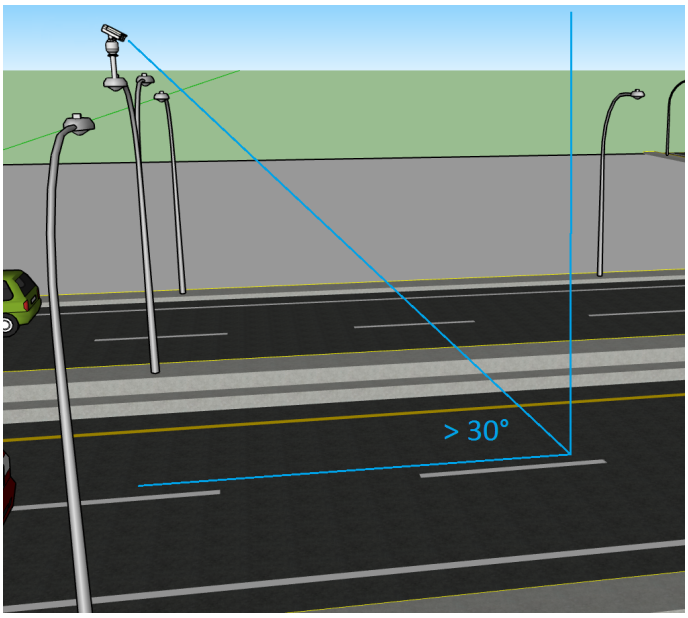

Figura 3.2: Ángulo de la cámara para una correcta visualización.

- Si bien no hay restricciones en cuanto al ángulo de la cámara respecto a las vías, es recomendable que la cámara enfoque en la dirección de las vías. Esto simplifica los cálculos realizados y permite el uso de algunas características exclusivas para este caso de visión frontal, que no están disponibles en otros.

Para ilustrar un poco mejor los puntos anteriores es necesario considerar diferentes puntos de vista, sobre un mismo escenario. En la Figura 3.3 puede apreciarse una vista frontal elevada de un cruce. En este caso de estudio resultaría fácil detectar, por ejemplo, cuando un vehículo no respeta el semáforo o dobla a la izquierda, para los vehículos que circulan en el sentido de la vista de la cámara. En cambio, para los vehículos que circulan por las vías laterales, resultaría más complicada la detección y el seguimiento debido a que se superponen fácilmente. Si bien esta vista resulta muy conveniente, casi ideal, cuenta con un problema, y es que la altura requerida para obtener una visión adecuada es muy grande. La mayoría de las calles no cuentan con columnas de alumbrado tan altas como para colocar una cámara que capture esta vista. Por esto, en la mayoría de los casos se cuenta con menos altura y ángulo, por ejemplo, la altura de un semáforo.

La siguiente vista, ilustrada en la Figura 3.4, es sobre el mismo cruce. La cámara se sitúa en una columna de alumbrado baja, a la misma altura de un semáforo. Puede apreciarse una disminución considerable del ángulo respecto al suelo, que de seguir igual que en la vista anterior, no permitiría visualizar más que un vehículo. Aunque el campo de visión es menor que en

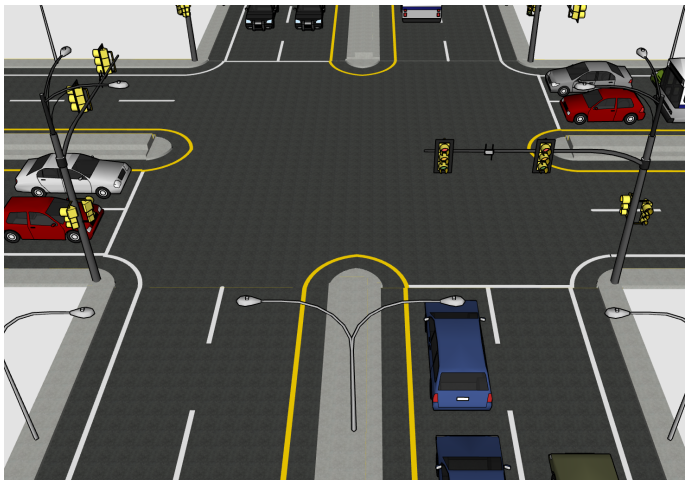

Figura 3.3: Vista frontal aérea de un cruce.

el ejemplo anterior, aún resulta posible detectar y seguir a los vehículos e incluso es posible detectar las infracciones mencionadas antes. El problema de esta vista está en las superposiciones de los vehículos que transitan por el mismo carril, la cual se hace más notoria a medida que los vehículos se alejan de la cámara o cuando un vehículo muy grande cruza por delante de la cámara y obstruye gran parte de la visión. Respecto al ejemplo anterior, esta vista tiene una ventaja y es la posibilidad de visualizar mejor a los vehículos, lo que implica por ejemplo, que sería posible tomar las matrículas siempre que la calidad del video lo permita.

Figura 3.4: Vista frontal de un semáforo.

La última vista, ilustrada por la Figura 3.5, no se sitúa en el cruce, sino un poco por detrás. La vista lateral aérea de dos carriles permite al sistema

detectar con más fiabilidad la velocidad de los vehículos y la proximidad entre ellos al utilizar el ancho de la cámara en vez del largo. Si bien esta vista no es la más recomendable porque no es frontal, resulta muy práctica para evitar las superposiciones entre los vehículos. El uso de este tipo de vistas se recomienda para las vías de alta velocidad, como carreteras o autopistas, que cuentan con infraestructuras grandes y columnas de alumbrado muy altas. Además, es en estas vías rápidas donde resulta más interesante el estudio de la velocidad y la distancia entre los vehículos.

Estas son algunas de las vistas más comunes en infraestructuras y las más recomendables para el sistema. Pueden existir casos especiales en los que resulten más o menos convenientes estos enfoques, pero lo importante a la hora de colocar una cámara es tener en cuenta los requisitos mínimos, junto con la superposición de vehículos y la cantidad de carriles que abarca la escena.

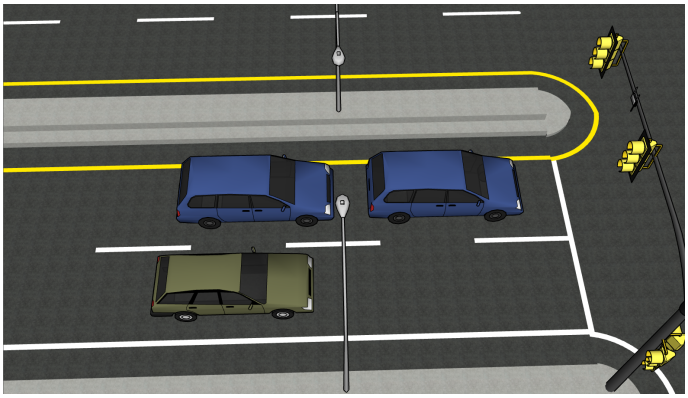

Figura 3.5: Vista lateral aérea de una vía.

3.2.2. Arquitectura de la Solución

La solución se basa en el modelo cliente/servidor. Consta de tres componentes principales:

- El servidor, que analiza los videos y almacena los datos extraídos en la base de datos de acuerdo a las solicitudes que recibe de los clientes.
- El cliente, una aplicación liviana que muestra la información procesada a los usuarios junto con los videos.
- La base de datos, que almacena los datos extraídos y contiene la información de control de cada video.

La Figura 3.6 es diagrama de la componentes de la arquitectura utilizada. Los clientes se comunican con el servidor mediante la interfaz de sockets de Java. En cuanto a la conexión con la base de datos, tanto clientes como servidor, utilizan Javax Persistence en su versión 2.1.0. La base de datos

utilizada es MySQL en su versión 5.6 y las aplicaciones se comunican con ella mediante la versión 5.1.6 del controlador.

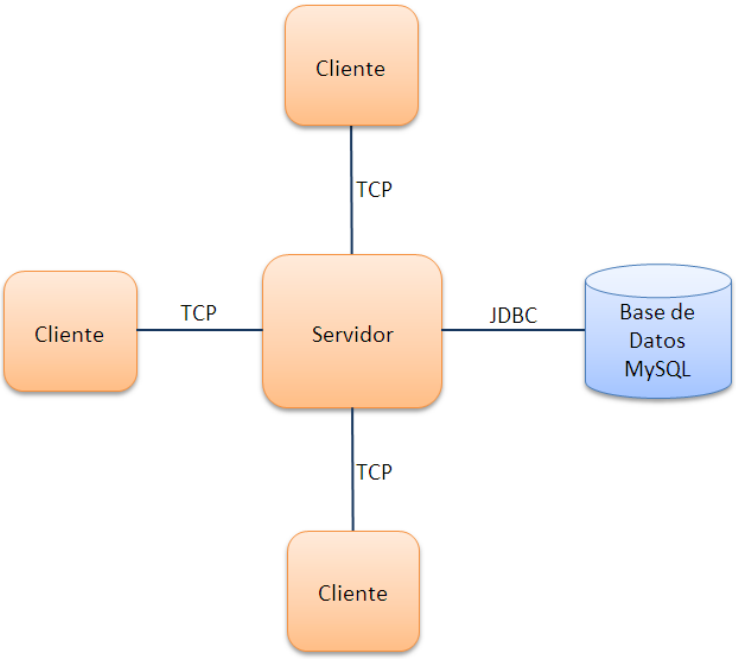

Figura 3.6: Diagrama de Componentes.

3.3. Cliente

El cliente permite al usuario la visualización de los resultados obtenidos, operando en cualquiera de los dos modos disponibles:

- Modo Servidor: el servidor y el cliente trabajan sincronizados. El usuario se conecta al servidor por medio del cliente y solicita que se analice un video. Los resultados son vistos por el usuario en tiempo real, a medida que son calculados por el servidor.
- Modo Base de Datos: se utiliza cuando un video ya fue procesado y tiene la ventaja de no involucrar al servidor. Requiere del archivo de video ya procesado y la conexión a la base de datos. El cliente consultará a la base de datos los resultados obtenidos para mostrar al usuario, a medida que vaya reproduciendo el video.

Se configura mediante el cuadro de dialogo mostrado en la Figura 3.7, dicho cuadro es accesible desde el menú Opciones del cliente. En el se especifica la información necesaria para utilizar el modo Servidor o el modo Base de Datos. En ambos casos, es necesario actualizar la lista de videos disponibles y seleccionar uno de ellos.

Por otro lado, el menú Cliente le permite al usuario pausar, reanudar y detener la reproducción del video, lo que puede resultar útil en caso de que desee observar algún detalle en el video o terminar la ejecución.

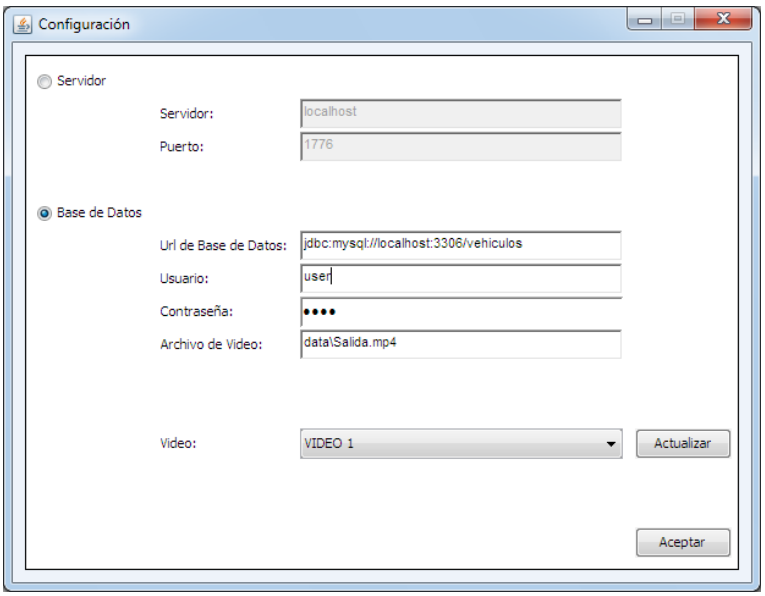

Figura 3.7: Cuadro de diálogo de configuración.

La información se muestra en tres paneles. El panel superior contiene la información agrupada en su total y por carril respecto al conteo de vehículos, los tipos de vehículos contados, los tipos de alertas y las velocidades máxima, mínima y promedio. El panel inferior reproduce el video y muestra el tiempo de la reproducción. Por último, el panel izquierdo contiene la información por vehículo, incluyendo una foto, un identificador, el carril por el que transita, el tipo de vehículo, la velocidad media a la que transitó y la lista de alertas generadas. La Figura 3.8 es una captura de pantalla de la aplicación cliente en funcionamiento.

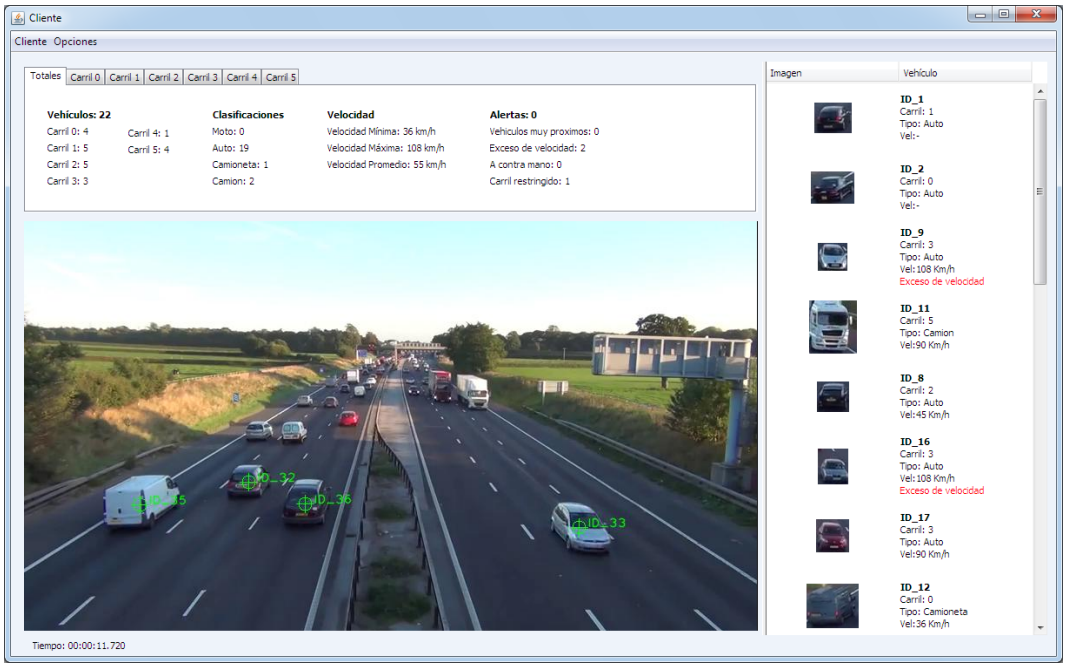

Figura 3.8: Captura de la aplicación cliente en funcionamiento.

3.4. Almacenamiento en Base de Datos

La persistencia de datos es de suma importancia para la realización de posteriores análisis de resultados. La forma más conveniente de persistencia es en una base de datos relacional. El sistema registra los datos relevantes sobre cada vehículo que se asocia a un video analizado. Estos datos relevantes son:

- Identificador de vehículo
- Instante de primera aparición en el video
- Identificador del cuadro de primera aparición
- Carriles transitados
- Clasificación del vehículo
- Velocidad calculada
- Imagen del vehículo tomada del video
- Alertas generadas por el vehículo

La información mencionada anteriormente se organiza en las siguientes tablas:

- Vehículos: contiene la información extraída para cada vehículo como identificador de video, identificador de vehículo, velocidad, instante de aparición, identificador de cuadro de aparición, carriles transitados, clasificación, velocidad e imagen de captura.

- Alertas-Vehículos: contiene la información de cada alerta generada por un vehículo en un video. Se almacenan los identificadores de video, vehículo y alerta, el instante de detección de la alerta y la descripción de la alerta.

También existen tablas de metadatos. Estas son consultadas por el cliente, para desplegar la información de los carriles y alertas de un video cuando se ejecuta en modo Base de Datos. Esto es necesario, porque al no ejecutar por intermedio del servidor, el cliente no conoce los detalles de configuración del video. Las tablas utilizadas para este fin son:

- Videos: almacena los nombres de los videos, que se utilizan como identificadores
- Carriles, Alertas y Clasificaciones: contienen información básica de los carriles, alertas o clasificaciones configurados. Utilizan el mismo formato, a saber, un identificador de video, otro de entidad (carril, alerta o clasificación) y el nombre de la entidad.

El esquema de base de datos utilizado puede apreciarse en la Figura 3.9.

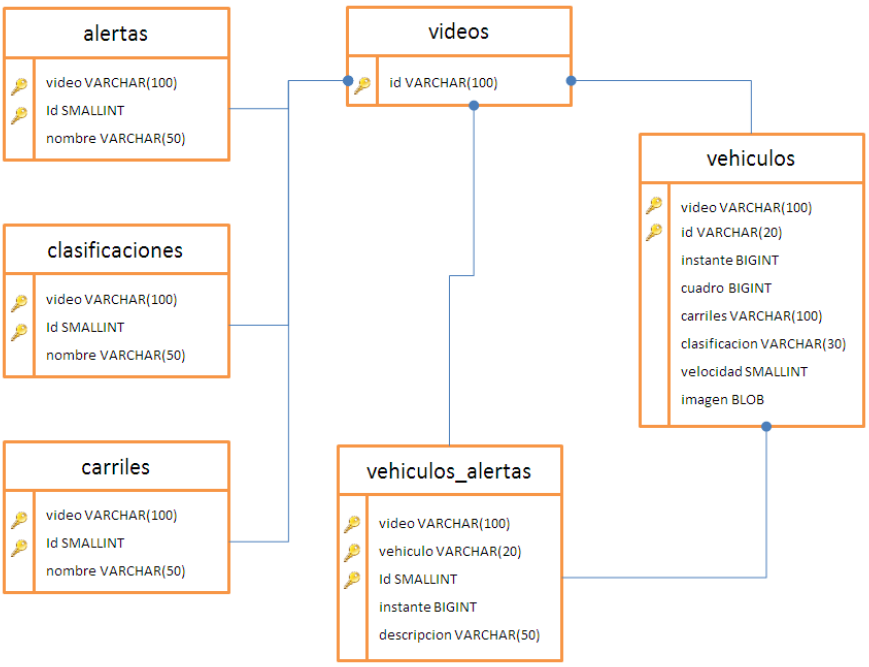

Figura 3.9: Esquema de base de datos.

3.5. Servidor Secuencial

3.5.1. Características Principales

A continuación se listan las principales características del servidor secuencial.

Manipulación de Video Lee los archivos de video para realizar el análisis y también puede guardar nuevos archivos de video como resultado de un análisis agregando recuadros sobre los vehículos, números de identificación, cantidad de vehículos y tiempo de procesamiento según especifique la configuración.

Detección y Seguimiento de Vehículos Los vehículos son detectados, ubicados y rastreados a lo largo de su trayectoria por la escena. Para lograr esto se utiliza el principio de localidad en cuadros de video sucesivos, es decir, un vehículo encontrado en el cuadro n+1 se reconoce en función de la cercanía con la última posición de un vehículo detectado en el cuadro n. En particular, si la nueva posición se encuentra a una distancia d predeterminada de la posición anterior, el reconocimiento es positivo. En el caso de que un vehículo no se pueda asociar a una posición anterior es porque se trata con un nuevo vehículo en la escena.

Reducción de Cuadros del Video El usuario puede permitir al servidor saltear cuadros de video en el procesamiento. Si bien esto puede provocar falta de precisión en la detección en videos con baja tasa de cuadros, por ejemplo en el cálculo de la velocidad media de un vehículo, también mejora considerablemente el rendimiento de la aplicación. La decisión del uso de esta opción queda a cargo del usuario, que puede elegir entre la cantidad de datos a extraer y la velocidad de respuesta del sistema. Un escenario donde puede ser útil la aplicación de la reducción de cuadros es en calles con velocidades máximas bajas.

Registro de Datos Cada vehículo es registrado en la base de datos con un identificador asociado al video, el instante en el que apareció por primera vez en la escena, los carriles por los que circuló, su tipo, su velocidad media y las alertas que generó. El almacenamiento de estos datos permite que el cliente pueda correr por separado, sin necesidad de conectarse al servidor. Además, permite consultar la información extraída sobre cada video y facilita la generación de reportes y estadísticas.

Opciones de Ejecución Para lograr un mejor rendimiento, la configuración permite al usuario especificar para cada video si desea almacenar los

datos extraídos en la base de datos, y si desea visualizar el video resultante en pantalla a medida que se procesa o si desea grabarlo.

Opciones de Prueba A modo de simplificar la tarea de prueba de configuraciones, el usuario puede especificar si desea ver o grabar el resultado de la Sustracción de Fondo, esto es, las imágenes binarias generadas por el algoritmo con fondo negro (sustraido) y los objetos detectados blancos. También se puede solicitar que se dibujen los objetos no confirmados y los carriles sobre la salida de video. De esta forma, los usuarios pueden comprobar la correctitud de la configuración elegida o probar alternativas. En la Figura 3.8 se muestran dos imágenes, una de ellas correspondiente a un cuadro de un video con los objetos detectados señalados y la otra es el resultado de la Sustracción de Fondo al mismo cuadro, al que luego se le aplicaron los filtros definidos en la configuración.

Cabe mencionar, que en la Figura 3.10 existen un par de incoherencias al comparar las dos imágenes. La primera es por qué algunos de los vehículos identificados (color blanco) en la imagen inferior no son recuadrados en la imagen superior, eso es porque existe una región de interés para el detector de vehículos, sólo los objetos que se encuentren dentro de esa región de la imagen serán dibujados. La razón del uso de esta región es evitar que los vehículos encontrados se encuentren parcialmente dentro de la escena. La segunda incoherencia es la diferencia entre la cantidad de vehículos observados en la imagen superior y la inferior o los vehículos que se ven cortados en la imagen inferior, y se da por un motivo similar a la anterior. Para este video, la configuración especifica el uso de un filtro máscara que restringe el área de búsqueda del algoritmo a los tramos de los 6 carriles que se encuentran más próximos a la cámara, lo cual permite reducir un poco los costos de procesamiento.

3.5.2. Configuración

El servidor permite procesar videos en dos modos. El primero en sincronización con el cliente, como fue explicado anteriormente. El segundo mediante la configuración del servidor, sin la intervención del cliente. Este último modo ofrece visibilidad reducida de la información extraída, pero resulta más práctico para realizar pruebas sobre el servidor y la configuración. En ambos modos, una instancia del servidor debe ser iniciada para analizar cada video, ya que el servidor finaliza su ejecución luego de haber finalizado el procesamiento.

Se utiliza un archivo para almacenar la configuración del servidor. Este archivo incluye entre otras cosas el modo de ejecución del servidor (con o sin cliente), el puerto del sistema sobre el que intercambiará datos y especificaciones para cada video que se va a analizar. Para cada video es necesario

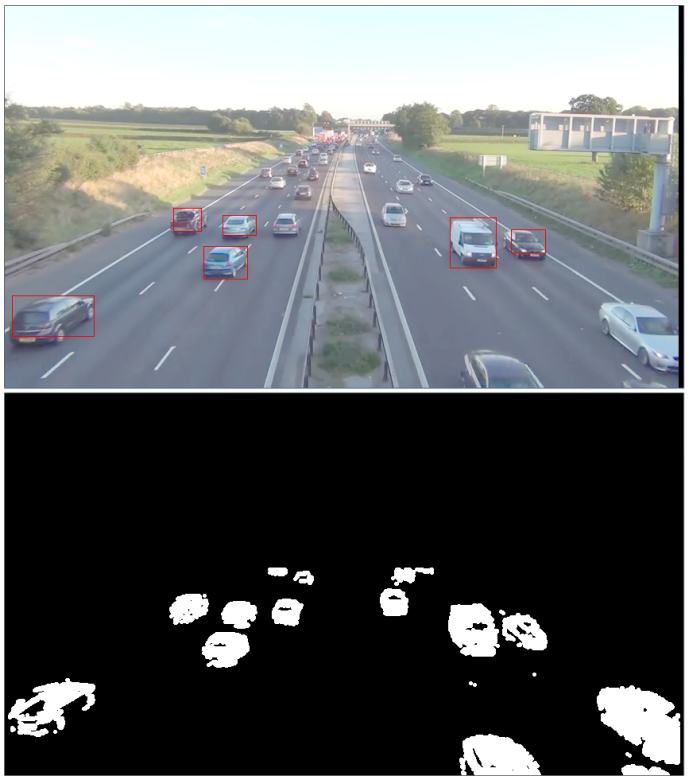

Figura 3.10: Ejemplo de salida de las opciones de prueba.

especificar varios parámetros, a continuación se explican los más importantes.

Especificaciones para la Detección y el Seguimiento El algoritmo de Sustracción de Fondo se configura mediante varios parámetros en la configuración. Distintas configuraciones producirán distintos resultados. Es conveniente realizar pruebas con varios valores antes de utilizar una configuración definitiva. En cuanto al seguimiento de los vehículos, se toman en cuenta varios parámetros como la cantidad de apariciones que debe tener un vehículo para considerarlo válido, la cantidad de cuadros que deben transcurrir para considerarlo fuera de la escena y el coeficiente de rastreo (varios parámetros) que determina el umbral máximo para considerar la asociación entre vehículos de cuadros consecutivos.

Filtros de Imágenes Los filtros de imágenes se aplican luego del algoritmo de Sustracción de Fondo sobre cada cuadro del video. Su función es mejorar la calidad de la imagen para detectar mejor los objetos de la escena en el paso siguiente. Los filtros disponibles son erosión, dilatación, gaussiano, de valor umbral y máscaras binarias and y or que se utilizan para restringirse

a un área de interés en la imagen.

Carriles Son la escencia de la extracción de información de los vehículos. Se delimitan mediante dos rectas, no se permiten curvas. Definen la distribución de las vías y restringen las zonas de circulación. También se utilizan para delimitar vehículos muy grandes y para detectar algunas infracciones.

Criterios de Detección Se aplican en el paso siguiente a la detección de objetos en la escena y previo al seguimiento de los vehículos. Se encargan de discriminar entre los objetos candidatos a vehículos. Para discriminar se basan en el tamaño de los objetos y su posición en la escena. Existen además criterios de corrección que se aplican para detectar superposiciones. Estos asumen que si un objeto es demasiado grande como para ocupar dos carriles es porque se trata de dos objetos superpuestos, en ese caso se divide al objeto por carril.

Patrones de Detección Son funciones que se aplican sobre los vehículos ya reconocidos para detectar distintas características. Cada patrón detecta una característica en particular como la velocidad media, el tipo de vehículo, el sentido de circulación, etc. Algunos patrones, como el de velocidad media o el de carril restringido pueden generar alertas sobre los vehículos. Esto ocurre cuando un vehículo transgrede una norma acordada en uno de los patrones. Las alertas programadas son exceso de velocidad, circulación en sentido contrario, circulación por carril restringido, vehículos muy próximos, maniobra prohibida y semáforo no respetado. Cada patrón posee su propia configuración, la cual le permite ser flexible según las diferentes escenas.

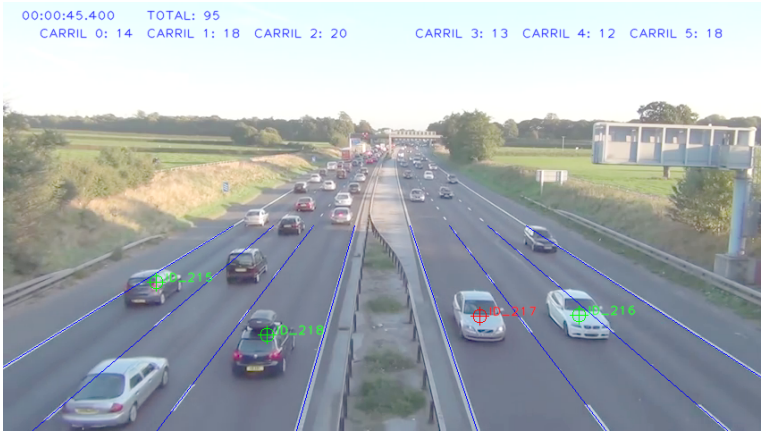

Figura 3.11: Captura de un cuadro procesado.

La Figura 3.11 ilustra algunos de los puntos mencionado anteriormente. Se trata de una captura de un cuadro procesado. En la parte superior

pueden apreciarse los datos del video como el tiempo transcurrido, el total de vehículos y los totales por carril. También pueden observarse los carriles dibujados por líneas, sobrepasando un poco la región de interés del detector de vehículos. Cada vehículo identificado que se encuentra en la región de interés está marcado por un círculo verde (una especie de mira) y un identificador a excepción de un vehículo que está marcado en color rojo, lo que significa que generó una alerta por algún comportamiento indebido, como exceso de velocidad por ejemplo.

Cabe destacar que la flexibilidad del sistema se debe a la amplitud de la configuración, la cual permite representar la mayoría de las infraestructuras viales. Además, permite al usuario especificar al sistema la información que le es relevante y por lo tanto omitir el procesamiento de la información que no lo es.

3.5.3. Diagrama de Flujo

La Figura 3.12 muestra el diagrama de flujo del servidor.

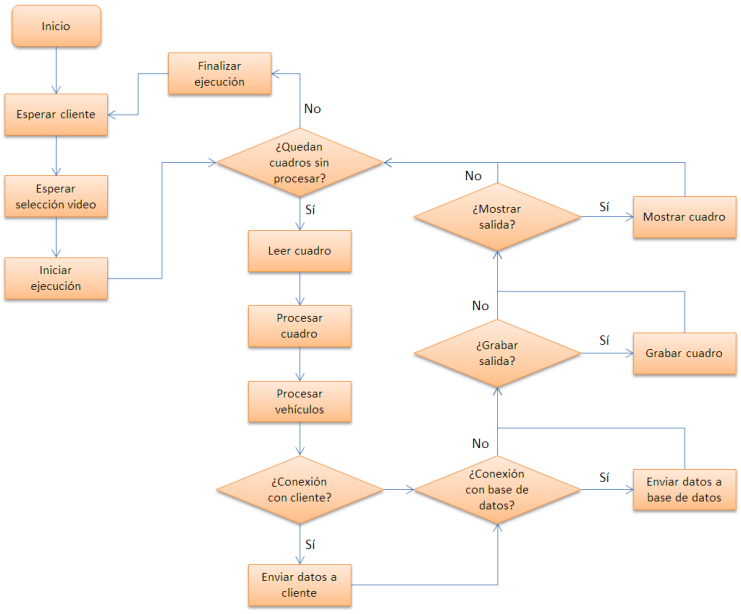

Figura 3.12: Diagrama de flujo del servidor.

3.5.4. Algoritmo

En principio, el algoritmo utilizado se basó completamente en el tercer prototipo construido. Luego de efectuadas varias pruebas sobre el sistema, se verificó que el algoritmo tardaba demasiado en procesar cada cuadro, lo que

no permitiría el procesamiento en tiempo real. Luego de un análisis más profundo, se concluyó que el algoritmo de detección de gotas era el culpable de las excesivas demoras. Se optó inmediatamente por descartar el algoritmo e incorporar la detección de contornos en su lugar. Los cambios en los tiempos de ejecución fueron notorios, la detección de contornos resultó 50 veces más rápida que la detección de gotas, lo que permitió al sistema el procesamiento en tiempo real. Si bien el algoritmo de detección de contornos no resultaba tan efectivo en comparación con su antecesor, aplicando algunas operaciones sobre los objetos detectados es posible mejorar la detección prácticamente sin agregar costos de procesamiento. El algoritmo puede dividirse en cuatro etapas, cada una encargada de distintas funciones, que deben ejecutarse en forma sucesiva. La separación se realizó con la intención de evitar grandes modificaciones al pasar a la versión paralela, aunque en esta versión todo la ejecución es secuencial. A continuación se describen todas ellas en orden y se muestran sus algoritmos.

Entrada de Video La primera etapa del algoritmo es la más simple de todas, tal como se muestra en el Algoritmo 4. Consiste en la lectura del video especificado y la aplicación de la reducción de cuadros del video si corresponde. El resultado de esta primer etapa es la extracción del próximo cuadro de video y su instante asociado.

```
si existen cuadros sin leer entonces
    para cada i en reduccionCuadros hacer
        video.leerCuadro();
    fin
    si existen cuadros sin leer entonces
        devolver [video.leerCuadro(), video.obtenerInstante()];
    fin
fin
```

Algoritmo 4: Extracción de un cuadro.

La función leerCuadro, lee el siguiente cuadro del archivo de video, y la función obtenerInstante devuelve el instante de tiempo asociado al último cuadro leído.

Procesamiento de Cuadros En la segunda etapa se extraen los objetos de la escena, según el Algoritmo 5. Primero se aplica el algoritmo de Sustracción de Fondo, luego se aplican los filtros de imágenes sobre el resultado y se detectan en la imagen filtrada los contornos, que corresponden a los objetos de la escena. Por último se analizan los objetos detectados, eliminando los que están incluidos dentro de otro objeto y fusionando aquellos que se

superponen. La entrada de esta etapa es un cuadro de video y la salida es una lista de rectángulos que contienen cada objeto detectado en la imagen.

```
cuadroObj = aplicarSustraccionFondo(cuadro);
para cada filtro en filtros hacer
    filtro.aplicar(imagen);
fin
objetos = aplicarDeteccionContornos(cuadroObj);
para cada objeto en objetos hacer
    para cada objeto2 en objetos hacer
        si objeto != objeto2 entonces
            si objeto incluido en objeto2 entonces
                objetos.remover(objeto);
            fin
            si objeto2 incluido en objeto entonces
                objetos.remover(objeto2);
            fin
            si objeto se superpone con objeto2 entonces
                objetos.remover(objeto);
                objetos.remover(objeto2);
                objetos.insertar(objeto + objeto2);
            fin
        fin
    fin
fin
devolver [cuadro, cuadroObj, instante, objetos];
```

Algoritmo 5: Procesamiento de un cuadro.

Aquí, la función aplicarSustraccionFondo, aplica el algoritmo de sustraccion de fondo sobre un cuadro, manteniendo actualizado el estado de aprendizaje del algoritmo, devuelve la imagen resultado como mapa de bits binario con los píxeles detectados en blanco y el resto en negro. La otra función, aplicarDeteccionContornos, aplica el algoritmo de detección de contornos sobre una imagen, devolviendo una lista de coordenadas de rectángulos que contienen cada objeto detectado. Por último, las funciones insertar y remover, insertan y remueven un objeto de una lista respectivamente.

Procesamiento de Datos La tercera etapa, tal como se describe en el Algoritmo 6, se encarga de discriminar los vehículos de los objetos que fueron encontrados en la etapa previa, para ello utiliza los criterios de detección. Luego se encarga de hacer el seguimiento a los vehículos encontrados en base al coeficiente de rastreo y de extraer la información correspondiente a cada uno mediante los patrones de detección. Por último, sincorniza la

información con la base de datos (opcional) y devuelve una lista de los vehículos y su información.

```
removerVehiculosQueTransitaron(vehiculos, instante);
objetos = aplicarCriterios(objetos);
objetos = aplicarCriteriosCorreccion(objetos);
para cada objeto en objetos hacer
    vehiculo = buscarVehiculo(objeto, instante, coeficienteRastreo);
    si no existe el vehiculo entonces
        vehiculo = crearVehiculo(objeto);
        vehiculos.insertar(vehiculo);
        vehiculo.imagen = tomarImagen(vehiculo, cuadro);
    fin
    asignarCarril(vehiculo, carriles);
fin
aplicarPatrones(vehiculos);
actualizarContadores(vehiculos, carriles);
si existe conexión a base de datos entonces
    enviarVehiculos(vehiculos);
fin
devolver [cuadro, instante, vehiculos];
```

Algoritmo 6: Procesamiento de vehículos de un cuadro.

La función removerVehiculosQueTransitaron, remueve de la lista de vehículos que circulan, aquellos que no volvieron a ser detectados en los últimos cuadros, según la configuración. Se utiliza también la función aplicarCriterios, que evalúa los criterios especificados en la configuración sobre los objetos detectados en la etapa anterior, descartando aquellos que no los cumplan. Luego, la función aplicarCriteriosCorreccion, que evalúa los criterios de corrección especificados en la configuración sobre los objetos detectados en la etapa anterior, corrigiendo a los objetos que los cumplan. La función buscarVehiculo, dado un objeto, el instante actual y el coeficiente de rastreo configurado, busca en la lista de vehículos en circulación la correspondencia con un vehículo que no haya sido asociado en dicho instante de acuerdo al coeficiente. Mediante crearVehiculo, se crea una instancia nueva de vehículo con un identificador generado a partir de un objeto y su ubicación. La función tomarImagen, toma una captura de un vehículo según su ubicación en el cuadro dado. A través de asignarCarril, se asigna el carril correspondiente a un vehiculo según su ubicación. La función aplicarPatrones, aplica todos los patrones especificados en la configuración a cada vehículo en la lista de vehículos que circulan, generando las alertas correspondientes. Luego está la función actualizarContadores, que actualiza los contadores de vehículos por carril y total según las últimas detecciones de vehículos. Por

último, la función enviarVehiculos, envía los datos de los nuevos vehículos detectados a la base de datos.

Salida de Video El Algoritmo 7, muestra la última etapa, la cual se encarga de la edición, grabación, difusión y proyección del video final, siendo opcionales estas últimas tres funciones. Esta etapa recibe los datos extraídos y el cuadro original del video, luego escribe sobre el cuadro la información que el usuario haya especificado en la configuración, como los identificadores de cada vehículo, los contadores de asociados a cada carril, el instante de tiempo, etc. Una vez terminado el proceso de edición, se procede a grabar el video resultado, mostrarlo en pantalla y/o enviarlo a la aplicación cliente, tal como el usuario haya especificado.

```
mostrarVehiculos(cuadro, vehiculos);
si se muestra la información entonces
    mostrarInformacion(cuadro, instante, carriles);
fin
si se muestra el video entonces
    ventana.mostrar(cuadro);
fin
si se graba el video entonces
    grabador.grabar(cuadro);
fin
si si existe la conexión con un cliente entonces
    conexion.enviar(cuadro, instante, vehiculos);
fin
```

Algoritmo 7: Procesamiento de la salida de un cuadro.

Esta etapa también se encarga de ejecutar las opciones de prueba como mostrar o grabar el video de los cuadros generados por la Sustracción de Fondo, y de mostrar características particulares de la detección como los recuadros asociados a los objetos o a los vehículos.

La función mostrarVehiculos, dibuja en el cuadro dado los vehículos detectados. Luego, la función mostrarInformación, dibuja el instante actual y los contadores de los carriles en el cuadro dado. También se utiliza la función mostrar, que muestra en la ventana de video el cuadro dado. La función grabar, agrega el cuadro a la salida de video. Por último, la función enviar, envia mediante la conexión TCP el cuadro, el instante y la información de los vehículos en circulación a la aplicación cliente.

3.5.5. Diagrama del Servidor

La Figura 3.13 muestra un diagrama del servidor. En el pueden observarse los cuatro procesadores alojados dentro de una ejecución, la cual se comunica con la aplicación cliente por medio de una conexión.

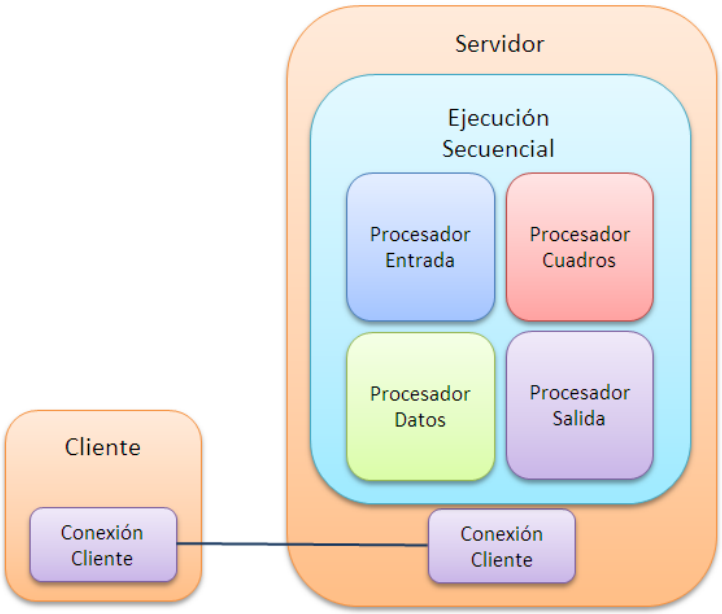

Figura 3.13: Diagrama del servidor secuencial.

3.6. Servidor en Paralelo

El servidor secuencial implementado es el punto de partida para la implementación del servidor en paralelo. Esta nueva implementación utiliza el mismo algoritmo como base y agrega varias funcionalidades:

3.6.1. Características Principales

A continuación se listan las principales características del servidor en paralelo. Que se agregan a las que posee el servidor secuencial.

Múltiples Ejecuciones Simultáneas El servidor en paralelo permite la conexión simultánea con varios clientes y el procesamiento simultáneo de varios videos. Para ello, cuenta con un pool de hilos que administra los hilos correspondientes a las diferentes ejecuciones. El pool crea hilos a demanda, a medida que se solicitan nuevas ejecuciones. Cuando una ejecución termina, el pool se encarga de destruir todos los hilos asignados a esa ejecución. El

tamaño del pool está acotado a un máximo de hilos, que se especifica en la configuración. Esta limitación de tamaño garantiza una restricción del uso de los recursos disponibles, pero también limita la cantidad de ejecuciones simultáneas. Cuando un cliente solicita una ejecución para la que no hay hilos disponibles suficientes, esta pasa a estado de espera hasta que existan hilos suficientes. Un mismo cliente no puede tener asociadas dos ejecuciones a la vez, pero sí puede solicitar otra ejecución luego de haber terminado una.

Soporte para Ejecuciones Secuenciales o Paralelas El servidor en paralelo permite dos tipos de ejecuciones: secuenciales o paralelas. Las ejecuciones secuenciales son exactamente iguales a las que implementaba el servidor secuencial y se ejecutan bajo un mismo hilo, de forma secuencial. En cambio, las ejecuciones paralelas se ejecutan en varios hilos, uno por cada procesador (entrada de video, cuadros, datos y salida de video), pudiendo utilizar más de un procesador de cuadros para una misma ejecución. La razón por la que interesa utilizar varios procesadores de cuadros es que los videos con resolución muy grande demandan una mayor capacidad de procesamiento en la parte del algoritmo de la que se encarga el procesador de cuadros. De esta forma, se logra evitar el cuello de botella que se forma en el algoritmo secuencial.

Soporte para Reconexión El servidor en paralelo permite al cliente retomar una transmisión luego de haber perdido la conexión entre ambos, aunque el servidor no esperará a que se conecte el cliente para seguir procesando, sino que continuará con la ejecución como si el cliente siguiera conectado. Para poder identificar las transmisiones, el cliente utiliza un nombre de usuario en conjunto con un número como identificador. El servidor asocia este identificador a una conexión, la cual está asociada a una ejecución. De esta forma, cuando el cliente pueda restablecer la conexión, puede solicitar al servidor seguir viendo el video. Esta opción no se encuentra predeterminada para evitar sobrecarga innecesaria en el servidor, debe especificarse desde el cliente, de otro modo el servidor descartará inmediatamente la ejecución si se pierde la conexión.

El servidor en paralelo requiere que la aplicación cliente le especifique un nombre de usuario, el tipo de ejecución y el modo de conexión. Es por ello que el cuadro de opciones de configuración se extiende según muestra la Figura 3.14.

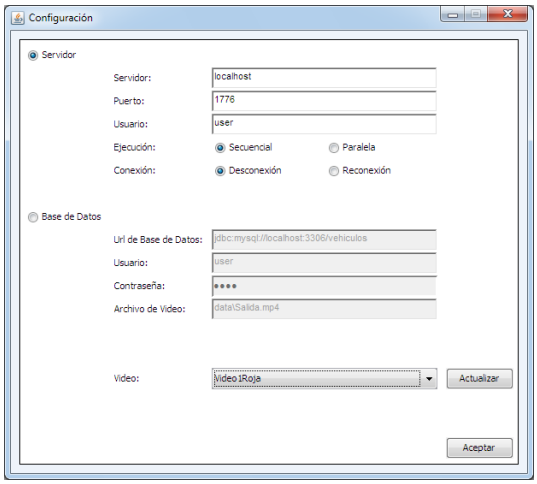

Figura 3.14: Cuadro de diálogo de configuración para el servidor en paralelo.

3.6.2. Algoritmo

El servidor en paralelo utiliza varios tipos de hilos para ejecutar. A continuación se describe cada uno y se muestran sus algoritmos.

Hilo Servidor o Principal En el Algoritmo 8, se puede ver el hilo Principal de la aplicación, que se encarga de supervisar al resto de los hilos. Administra las ejecuciones y sus procesadores a través del pool de hilos. Las ejecuciones arriban en orden al hilo Principal a través de una cola sincronizada, luego este las atiende de a una a la vez, siempre que haya espacio suficiente en el pool de hilos. Además, el hilo Principal mantiene un diccionario de ejecuciones identificadas por su cliente, lo que permite restablecer una conexión interrumpida con dicho cliente.

```
inicializar();
mientras verdadero hacer
    si no existe una ejecución entonces
        ejecucion = colaEjecucionesPendientes.primera();
    fin
    si hay hilos disponibles en el pool entonces
        si la ejecución no está iniciada entonces
            diccionarioEjecuciones.agregar(ejecucion.cliente,
            ejecucion);
            pool.ejecutar(ejecucion);
        fin
        si la ejecución es paralela entonces
            si existen hilos disponibles suficientes en el pool entonces
                pool.ejecutar(ejecucion.procesadorEntrada);
                para cada p en ejecuciones.procesadoresCuadros
                hacer
                    pool.ejecutar(p);
                fin
                pool.ejecutar(ejecucion.procesadorDatos);
                pool.ejecutar(ejecucion.procesadorSalida);
                ejecucion = {};
            fin
        en otro caso
            ejecucion = {};
        fin
    fin
    para cada e en diccionarioEjecuciones hacer
        si e.terminada entonces
            diccionarioEjecuciones.remover(e);
        fin
    fin
fin
```

Algoritmo 8: Algoritmo del Hilo Servidor o Principal.

La función primera, devuelve la primera ejecución de la cola. Sobre el diccionario, operan la función agregar, que agrega al diccionario la clave cliente junto con su ejecución, y remover, que remueve una ejecución del diccionario. La función ejecutar, hace que el pool de hilos cree un nuevo hilo y le asigna una ejecución o un procesador.

Hilo de Conexion de Servidor Este hilo se encarga de recibir las conexiones de los clientes, el mismo se ilustra en el Algoritmo 9. Al recibir una conexión, espera por un mensaje del cliente, el cual puede contener una so-

licitud de ejecución o una solicitud de reconexión a una ejecución existente. Este hilo comparte las estructuras diccionario de ejecuciones en progreso y cola de ejecuciones pendientes con el hilo Principal. En la primera de ellas coloca las ejecuciones en orden de llegada para que sean atendidas luego por el hilo Principal. La segunda la utiliza para reconectar a los clientes con sus ejecuciones cuando se pierde la conexión. Este hilo permanece siempre activo.

```
mientras no finaliza la ejecución hacer
    conexionCliente = recibirCliente();
    mensaje = conexionCliente.recibirMensaje();
    cliente = mensaje.cliente;
    si si el mensaje solicita ejecución secuencial entonces
        colaEjecucionesPendientes.encolar(crearEjecucionSecuencial(cliente,
        conexionCliente));
    fin
    si si el mensaje solicita ejecución paralela entonces
        colaEjecucionesPendientes.encolar(crearEjecucionParalela(cliente,
        conexionCliente));
    fin
    si si el mensaje solicita reconexión y existe su ejecución entonces
        ejecucion = diccionarioEjecuciones.obtener(cliente);
        ejecucion.reconectar(conexionCliente);
    fin
fin
```

Algoritmo 9: Algoritmo del Hilo Conexión de Servidor.

La función recibirCliente, acepta una conexión TCP de la aplicación cliente para servir. La función recibirMensaje, espera y recibe un mensaje de parte de la aplicación cliente. Para colocar las ejecuciones al final de la cola se utiliza la función encolar. Luego, la función obtener, devuelve una ejecución con clave cliente. Por último, la función reconectar, restablece una conexión TCP con la aplicación cliente que tenía una ejecución en curso, luego de ocurrida una desconexión.

Hilo de Ejecución Secuencial El hilo de Ejecución Secuencial ejecuta prácticamente el mismo código que la versión secuencial del servidor, tal como lo muestra el Algoritmo 10. Utiliza un procesador de cada tipo y los ejecuta en el mismo orden en cada iteración, siendo el resultado de un procesador la entrada que toma el siguiente procesador. Una ejecución se crea cuando un cliente hace una solicitud para procesar un video. Cuando el hilo Principal inicia el hilo de una ejecución, lo primero que este debe hacer es enviarle los datos al cliente sobre los videos disponibles en el servidor y

esperar a que el usuario escoja uno de ellos. Al tener un video seleccionado, la ejecución incializa todos los procesadores, basándose en la configuración existente en el servidor. Una vez finalizada la ejecución el hilo deja de existir, liberando espacio en el pool de hilos del servidor para nuevas ejecuciones.

```
video = conexion.incializar(obtenerVideosDisponibles());
si el video seleccionado existe entonces
    incializarProcesadores(video);
    mientras mientras existan cuadros para procesar hacer
        cuadro = entrada.procesar({});
        cuadros.procesar(cuadro);
        datos.procesar(cuadro);
        salida.procesar(cuadro);
    fin
    conexion.finalizar();
    entrada.finalizar();
    datos.finalizar();
    salida.finalizar();
fin
```

Algoritmo 10: Algoritmo del Hilo Ejecución Secuencial.

La función obtenerVideosDisponibles, devuelve los videos disponibles en el servidor, especificados en la configuración. Luego, la función inicializar, inicializa una conexión con el cliente dada una lista de videos disponibles, de la cual el cliente escoge cual reproducir. Para crear las estructuras necesarias para iniciar la ejecución se utiliza la función incializarProcesadores. La función procesar es genérica, tal que dado un objeto del tipo cuadro, que contiene imagen, imagen resultado de la sustracción de fondo, instante, objetos y vehículos, realiza la acción específica del procesador en cuestión. Por último, finalizar, es otra función genérica que destruye las estructuras que fueron utilizadas.

Hilo de Ejecución Paralela El Algoritmo 11, muestra el hilo de Ejecución Paralela, el cual se inicializa de la misma forma que el de Ejecución Secuencial. Su diferencia está en que este hilo no es responsable de la ejecución de los procesadores, sino que cada procesador se ejecuta en un nuevo hilo denominado Contenedor. El hilo de Ejecución Paralela contiene las referencias a cada hilo Contenedor y semáforos que le indican cuando al menos uno de estos hilos terminó y cuando cada uno de ellos terminó. Mediante el uso de estos semáforos, este hilo actúa como maestro y ordena a sus esclavos que finalicen su ejecución si detecta que hubo un error o que el video ya fue

procesado completamente.

```
video = conexion.incializar(obtenerVideosDisponibles());
si el video seleccionado existe entonces
    incializarContenedores(video);
    esperarSemaforoFin();
    si no se detectó error entonces
        notificarFinATodos();
    en otro caso
        notificarErrorATodos();
    fin
    esperarSemaforos();
    conexion.finalizar();
fin
```

Algoritmo 11: Algoritmo del Hilo Ejecución Paralela.

La función incializarContenedores, inicializa e inicia la ejecución de los hilos contenedores. Luego, la función esperarSemaforoFin, espera por el semáforo que determina el fin de uno de los contenedores. La función notificarFinATodos, notifica el correcto fin de la ejecución al resto de los procesadores. También se utiliza la función notificarErrorATodos, que notifica el fin abrupto de la ejecución causado por un error del que no hay recuperación. Finalmente, la función esperarSemaforos, espera por el semáforo que determina el fin de todos los contenedores.

Hilo Contenedor El hilo Contenedor, tal y como lo dice su nombre, oficia de contenedor para un procesador cualquiera. Los procesadores implementan una interfaz genérica que permite tratarlos a todos por igual, independientemente de su tipo. De esta forma, el código de los Contenedores resulta sumamente simple, tal como lo muestra el Algoritmo 12. Estos hilos utilizan una cola como entrada y otra como salida, tomando de la cola de entrada (si existe) la entrada para el procesador y colocando en la cola de salida (si existe) la salida del procesador. De esta forma se crea la misma cadena de procesamiento que en el caso secuencial. Los Contenedores se detienen ante la aparición de errores en la cadena de procesamiento o cuando se termina el procesamiento, por eso cuentan con métodos sincronizados que le permiten al hilo de Ejecución Paralela notificar un error o el fin de la ejecución.

```
procesador.iniciar();
mientras no hay error y (no ha terminado o hay cuadros) hacer
    si si existe entrada entonces
        previo = entrada.primerCuadro();
    fin
    resultado = procesador.procesar(previo);
    si existen salida y resultado entonces
        salida.encolar(resultado);
    fin
fin
procesador.finalizar();
semaforoTodos.permitir();
semaforoIndividual.permitir();
```

Algoritmo 12: Algoritmo del Hilo Contenedor.

La función iniciar, inicializa las estructuras necesarias para la posterior ejecución del procesador. Para extraer el primer cuadro de la cola de entrada se utiliza la función primerCuadro, mientras que para colocar un cuadro al final de la cola de salida se utiliza la función encolar. La función procesar, procesa el cuadro de entrada, si corresponde según el tipo procesador, generando un nuevo cuadro como resultado, siempre que corresponda con el tipo de procesador. Luego está, finalizar, una función genérica que destruye las estructuras que fueron utilizadas. Por último, la función permitir, habilita la ejecución a los procesos que esperan por el semáforo.

Las dos siguientes figuras ilustran mejor el funcionamiento del servidor. La Figura 3.15 muestra un diagrama con todos los tipos de hilos y como se relacionan entre sí. Pueden apreciarse en ella los hilos Principal y Conexión Servidor asociados entre sí, y el primero de ellos asociado a todas las Ejecuciones activas o pendientes. También pueden apreciarse dos Clientes asociados a sus Ejecuciones, siendo una de ellas Secuencial y la otra Paralela, esta última supervisando varios Contenedores.

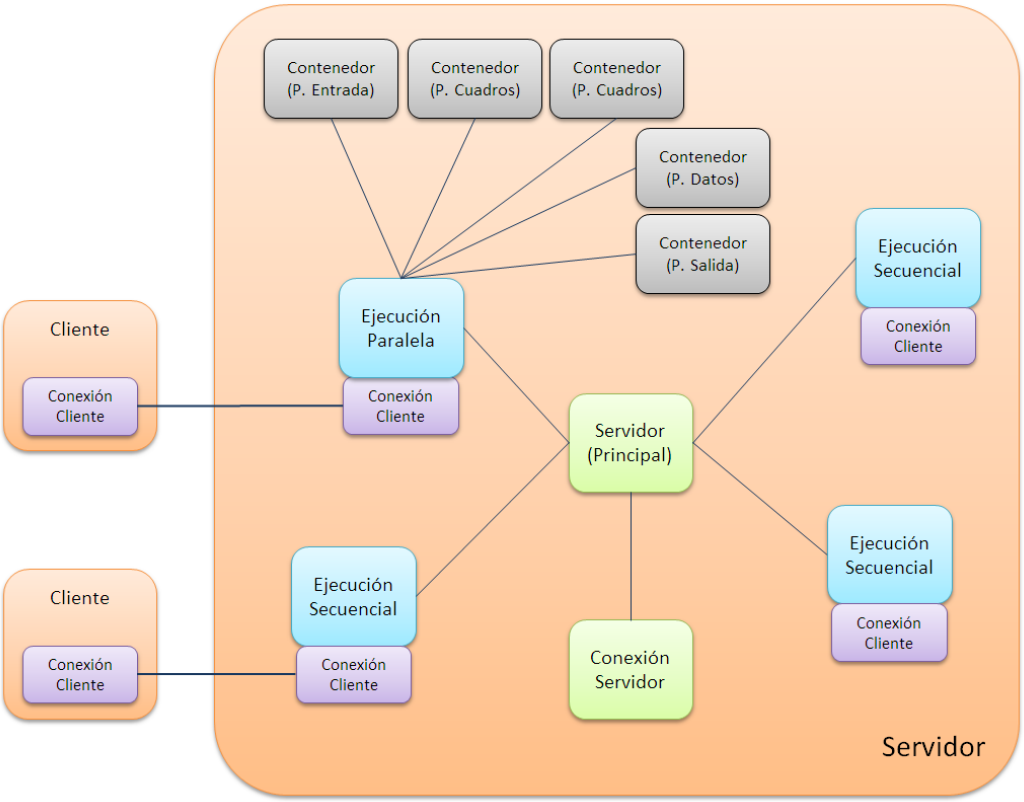

Figura 3.15: Diagrama de hilos para tres ejecuciones secuenciales y una paralela.

La Figura 3.16 muestra un diagrama de la interacción entre los contenedores y las colas de cuadros. Puede observarse un contenedor con un Procesador de Entrada que ejecuta primero leyendo los cuadros y colocándolos en la primera cola. Luego los Procesadores de Cuadros, ejecutando en paralelo, que toman estos cuadros y extraen los objetos de la escena, los cuales se asocian al cuadro y son enviados por la segunda cola. En la salida de esa cola se encuentra el procesador de Datos, que debe reordenar los cuadros y procesar los objetos detectados para determinar si corresponden a vehículos. También debe extraer sus datos y persistirlos si corresponde, y colocar los resultados en la última cola. Por último el procesador de salida se encarga de procesar el cuadro extraído imprimiendo la información requerida en el y emitir la transmisión o almacenarla en disco, según lo indique la configuración.

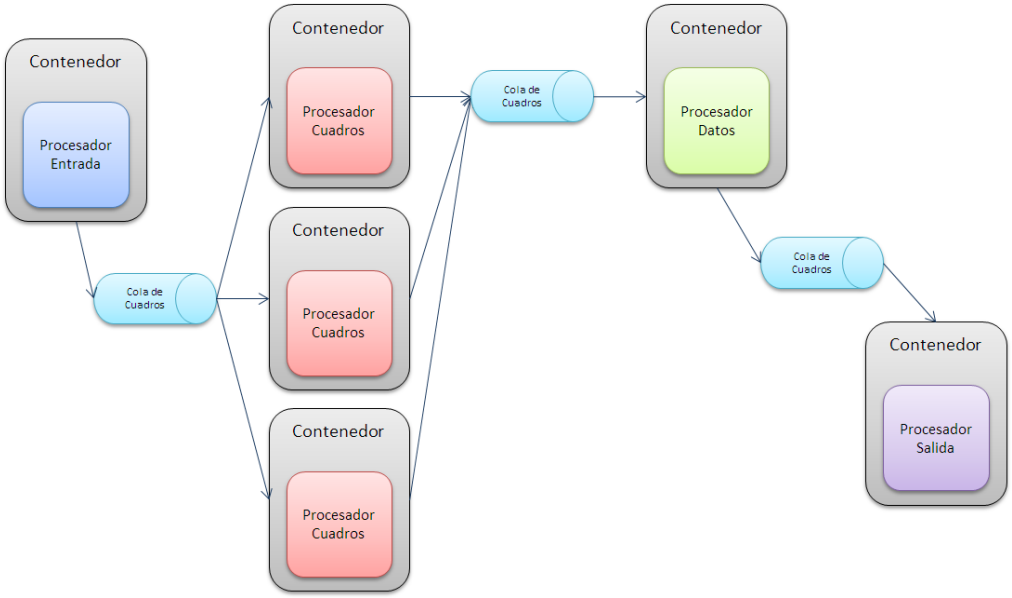

Figura 3.16: Diagrama de interacción entre contenedores y colas en una ejecución paralela.

Capítulo 4

Evaluación Experimental

4.1. Implantación en Ambiente de Pruebas

Para comenzar a realizar las pruebas es necesario adecuar un ambiente de trabajo con los requerimientos necesarios tanto para las bibliotecas que son utilizadas por el sistema como para las tecnologías con las que se construyó el mismo.

El único requerimiento solicitado por JavaCV para las funciones utilizadas en el proyecto es una implementación de Java SE 6 o una versión más reciente. La primera alternativa considerada para la implantación fue la infraestructura de Cluster Fing [20], que cuenta con un sistema operativo CentOS versión 6.5 para 64 bits, la versión de Java es 1.7 por lo que cumple con los requerimientos especificados por la biblioteca.

En una primera etapa se intentaron realizar las pruebas funcionales y de rendimiento sobre los equipos de Cluster Fing pero no se lograron resultados. La biblioteca retorna un error que alude a que ciertos elementos que utiliza la aplicación no son "visibles", por así decirlo. Se constató en todo momento que los elementos mencionados sí existían y la aplicación podía acceder directamente a los mismos identificando el problema como un posible error de referenciación a la hora de ensamblar los archivos binarios del sistema.

En foros de discusión sobre JavaCV [6] se mencionó que la versión más reciente de la biblioteca contemplaba éste y otros errores, que la fuente del error hace referencia a que el sistema operativo no es capaz de hallar las bibliotecas nativas que JavaCV necesita. Hecho el pasaje de versiones, se volvió a intentar pero nuevamente sin éxito. Otra alternativa detallada por el creador de JavaCV consiste en utilizar Maven [31] para realizar el manejo de la gestión de dependencias y la instalación de las mismas, opción que tampoco arrojo resultados positivos. También se mencionó actualizar varias bibliotecas nativas para evitar el problema, pero el usuario brindado para acceder a Cluster Fing no cuenta con privilegios para hacerlo. Otra solución

sugerida en otros foros de discusión, como Stackoverflow [30], explicaba que el problema se debe a la falta de una interfaz gráfica de usuario, las bibliotecas del entorno gráfico del sistema operativo. Esto también requiere de privilegios de administrador por lo que no se pudo probar.

También se investigaron opciones de contenedores de código para poder portar los ejecutables de la aplicación sin importar el ambiente en el que se use. Opciones como Docker [21] y RKT [29] fueron tenidas en cuenta. Básicamente estas herramientas encapsulan las aplicaciones en un contenedor y este puede ser transportado a cualquier equipo que cuente con un sistema operativo Linux. Para el caso de Docker es necesario contar con permisos de administrador por lo que tampoco fue posible utilizarla. Para RKT sucedió algo muy similar. Al momento de querer ejecutar la herramienta se advierte que la versión de la biblioteca glibc [25] es demasiado antigua y no es soportada por la misma. Esta biblioteca es una parte crítica de cualquier sistema operativo Linux y para actualizarla es necesario contar con privilegios de administrador.

Finalmente se tomó la decisión de dejar de tener en cuenta Cluster Fing y buscar servidores de alto rendimiento públicos donde fuera posible ejecutar la aplicación de forma satisfactoria. Algunas alternativas vistas fueron Linode [28], Amazon Web Services EC2 de Amazon [19] y Compute Engine de Google [26]. Ambas ofrecen virtualización de equipos en la nube pudiendo optar por varias configuraciones posibles variando tanto en cantidad y tipo de procesadores como en memoria, capacidad y tipo de discos duros. La siguiente tabla realiza una comparación con las prestaciones computacionales ofrecidas en lo que respecta a procesadores por los servicios y las compara con los utilizados por Clúster Fing para ver de forma explícita las diferencias y similitudes entre ellas.

Tanto Google Compute Engine como Amazon Web Services y Linode son infraestructuras pagas, cada una de ellas tiene un sistema de facturación distinto pero todas dependen del tiempo de uso que se emplee en determinado periodo de tiempo. Estas utilizan procesadores muy similares alcanzando una cantidad muy superior de núcleos a la brindada por Cluster Fing. En cuanto a los precios de cada una, la más económica es Google seguida por Amazon y Linode la cual realiza su facturación de forma mensual sin importar cuanto tiempo hayan sido utilizadas las instancias contratadas como lo hacen Google y Amazon. Finalmente la opción elegida fue Google debido a la alta disponibilidad de recursos, la posibilidad de configurar cada instancia a demanda, las herramientas proporcionadas y a los tiempos obtenidos en pruebas generando pequeñas diferencias entre cada proveedor. La Tabla 4.1 muestra una comparación entre las opciones contempladas.

Nube	#	Procesador	#Cores	Total
Cluster Fing	2	Intel Xeon E5 430 2.66GHz	8	16
	2	Intel Xeon E5 520 2.26GHz	4	8
	2	AMD Opteron 6172 2.10GHz	12	24
	4	AMD Opteron 6272 2.09GHz	16	64
	2	Intel Xeon E5 2650 2.00GHz	8	16
	1	Intel Xeon E5 530 2.40GHz	16	16
Google Compute Engine*	1-32	Intel Xeon E5 Intel Xeon E5 V2 Intel Xeon E5 V3	8	8-256
Amazon Web Services EC2	2-40	Intel Xeon E5-2676 v3	12	24-480
	1-8	Intel Xeon E5-2670 v2	10	1-80
	2-36	Intel Xeon E5-2666 v3	10	20-360
	2-32	Intel Xeon E5-2670	8	16-128
	2-32	Intel Xeon E5-2680 v2	10	20-320
Linode	1-20	Intel Xeon E5-2670 v2	10	10-200

Tabla 4.1: Comparación de recursos ofrecidos por los servicios de Cloud Computing. Cada instancia de (*) puede tener uno de los tipos de procesadores indicados, esta elección no es realizada por el usuario, el sistema automáticamente asigna los recursos.

4.2. Recursos de Video

Generalmente, al momento de evaluar un software, las entradas utilizadas en las pruebas deben abarcar todos los casos posibles para lograr cierto nivel de conformidad. Para este sistema en particular, es imposible abarcar las infinitas posibilidades de entrada que podrían utilizarse para realizar pruebas sobre el software. Por lo tanto, es necesario hacer que las pruebas sean lo más exhaustivas posible, y para ello es necesario contar con cierta cantidad de videos, con diversidad de escenas y situaciones. Por esta razón, y ante la dificultad de conseguir videos de tránsito por otros medios, es que se grabaron varios videos de tránsito en la ciudad de Montevideo y sus alrededores, aprovechando infraestructuras accesibles públicamente como puentes y edificios, para situar una cámara de video o un teléfono celular. La Tabla 4.2 contiene las características más relevantes de los videos utilizados para las evaluaciones. Todos los videos se encuentran en formato MP4. Todos reproducen 30 cuadros por segundo y son propios, a excepción de M6, que reproduce 25 cuadros por segundo y fue extraído de Youtube [22]. La Figura 4.1 muestra una captura de cada uno de los videos de la Tabla 4.2. Algunos de estos videos se encuentran disponibles en la página web del proyecto.

Nombre	Ubicación	Resolución	Duración	Altura	Ángulo	Vista
Italia 1	Puente de la calle Paul Harris sobre Av. Italia, Montevideo	848x480	00:08:12	8 metros	40º	Frontal
Italia 2	Puente de la calle Paul Harris sobre Av. Italia, Montevideo	848x480	00:11:24	8 metros	40º	Trasera
Giannattasio	Puente peatonal por Av. Giannattasio a metros de Buenos Aires, Lagomar, Canelones	1280x720	00:01:37	7 metros	35º	Ambas
Sarmiento 1	Puente de la Av. Sarmiento sobre Bv. Artigas, Montevideo	848x480	00:08:14	7 metros	35º	Frontal
Sarmiento 2	Puente de la Av. Sarmiento sobre Bv. Artigas, Montevideo	848x480	00:09:39	7 metros	35º	Trasera
Puerto	Edificio en Rambla Sud América esquina Colombia, Montevideo	1280x720	00:00:54	11vo piso, zoom x4	75º	Lateral
Canteras	Puente peatonal sobre Av. Dr. Juan Cachón, Canteras del Parque Rodó, Montevideo	1280x720	00:00:51	10 metros	60º	Frontal
M6	Autopista M6, Gran Bretaña	854x480	00:15:03	8 metros	35º	Ambas

Tabla 4.2: Características de los videos utilizados.

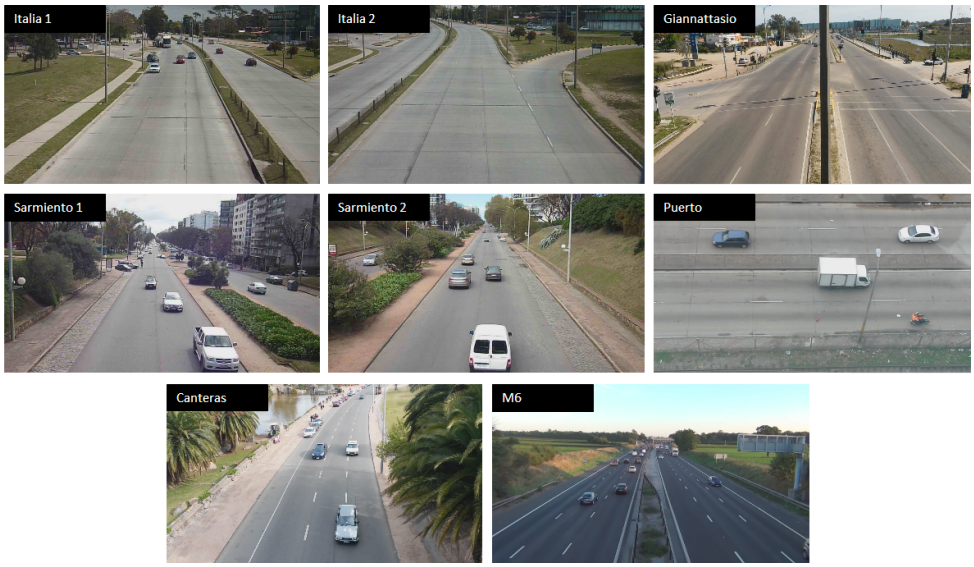

Figura 4.1: Captura de un cuadro de cada video.

4.3. Pruebas Realizadas

4.3.1. Pruebas Funcionales

Las pruebas funcionales son llevadas a cabo con la intención de evaluar las capacidades de detección y seguimiento de vehículos y la fiabilidad de los patrones de detección implementados. Para llevar a cabo esta evaluación, se realizan varias pruebas, cada una orientada a evaluar una de estas características y se aplicada sobre determinado subconjunto de los videos disponibles. El resto de esta sección se dedica a describir, mostrar los resultados y comentar sobre cada una de estas pruebas.

Pruebas de Conteo La primera prueba funcional tiene como cometido calificar la capacidad de detección que posee el sistema, comparando la cantidad de vehículos detectados con la cantidad real de vehículos que transitan por la vía en cada video. Debido a que el resto de las funcionalidades del sistema se basan en la detección de vehículos para funcionar, esta prueba es quizás la más importante de todas las de este tipo. En la Tabla 4.3 pueden apreciarse los resultados por video y el total.

Video	**Vehículos que Transitan**	**Vehículos Detectados**	**Aciertos**
Italia 1	220	193	87,73 %
Italia 2	285	272	95,44 %
Sarmiento 1	92	92	100,00 %
Sarmiento 2	103	98	95,15 %
M6	427	400	93,68 %
Total	1127	1055	93,61 %

Tabla 4.3: Resultado de las pruebas de conteo

Los resultados son prometedores. Se obtuvo un 93,61 % de aciertos en 1127 vehículos. Está claro que el error de 6,43 % en el conteo se debe a impresiciones en la detección. Estos errores de detección ocurren, generalmente, cuando el sustractor de fondo omite algunos píxeles de los objetos detectados en la imagen resultado. Luego, el detector de contornos no puede distinguir completamente entre las formas que no se encuentran cerradas, lo que impide que tome en cuenta esos vehículos.

Pruebas de Clasificación Siguiendo el mismo esquema de evaluación, la segunda prueba consiste en comparar la cantidad de vehículos clasificados en total y según clasificación, y compararlos contra la realidad que presenta el video. La Tabla 4.4 muestra el total de vehículos que transitaron y los que fueron clasificados para cada categoría, y para los totales por categorías, por video y general. La columna detectados, presente solo en los totales, indica

la cantidad de vehículos que fueron contados por el sistema, según la prueba anterior. Es importante tener en cuenta que el algoritmo de clasificación depende del algoritmo de detección, por lo tanto, si el algoritmo de detección falla, el algoritmo de clasificación no podrá tener en cuenta a los vehículos no detectados.

	Moto/Bicicleta		**Auto/Camioneta**		**Ómnibus/Camión**		**Totales**		
Video	**Transita**	**Clasifica**	**Transita**	**Clasifica**	**Transita**	**Clasifica**	**Transita**	**Detecta**	**Clasifica**
Italia 1	20	15	191	147	9	15	220	193	177
Italia 2	13	12	262	234	10	10	285	272	256
Sarmiento 1	6	5	80	76	6	5	92	92	86
Sarmiento 2	5	5	96	84	2	2	103	98	91
Total	44	37	629	541	27	32	700	655	610

Tabla 4.4: Resultados de clasificación de vehículos

Como es de suponer, la cantidad de vehículos clasificados es siempre menor o igual a la cantidad de vehículos de cada clasificación que transitaron, salvo por un caso. En el caso del video Italia 1, se clasificaron más camiones y ómnibus de los que realmente transitaron. Esto se debió a que en varias oportunidades, dos o más vehículos se superpusieron y el sistema no lo detectó, creyendo así que el objeto mal identificado tenía el tamaño acorde a un ómnibus o camión.

Partiendo de la tabla anterior, con todos los resultados, se pueden calcular el porcentaje de aciertos y el error en cada video, categoría y en los totales. Los aciertos se definen como el porcentaje de videos bien clasificados respecto al total de vehículos que transitaron y el error como el porcentaje de videos mal clasificados respecto a los vehículos que transitaron. Por último, los aciertos condicionados, se calculan solo para el total de las categorías, y se definen como el porcentaje de aciertos dada la cantidad de vehículos detectados. porque la clasificación depende de la detección.

La Tabla 4.5 resume toda esa información, en un formato similar al de la Tabla 4.4.

	Moto/Bicicleta		**Auto/Camioneta**		**Ómnibus/Camión**		**Totales**		
Video	**Aciertos**	**Error**	**Aciertos**	**Error**	**Aciertos**	**Error**	**Aciertos**	**Error**	**Aciertos Condicionados**
Italia 1	75,00 %	25,00 %	76,96 %	23,04 %	100,00 %	66,67 %	80,46 %	19,54 %	91,71 %
Italia 2	92,31 %	7,69 %	89,31 %	10,69 %	100,00 %	0,00 %	89,83 %	10,17 %	94,12 %
Sarmiento 1	83,33 %	16,67 %	95,00 %	5,00 %	83,33 %	16,67 %	93,48 %	6,52 %	93,48 %
Sarmiento 2	100,00 %	0,00 %	87,50 %	12,50 %	100,00 %	0,00 %	88,35 %	11,65 %	92,89 %
Total	84,10 %	15,90 %	86,01 %	13,99 %	96,23 %	3,77 %	86,29 %	13,71 %	92,21 %

Tabla 4.5: Porcentaje de aciertos para clasificación de vehículos

Los resultados obtenidos son peores que los de la primera prueba. Se obtuvo un 86,29 % aciertos en 700 vehículos. Pero, considerando solo los vehículos que fueron detectados, la tasa de aciertos sería de 92,21 % en 655 vehículos, por lo que el error neto del algoritmo de clasificación sería de 7,79 %, que de todas formas es mayor al error de la prueba anterior.

Se pueden apreciar dos tipos de error en esta prueba. El primero de ellos es la clasificación equivocada y se debe a la limitación en la implementación

de esta característica, ya que cada vehículo se evalúa según su largo, su alto y/o su área. Esto, en combinación con errores de detección, como sombras no excluidas del objeto o superposición de varios objetos o sombras, pueden hacer que los objetos parezcan más grandes de lo que en verdad son y confundan la detección. El otro error es la no clasificación de vehículos y suele darse porque los vehículos fueron detectados solamente fuera del área definida para la clasificación, parámetro que se define en la configuración, junto con las clasificaciones.

Pruebas de Medición de Velocidad Para esta característica se realizan dos tipos de pruebas. La primera se basa en evaluar la cantidad de vehículos a los que se les calcula la velocidad y sus resultados pueden apreciarse en la Tabla 4.6.

Video	Vehículos que Transitaron	Vehículos Detectados	Vehículos con Velocidad Medida	Aciertos	Aciertos Condicionados
Italia 1	220	193	188	85,46 %	97,41 %
Italia 2	285	272	267	93,68 %	98,16 %
Sarmiento 1	92	92	89	96,74 %	96,74 %
Sarmiento 2	103	98	95	92,23 %	96,94 %
Total	700	655	639	91,29 %	97,56 %

Tabla 4.6: Resultado de vehículos utilizados para la medición de velocidad

Al 91,29 % de los vehículos se les calcula la velocidad. Dejando de lado los vehículos no detectados por el sistema, el 97,56 % de los vehículos es medido. El error del 2,44 % se debe a que algunos de los vehículos no fueron detectados durante todo su tránsito por la zona del video en la que se mide la velocidad, parámetro que se especifica en la configuración.

La segunda prueba evalúa la veracidad de la velocidad calculada. Para ello, se utilizan videos iguales a Sarmiento 2, en los que circula un vehículo a una velocidad conocida a través de un tramo de longitud conocido en la vía. La prueba consiste en comprobar si el sistema puede detectar correctamente la velocidad a la que circula dicho vehículo. La Tabla 4.7 muestra, para cada velocidad medida, la velocidad real y el error en la medición.

Velocidad Real	Velocidad Detectada	Error
15km/h	15km/h	0,00 %
30km/h	30km/h	0,00 %
45km/h	44km/h	2,22 %
60km/h	58km/h	5,33 %

Tabla 4.7: Porcentaje de acierto en el cálculo de velocidad de los vehículos

Si bien el margen de error es reducido puede apreciarse que el error en la segunda prueba es creciente, a medida que aumenta la velocidad. Para comprender el origen del error es necesario conocer como funciona el cálculo de la velocidad media. En la figura 4.2 puede verse un diagrama, que ilustra

como el sistema calcula la velocidad media entre dos rectas perpendiculares a la dirección de las vías. Las posiciones tomadas son la previa y la posterior inmediatas fuera del área considerada. Es posible que un vehículo no sea detectado cerca de las rectas limítrofes, haciendo que la distancia asumida para realizar la medición sea mayor a la conocida. Esto ocurre porque la representación de video es discreta y la cantidad de cuadros por segundo es la limitante en la apreciación de las posiciones de los vehículos. De esta forma se pueden obtener velocidades menores a las reales, que si bien es incorrecto, brinda la garantía de que la velocidad detectada, sea o no correcta, nunca será mayor a la real, lo que permite generar alertas de exceso de velocidad fiables. La alerta de velocidad se dispara cuando la velocidad supera el máximo establecido en la configuración. No tiene sentido realizar pruebas sobre esto, ya que lo que determina si una alerta de velocidad es correcta, es la correctitud del cálculo de la velocidad.

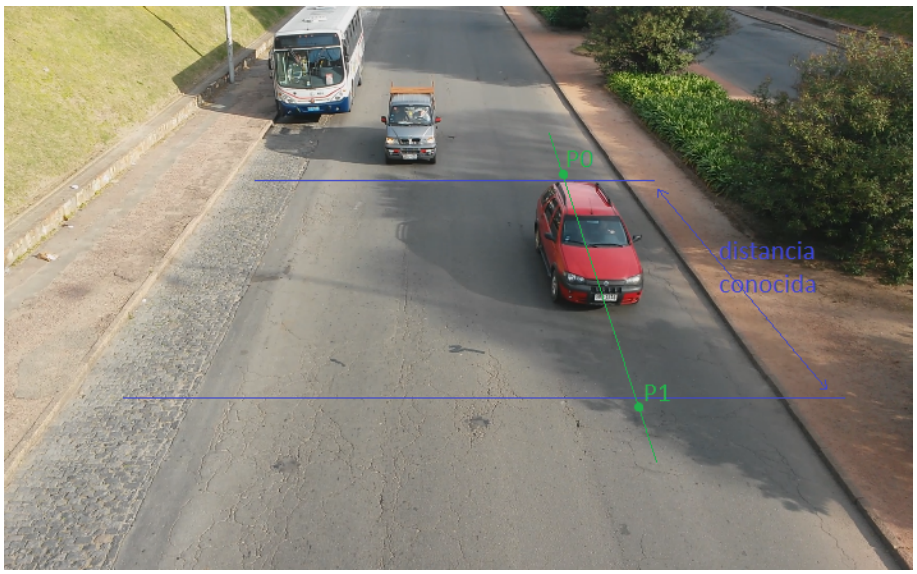

Figura 4.2: Elementos para el cálculo de la velocidad media.

Pruebas de Circulación en Sentido Contrario Probar esta característica de forma tradicional no resulta posible, dado que no se dispone de videos en los que un vehículo circule en sentido contrario por alguna vía y realizarlo con un vehículo propio implicaría cometer una infracción. Pero de todas formas es posible evaluar esta característica, definiendo en la configuración del sistema el sentido contrario para los carriles. La Tabla 4.8 muestra los resultados obtenidos para estas pruebas.

Video	Vehículos Transitados	Vehículos Detectados	Vehículos Detectados Sentido Contrario	Aciertos	Aciertos Condicionados
Italia 1	220	193	193	87,73 %	100,00 %
Italia 2	285	272	272	95,44 %	100,00 %
Sarmiento 1	92	92	92	100,00 %	100,00 %
Sarmiento 2	103	98	89	95,15 %	100,00 %
M6	427	400	400	93,68 %	100,00 %
Total	1127	1055	1055	93,61 %	100,00 %

Tabla 4.8: Vehículos circulando en sentido contrario

Dejando de lado los errores de detección, los resultados de estas pruebas son perfectos, según la tabla. El cálculo del sentido se hace en función de la dirección y el sentido de la trayectoria del vehículo. Además, un vehículo se considera si es detectado al menos dos veces (o más, según la configuración), lo que basta para calcular la dirección y el sentido de su desplazamiento. Esto quiere decir que si la configuración y la detección son correctas, entonces el sentido calculado también lo es, y por lo tanto las alertas de circulación en sentido contrario también lo serán.

Pruebas de Carril Restringido Al igual que el sentido de un carril, las restricciones de vehículos que pueden circular por un carril se definen en la configuración del sistema. Para realizar esta prueba es necesario definir carriles restringidos para algún tipo de vehículo definido en las clasificaciones de vehículos. Esta prueba compara la cantidad de vehículos que se detectan circulando por un carril que no les corresponde, contra la cantidad real de vehículos que lo hacen. Las pruebas se realizaron sobre los videos Italia 1 e Italia 2, definiendo para ambos, el carril derecho para vehículos grandes exclusivamente, y los carriles medio e izquierdo para el resto de los vehículos. La Tabla 4.9, muestra en la primera columna la cantidad real de vehículos que transitaron por un carril no permitido. En la segunda muestra de esos vehículos, los vehículos que fueron detectados y clasificados correctamente. La tercera columna, muestra la cantidad de vehículos que fueron detectados circulando en un carril no permitido según su clasificación.

Video	Vehículos Transitados	Vehículos Clasificados	Vehículos en Carril Restringido	Aciertos	Acier. Cond. Condicionados
Italia 1	50	46	46	92,00 %	100,00 %
Italia 2	20	19	19	95,00 %	100,00 %
Total	70	65	65	92,86 %	100,00 %

Tabla 4.9: Vehículos circulando por carriles restringidos

Como muestra el tabla 4.9, los resultados de esta prueba también son perfectos, siempre que los vehículos sean detectados y clasificados correctamente. El caso es similar al de la prueba de circulación en sentido contrario, salvo porque esta funcionalidad depende además de la clasificación de los vehículos.

Pruebas de Maniobras Prohibidas Como en las últimas dos funcionalidades evaluadas, las maniobras prohibidas se definen en la configuración. Para llevar a cabo esta prueba se define un conjunto de maniobras prohibidas específicas para cada video. Luego se evalúa la cantidad de vehículos correctamente detectados en cada uno de ellos. Los videos utilizados en esta prueba son Italia 2 y Giannattasio. En el primero de ellos, se definen dos maniobras prohibidas, la entrada y salida del carril derecho. En el segundo, se define la maniobra prohibida como el giro a la izquierda en el semáforo. En la primer columna de la Tabla 4.10 se indica la cantidad real de vehículos que realizan la maniobra prohibida. En la segunda, se indica, de esos vehículos, cuales han sido detectados y en la tercera, los que fueron detectados realizando una maniobra prohibida.

Video	**Vehículos Totales**	**Vehículos Detectados**	**Vehículos con Maniobra Prohibida Detectada**	**Aciertos**	**Aciertos Condicionados**
Italia 2	14	13	13	92,86 %	100,00 %
Giannattasio	3	3	3	100,00 %	100,00 %
Total	17	16	16	94,12 %	100,00 %

Tabla 4.10: Vehículos circulando por carriles restringidos

Este patrón también obtuvo un resultado perfecto, condicionado a la correcta detección del vehículo.

Prueba con Vista Lateral Esta prueba se toma por separado debido a que las funcionalidades son acotadas para este tipo de vista. Las funcionalidades que se evalúan son conteo, clasificación, medición de velocidad, sentido contrario y carril restringido. El video utilizado es Puerto. Las filas de la Tabla 4.11 corresponden a la evaluación de cada característica.

Característica	**Vehículos Totales**	**Vehículos Detectados**	**Aciertos**	**Aciertos Condicionados**
Conteo	44	40	90,91 %	-
Clasificación	44	38	86,37 %	95,00 %
Velocidad	44	39	88,64 %	97,50 %
Sentido Contr.	44	40	90,91 %	100,00 %
Carril Restr.	44	38	86,37 %	100,00 %

Tabla 4.11: Resultados de pruebas de vista lateral

Dejando de lado la inferior cantidad de vehículos evaluados, los resultados de esta prueba son muy similares a los obtenidos en el resto de las pruebas, que se realizaron con videos con vista frontal de las vías.

Prueba con Escenas Nocturnas Esta prueba utiliza escenas tomadas durante la noche. En estas escenas, las vías se encuentran iluminadas por la red de alumbrado público, factor crucial para que la cámara utilizada pueda

capturar a los vehículos que circulan. Para esta prueba, fueron tomados dos videos con las mismas características que Sarmiento 1 y 2, pero con duración menor, de 4 minutos aproximadamente. La Figura 4.3 muestra capturas de ambos videos y la Tabla 4.12 muestra los resultados de las pruebas.

Figura 4.3: Capturas de los videos Sarmiento 3 y 4 respectivamente.

Video	**Vehículos Totales**	**Vehículos Detectados**	**Aciertos**
Sarmiento 3	3	30	10,00 %
Sarmiento 4	33	44	75,00 %

Tabla 4.12: Resultado de pruebas sobre escenas nocturnas

No hay dudas de que esta es la prueba que arrojó peores resultados. En el video Sarmiento 3, se observaron resultados pésimos, con un 10 % de aciertos. El problema con este video, es que las luces de los vehículos se proyectan en el suelo, lo que hace que el detector asocie estas zonas iluminadas como parte de los vehículos o si la luz encandila a la cámara, que el algoritmo solo detecte los focos de luz y no todo el vehículo. Para Sarmiento 4, los resultados son mucho mejores, con un 75 % de aciertos, tasa que de todas formas, se encuentra más de un 15 % por debajo de las obtenidas para el resto de los videos. La clasificación de vehículos y la medición de velocidad no fueron tomadas en cuenta para esta prueba porque el algoritmo confunde las luces con los vehículos según la posición de estos, lo que no permite una detección uniforme de los vehículos y por lo tanto distorsiona estas características.

Prueba con Semáforos La última de las pruebas funcionales busca evaluar el patrón de detección basado en semáforos. En la configuración se especifican 3 patrones, uno por cada luz del semáforo, a saber, roja, amarilla y verde. Por lo tanto, todos los vehículos que circulen por la zona de intersección de las calles, que es regulada por el semáforo, deberían dar origen a una alerta. Cada alerta corresponderá con la luz que estaba activa en el momento en que el vehículo atraviesa la zona configurada. El video utilizado es Giannattasio. Cada fila de la Tabla 4.13, corresponde con una de las alertas configuradas.

Alerta	**Vehículos Totales**	**Vehículos Detectados**	**Vehículos con Luz**	**Aciertos**	**Aciertos Condicionados**
Luz Verde	24	20	20	83,33 %	100,00 %
Luz Amarilla	3	2	2	66,67 %	100,00 %
Luz Roja	0	0	0	100,00 %	100,00 %

Tabla 4.13: Evaluación de patrón Semáforos

Si bien, la cantidad de vehículos es muy reducida en esta prueba, los resultados observados son perfectos, dejando de lado el error en la detección de los vehículos. Este patrón se basa en la detección de la posición y del color de la luz del semáforo en el momento en que el vehículo atraviesa el área configurada.

Finalizando esta primera etapa de pruebas, es importante destacar los tres puntos débiles encontrados hasta el momento en la implementación. El primero de ellos, es la causa del error de 6,39 % en la prueba de conteo de vehículos. Este defecto del algoritmo de detección y seguimiento, repercute negativamente en la detección del resto de las características de los vehículos y podría considerarse como el mayor de todos.

El siguiente defecto se encuentra en el algoritmo de clasificación de vehículos, con un error de 8,79 % para la característica individual y 13,71 % en acumulación con el primer defecto mencionado. Si bien el margen de error es mayor para este defecto, son menos funcionalidades las que se ven afectadas por el. Para mejorar este defecto sería necesario que el algoritmo pueda ser más preciso a la hora de delimitar los objetos de la escena y pueda detectar aún mejor las sombras proyectadas por estos. También se podrían utilizar técnicas basadas en redes neuronales para identificar los contornos hallados y sumar otro criterio de clasificación para hacer más fiable el algoritmo. Si bien esta temática fue estudiada y mencionada en el informe, no fue posible su adopción por falta de tiempo.

El último de los defectos, se encuentra también en el algoritmo de detección, y se da solo en los videos en los que la tenue iluminación de la noche, en conjunción con las proyecciones de las luces de los vehículos sobre el suelo, engañan al sistema. Estos escenarios, acotan las capacidades de detección alrededor de un 20 % cuando los vehículos son capturados por detrás, y vuelven inutilizable al sistema cuando son capturados frontalmente. Resulta necesario mejorar estos aspectos a futuro, si se pretende construir un sistema capaz de funcionar con videos tomados durante la noche. En este proyecto, por falta de tiempo, no es posible abordar estos problemas.

4.3.2. Pruebas de Rendimiento

La evaluación de rendimiento pretende medir las velocidades de procesamiento y respuesta que tiene el sistema en un ambiente de alto rendimiento en el que pueda ser implantado para su uso en producción. Particularmente, interesa determinar la cantidad de videos que pueden ser procesados simultáneamente, sin que se vean afectadas la transmisión en tiempo real y las velocidades de almacenamiento y procesamiento de datos. Otro factor a evaluar es la variación de los tiempos de respuesta para videos de diferentes resoluciones. Por último, interesa comparar los rendimientos entre los dos tipos de ejecuciones posibles, secuencial y paralela. Para llevar a cabo estas pruebas, es necesario ensamblar varios conjuntos de videos de prueba, que deben ser ejecutados concurrentemente desde varias instancias remotas de la aplicación cliente.

Para estas pruebas se utilizaron los servicios brindados por Google Compute Engine. Se creó y configuró una máquina virtual que cuenta con 8 procesadores Intel Xeon E5 V2, de 8 núcleos cada uno, con 8GB de memoria RAM y 20GB de disco duro. Para automatizar las ejecuciones de las diferentes instancias de los clientes, se implementó un programa especifico para la simulación de envío y recepción de mensajes con el servidor. De esta forma la inicialización de los clientes y la ejecución de los videos se puede iniciar de forma muy simple y rápida. El mecanismo consiste en generar cierta cantidad de usuarios a una tasa constante y ejecutar video determinado en el servidor. Estos parámetros son ingresados por el usuario al momento de querer ejecutar cualquier prueba donde la cantidad de usuarios es un entero, la tasa de nacimiento de los mismos se define por medio de un entero que indica la cantidad de segundos que habrá entre cada nuevo nacimiento y un identificador para el video que se quiere utilizar. Para comparar los resultados obtenidos entre las ejecuciones paralelas y secuenciales, se compara la diferencia entre ejecuciones de un mismo video realizando variaciones sobre la resolución de los mismos y la carga de usuarios concurrentes se definieron como videos de prueba Italia 1 y Giannattasio 1.

A continuación se realizará un análisis sobre los resultados obtenidos para cada una de las pruebas mencionadas anteriormente. Estos datos se encuentran en la sección de anexos y para las representaciones gráficas se utilizaron valores promedio de las mediciones obtenidas.

Ejecuciones Secuenciales y Paralelas En la Figura 4.4 se observa el tiempo de respuesta del servidor que insume la ejecución de un video durante una ejecución secuencial. Se ve claramente como al aumentar la cantidad de usuarios concurrentes los tiempos comienzan a crecer hasta llegar al punto en que la duración del video sometido a la ejecución es menor al consumido por el sistema. Este caso ocurre con 6 usuarios simultáneos donde se alcanza un tiempo de casi 540 segundos, mientras que el video tiene una duración

total de 492 segundos.

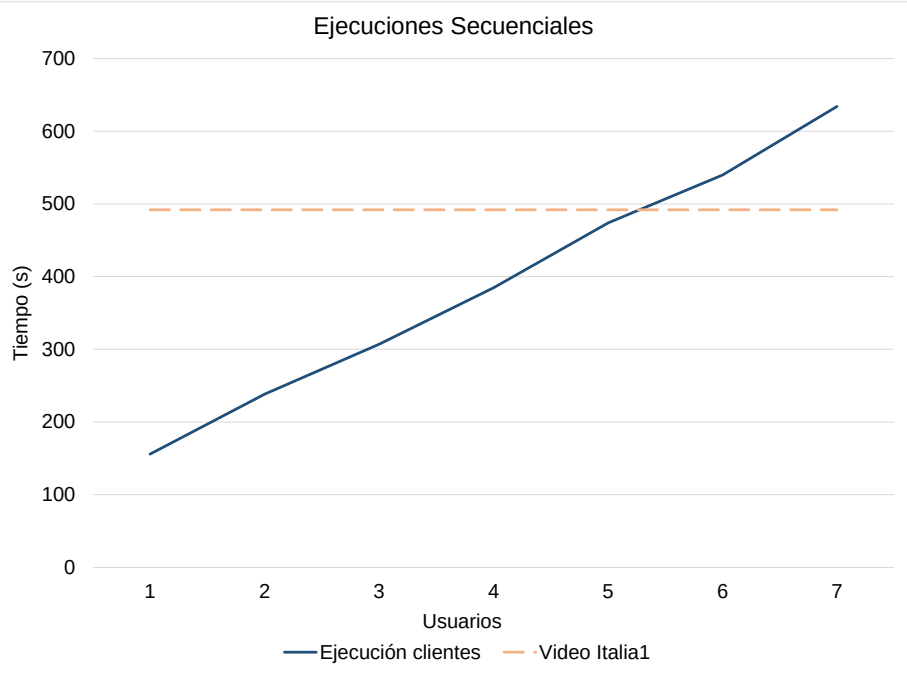

Figura 4.4: Resultados de ejecuciones secuenciales sobre video Italia 1.

Usuarios	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Tiempo (s)	156,0	238,4	307,1	384,8	474,0	539,7	634,1

Tabla 4.14: Tiempos promedio de ejecuciones secuenciales

En la Figura 4.5 se puede ver como también los tiempos aumentan al incrementarse la cantidad de usuarios concurrentes pero existe una diferencia clara con el caso anterior donde el sistema tolera unicamente 5 usuarios para mantener la condición de procesamiento y visualización en tiempo real. En este caso la cantidad de usuarios soportados para ejecutar concurrentemente es 10. Si bien en ese caso el valor obtenido es superior a la duración del video sn unos pocos segundos (cercano al 1 % del tiempo total) que en el correr de toda la ejecución terminan siendo imperceptibles para el usuario.

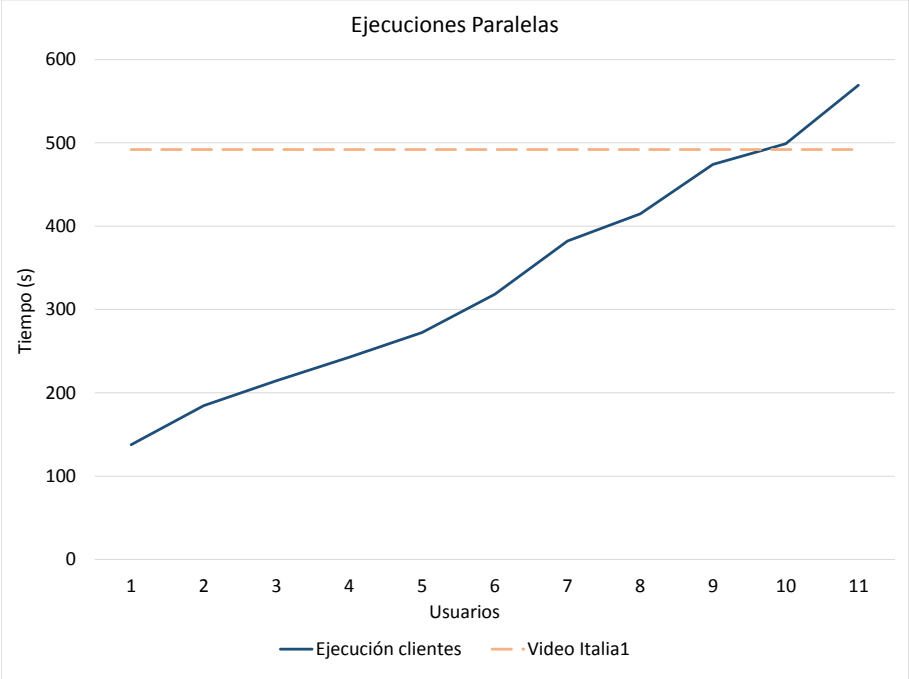

Figura 4.5: Resultados de ejecuciones paralelas sobre video Italia 1.

Usuarios	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Tiempo (s)	137,8	184,5	214,4	242,6	272,1	318,1	381,9	414,9	473,9	498,8	568,9

Tabla 4.15: Tiempos promedio de ejecuciones paralelas

Esta comparación se hace mas notoria incluso cuando se representan gráficamente ambos resultados. En la Figura 4.6 se expone este contraste.

En ambos casos se observa un crecimiento lineal sobre la variación en la cantidad de usuarios concurrentes del sistema pero también queda clara la diferencia que existe entre una ejecución y otra. La ejecución paralela es notoriamente superior a la hora de procesar videos de forma concurrente ya que utiliza de forma mas eficiente los recursos con los que se dispone para realizar el cómputo en cada ejecución a diferencia de la modalidad secuencial. La Tabla 4.16 muestra el cálculo de los diferentes speedups según la cantidad de usuarios. Este cálculo se realizó utilizando la ecuación (4.1) donde el tiempo generado por cada ejecución paralela es representado por TNew y el tiempo generado por las ejecuciones secuenciales por TOld.

$$speedUp = \frac{TOld}{TNew} \tag{4.1}$$

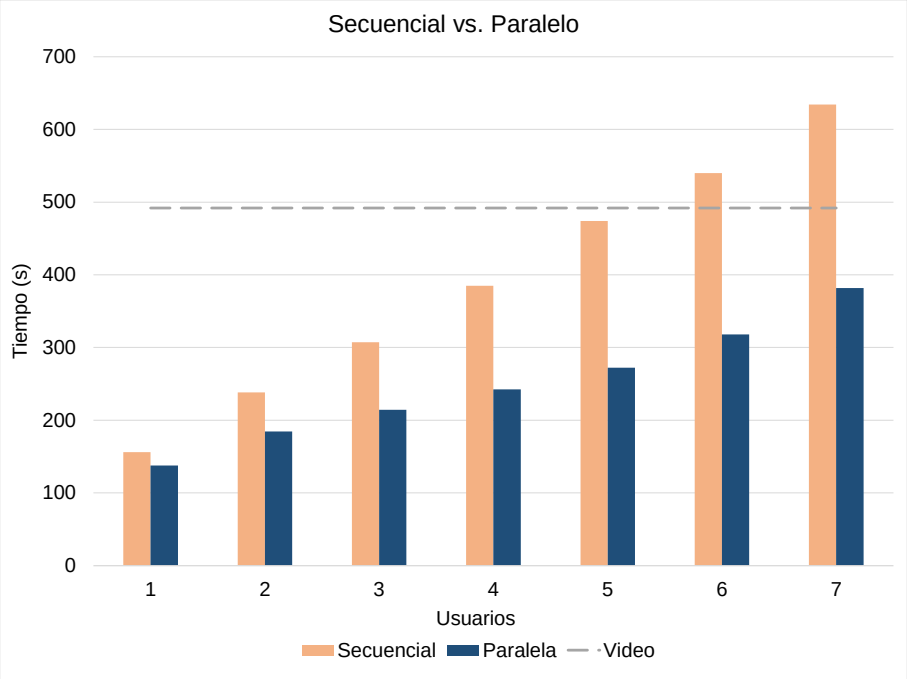

Figura 4.6: Comparación entre ejecuciones secuenciales y paralelas sobre el video Italia 1.

El mejor speedup registrado fue de 1.74, para 5 usuarios concurrentes, que representa una mejora del 74 % en la velocidad del algoritmo paralelo sobre el algoritmo secuencial.

Usuarios	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Speedup	1,13	1,29	1,43	1,59	1.74	1,70	1,66

Tabla 4.16: Cálculo de speedup entre ejecuciones secuenciales y paralelas

Variación de Resolución en Videos Para evaluar esta característica se utilizó como base el video Giannattasio 1 que tiene una resolución de 720p y una duración aproximada de 59 segundos. Se generaron y compararon los resultados obtenidos con el mismo video pero realizando una variación en la resolución dejándola en 480p y 360p. La decisión de utilizar este video se debe a la calidad inicial y a su corta duración. Si bien la duración puede llegar a jugar un rol importante en estas pruebas, en caso de que sea demasiado breve, la tasa de nacimiento de los clientes es más veloz que la duración del video, de este modo se puede contar con la carga completa durante gran parte de la ejecución. La cantidad de cuadros por segundo se mantuvo igual así como también la cantidad de cuadros total del video original. Se utilizó la modalidad de ejecución paralela que fue destacada en la sección anterior.

La Tabla 4.17 contiene una versión resumida de los datos obtenidos en las pruebas al variar la resolución del video. Se puede ver numéricamente la gran diferencia que existe entre los diferentes escenarios. Luego, se presenta en la Figura 4.7, una comparación de los resultados obtenidos para cada una de las resoluciones al variar la carga de usuarios concurrentes.

Usuarios	**360p**	**480p**	**720p**
1	3,91	7,80	30,80
2	4,80	8,40	34,06
4	6,60	11,65	40,86
8	10,90	18,97	58,41
16	21,51	41,46	
24	37,55	56,59	
32	49,73		
36	56,65		

Tabla 4.17: Resultados promedio en segundos para ejecuciones paralelas al variar la resolución

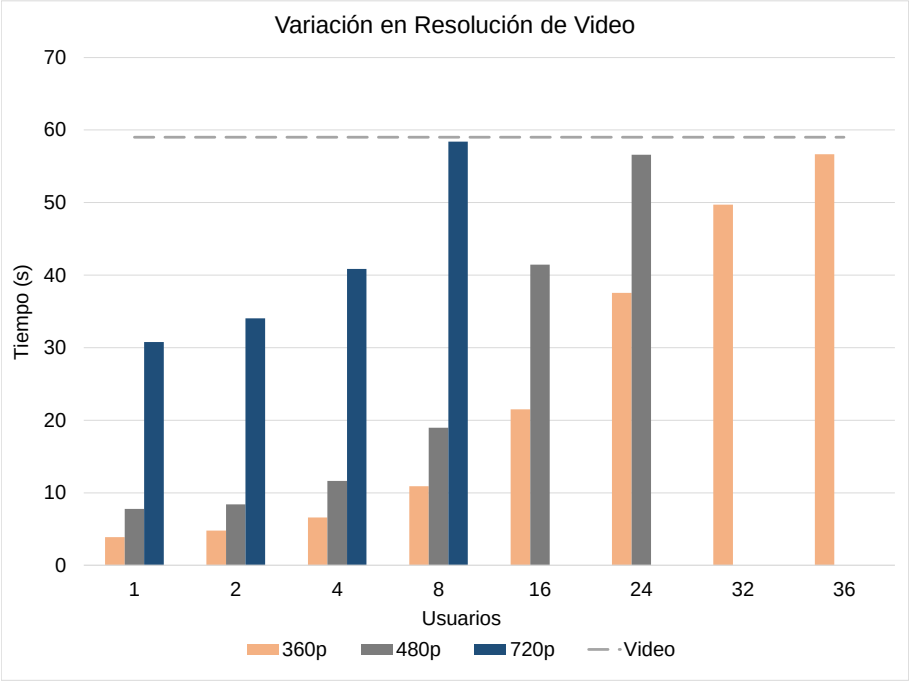

Figura 4.7: Comparación de resultados al variar la cantidad de usuarios y la resolución del video Giannatassio 1.

El cambio de resolución genera una gran diferencia en cuanto a la cantidad soportada para lograr que toda la carga sea procesada en tiempo real.

La cantidad máxima de clientes simultáneos que se logró alcanzar fueron 36 en el caso de la resolución 360p mientras que para el video de 480p y 720p fue de 24 y 8 respectivamente. Esto remarca el efecto que puede llegar a tener esta característica del video sobre el rendimiento y capacidad de computo del sistema. Se puede observar la gran diferencia al comenzar la prueba con un único usuario utilizando el sistema, los tiempos generados son demasiados distantes mientras más alta sea la resolución.

Capítulo 5

Conclusiones y Trabajo Futuro

En esta sección se explicitarán las conclusiones generadas a lo largo de todo el trabajo realizado, analizarán los aspectos mas relevantes del sistema y se detallarán varios puntos que han sido dejados de lado, aspectos que serán tenidos en cuenta como trabajo futuro de la aplicación, así como también aquellos que fueron surgiendo durante todo el proceso de diseño, análisis y construcción del sistema.

5.1. Conclusiones

La arquitectura distribuida utilizada, permite atender a varios usuarios simultáneamente, desde una misma instancia servidor, que a través de la difusión en flujo permite mostrar los resultados obtenidos en tiempo real. Característica que resulta fundamental para cualquier software de monitoreo, al momento de asistir al usuario en su tarea. El sistema construido es sumamente flexible al soportar una gran diversidad de infraestructuras viales, debido al amplio soporte que ofrece su configuración. Es capaz de detectar, clasificar y hacer el seguimiento de vehículos en los videos de tránsito, y también puede detectar varios comportamientos inapropiados en el tránsito, a través de los patrones de detección implementados.

Los resultados de la evaluación funcional han arrojado que el sistema es confiable, con una tasa de aciertos del 93,61 % para la detección y el seguimiento de los vehículos, y con una de 92,21 % para la clasificación de los mismos, que representan los dos mayores puntos de fallos del sistema. Podría decirse que el resto de las funcionalidades son casi infalibles, dejando de lado su dependencia de estas dos. Las pruebas también arrojaron que el sistema tolera la falta de iluminación hasta cierto punto. Solo en los videos en los que el alumbrado público ilumina en cierta medida las vías y el tránsito se observa desde atrás, es posible realizar las detecciones con un 75 % de

confiabilidad, un 18 % menos que en las escenas tomadas durante el día.

En cuanto a la carga tolerada por el sistema, se obtuvieron valores más que satisfactorios. Fue posible reproducir hasta 36 videos de resolución 360p de forma simultánea y en tiempo real. Para videos de calidad 480p y 720p, fue posible hacerlo para 24 y 8 usuarios, respectivamente. Queda muy clara la relación de proporcionalidad inversa que existe entre la cantidad de usuarios soportados por la aplicación y la resolución de los videos utilizados en las ejecuciones. Esta relación tiene sentido, ya que al aumentar la resolución, la cantidad de píxeles de cada cuadro del video es significativamente mayor, demandando al servidor una mayor capacidad de computo para procesar la ejecución. Por otro lado, la gran diferencia que existe entre una ejecución secuencial y una paralela también quedó remarcada en los resultados gracias a las pruebas realizadas. Se obtuvo una mejora máxima del 74 % en el rendimiento de la aplicación paralela respecto a la versión secuencial. Esta mejora resulta vital para poder utilizar el sistema en un ambiente donde la cantidad de ejecuciones simultáneas es un aspecto importante.

Concluyendo este trabajo, es posible afirmar que se han combinado satisfactoriamente los conceptos estudiados de visión por computadora junto con el paradigma de programación multihilo, para crear un sistema de software distribuido que implementa las funcionalidades esperadas, y que además, permite almacenar y reproducir los resultados obtenidos de una forma cómoda para el usuario.

5.2. Trabajo Futuro

Durante el desarrollo de este proyecto, surgieron diferentes ideas o aspectos, que podrían haber sido incorporados al sistema final. Muchos no lo fueron por diferentes razones, como la definición del alcance o el objetivo principal del proyecto, y fueron registrados como trabajos a ser realizados en el futuro.

Para que este proyecto pueda convertirse en un sistema apto para realizar la vigilancia permanente del tránsito, sería necesario efectuar algunas correcciones y mejoras. Para empezar, es crítico mejorar el algoritmo de detección en cuanto a la iluminación, para que este soporte totalmente las escenas tomadas en la noche. También debe modificarse el algoritmo para reducir los errores de detección y clasificación de vehículos, que son los dos principales puntos de fallo del sistema.

Se pueden agregar, muy fácilmente, nuevos patrones de detección que no fueron tenidos en cuenta en este proyecto. Algunos ejemplos interesante y muy comunes son el control de señales de tránsito como pare, ceda el paso, cruces peatonales con preferencia, etc.

Otro aspecto que fue muy discutido para ser incorporado en esta versión fue la detección de matrículas de los vehículos detectados. Si bien es una

funcionalidad que podría aportar información muy valiosa a la hora de la generación de alertas también es muy costoso en cuestiones de cómputo y terminaría condicionando aún mas las características de la filmación, ya sea por su ubicación ó su resolución de imagen.

Otra mejora fundamental, sería el pasaje de la configuración o metadata del sistema a la base de datos. La configuración por archivo resulta muy poco práctica cuando se manejan muchas escenas, por lo que resultaría necesario contar con una interfaz de usuario para configurar las ejecuciones. Incluso sería muy útil contar con herramientas gráficas de dibujo para facilitar al usuario la delimitación de carriles o la región de interés sobre la cual realizar la detección, por ejemplo.

Por otro lado, realizar streaming directamente desde las cámaras ubicadas en la vía pública también sería un requisito necesario, esta funcionalidad no ha sido tomada en cuenta para el sistema, pero no sería muy difícil incorporarla debido a que la biblioteca FFmpeg ya ofrece la posibilidad.

La siguiente mejora, que puede afectar un poco el rendimiento al agregar algo de sobrecarga, sería incluir el sistema o una versión ligeramente modificada, dentro de un servidor de aplicaciones, para darle mayor robustez y hacerlo más escalable. Los servidores de aplicaciones ofrecen entornos de ejecución confiables y recuperación de errores, aspectos que no fueron tomados en cuenta del todo en este proyecto. Luego, para aumentar el volumen de ejecuciones concurrentes, se podría utilizar un balanceador de carga externo, en conjunción con varias instancias del sistema. Para el usuario sería totalmente transparente, y a nivel de los servidores, la disponibilidad de los recursos podría llegar a ser mejor administrada.

Sería necesario añadir la gestión de usuarios y roles, con un flujo de trabajo que permita, entre otras cosas, alertar a los usuarios al detectar infracciones, revisar las infracciones detectadas por el sistema, programar ejecuciones. Luego, para visualizar toda la información y generar reportes de valor, de la forma más cómoda posible, se necesitaría de una aplicación web. De esta forma se podrían monitorear eficientemente varios puntos de una ciudad, por ejemplo, extrayendo datos en tiempo real del volumen de tránsito o realizando estadísticas sobre las infracciones cometidas o el tipo de vehículos que transitan.

En un ambiente de producción, la generación de datos registrados por el sistema comenzaría a crecer velozmente, debido al gran volumen de vehículos que circulan en la mayoría de las calles de una ciudad. Finalmente, sería necesario entonces, utilizar técnicas de Big Data para el almacenamiento de la información o para poder analizarlos de forma rápida y confiable. Luego, teniendo ya esta cantidad considerable de datos, los mismos se podrían explotar con técnicas de Data Mining, aportando gran valor a futuros proyectos relacionados con el tránsito.

Apéndice A

Resultados de Pruebas de Carga

A.1. Italia 1 - Ejecuciones secuenciales

	1	2	3	4	5	6	7	8	9	10
1	147285	164906	151466	156981	150752	156153	151398	152617	156564	155612
2	238139	235514	235000	236952	238159	240832	239330	239875	240550	239621
	235360	240612	235854	236707	237094	235927	240846	240057	239941	240584
3	306855	308158	306742	307937	305281	306184	304414	309963	310003	306659
	307197	308981	304435	306232	306735	309760	304700	308881	305714	309236
	307198	307389	307937	305728	304511	307490	307174	308110	308406	308404
4	384040	384504	384532	383983	384815	384296	385206	385651	385080	383930
	385509	383852	385052	385588	384030	385461	384300	384297	384986	384592
	385049	385256	384135	385130	385687	384411	385417	384430	384929	385282
	383923	384738	385445	385113	385245	385371	383884	384823	385818	384316
5	474021	474895	473750	473300	474957	473558	474962	474950	474590	474668
	473567	474360	473716	475096	474148	474033	473766	473468	473439	474735
	473165	474376	473990	473816	474872	473208	473246	473285	474148	473552
	473411	474612	473671	474788	474511	474119	473891	473285	473362	473253
	474662	473540	473962	474516	473382	474170	473557	473312	474489	473964
6	542896	542284	537995	535882	537032	538327	535879	538294	536994	539105
	542644	540635	542061	538816	539065	540191	536901	542818	542938	538821
	541645	539363	538532	542595	540904	541013	540333	540196	537024	537594
	538325	538823	538174	541819	541229	538469	540378	539557	538686	540389
	542572	541621	539779	535985	540847	539346	536678	536533	542068	539375
	536540	540976	539994	537138	542592	537169	541352	540628	537502	540222
7	631457	632116	633036	633723	636299	636410	635897	630398	631577	637456
	634985	637879	631042	630851	631201	634717	635643	636248	630761	631563
	632710	635877	634297	635601	631438	631059	635820	634741	633681	638159
	630686	635451	633923	637167	638257	630780	631621	636695	631889	630676
	636820	634220	632454	629842	631557	633841	630036	632244	630628	629975
	632807	636747	631058	634683	629810	635099	635035	635307	632734	635895
	636719	634815	638219	633708	633161	634759	630152	636034	636788	636544

Tabla A.1: Tiempos en milisegundos generados por 10 ejecuciones secuenciales utilizando entre 1 y 7 clientes concurrentes

A.2. Italia 1 - Ejecuciones paralelas

	1	2	3	4	5	6	7	8	9	10
1	148708	144831	148111	149173	146182	145570	142710	143617	150014	146155
2	183302	188476	188832	189228	189016	180153	183056	180894	184422	186776
	179945	182952	184313	187332	184582	186560	185602	185744	187458	184147
3	211839	214572	213837	216006	217826	215308	215089	217812	216922	216429
	213177	211324	216524	214657	210480	211190	214737	211258	214813	213440
	216386	214507	214451	217543	214256	211492	217609	212566	215793	210508
4	240927	240126	239015	240033	244380	241304	239140	242286	242081	240345
	243610	241809	243374	239235	243298	240965	239408	246443	246223	245717
	239955	239550	245860	246177	240366	242380	240555	244165	246283	241618
	244558	240321	246399	241751	242250	243301	242846	242587	244666	242336
5	270259	267867	274110	271948	271873	269368	268986	273202	269565	275696
	272215	268639	274273	267629	269843	273203	273685	275296	272740	271139
	268944	269671	276908	273713	275528	270290	270688	269996	277103	271964
	274527	269478	269683	269103	277561	276223	271411	272120	274415	272424
	269963	275190	270692	273559	267651	267914	268275	269122	272266	269608
6	317262	314526	322456	322484	317884	318841	317653	318006	321439	319609
	320850	316390	319986	314798	316352	316975	314635	316340	315168	322478
	315418	314110	315194	320330	315037	318512	320597	322508	317573	321735
	313885	321698	321476	322087	317976	318123	313502	316128	314008	317315
	315252	313467	321373	320231	315091	313904	318251	321560	319360	320660
	314380	317129	319795	322968	319110	317582	320092	319081	314145	316516
7	384189	382927	379094	379635	381446	380690	383124	383591	383782	379458
	378973	380199	384434	382204	381382	380706	383240	382219	379298	381091
	383962	383861	382911	380069	379539	381844	380623	383325	382856	383163
	379665	381102	378902	380503	382924	384361	378809	382250	384661	383512
	384061	384021	379364	382854	382130	381488	384690	378882	381018	378857
	380982	381465	383523	381758	383029	383921	384156	383964	382157	382207
	380701	379035	382524	384749	380278	381241	384585	382713	383904	379198
8	414178	414320	414125	415865	416316	414944	415283	413989	415042	413464
	415548	414537	415993	415810	414996	413909	413799	414961	415651	414398
	416037	416164	415438	415991	413516	415922	415588	414079	414616	415433
	413953	416090	416241	415065	414007	415625	413374	413832	416305	414341
	414922	414845	414220	414240	414031	413328	416101	413602	416139	415113
	415193	413698	415715	414107	415895	413709	415254	415574	414660	416181
	413703	415416	415061	413514	413978	416211	415844	415419	415772	415709
	415113	413758	413677	415549	413489	414571	416015	413730	414451	415302
9	472617	474222	475335	475275	472574	473988	475398	474656	473816	475331
	475145	473770	473745	472715	472723	473914	473586	474639	474644	475264
	473984	474152	474644	472511	475341	472438	475379	472842	472456	472613
	474872	473503	474187	473245	473953	474407	473446	473409	474487	475335
	472680	474805	473737	473927	473312	473770	472996	473586	474942	474500
	472840	473979	474354	472920	472699	475203	474909	473350	474372	475109
	473728	474461	474077	472511	473752	474621	475030	475416	473102	473107
	473863	473043	472731	475179	475371	472686	475223	474874	473859	472556
	475411	473267	474586	475337	473344	473428	474804	473681	472446	473901

Tabla A.2: Tiempos en milisegundos generados por 10 ejecuciones paralelas utilizando entre 1 y 11 clientes concurrentes

	1	2	3	4	5	6	7	8	9	10
10	499232	498548	499320	499363	498481	499532	498559	499541	499373	498434
	498084	499212	498998	498706	498128	499150	498136	499388	498197	499227
	498585	499029	498775	499304	499404	498731	498748	498154	499059	498270
	498510	498360	498940	498557	498504	499101	498819	498998	498848	499075
	498074	498497	499017	498767	499429	498472	499281	498818	498786	499277
	499468	498932	499190	498980	499515	498921	498544	498790	499473	498778
	498928	499419	498586	499084	498201	499008	498870	498302	498211	498800
	498797	498393	499513	498595	498897	499151	498781	499228	498801	498157
	498984	498353	498868	499440	498635	498264	499162	499157	498991	498805
	498438	498195	498854	499024	498469	499098	498319	499360	498226	498931
11	568471	569396	568179	568558	569867	569211	569928	569142	567632	567967
	569744	568427	567697	567219	570224	570344	569930	568953	568636	568742
	569182	568246	568695	568450	568350	568609	570151	569748	569115	570014
	568967	568244	569404	567321	568467	569994	569527	569840	570107	570168
	567446	567790	568701	568366	569060	569356	567818	570020	567986	568192
	568844	567816	570155	568876	568676	567546	567278	569041	570230	568436
	568706	569814	568141	569908	567403	570213	567923	569783	568189	570372
	569535	569848	568356	568906	568254	569080	570231	568076	567510	568090
	568242	570015	567202	569053	567473	568089	569655	567926	568409	570354
	567144	569256	569618	569614	567027	567654	570086	570381	568018	569549
	568454	568572	567103	567729	568443	568139	569665	570068	569025	569449

Tabla A.3: Continuación de cuadro A.2 para 10 y 11 clientes concurrentes

A.3. Ginnattasio 1 - Resolución 720p

	1	2	3	4	5	6	7	8	9	10
1	30477	31221	30415	30775	31009	30319	31263	30787	31075	30609
2	33948	34538	33594	33700	34056	33619	34273	34085	34441	34097
	34223	34459	33719	34079	34225	33940	33643	34144	33924	34550
4	40479	40510	40715	40466	40636	40690	40471	40831	40615	41206
	41209	40542	41097	41190	40983	40676	41217	41172	41057	41058
	40405	40788	40509	40711	41289	40532	40399	40947	40776	41286
	41323	41376	41010	41335	40597	40427	41240	40565	41189	40792
8	58779	58659	58299	58543	58501	58909	58664	58176	58091	58663
	58491	58516	58062	58567	58037	58600	58002	57972	58089	58662
	58813	58326	58393	58637	58655	58223	58646	58923	58800	58143
	58663	58189	58411	58382	58442	58354	58060	58446	58903	58182
	58156	58099	58687	58518	58366	58897	58362	58483	58816	58503
	58080	58538	58416	58761	58691	58435	58076	58070	58096	58674
	58541	57954	58611	58026	58689	58320	58098	58099	58078	58937
	58227	58587	58292	58100	58551	58330	58072	58225	58508	58204

Tabla A.4: Tiempos en milisegundos generados por 10 ejecuciones paralelas utilizando entre 1 y 8 clientes concurrentes con resolución 720p

A.4. Ginnattasio 1 - Resolución 480p

	1	2	3	4	5	6	7	8	9	10
1	7903	8295	7694	7403	7535	7728	7598	7669	7899	8224
2	8379	8747	8296	8154	8194	8175	8079	8148	7961	8801
	8736	8057	8492	8275	8206	8598	8742	8500	8568	8845
4	11285	12126	11847	11309	12033	11522	11651	11262	11342	11736
	11323	11411	11491	11449	11316	11498	11783	11404	11676	11724
	12064	11684	11393	11727	11699	11764	12211	11563	12089	11328
	11370	12128	11991	11348	11348	12079	11646	11741	11829	11599
8	19061	18809	19117	18940	19294	18941	19230	19338	19213	19474
	19321	18711	19219	19156	18623	19118	18772	18827	19043	19140
	19487	18548	18575	19408	18921	19442	18527	18974	18524	18556
	18840	19094	18594	19243	18558	19150	18595	19405	18621	18711
	18779	18903	19089	18850	18932	18746	18501	19321	18856	18511
	18873	19058	18806	19311	18630	19000	19346	18606	18837	19137
	18949	18712	19010	19274	18929	18632	18793	18557	19382	18848
	19487	19210	18807	19237	18527	18923	18908	19139	19402	19388
16	40715	40757	40296	40794	40213	40305	40386	40574	40965	40205
	40219	40203	40247	40966	40420	40082	40367	40781	40515	40506
	40920	40289	40814	40401	40403	40628	40372	40164	40320	40442
	40169	40660	40181	40152	40930	40357	40763	40920	40160	40599
	40647	40229	40677	40400	40499	40941	40585	40183	40869	40781
	40865	40034	40075	40578	40365	40652	40140	40151	40127	40024
	40081	40849	40679	40827	40341	40632	40019	40501	40378	40638
	40613	40908	40594	40769	40246	40647	40985	40676	40995	40910
	40469	40312	40944	40561	40591	40378	40268	40091	40339	40327
	40780	40859	40236	40501	40239	40655	40996	40256	40710	40529
	40521	40868	40347	40879	40533	40446	40463	40942	40963	40132
	40286	40711	40113	40162	40711	40483	40946	40747	40930	40807
	40594	40637	40762	40530	40626	40409	40298	40219	40192	40555
	40698	40974	40947	40695	40948	40144	40978	40787	40035	40816
	40090	40651	40256	40118	40448	40849	40172	40431	40597	40365
	40083	40338	40393	40591	40982	40104	40859	40654	40338	40871
24	56739	56249	56129	56590	56602	56206	56623	56428	56848	56442
	56213	56934	56845	56543	56404	56660	56873	56843	56886	56931
	56583	56348	56442	56228	56438	56815	56952	56781	56654	56984
	56888	56452	56325	56233	56261	56536	56557	56335	56914	56490
	56448	57038	56367	56797	56761	56637	56271	56632	56996	57021
	56574	56417	56688	56765	56501	56679	56677	56241	56432	56544
	57027	56753	56682	56334	56218	56318	56224	56727	56750	56969
	56747	56105	56219	56987	56315	56138	56339	56154	56228	56166
	56566	56464	56313	56196	56566	56658	56419	56469	56726	56277
	56128	56299	56894	56744	56104	56774	56691	56981	56881	56100
	56892	56373	56223	56545	57074	56530	56588	56720	56980	56709
	56391	56533	56558	56368	56536	56818	56187	56827	57058	57058
	56103	56200	57008	56638	56377	57074	56991	56822	56367	56952
	56912	56218	56624	56728	56401	57029	56738	56127	56895	56267
	57050	56918	56470	56858	56230	56137	56181	56301	56110	56884
	56628	56104	56537	56889	56741	56444	56960	56803	56780	57019
	56286	56173	56497	56552	56273	57019	56124	56486	56420	56557
	56494	56442	56247	56244	56795	56441	56381	56962	56323	56978
	57010	56809	56964	56222	56628	57051	56451	56932	56849	56472
	56563	56453	56945	57086	56872	56987	56736	56904	56915	56838
	56530	56631	56986	56596	56395	56751	56126	56731	56547	56382
	56096	57005	56781	56651	57071	56204	56460	56382	56797	56963
	56529	56647	56499	56759	56720	56496	56523	56511	56577	56419
	56201	56455	57053	56787	56990	56198	56347	56646	56276	56678

Tabla A.5: Tiempos en milisegundos generados por 10 ejecuciones paralelas utilizando entre 1 y 24 clientes concurrentes con resolución 480p

A.5. Ginnattasio 1 - Resolución 360p

	1	2	3	4	5	6	7	8	9	10
1	3848	4214	3550	3552	4345	4179	3980	3671	4159	3557
2	4882	4614	4359	5038	5032	4849	4654	4968	4697	4371
	5188	4851	4609	4959	4748	4423	5049	4742	4865	5151
4	6951	7020	6756	6812	6279	6781	6980	7019	6854	6198
	6135	6416	6817	6473	6904	6228	6402	6611	6162	6493
	6515	6247	6649	6493	6429	6305	6323	6293	6485	7043
	6354	6434	6639	6988	6322	6432	6911	6852	7098	6775
8	10693	10764	11294	10560	11146	10534	10422	11263	11054	10785
	11229	10574	11036	10667	10594	10549	10857	10882	10434	11386
	10588	10558	11109	10808	11222	10494	10723	10919	10832	11284
	11027	10724	10551	10964	11155	10738	10688	11147	11100	11293
	10439	11374	10670	11359	11180	11293	11240	11235	11082	11370
	10403	11137	10492	11206	10904	10609	10478	10720	10871	10770
	11209	11210	10844	10844	10577	10529	11152	11368	10785	11046
	10531	10475	10924	10555	11251	11305	11197	11286	10776	10732
16	20957	20664	20844	20142	20439	20721	20522	20929	20473	20481
	20475	20435	20835	20848	20377	20458	20334	20535	20779	20895
	20211	20061	20767	20773	20691	20422	20073	20363	20536	20676
	20162	20560	20470	20874	20048	20926	20577	20924	20869	20188
	20119	20103	20402	20348	20790	20755	20085	20810	20585	20441
	20140	20268	20253	20727	20804	20683	20498	20704	20080	20988
	20614	20072	20285	20613	20843	20985	20893	20268	20698	20757
	20649	20649	20541	20816	20112	20586	20247	20184	20157	20594
	20685	20802	21002	20282	20436	20682	20235	20502	20231	20938
	20365	20407	20332	20118	20658	20514	20184	20827	20956	20210
	20900	20898	20084	20162	20490	20412	20812	20725	20178	20425
	20545	20613	20067	20856	20147	20306	20049	20582	20812	20033
	20237	20065	20160	20292	20509	20660	20226	20808	20822	20093
	20811	20182	20718	20663	20470	20484	20078	20546	20356	20781
	20596	20358	20331	20395	20605	20367	20560	20317	20927	20174
	20492	20927	20349	20636	20497	20628	20056	20723	20028	20547
24	37967	37253	37119	37261	37930	37295	37581	37805	37620	37972
	37460	37370	37900	37709	38045	37861	37222	37689	37994	37084
	37956	37628	37902	37466	37235	37547	37377	37530	37089	37697
	37908	37852	37983	37345	37465	37965	37837	37594	37895	37240
	37964	37675	37983	37562	37787	37192	37371	37401	37380	37459
	37396	37476	37423	37427	38015	37502	38034	37876	37114	37181
	37190	37197	37899	37360	37953	37528	37185	37952	37086	37153
	37606	37730	37305	37539	37673	37163	37545	37553	37719	37279
	37395	37845	37226	37981	37244	37723	37685	37735	37516	37712
	38004	37104	37953	38048	37757	37819	38026	37244	37245	37634
	37247	37884	37269	37241	37708	37323	37356	37236	37199	37287
	37442	37821	37851	37119	37870	37872	37110	37248	37320	37088
	37097	37623	37850	37907	37293	37940	37183	37901	38009	37740
	37979	37367	37119	37218	37274	37590	37754	37700	37688	38018
	38031	37271	37277	37959	37935	37291	37237	37885	37674	38019
	37795	37804	37864	38048	37477	37744	37054	37318	37874	37387
	37901	37147	37092	37396	37391	37274	37954	37943	37700	37081
	37940	37564	37059	37317	37128	37881	37175	37568	37451	37114
	37835	37674	37617	37585	37115	37223	37892	37468	37122	37156
	37805	37048	37829	37231	37491	37172	37548	37084	38025	37541
	37361	37651	37231	37404	37344	37188	37936	37107	37317	37440
	37106	38007	37689	37271	37567	37489	37067	37744	37146	37635
	37829	37392	37136	37681	37750	37860	37717	37757	37452	37602
	37723	38029	37637	38017	37397	37989	37589	37095	37750	38021

Tabla A.6: Tiempos en milisegundos generados por 10 ejecuciones paralelas utilizando entre 1 y 36 clientes concurrentes con resolución 360p

	1	2	3	4	5	6	7	8	9	10
32	49857	49493	49445	49312	50068	49463	49652	50246	49446	49994
	50097	49947	49653	49950	49370	50163	49486	49586	50202	49753
	49682	49755	49613	50061	49311	49808	50124	50159	49351	49494
	49932	49690	49841	49387	49897	50097	49360	50228	49515	49867
	49781	50015	49431	50242	50109	50034	49275	49668	49764	49931
	49852	49540	49930	49359	49621	50035	49473	50180	49503	50270
	49695	50208	50004	49425	49948	49685	49705	49890	50034	49897
	49911	49804	50049	49868	49309	49640	49479	49767	49959	50145
	49362	49422	49381	49958	49536	49748	50103	49393	49458	49463
	49714	49825	49793	49860	49989	49280	49920	49951	49326	49989
	49469	49804	49406	50155	49333	50268	49338	50134	49838	49654
	50248	49682	49593	49317	50095	49335	50180	49849	49278	49879
	49644	49613	49312	50253	49375	49331	49341	49631	49887	49761
	49365	50015	49399	49967	49500	49784	49891	49550	49425	49615
	49272	49426	49534	49340	49371	50156	49575	49283	49540	49474
	49289	49414	49951	49753	49907	49447	49712	49749	49361	50072
	49591	49376	50009	49447	49536	49283	49661	49810	50084	49764
	49283	50083	49875	49862	49832	50256	49871	49497	49616	50055
	49647	49342	49287	49380	49779	49749	49725	49454	50242	49926
	50150	50030	49413	49618	50161	49279	49323	49726	50011	49551
	49597	50179	49886	49317	49611	49919	49454	49674	49996	50009
	49797	49707	49325	49436	50074	49830	49786	49791	50016	49911
	49436	49284	49909	49480	50098	49667	50029	49362	49350	50084
	49954	50165	50010	49760	49705	50165	49299	49361	49690	49488
	49792	49778	49340	50126	50266	50268	49625	49465	49370	50128
	49805	49723	50179	50054	49842	49453	49446	50048	50022	49761
	49835	49323	50006	49759	49706	49747	49749	50108	50011	49948
	49749	49699	49644	49889	49371	49349	50188	49389	49722	50010
	49543	50163	50106	49878	49310	49400	49395	49641	50006	49758
	49960	50042	49291	49429	49694	49344	49906	49661	49379	50075
	49338	50115	50056	49836	49280	49628	49863	49382	49470	49296
	50074	49871	49955	49649	49558	50101	50164	50048	49409	49564

Tabla A.7: Continuación de cuadro A.6 para 32 clientes concurrentes con resolución 360p

	1	2	3	4	5	6	7	8	9	10
36	56576	56259	56813	56205	56359	56449	56780	56866	56781	56294
	56572	56841	56654	57074	56621	56405	56960	57120	56316	56376
	56937	56394	56784	56744	56975	56229	57132	56708	56895	56560
	56204	56638	56226	56460	56462	56476	56284	56163	56736	56275
	56916	56530	56731	56723	56985	56573	56763	56181	56369	56688
	56408	56191	56575	56830	56694	56254	56426	57097	56656	56387
	56405	56878	56880	56295	56249	56873	56991	56822	56719	56419
	56324	57136	56882	56778	56242	56337	57035	56732	56218	56204
	56508	56295	56986	56959	56370	57058	56300	56979	56753	57078
	56556	57092	56858	56963	56288	56590	56847	56776	56504	56991
	56641	56811	56926	56505	56616	57113	56715	56522	56560	56843
	56268	56868	56826	56263	56552	56886	56283	56344	56780	56235
	57016	56441	56791	56686	56349	57059	56668	56849	56168	56730
	57074	56895	56354	56935	56930	56514	57038	56633	57132	57057
	56729	57094	56718	56925	56359	56472	56159	57103	56209	56418
	56214	56141	56541	56654	56279	56639	57059	56724	56546	57048
	56173	56186	57086	56444	56932	57093	56642	56682	56202	57046
	56251	56307	56369	56461	56205	56946	56935	56328	56782	56279
	56781	56283	56676	56278	56294	56993	56494	56843	56275	56746
	56562	56509	56881	56187	56984	56306	56950	56284	56902	56308
	56320	56303	56303	56874	56229	56704	56513	56602	56213	56683
	56540	56891	56835	56266	56669	56628	56869	56393	56772	56191
	56333	56589	56485	56282	56393	56637	56838	56517	56212	56753
	56948	57089	56178	56957	57051	56178	57023	56403	56900	56511
	56735	57072	56291	56908	56341	56548	57096	56447	56268	56224
	56167	56962	56341	56603	56352	56838	56521	56335	56222	56988
	56553	56153	56943	57067	56408	56706	57022	56986	56610	56927
	56710	56155	56963	57127	56932	56277	57063	56449	56990	56549
	56555	56916	57006	56658	56681	56983	56643	56848	56407	56682
	56766	56177	56741	56359	56180	56308	56407	56615	57062	56874
	56762	56366	56323	57115	56544	56634	56954	56756	56393	56754
	56362	56696	57117	56941	56767	56421	57131	56716	56799	56739
	56771	56965	56638	56759	56349	56786	56880	56603	57109	56882
	57069	56626	56579	56884	56863	56686	56365	56667	56818	56843
	56840	56946	57131	56225	56815	56148	56808	56814	56822	56771
	56719	57091	57139	56805	56922	57114	56252	56612	57131	56698

Tabla A.8: Continuación de cuadro A.6 para 36 clientes concurrentes con resolución 360p

Bibliografía

[1] A. Basu and X. Li, *Computer Vision: Systems, Theory and Applications*, 1st ed. Singapore: World Scientific Publishing Co. Pte. Ltd., 1993.

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *8th European Conference on Computer Vision*, Prague, 2004, pp. 25–36.

[3] W. Burger and M. J. Burge, *Digital Image Processing: An Algorithmic Introduction Using Java*, 1st ed. New York: Springer Science+Business Media, 2008.

[4] W. Burger and M. J. Burge, *Principles of Digital Image Processing: Advanced Methods*, 1st ed. London, United Kingdom: Springer-Verlag, 2013.

[5] Bytedeco, "JavaCV," 2015. [En línea]. Disponible: https://github.com/bytedeco/javacv Acceso: setiembre, 2015.

[6] Bytedeco, "JavaCV Forum," 2015. [En línea]. Disponible: https://groups.google.com/forum/!forum/javacv Acceso: setiembre, 2015.

[7] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1337–1342, 2003.

[8] J. J. Esqueda and L. E. Palafox, *Fundamentos para el procesamiento de imágenes*, 1st ed. Mexicali, México: Universidad Autónoma de Baja California, 2005.

[9] B. Furht and A. Escalante, *Handboook of Cloud Computing*, 1st ed. New York: Springer Science+Business Media, 2010.

[10] M. Ghanbari and I. of Electrical Engineers, *Standard Codecs: Image Compression to Advanced Video Coding*, London, United Kingdom, 2003.

[11] A. F. Granados and J. I. Marin, "Detección de flujo vehicular basado en visión artificial," *Scientia et Technica*, vol. XIII, no. 35, pp. 163–168, 2007.

[12] B. Horn and B. G. Schunck, "Determining optical flow," Cambridge, MA, Tech. Rep., 1980.

[13] Itseez, "OpenCV," 2015. [En línea]. Disponible: http://opencv.org/ Acceso: setiembre, 2015.

[14] S. Jayaraman, S. Esakkirajan, and T. Veerakumar, *Digital Image Processing*, 1st ed. New Delhi, India: Tata McGraw-Hill Education, 2009.

[15] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook*, 1st ed. Birmingham: Packt Publishing Ltd., 2011.

[16] R. Lienhart, E. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," in *DAGM 25th Pattern Recognition Symposium*, Magdeburg, 2003, pp. 297–304.

[17] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," *IEEE ICIP*, 2002.

[18] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, 1981, pp. 674–679.

[19] Amazon Web Services, Inc., "Aws — elastic compute cloud (EC2) de capacidad modificable en la nube," 2015. [En línea]. Disponible: http://aws.amazon.com/es/ec2/ Acceso: setiembre, 2015.

[20] Cluster FING, "Cluster FING," 2015. [En línea]. Disponible: https://www.fing.edu.uy/cluster/index.php Acceso: setiembre, 2015.

[21] Docker, Inc., "Docker - build, ship and run any app, anywhere," 2015. [En línea]. Disponible: https://www.docker.com Acceso: setiembre, 2015.

[22] Drive Cam UK, "M6 motorway traffic," 2013. [En línea]. Disponible: https://www.youtube.com/watch?v=PNCJQkvALVc Acceso: setiembre, 2015.

[23] El País, "La imm controlará el tránsito con cámaras y semáforos inteligentes," 2015. [En línea]. Disponible: http://www.elpais.com.uy/informacion/imm-controlara-transito-camaras-semaforos-inteligentes.html Acceso: noviembre, 2015.

[24] El País, "Maldonado cobrará un impuesto a la seguridad para financiar cámaras," 2015. [En línea]. Disponible: http://www.elpais.com.uy/informacion/maldonado-cobrara-impuesto-seguridad-financiar.html Acceso: noviembre, 2015.

[25] Free Software Foundation, Inc., "The GNU C Library," 2015. [En línea]. Disponible: http://www.gnu.org/software/libc/ Acceso: setiembre, 2015.

[26] Google Inc., "Google Compute Engine - Cloud Computing and IAAS — Google Cloud Platform," 2015. [En línea]. Disponible: https://cloud.google.com/compute/ Acceso: setiembre, 2015.

[27] Instituto Nacional de Estadística, "Uruguay en cifras 2014," Montevideo, Uruguay, 2015. [En línea]. Disponible: http://www.ine.gub.uy/biblioteca/uruguayencifras2014/Uruguay_en_cifras_2014_Cap_11.pdf Acceso: setiembre, 2015.

[28] Linode, LLC, "SSD Cloud Hosting - Linode," 2015. [En línea]. Disponible: https://www.linode.com/ Acceso: setiembre, 2015.

[29] Rocket Community, "rocket," 2015. [En línea]. Disponible: http://rocket.readthedocs.org/en/latest/ Acceso: setiembre, 2015.

[30] Stack Exchange Inc, "Stackoverflow," 2015. [En línea]. Disponible: http://stackoverflow.com/questions/tagged/javacv Acceso: setiembre, 2015.

[31] The Apache Software Foundation, "Apache maven project," 2015. [En línea]. Disponible: https://maven.apache.org/ Acceso: setiembre, 2015.

[32] Unidad Nacional de Seguridad Vial, "Siniestralidad vial en uruguay," Montevideo, Uruguay, 2015. [En línea]. Disponible: http://unasev.gub.uy/inicio/sinatran/informes_siniestralidad_vial_uruguay/ Acceso: setiembre, 2015.

[33] D. Mora, A. Páez, and J. Q. Sepúlveda, "Detección de objetos móviles en una escena utilizando flujo Óptico," in *XIV Simposio de Tratamiento de Señales, Imágenes y Visión Artificial*, Pereira, 2009.

[34] M. Piccardi, "Background subtraction techniques: a review," in *IEEE International Conference on Systems, Man and Cybernetics, 2004*, The Hague, pp. 3099–3104.

[35] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998.

[36] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition, 1994*, Seattle, pp. 593–600.

[37] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, 4th ed. Boston: Cengage Learning, 2014.

[38] J. M. B. Stephen M. Smith, "Susan—a new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[39] F. D. Team, "FFmpeg," 2015. [En línea]. Disponible: http://ffmpeg.org/ Acceso: setiembre, 2015.

[40] L.-W. Tsai, J.-W. Hsieh, and K.-C. Fan, "Vehicle detection using normalized color and edge map," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 850–864, 2007.

[41] G. E. Urrego, F. C. Calderón, A. Forero, and J. A. Quiroga, "Adquisición de variables de tráfico vehicular usando visión por computador," *Revista de Ingeniería*, no. 30, pp. 7–15, 2009.

[42] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Conference on Computer Vision and Pattern Recognition, 2001*, Kauai, pp. 511–518.

[43] W. Zhan and X. Ji, "Algorithm research on moving vehicles detection," *Procedia Engineering*, vol. 15, pp. 5483–5487, 2011.