ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"MONITOREO AUTOMÁTICO DE CARRETERAS MEDIANTE EL USO DE UN SISTEMA DE DETECCIÓN, SEGUIMIENTO Y EXTRACCIÓN DE CARACTERÍSTICAS DE VEHÍCULOS CON TÉCNICAS DE VISIÓN POR COMPUTADOR"

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN EN SISTEMAS TECNOLÓGICOS

INGENIERO EN COMPUTACIÓN EN SISTEMAS MULTIMEDIA

Presentado por

Jorge E. Faytong Real

Giancarlo J. Moggia Cucalón

Guayaquil - Ecuador

2008

AGRADECIMIENTOS

Agradecemos a Dios en primer lugar. A nuestras familias, que son el constante apoyo y puerto seguro cuando nos desalentamos. Nuestra gratitud más sincera al invaluable y constante apoyo que nos ofreció Boris Vintimilla B. Ph. D, nuestro director de tesis, desde habernos brindado la oportunidad de trabajar con él en el Centro de Visión y Robótica hasta su dirección para este trabajo. Al Ing. Miguel Realpe, por sus consejos y apoyo incondicional. A todos nuestros maestros, profesores, compañeros y amigos, por compartir su tiempo y alegría con nosotros

.

A Bulún

A mis padres.

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

RESUMEN

Las vías rápidas o autopistas son una parte importante del sistema de progreso y desarrollo de un país. La función de las autopistas o carreteras como caminos arteriales entre centros urbanos individuales o ciudades será cada vez más importante para el manejo de flujos de enlace hacia y dentro de las ciudades. Sin embargo, cuando el parque automotor local experimenta un crecimiento acelerado, un problema que puede derivarse es que usualmente la cantidad de vigilantes de tránsito o monitoreo vial es insuficiente para el debido control vehicular. Por otra parte, las empresas privadas dedicadas a la concesión vial necesitan herramientas para elaborar estudios de mantenimiento o negociación de las vías.

En ambos casos, la tarea de analizar la cantidad de vehículos circulando en carreteras se lo ha hecho usualmente de manera manual. Esto, aunque algo reducido en costo, es poco eficiente ya que la capacidad de concentración del ser humano disminuye en jornadas prolongadas, acentuándose más el cansancio de la persona encargada a medida que avanza el tiempo. Adicional a este problema, se puede también destacar que este tipo de control no se lo hace permanentemente, es decir, las 24 horas y 7 días a la semana; sino por el contrario, el trabajo de control manual se lo hace en determinados días y durante ciertas horas. Después de que los datos estadísticos de control de vehículos han sido obtenidos de forma manual, esta información es llevada a un profesional o empresa especializada para extraer datos de interés para el concesionario o la entidad pública.

Otros sistemas que también son usados para realizar trabajos similares a la de este proyecto poseen un alto costo, debido a sus componentes o necesidad en grandes cantidades de los mismos. Un ejemplo claro de esto son los sistemas de detectores por láser.

En este proyecto se pretende implementar un sistema que usa técnicas de visión por computador para detectar, seguir y extraer características básicas de vehículos que se encuentran circulando en carreteras o autopistas. Para probar el sistema implementado se prevé usar como banco de prueba el uso de videos que han sido generados en vías rápidas o carreteras a través del uso de cámaras. Los resultados obtenidos con el sistema propuesto ofrecen información básica de los vehículos, tales como: dirección de su movimiento, carril en el que se desplaza, tamaño de la caja englobante que lo contiene, entre otros.

La información que se genera de este proyecto podría servir como base para ayudar a monitorear el flujo vial en carreteras o autopistas, lo cual posteriormente facilitará las labores de control vial de los servidores públicos y la empresa privada. No es arriesgado pensar que en un futuro cercano se podría reutilizar o adaptar el sistema para realizar seguimiento vehicular en zonas urbanas. Adicionalmente, podría también considerarse su uso total o parcial para proyectos similares en túneles, vigilancia en video, sistemas de llamadas de emergencia, sistemas de peaje, entre otras aplicaciones afines.

Para llevar a cabo la implementación de este proyecto, inicialmente se realizó un estudio del estado del arte de las diferentes técnicas existentes en los campos de detección de objetos en movimiento y de seguimiento de objetos en movimiento. Con este estudio se definió cuales técnicas serían usadas o adaptadas a la solución de nuestro problema. Posteriormente, se implementaron los respectivos módulos de detección y seguimiento de vehículos en movimiento previamente definidos. Finalmente, para darle robustez al sistema y veracidad a los resultados obtenidos con el módulo de seguimiento de vehículos, un filtro de predicción adaptativo fue también desarrollado como complemento de este módulo.

INDICE GENERAL

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

1. INTRODUCCIÓN

Hoy en día el tráfico vehicular es complicado en la mayoría de las áreas metropolitanas y rurales de casi todos los países en los que la congestión y el mantenimiento de las vías se han convertido en problemas cotidianos de difícil solución. Estos problemas son algunos de los causantes de efectos indeseados en la movilidad de los conductores y peatones. Adicionado a esto, el pésimo o inexistente monitoreo vehicular para generar estadísticas de la cantidad de vehículos que circulan por una determinada carretera muchas veces incide en que la planificación para el mantenimiento de dichas vías no sea el más adecuado, lo que atrasa la transportación y por ende a la sociedad en aspectos económicos, sociales, de salud y comunicación. Todo ello redunda en una disminución evidente del bienestar de la población.

Una de las respuestas más eficientes al problema del mantenimiento de vías y la congestión radica en el uso intensivo de sistemas informáticos y de las telecomunicaciones aplicadas a la gestión del tráfico. En efecto, los denominados *Sistemas Inteligentes de Transporte* (SIT), son un ejemplo que incluye electrónica avanzada, comunicaciones y sistemas informáticos para aumentar la eficiencia y seguridad del transporte por carretera y su mantenimiento.

El monitoreo de tránsito vehicular es un área en la que se están utilizando sistemas avanzados de visión por computador y electrónica para mejorar el control del tráfico y

para realizar un mejor estudio de necesidades para los programas de manutención vial. En algunas ciudades, la circulación de las principales carreteras se controla mediante cámaras de vídeo, radares o sensores en la propia carretera. Un sistema de visión por computador analiza la información. Esta tecnología de la información se ha convertido en un eficiente apoyo para el ciudadano y para las instituciones públicas en el intento de paliar los problemas de congestión de los transportes urbanos e interurbanos, no solamente ayudando a mejorar su movilidad sino haciéndola más sostenible. En este sentido, un sistema de transporte inteligente para monitoreo vial mediante detección, seguimiento y extracción de características básicas de vehículos en movimiento en vías rurales será el objetivo de este trabajo.

Para obtener un monitoreo personalizado y robusto, con extracción automática de características, los sistemas de transporte inteligente poseen un sinnúmero de técnicas a su disposición. Los *métodos basados en conocimiento*, *véase [18, 19, 20, 22]*, tienen la ventaja de poseer un costo de implementación relativamente más bajo. Sin embargo, poseen la tremenda desventaja de la eliminación de regiones de interés en su etapa de detección, por lo que brindan resultados no muy concisos y específicos. Además tienen la característica de ser muy rígidos y consecuentemente no son muy adaptables.

Por otro lado, los sistemas que emplean *técnicas de reconocimiento en estéreo*, *véase [18, 25, 26],* ofrecen resultados sumamente concisos y exactos. Lastimosamente tienen como desventaja que los recursos computacionales demandados son sumamente altos. El

modelo a su vez necesita de entradas con un alto grado de resolución, lo cual incurre en un alto precio a su implementación.

Finalmente se encuentran los sistemas con *detección basados en movimiento*. Estos son capaces de encontrar objetos y obstáculos móviles utilizando las técnicas de *diferenciación temporal, flujo óptico* y *background subtraction*, *véase [13, 14, 17, 24, 25]*. Existen sistemas y aplicaciones con este método en forma tanto discreta como continua. Es un hecho que las formas discretas de implementación de esta técnica son las que mejores y más eficientes resultados ofrecen. Los sistemas que utilizan el background subtraction tienen múltiples ventajas. Algunas válidas de mencionar son que ofrecen resultados excepcionales y altamente descriptivos, no introducen ruido al proceso en cada una de las ejecuciones del sistema, es también extremadamente adaptable a las necesidades y condiciones para las cuales tengamos en mente darle uso y su carga computacional es sumamente modesta.

El sistema desarrollado en nuestro proyecto incluye la técnica de background subtraction por las ventajas mencionadas arriba y descritas en detalle en el capítulo 2. Adicional a esta técnica se implementó un algoritmo para seguir a los vehículos en movimiento, en nuestro caso una combinación de mezclas Gaussianas. Gracias a esta técnica de seguimiento de vehículos, el sistema obtuvo una mayor calidad en los resultados al ser más explícitos. El sistema es también capaz de determinar ciertas características básicas de los vehículos mediante cálculos simples pero eficaces. Nuestra investigación se vio

reforzada incluso un poco más con la acertada inclusión de algoritmos adaptativos lineales de predicción del movimiento.

Las ventajas de nuestro sistema en relación a otros que hemos podido evaluar no son demasiado numerosas, pero si importantes. En primer lugar contamos con un módulo de pregrabado de la secuencia de imágenes de entrada, siendo esta una ventaja sobresaliente con respecto a otros sistemas similares. Otros sistemas se enfocan en mejorar la eficiencia del procesamiento frame a frame, en lugar de reducir el material de entrada con el que se trabaja. Nosotros ahorramos horas de video y de procesamiento superfluo con este práctico módulo de eliminación de material de entrada insignificante. Además incluimos un resumen de características básicas, como dirección del movimiento, carril en que se encuentra, área del vehículo, entre otros, lo cual hace más completa la información retornada por el sistema.

Es beneficioso que nuestro sistema no necesita trabajar en tiempo real, sino que solo se requiere aplicarse sobre videos almacenados en disco, por lo cual no se necesitan algoritmos demasiado complicados que ofrezcan un fuerte desempeño en tiempo real. A esto se lo considera como una gran ventaja al momento de la implementación del algoritmo del prototipo.

La posibilidad de ingresar diferentes tipos de entradas es también una característica valiosa sobre otros sistemas. Es decir, aunque nuestro sistema está optimizado para

pero no recomendado para óptimos resultados, utilizar una señal de video capturada directamente por una cámara, al igual que imágenes simples. Esto permite la adaptación de nuestra interfaz para varios algoritmos, ya sea para realizar pruebas, o tener diferentes alternativas del sistema en tiempo de ejecución.

La Figura 1 muestra un diagrama explicativo de los módulos del sistema desarrollado y sus correspondientes relaciones.

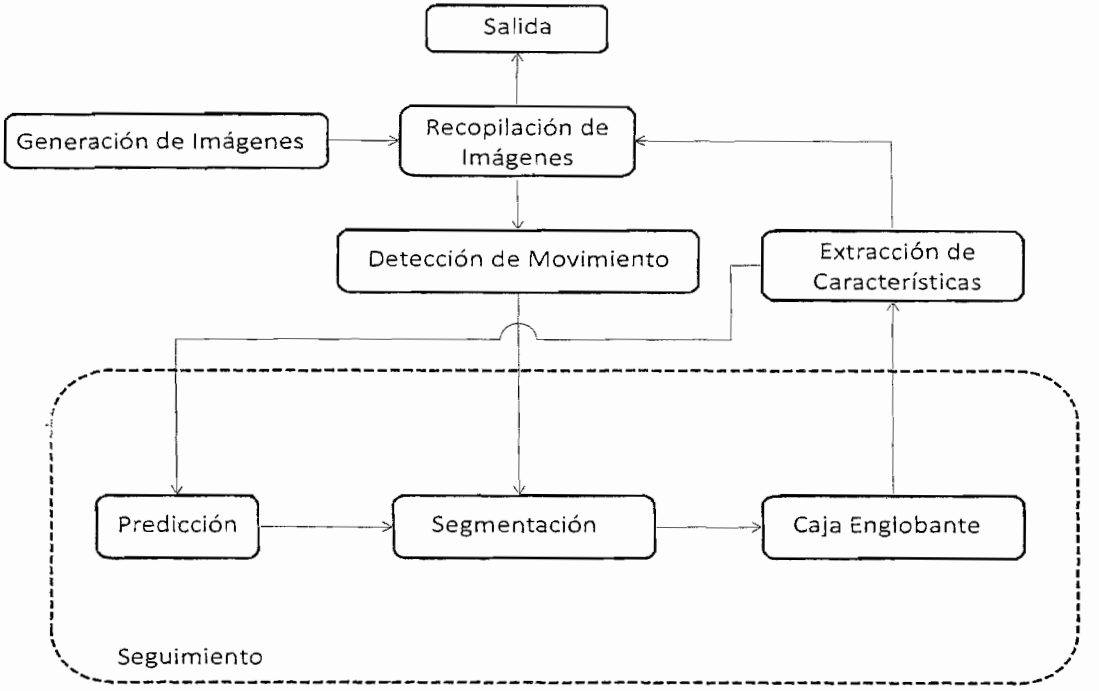

Figura 1 Diagrama del sistema desarrollado: Detección, seguimiento y extracción de características básicas de vehículos en movimiento.

La información contenida en los diferentes capítulos de este documento puede ser resumida de la siguiente forma: el Capítulo 1 incluye una breve introducción del sistema en general. El Capítulo 2 expone un estudio bibliográfico de los trabajos previos que tienen relación con nuestros proyectos. Los módulos de generación de imágenes a procesar y de detección de vehículos en movimiento mediante la técnica de background subtraction son descritos en el Capítulo 3. En el Capítulo consecuente, desarrollaremos cómo implementamos el seguimiento de los vehículos previamente detectados y definidos como en movimiento. En este capítulo se explicará nuestro trabajo con la técnica de la combinación de mezclas Gaussianas para la segmentación y seguimiento de los vehículos en movimiento, extrayendo sus características básicas mediante técnicas de extracción de blobs, caja englobante y determinación de centroide. Se procederá entonces a detallar cómo nuestro sistema maneja la predicción de estados subsecuentes de los vehículos bajo seguimiento utilizando los filtros adaptativos lineales de Kalman y del desplazamiento previo. Concluida esta ilustración indicaremos cómo se completa el ciclo de procesamiento de datos del sistema al generar un video de salida con toda la información detectada a través de cada uno de los módulos previos. A continuación en el Capítulo 5 incluiremos los resultados obtenidos en las pruebas experimentales con su respectivo análisis. Finalmente agregaremos conclusiones y recomendaciones extraídas gracias al trabajo realizado en este proyecto.

2. TRABAJOS RELACIONADOS

En el presente capítulo se realiza un análisis bibliográfico de los distintos métodos existentes para detectar y seguir objetos en movimiento en una secuencia de imágenes o videos. Primero se explicarán diferentes técnicas de detección de objetos en movimiento usando visión por computador, mientras que en la segunda instancia se mostrarán trabajos y técnicas relacionadas con el seguimiento de los mismos objetos

2.1 INTRODUCCIÓN

A continuación explicaremos con breve detalle algunos trabajos publicados en la literatura que están relacionados con la detección y seguimiento de objetos móviles mediante el uso de herramientas de visión por computador. Analizaremos diferentes métodos que han sido desarrollados tanto para la detección como para el seguimiento de objetos en una secuencia de imágenes. Es muy importante exponer estos trabajos desde un inicio, para que el lector empiece a conocer información valiosa, necesaria para entender el porqué de las de las decisiones tomadas por nosotros y para una completa comprensión de los temas desarrollados. Al final del capítulo definiremos que técnicas de detección y seguimiento se implementarán en este proyecto, en base al estudio bibliográfico realizado.

2.2 DETECCION DE OBJETOS EN MOVIMIENTO

Usualmente un sistema de vigilancia de objetos en movimiento a través de visión por computador está formado por 3 módulos, que son: detección de objetos en movimiento, seguimiento de los objetos en movimiento y análisis del movimiento. Para nuestro estudio bibliográfico hemos considerado que un objeto moviéndose representa a un vehículo. En algunos estudios también se han considerado a seres humanos, estos casos especiales serán debidamente señalados.

De esta forma, la detección de objetos en movimiento es el primer paso a considerar en el sistema de visión por computador. Este módulo es una parte esencial dentro del procesamiento, debido a que con los datos generados en esta fase el sistema trabajará para devolver información para el posterior seguimiento y análisis. Está bastante claro que si la detección es defectuosa la información extraída por el sistema tendrá poco valor, al ser inexacta y en muchos casos inútil.

En la bibliografía varios métodos de detección de movimiento han sido propuestos y pueden ser divididos en tres categorías:

- Basados en el conocimiento
- Basados en estéreo
- Basados en el movimiento

Algunos ejemplos de métodos conocidos basados en el conocimiento son explicados a continuación. Bertozzi *et al [18]* propuso un método produciendo un mapa simétrico mediante la combinación entre las simetrías a nivel de grises, de ejes horizontales y verticales. Este mapa simétrico define posiciones donde se encuentra una diferencia de brillo acentuada y sus correspondientes separaciones a los ejes de simetría. Una vez que una posición de simetría y el ancho eran detectados, una nueva búsqueda comenzaba. La nueva búsqueda apuntaba a la detección de las dos esquinas inferiores de la caja englobante. Finalmente, se buscaba el borde del tope del carro y se terminaba la definición de la caja englobante. De esta forma se definía una posición bastante exacta de la ubicación del vehículo, encerrando en una caja englobante la mayor parte, sino en su totalidad, su perímetro. El método ofrece una gran exactitud en la detección en situaciones aisladas, con secuencias de imagen sin presencia de ruido. Sin embargo, tiene defectos en casos especiales, como cuando cruzan dos carros y uno traslapa a otro en la imagen, creándose entonces falsas detecciones y cajas englobantes inadecuadas.

Papanikopoulus *et al [19]* mejoró la técnica de Bertozzi y usó la ventaja de la simetría de contorno estable en vez de una simetría de intensidad, muy propensa a ruido y errores. Inicialmente se usó el operador de Sobel para generar contornos en imágenes binarias. En un segundo intento, seleccionó dos umbrales de binarización o *thresholds* para determinar probables rango de anchos de los objetos. Finalmente empleó una técnica de escrutinio para detectar el eje de simetría vertical del objeto. Aunque esta técnica mejora

en gran escala la robustez del modelo previamente descrito por Bertozzi, agregándole mayor aplicación en casos reales, demuestra aún una dependencia marcada de la secuencia de imágenes sin ruido para el adecuado funcionamiento. Por otro lado, este método aumenta los recursos computacionales requeridos, debido a la reestructuración del método original y la implicación de un número mayor de operaciones realizadas por corrida.

Hoffman *et al [20]* por otro lado utilizó la extracción de características a partir de imágenes monocromáticas. El proceso de detección utilizó características de sombras y simetrías para generar hipótesis. Se llegó a una fórmula:

1)

$$S(x_s, w) = \frac{E_e^{'}\ \ e(x_s, w) - \ E_o(x_s, w)}{E_e^{'}\ \ e(x_s, w) + \ E_o(x_s, w)}$$

El rango de la medida de simetría es (-1,1). Si la región es totalmente simétrica, S = 1, en caso de ser completamente asimétrica, S = -1. Donde x_s es el centro de simetría, w el ancho de la ventana de captura,$e(x_s, w) = \frac{1}{2}\,(g(x_s + w) + g(x_s - w))$, $o(x_s, w) = \frac{1}{2}\,(g(x_s + w) - g(x_s - w))$, siendo g el operador de Sobel,$E_e(x_s, w) = \sum_{u=0}^{w/2} e^2\,(x_s, w)$ y $E_o(x_s, w) = \sum_{u=0}^{w/2} o^2\,(x_s, w)$.

Gracias al operador de Sobel, un gran porcentaje del ruido se elimina, pero se pierden también ciertas regiones de interés importantes. Es debido a esto que tiene una mayor aplicación en la vida real que los métodos de Bertozzi y Papanikopoulus. Sin embargo, al perder ciertas partes del objeto detectado al aplicarle el filtro de Sobel a la imagen monocromática del objeto, se pierde la exactitud en la forma del vehículo.

Huang *et al [22]* utilizó un método de reconocimiento de carriles empleando un filtro Gaussiano, procedimientos de búsqueda de picos y agrupamiento de segmentos de líneas para encontrar marcadores de tierra. Los objetos, en este caso vehículos, se encontraron mediante la búsqueda del eje vertical y una propiedad de simetría. Para encontrar donde se encontraba el vehículo se calculaba el histograma de la proyección de eje horizontal y se comparaban con los ángulos verticales. Una vez realizada esta tarea, se definía el piso mediante un algoritmo de búsqueda de abajo hacia arriba. Este filtro tiene como ventaja la abstracción de formas en su método de detección, haciéndolo ideal para videos o secuencias de imágenes donde se garantiza la presencia de una única forma u objetos, como en una fábrica de botones. Sin embargo, los videos de vehículos no siempre incluyen solo automotores, sino que manifestaciones de seres humanos, animales, entre otros, también se hacen presentes. Debido a la falta de parámetros en el método para regular las formas de los objetos detectados, se hace bastante difícil moldear el método para una efectiva detección exclusiva de vehículos.

Los métodos basados en estéreo toman mucha ventaja del mapeo de la perspectiva inversa (IPM) para estimar las posiciones de los objetos y obstáculos en una imagen. Bertozzi *et al [18]* computó el IPM de las imágenes de izquierda a derecha, teniendo incluso la ventaja de encontrar objetos que no estaban en el piso. Esta aproximación le ofrece al método basado en información descrito previamente por Bertozzi una mayor robustez al momento de la detección, haciendo que el error en la detección causado por la solapamiento de dos vehículos se disminuya notablemente. Sin embargo, este método aún tiene el defecto de reacción impredecible a la introducción de ruido no esperado al sistema.

Por otro lado, Suhr *et al [25]* empleó un lector láser para modelar en tres dimensiones el mapa global. Se definen puntos de referencias, a partir de los cuales se aplican métodos de de-rotacion y se analizan con un mosaico de 3D para estimar un parecido a una forma ya predefinida. Este método no utiliza odometría. Esta técnica incurre en una utilización de gran cantidad de sensores láser de alto costo, ubicados en distintas posiciones y comunicados con el sistema principal, lo que hace a su valor de implementación bastante elevado. Sin embargo, su efectividad es muy alta y es recomendable en caso de poseer un presupuesto cuantioso.

Argyros *et al [26]* utilizó el filtrado de los histogramas de las distribuciones normales para eliminar la profundidad en objetos en 3D previamente establecidos. En este trabajo, la detección independiente de movimiento es formulada como un parámetro robusto de

estimación aplicada a la entrada visual por un observador binocular moviéndose rígidamente.

Las mediciones de profundidad y movimiento son combinadas en un modelo linear. Los parámetros de este modelo están relacionados a los parámetros de egomoción y a los parámetros de la configuración estereoscópica del observador.

La estimación robusta de este modelo lleva a una segmentación de la escena basada en movimientos en 3D, lo cual es ventajoso. A su vez, evita el problema de correspondencia mediante el uso de campos de flujo de distribución normal. Los resultados experimentales demuestran que el método es efectivo para la detección independiente de escenas en movimiento con grandes variaciones de profundidad sin restricciones impuestas por el observador. Sin embargo, los recursos computacionales demandados por este modelo son sumamente altos. El modelo a su vez necesita de entradas con un alto grado de resolución, lo cual incurre en un alto precio a su implementación.

Finalmente, los métodos de detección basados en movimiento son capaces de encontrar objetos y obstáculos utilizando la técnica de flujo óptico. Esta técnica consiste en detectar el movimiento aparente de patrones de brillo en una imagen, mediante la generación de un vector de desplazamiento para cada pixel (aproximación continua). La principal desventaja de estos sistemas es que son imprácticos para tiempo real y de alto consumo de tiempo y recursos. En contraste con los métodos continuos, los discretos reportaron

mejores resultados utilizando características de la imagen, tales como segmentos o blobs de colores o intensidades locales máximas y mínimas.

Seol *et al [24]* presentó un modelo de estimación de movimiento de doble diferenciación basado en el enfoque de detectar y seguir objetos en movimiento en secuencias de imágenes. En una primera instancia el método de doble diferenciación operaba en dos imágenes de diferencia con un operador lógico AND, para después acumular esta información y descartar la información sin textura en la secuencia de imágenes. En una segunda instancia, se seleccionaban regiones candidatas para que aparezcan objetos en movimiento. El movimiento basado en bloques estima el desplazamiento de las regiones candidatas. Esto daba paso a eliminar regiones candidatas que no cubrían un desplazamiento superior al del *threshold*. Finalmente las regiones candidatas finales eran agrupadas por un algoritmo de *clustering* o agrupamiento. Estos objetos o bloques eran tomados como imágenes en movimiento. Este método es bastante útil y aplicable en la vida real, por lo que su uso es cada vez más frecuente. A su vez, utiliza módulos simples con procesamiento local bastante sencillo, lo cual abarata su implementación. Presenta como desventaja más sobresaliente la característica que resulta imposible delimitar un parámetro de sensibilidad a las texturas, por lo que se corre un alto riesgo de que objetos queden sin detectar y no se puede hacer algo al respecto.

Las técnicas basadas en el flujo óptico son también muy populares y efectivas para el proceso de detección. El flujo óptico es en si el patrón de movimiento aparente de los

objetos, superficies y bordes en una escena visual causada por el movimiento relativo entre un observador (un ojo o cámara) y la escena. Las técnicas de flujo óptico, como de detección de movimiento, segmentación de objetos, cálculos de tiempo de colisión y de expansión de la región de interés y mediciones estéreo de la disparidad del movimiento, utilizan este patrón de movimiento.

Es necesario mencionar que además de la clasificación de las 3 categorías para detectar movimientos que fueron descritos arriba, en la bibliografía existen también otras clasificaciones para estos métodos. Así por ejemplo, Fujiyoshi, Lipton y Kanade *et al [17]* mencionan en su artículo otra interesante clasificación para detectar objetos en movimiento. A pesar de que su trabajo está enfocado principalmente a la detección de personas en movimiento ellos mencionan la existencia de tres aproximaciones convencionales para la detección del movimiento:

- Diferenciación temporal (de dos o tres frames), véase Anderson *et al [35],* para una explicación más detallada.
- Background subtraction, Haritaoglu *et al [37]* y Wren *et al [14],* exponen este método de forma interesante.
- Flujo óptico, detalles en Barron *et al [36]*.

Las *técnicas de diferenciación temporal* son muy adaptables a ambientes dinámicos. Sin embargo, realizan generalmente un desempeño pobre en la extracción de todos los píxeles con características relevantes. El método de *background subtraction,* o también conocido como *substracción de fondo,* es una técnica muy usada y sencilla de implementar. Esta técnica provee la más extensa colección de datos de los píxeles en movimiento con características relevantes, pero es sensible a los cambios dinámicos de escena, generalmente causados por variaciones en la iluminación y factores externos. No obstante, varios algoritmos han sido propuestos para resolver estos problemas. Finalmente, el flujo óptico puede ser utilizado para detectar objetos que se mueven independientemente el uno del otro en la presencia de una cámara. Sin embargo, la mayoría de los métodos computacionales que utilizan esta técnica se basan en procesamientos demasiado complejos y son prácticamente imposibles de aplicar en algoritmos de tiempo real sin hardware especializado.

En la literatura de detección de objetos en movimiento dos nombres de imágenes que son muy usados son el *foreground* y el *background*. Por un lado, el foreground representa a la imagen de primer plano y contiene a los objetos que están en movimiento. Mientras que, el background representa a la imagen de fondo o segundo plano, la cual contiene a todos los objetos estacionarios (que no se mueven). Los objetos de la imagen de background pueden permanecer estacionarios permanentemente o pueden también variar cada cierto

tiempo, para luego permanecer estacionarios. Es debido a estos casos que el background debe ser actualizado cada cierto tiempo.

Shih y Chang *et al [29]* exponen un método para la detección de objetos en fondos no estacionarios. Los movimientos concurrentes de los pixels del foreground y background hacen que el mantenimiento plausible de un modelo de fondo para background subtraction sea sumamente complicado. Este método utiliza campos de movimiento alineados a frames vecinos, los cuales son después fusionados para reducir efectos de parallax en la detección de blobs. Se desarrolla después un modelo de fondo con un color fusionado para refinar las formas de los objetos detectados. Finalmente, la información de los blobs en movimiento es incorporada al proceso de adaptación del modelo de fondo. Solo los píxeles marcados como de fondo son adaptados en los modelos de background con cada frame entrante. Este modelo tiene muchísimas ventajas y la aplicación de cada uno de los métodos ideados para la implementación concluyen con que sea bastante sencillo, si se sabe lo que se hace, elaborar un modelo similar. Aprovecha en gran manera sus fortalezas y sus debilidades, como la correcta extracción de un fondo, pueden ser manejados por otros sistemas de baja demanda computacional. Tiene como desventaja principal la falta de adaptación del fondo, ya que no se trabaja de manera constante de frame a frame sino que se modela un fondo inicial y solo bajo ciertas circunstancias este es modificado con nueva información.

Existen un sin número de métodos que emplean la técnica de background subtraction. A continuación mencionaremos algunos de los más importantes en nuestra investigación. Heikilla y Olli *et al [27]* especifican que si un pixel cumple con

2)

Entonces pertenece al foreground. Donde es un umbral predefinido. La umbralización obtenida es después perfeccionada con una operación de cierre morfológico con un kernel de 3x3, descartando de esta forma las regiones pequeñas.

La actualización del background está regida por

3)

Donde se mantiene pequeña para prevenir que "colas" artificiales se formen detrás de objetos en movimiento.

Luego se aplican dos correcciones de background, dependiendo del estado del pixel. Si el pixel está marcado como foreground para un número más que m de los últimos M frames, entonces el backgound se actualiza como . Esta corrección está

diseñada para compensar cambios súbitos de iluminación y la aparición de nuevos objetos estáticos. Si por otro lado el pixel cambia de estado de foreground a background constantemente, se lo enmascara fuera de inclusión en el foreground. Esta acción se realiza para compensar la iluminación fluctuante, como la proveniente de ramas en movimiento. Tiene como gran ventaja la consideración del cambio del valor de brillo en cada pixel. Esto hace que las detecciones sean sumamente efectivas y exactas. Sin embargo, al no considerar ciertas regiones pequeñas, se pierden detalles de la forma de los vehículos y en un posible caso, partes o vehículos enteros.

En el sistema Pfinder, desarrollado por *Wren et al [14*] se usa un método que emplea un esquema bastante simple, donde los píxeles del background son modelados por un valor único, actualizado por

$$B_t = (1 - \alpha)B_{t-1} + \alpha I_t \quad 4)$$

Los píxeles del foreground son modelados explícitamente por una media y covarianza, que son actualizadas recursivamente. Requiere de una escena en blanco como frame inicial. Aunque resulta un método efectivo si se le brindan todas las restricciones que solicita en las especificaciones del modelamiento, carece de robustez aceptable. Al

momento del cálculo de nuevas medias y covarianzas, es necesario monitorear los resultados obtenidos para evitar que se generen datos de entrada con ruido, al poseer solo una fórmula de modelamiento. Esto da cabida a que por el cambio rápido de iluminación, como una nube repentina o una rama, los resultados se afecten increíblemente.

Halevy *et al [13]* indica que el background es actualizado por

5)

$$B_{t+1} = \alpha S(I_t) + (1 - \alpha) B_t$$

A todos los píxeles, donde $S(I_t)$ es una versión suavizada de I_t . Los píxeles de foreground son identificados mediante el seguimiento del máximo de $S(I_t - B_t)$, a la inversa de una umbralización o thresholding. Se utiliza un valor de $\alpha = [0{,}3 \ldots 0{,}5]$. Se toma también un valor de $(1\text{-}\alpha)^t < 0{,}1$ como indicador de los frames necesarios para que el background se asiente después de la inicialización.

El proceso de Halevy es mucho más flexible que los de Heikilla y Pfinder. Esto se debe al proceso de comparación con un modelo ya suavizado, lo que brinda como resultado una imagen sin cambios bruscos en los valores de brillo, haciendo que aunque se defina una mayor área inicial en movimiento, no se pierdan detalles. Esta área es después reducida mediante filtros morfológicos. Como desventaja se tiene a la necesidad de

esperar unos cuantos frames hasta que el sistema esté debidamente asentado y listo para la detección, perdiéndose así, información importante al inicio del sistema.

.

2.3 SEGUIMIENTO DE OBJETOS EN MOVIMIENTO

Cuando uno o varios objetos en movimiento han sido localizados o detectados dentro de la imagen que se está analizando es necesario tomar ventaja de esta información para realizar un adecuado seguimiento de estos objetos a lo largo de la secuencia de imágenes. De no cumplir con esta parte, mucha información con respecto al movimiento descrito por el objeto, como velocidad y dirección del movimiento, no podría ser obtenida. En este sentido, el módulo de seguimiento de objetos en movimiento ofrece datos necesarios para modelar el comportamiento y trayectoria de los mismos.

Una vez establecidos los objetos o bloques en movimiento, se los debe rastrear en las imágenes o frames consecuentes. Dos propósitos principales para el seguimiento de objetos han sido definidos en la bibliografía:

- Se puede mejorar la detección de objetos cerca de otros objetos en cada frame
- La detección más robusta de objetos en movimiento es posible.

Fujiyoshi, Lipton y Kanade *et al [17]* hacen hincapié en el seguimiento de objetos, en su caso de humanos, y que contribuyen a la generación de modelos de patrones de movimiento de diferentes objetos, información sumamente importante para una posterior

clasificación acertada con procesamiento no tan extenso, así como también para un posterior análisis del movimiento. Es importante mencionar que para movimientos de objetos más complejos es sumamente útil la técnica propuesta por ellos, sin embargo, al nosotros manejar bloques con movimiento en 2D, no es necesario utilizar los métodos y aproximaciones por ellos implementados.

Hu *et al [21]* presentó con el valor de estimación de vitalidad (VEV) un método capaz de describir el estado de objetos seguidos y con la confiabilidad de estimación de vitalidad (REV) una medida de incertidumbre de confiabilidad del objeto en movimiento. Si un objeto tiene VEV = 0, su seguimiento terminará. El valor del VEV y su correspondiente REV varían dependiendo de las apariciones de un objeto en una secuencia de frames.

6)

$$Vev(t_i) = \int_{t_0}^{t_i} \sigma_m(\tau)\, Re\ v(\tau) d\tau$$

Donde $\sigma_m(\tau)$ es el resultado del matching entre plantillas, siendo 1 cuando hay match y -1 cuando no falla. *Rev* resulta de la combinación de tres factores ponderados, los cuales son el promedio de la tasa de error de intensidad en el matching de plantillas, el factor de confinamiento de área y el factor de movimiento inercial. Tiene como principal ventaja ofrecer un valor de incertidumbre en los resultados obtenidos, por lo que el investigador puede saber a ciencia cierta con que grado de confiabilidad está trabajando. Sin embargo, la definición y obtención de un exacto valor de medida de incertidumbre es

tremendamente difícil de obtener, ya que depende de un sin número de observaciones, que si son hechas con herramientas no adecuadas, no debidamente calibradas, en desorden o con poca precisión, alterarán de manera fatal los resultados y su correspondiente grado de incertidumbre.

Hoffman *et al [20]* utilizó el filtro de modelos interactivos múltiples (IMM) para combinar varios modelos mediante el peso entregado por correspondientes filtros de Kalman. El utilizó dos modelos en IMM. La siguiente ecuación de estado describe ambos modelos:

7

$$x_k^{(i)} = A^{(i)} x_{k-1}^{(i)} + w_{k-1}^{(i)}$$

Donde i = 1, 2, denotando el primero y segundo modelo respectivamente. En esta ecuación, generalmente *w* es ruido blanco de distribución normal con media cero. El filtro entonces definido por este método es una matriz A de 4 x 4 y un vector.

$$x^{(1)} = [X\ Y\ Z\ A]^T$$

$$A^{(1)} = \begin{matrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{matrix}$$

Tiene como ventaja sobresaliente la falta de restricciones y el poco uso de recursos computacionales para la estimación de un modelo aproximado del movimiento descrito. Sin embargo, al utilizar filtros de Kalman para la definición de los pesos de los modelos, se corre el tremendo riesgo de obtener un modelo desfasado e ilógico debido a la mala definición de un estado inicial o del inadecuado cambio con las mediciones de actualización.

Cohen y Medioni *et al [38]* basan su análisis en la representación gráfica de objetos en movimiento, lo cual permite derivar y mantener una plantilla dinámica de cada objeto mediante el apoyo a su congruencia temporal. Esta plantilla junto con la representación gráfica les permite caracterizar la trayectoria descrita por el objeto como una ruta óptima en la gráfica. Básicamente, una vez detectadas pequeñas regiones en movimiento, se aplica una técnica de clustering para agrupar y fusionar las regiones que tengan congruencia de movimiento y ubicación relativa. Esta información de actualización se entrega a la representación gráfica de la trayectoria del objeto en movimiento y modifica

la plantilla dinámica, necesaria para encontrar una congruencia lógica entre la posición actual y la trayectoria previamente descrita por el objeto. Tiene como ventaja que se puede aplicar en un gran número de situaciones, donde el modelo a describir no puede definirse con exactitud. Esto se debe a que detecta pequeñas regiones de movimiento y después mediante el clustering agrupa las regiones en áreas más grandes. Sin embargo, tiene como gran desventaja la falta de flexibilidad al cambio de clima, ya que en el caso de la lluvia, se ve gravemente afectada la detección de pequeñas regiones en movimiento y consecuentemente el sistema describe posiciones erróneas al agrupar de manera desacertada regiones ruidosas.

Por otro lado, Koller *et al [33]* utiliza un análisis de píxeles utilizando una función discreta para determinar cuáles píxeles han cambiado de valor de intensidad. Se define entonces un perímetro preliminar, definiendo los bordes de tal manera que sean totalmente paralelos y para el caso de la anchura, totalmente ortogonal, con respecto al eje que definen los píxeles detectados como en movimiento. Una vez realizado esto, se procede a una comparación entre modelos preestablecidos y previamente adquiridos (en frames anteriores), para establecer a que objeto previo le pertenece este modelo. Es entonces que se define una aparición de un objeto en movimiento previamente descrito o de un nuevo objeto en el frame en ese momento analizado. Este método, basado en plantillas adaptativas, tiene la ventaja de un bajo costo computacional y relativa fácil implementación. Sin embargo, tiene como punto débil la rigidez al momento de detectar

qué píxeles se mueven. Un ejemplo de esto es cuando un auto cruza atrás de un poste. En este caso el sistema detectaría dos partes en movimiento cuando solo hay uno, ya que no ofrece un método para el agrupamiento apropiado de píxeles en movimiento.

El modelo Gaussiano de actualización de background consiste en un algoritmo de actualización inteligente de la imagen utilizada como fondo en el seguimiento de un vehículo en un video.

Se basa en una mezcla de parámetros estadísticos que pueden ser configurados con diferentes valores dependiendo de la situación, que según determinados cambios o transcurso de tiempo, indican el cambio de los píxeles que se utilizan como fondo. Es entonces, en resumen, una herramienta que utiliza una mezcla de variables con distribución Gaussiana para la aplicación del filtro de actualización.

A continuación describiremos el algoritmo del modelo Gaussiano simple (SGM) propuesto por Wren *et al [14]*. En este método, la intensidad y el color de cada pixel es representado por un vector [Y,U,V] *T*. Se asume que solo existen cambios lentos en las escenas. La media y la covarianza de cada pixel A puede ser recursivamente actualizado de esta forma:

8)

$$\mu^{t}(\varphi) \quad = (1-\alpha)\mu^{t-1}(\varphi) + \alpha I^{t}(\varphi)$$

9)

$$\mathrm{U}^{\mathrm{t}}(\varphi) \quad = (1-\alpha)U^{t-1}(\varphi) + \alpha v \quad (\varphi)v(\varphi)^{T}$$

Donde I^{t} (□) es el pixel del frame actual en un espacio de color Y-UV y α es la tasa de aprendizaje.

10)

$$\mathrm{l}(\varphi) = -\frac{1}{2}v(\varphi)^{T}(U^{t})^{-1}v(\varphi) - \frac{1}{2}\ln|\mathrm{U}^{\mathrm{t}}| - \frac{3}{2}\ln 2\pi$$

Después de una adaptación de fondo, computamos todas las posiciones de las imágenes □ y la diferencia entre la imagen actual y el fondo. Este valor da origen a una clasificación individual de píxeles como fondo o foreground.

Un pixel α es clasificado como foreground si $\mathrm{l}(\varphi) < n_{c}$. En otro caso es parte del fondo. Después de esto se pasa a detectar los puntos del foreground.

Por otro lado, el modelo Gaussiano múltiple utiliza combinaciones de modelos Gaussianos de fondo, los cuales han sido una opción muy popular para modelar fondos complejos y variantes en el tiempo. En este algoritmo el procesamiento de píxeles es

considerado una serie de tiempo de vectores para imágenes de color. La historia de un pixel α en particular está dado por

$$\{X_1, \ldots X_t\} = \{I(\alpha, i) : 1 \leq i \leq t\}$$

Donde I es la secuencia de imagenes. El algoritmo modela la historia reciente de cada pixel como una mezcla de K distribuciones Gaussianas. De la misma forma, la probabilidad de observar el valor actual del pixel es:

11)

$$\mathrm{P}(\mathrm{X_t}) = \sum_{i=1}^{K} w_{i,t} * \acute{\eta}(X_t, \mu_{i,t}, U_{i,t}) \ldots$$

Donde K es el número de distribuciones, $w_{i,t}$ es el peso estimado del i-ésimo Gaussiano en la mezcla en un tiempo t y $\mathrm{U}_{i,t}$ y $\mu_{i,t}$son la media y la covarianza de la matriz del i-ésimo Gaussiano en un tiempo t y n la función de densidad de la probabilidad Gaussiana.

12)

$$\acute{\eta}(X_t, \mu, U) = \frac{1}{(2\pi)^{\frac{n}{2}} |U|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu)^T U^{-1}(X_t-\mu)}$$

El algoritmo asume que el rojo, azul y verde son canales independientes y que cada pixel no es estacionario. Estos supuestos dan como resultado K medias de aproximación en tiempo real para el algoritmo de mezcla. En la aproximación en tiempo real, cada nuevo pixel X_t es chequeado con un K ya existente de la distribución Gaussiana. Un valor de respuesta es encontrado si el valor del pixel está dentro de la desviación estándar de una distribución.

Es en sí, un threshold bastante efectivo para cada pixel y puede ser utilizado para modelar regiones que presentan cambios periódicos en las condiciones de alumbrado. Si el pixel actual no coincide con las distribuciones, se reemplaza a la distribución menos probable con este valor, agregándole una alta varianza y un peso de prioridad bajo. Este algoritmo es sumamente eficiente y a la vez adaptable a una gran variedad de situaciones. La sensibilidad a los cambios de brillo es aún la desventaja más prominente. Sin embargo, gracias al procesamiento interno del algoritmo, aparece en un menor grado que en otras técnicas similares.

McIvor *et al [31]* explica el uso de una mezcla de distribuciones adaptativas Gaussianas para la detección de un objeto en movimiento. Se definen un conjunto de parámetros para establecer una distribución Gaussiana que definirá de manera adaptativa una imagen de fondo. Esta mezcla detecta cada uno de los píxeles en movimiento y a cada uno se le aplica una distribución Gaussiana, frame a frame. Se la compara entonces a un fondo

predefinido, considerando cambios de iluminación, sombras y otros causados por diferentes tipos de movimiento. Este método es en sí un método de detección de objetos en movimiento que toma en consideración los cambios de brillo. Solamente se ve sobrepasado en eficacia y exactitud por el flujo óptico, pero este método es demasiado costoso computacionalmente. Tiene como desventaja que depende de los parámetros de inicialización, donde dado el caso de una mala definición de este conjunto datos de entrada, se obtiene un resultado totalmente inútil e inaceptable.

Stauffer y Grimson *et al [32],* desarrollaron un modelo de actualización de fondo adaptivo basado en mezclas Gaussianas, donde la detección y el seguimiento de objetos en movimiento están estrechamente relacionadas. Ellos indican que en el pasado, las barreras computacionales habían limitado la complejidad de las aplicaciones de procesamiento de imágenes en tiempo real. Como consecuencia de esto, la mayoría de los sistemas eran demasiado lentos por ser prácticos, o tenían éxito si se restringían a situaciones muy controladas. En lo que se refiere a vigilancia y monitoreo por video, un sistema robusto no debería depende de cámaras cuidadosamente ubicadas. Deben también considerarse los objetos en su campo visual, y efectos en la iluminación. A la vez deben también considerarse objetos que se sobreponen, sombras, y efectos causados por diferentes movimientos. Acercamientos típicos basados en substracción de fondos fallan en estas situaciones. El objetivo es crear un sistema robusto y adaptativo que sea lo

suficientemente flexible para soportar cambios en la iluminación, y otros cambios arbitrarios.

El acercamiento varía un poco de la implementación general del modelo de fondo adaptivo Gaussiano, y en lugar de modelar explícitamente los valores de los píxeles como un tipo particular de distribución, simplemente modelan los valores de un píxel en particular como una mezcla de distribuciones Gaussianas. Basado en la persistencia y la varianza de cada una de las distribuciones Gaussianas de la mezcla, determinaron cual distribución Gaussiana podía corresponder a los colores del fondo. Los valores de píxel que no encajan con las distribuciones del fondo son considerados como foreground hasta que haya alguna distribución Gaussiana que los incluya con suficiente evidencia consistente apoyándola.

El método consiste en que si cada píxel resulta de una superficie particular bajo una iluminación particular, una sola distribución Gaussiana sería suficiente para modelar el valor del píxel teniendo en cuenta el ruido de adquisición. Si solo la iluminación cambiase a través del tiempo, una sola distribución adaptativa Gaussiana por píxel sería suficiente. En la práctica, múltiples superficies usualmente aparecen en el volumen de visualización de una cámara en un punto determinado, desde la vista de un píxel en particular y si las condiciones de iluminación cambian. Por lo tanto, múltiples

distribuciones adaptivas Gaussianas son necesarias. Se usa una mezcla de Gaussianas adaptivas para aproximar este proceso.

Cada vez que los parámetros de las distribuciones Gaussianas son actualizados, las distribuciones son evaluadas usando una heurística simple para generar una hipótesis acerca de cuáles tienen mayor probabilidad de ser parte del "proceso de fondo". Los valores de píxel que no corresponden con una de las distribuciones del "fondo" del píxel son agrupados usando componentes conectados. Finalmente, los componentes conectados son detectados cuadro a cuadro usando un detector de múltiples hipótesis.

La estimación del modelo de fondo fue realizada de la siguiente forma. Debido a que los parámetros de la mezcla del modelo de cada píxel cambian, es aconsejable determinar cuáles de las distribuciones Gaussianas de la mezcla tienen mayores probabilidades de ser producidas por procesos de fondo. Heurísticamente, el interés está en las distribuciones Gaussianas que tienen la mejor evidencia, es decir, la que mejor apoyo tiene, y que tenga el menor valor de varianza.

En el análisis de los resultados, se destaca que en una SGI O2 con un procesador R1000, este método puede procesar de 11 a 13 imágenes por segundo, con imágenes de una resolución de 160x120 píxeles. La variación en la tasa de imágenes se da por la variación de la cantidad de foreground presente, es decir, la cantidad de movimiento. Este sistema de detección ha estado guardando efectivamente información de detección para cinco

escenas durante más de 16 meses. Por otro lado, cambios bruscos, como en cobertura del frame por nubes pueden necesitar un nuevo conjunto de distribuciones de fondo, donde e; sistema se estabilizará dentro de 10 a 20 segundos y la detección continuará inafectada.

Se ha mostrado entonces un novedoso método probabilístico para la substracción de movimiento. Implica modelar cada píxel como un modelo de mezcla separado. De esta forma se aísla cada región hasta su más mínima expresión, el pixel. Se trata de una verdadera implementación del dicho "divide y conquista". Se implementó un sistema en tiempo real estable y robusto.

Este método trata con cambios lentos de iluminación adaptando lentamente los valores de las Gaussianas. También trata con distribuciones multimodales causadas por sombras, ramas de árboles, y otras similares. Se recupera rápidamente cuando el fondo reaparece y tiene un umbral automático. Todos estos factores han hecho a este detector una parte esencial de esta actividad y de la investigación de clasificación de objetos.

Este es un sistema que ha sido utilizado exitosamente para detectar personas en ambientes interiores, personas y carros en ambientes exteriores. Todos los ambientes probados con distintas iluminaciones. El sistema consigue todas las metas de rendimiento en tiempo real durante periodos extendidos de tiempo sin la intervención humana.

La única desventaja está en la definición de los parámetros de inicialización de la mezcla, que podrían necesitar ajustes, pero aparte de eso, con buenos parámetros, el algoritmo

proporciona los resultados deseados. Está claro que son más ventajas que desventajas que presenta este método.

Para la segmentación, parte fundamental en el seguimiento de objetos en movimiento, se estudiaron algunas técnicas muy populares y eficientes. De entre todos estos alcances y técnicas, podemos destacar el de Beymer *et al [28]* y el de Lucas-Kanade *et al [29]*. *Este* último define unas características de interés sobre el objeto en movimiento y sigue a estas características durante toda la secuencia de las imágenes. Luego, las características son agrupadas utilizando pistas de su movimiento para segmentar a los objetos. La distancia entre pares de características, como también sus velocidades, es calculada desde una coordenada fija global utilizando homografía simple entre lo fijo y el plano de imagen. Debido a esta naturaleza, es únicamente aplicable a situaciones de un alto ángulo. Es muy ventajoso utilizar esta aproximación, ya que al estar en una posición un tanto omnisciente, se obtiene un gran detalle al estar la capacidad de percepción aumentada al máximo. La mayor desventaja consiste en encontrar un lugar apropiado para la colocación de la cámara y el costo de mantenimiento al sistema, al verse localizado a una altura considerable.

Para nuestro conocimiento, la única técnica de ángulo bajo es la de Kamijo *et al [30]*. En su trabajo, la imagen está dividida en bloques de 8x8 píxeles y un campo espacio termporal aleatorio de Markov es utilizado para actualizar un vehículo, utilizando la imagen previa y actual. Tiene la obvia ventaja del bajo ángulo, lo que hace a su

mantenimiento y corrección rápido y barato. Sin embargo esto también le trae desventajas al dificultársele la tarea de extracción de características, ya que en muchos casos, no recepta todos los atributos presentes en los vehículos.

En nuestro caso, la segmentación se debe enfocar en extraer conjuntos de píxeles que formen masas que potencialmente sean vehículos en la imagen de movimiento. Estas masas obtenidas se las conoce en el campo de la visión por computadora como blobs. Por definición, un blob, es un conjunto de píxeles juntos, que se tratan como un solo objeto.

Boreczky *et al [34]* mencionan el uso de modelos de Markov escondidos para la apropiada segmentación. El video es segmentado en regiones definidas por tomar, limites de las tomas y los movimiento de cámara dentro de las tomas. Los parámetros para la segmentación incluyen una distancia basada en imágenes entre frames de video adyacentes, una distancia basada en audio mediante la diferencia acústica de los intervalos justo y después de cada frame y una estimación de desplazamiento entre los dos frames. Este algoritmo clasifica las regiones computando la diferencia de las imágenes y el fondo. Los típicos algoritmos de segmentacion separan las diferencias mencionadas del proceso de segmentación. Este trabajo permite incluir esta informacion dentro de cada modelo de Markov escondido. Una ventaja de esta técnica es que no requiere la definición de un threshold. Sin embargo, para implementar la obtención de una distancia acústica, es necesario realizar un proceso previo de análisis de señales de sonido para su consecuente comparación y obtención de la diferencia. Este

procesamiento extra retrasa de manera notable el tiempo de respuesta del sistema, por lo que hasta ahora solo ha podido aplicarse en casos bastante simples.

2.3.1 ALGORITMOS DE PREDICCIÓN DE POSICIONES

En nuestro estudio bibliográfico hemos observado que existen métodos de detección y seguimiento de objetivos en que incluyen el uso de algoritmos de predicción de posición para brindar un grado mayor de robustez a su sistema. El objetivo de introducir un módulo de predicción de movimiento es el de ofrecerle una posible respuesta a la incógnita de la posición real del objeto en seguimiento en casos de que el seguimiento de los objetos se haya visto corrompido o sea imposible obtener un resultado coherente. Es decir, esta implementación adicional toma parte en el sistema al predecir mediante la información previamente obtenida, una posible ubicación del objeto bajo seguimiento en un punto futuro.

Los filtros de predicción de estados posteriores de un objeto afectado por un sistema han jugado un rol importantísimo en el avance de los productos y servicios que ahora disfrutamos en muchos aspectos de nuestras vidas. Gran parte del esfuerzo de desarrollo de estos métodos ha sido volcado a la parte frecuencial de la señal generada, llevándose la transformada de Fourier una parte estelar en esta área de desarrollo. Sin embargo,

recientemente se ha invertido más atención en el estudio de los filtros que trabajan en el campo espacial, usando métodos probabilísticos para sus cálculos y aproximaciones.

A pesar de que las implementaciones y algoritmos empleados para la detección de movimiento tienen un porcentaje de error bastante reducido, se da el caso en que el sistema es incapaz de detectar el objeto en movimiento y por ende cercarlo dentro de una caja englobante. Este problema genera entonces un desfase en la recolección de datos, lo que provoca incongruencia con la información generada y dificulta su posterior análisis.
Independientemente del tipo de cámaras empleadas o el desarrollo ingenieril adaptado, los errores de este tipo pueden ocurrir y por lo tanto es necesario añadir módulos de soporte y manejo de casos especiales, para de esta forma agregar robustez al sistema y brindar confiabilidad en los resultados que éste devuelve.
Dentro de los filtros de predicción espaciales, los filtros digitales y los adaptativos son los que más destacan. Estos algoritmos son importantes debido a su velocidad de cálculo y su buen rendimiento. La idea básica que hay detrás de la codificación de predicción adaptativa (APC) es que una muestra de la señal o instancia del estado de un objeto se puede aproximar como una combinación lineal de las pasadas p muestras anteriores. Minimizando el cuadrado de la diferencia entre las muestras actuales de la señal o estado y las predichas linealmente, se pueden determinar los coeficientes del predictor. Es decir, los coeficientes de peso de la combinación lineal.

El hecho de que estos filtros no sean invariantes temporales y que tampoco sean lineales hace que su estudio sea más complejo que el de un filtro digital, ya que no se pueden aplicar, salvo en un par de excepciones, las transformaciones en frecuencia, dominio Z, etc.

Las causas más comunes por las que ocurren errores en los sistemas de seguimiento y que pueden ser solucionados usando filtros espaciales son:

- Movimiento constante repetitivo (jittering)
- Bajo contraste en los valores de los píxeles vecinos (bordes)

El uso de filtros espaciales de predicción ha sido la solución más adecuada para responder al pedido demandado por el sistema. Existe una amplia gama de procesadores de información de éste tipo, mas es necesario escoger los adecuados filtros para obtener el máximo provecho de los recursos disponibles, teniendo en cuenta el factor de la velocidad de cálculo y la exactitud de los resultados retornados como índices sujetos a calificación para su selección. Dentro de las más prominentes técnicas de predicción encontramos a:

El observador de Luenberger, mencionado por Bundhwar *et al [5]*, creado por David Luenberger, un profesor de Stanford, que desarrolló el algoritmo que lleva su nombre para su tesis doctoral en 1963.

El observador de Luenberger “clona” el sistema evaluado, creando otro sistema, el cual es conocido, e introduciendo las mismas entradas a ambos. A partir de la salida generada por el sistema en evaluación, el algoritmo modifica la salida obtenida por el clon, adaptando este último resultado al primero, lo cual generalmente es la salida del clon más la diferencia entre la salida de los dos sistemas multiplicada por una constante. El objetivo final, es poder modelar un sistema desconocido a partir de uno conocido y así definir un comportamiento esperado para el sistema en estudio.

Este método puede presentar resultados incongruentes en sus inicios, ya que el ruido no estimado que afecta al sistema puede distorsionar increíblemente su siguiente predicción.

El algoritmo LMS o least mean square, mencionado también por Bundhwar *et al [5]*, donde este algoritmo permite obtener el valor esperado mínimo del cuadrado de la señal de error, consiste en dos partes:

- Proceso de filtrado, donde se obtiene la medición real del sistema y se calcula mediante un filtro lineal una posible salida. Posteriormente se procede a estimar el error comparando la salida real con la generada por el filtro lineal
- Proceso de ajuste, donde se adapta al filtro de acuerdo a los datos generados por la estimación realizada en el proceso de filtrado.

Este método es muy útil mientras se defina un filtro lineal que brinde resultados parecidos a los del sistema en análisis. En caso contrario, los ajustes efectuados serían demasiado grandes y afectarían al método de tal forma que la probabilidad de convergencia con resultados reales estaría muy por debajo de lo necesario para brindar aproximaciones útiles. Como conclusión se obtiene que mientras más datos previos reales puedan ser capturados, mejor será el resultado de la aproximación. Tiene la desventaja de no poseer filtro alguno o capacidad de adaptación para las entradas con ruido.

La red neuronal artificial, propuesta por Pollock *et al [11]* es una red de procesamiento automático que colabora para ofrecer un resultado. Está inspirada en cómo funciona el cerebro humano, el cual tiene neuronas interconectadas. Una fortaleza de este tipo de sistemas es que al parecerse su procesamiento al de un cerebro humano, brinda respuestas generales y robustas. La Figura 2 muestra un modelo general de un positrón.

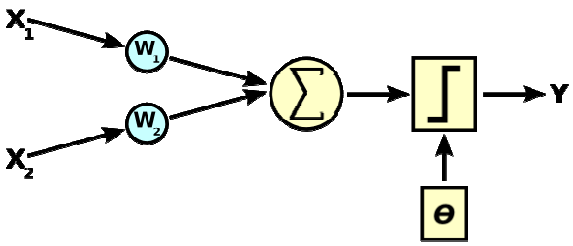

Figura 2 Modelo de un positrón básico

Para el caso de estimación de un nuevo estado para un sistema dado, el sistema necesita de una función de propagación, una de activación y finalmente una de transferencia. Al no poder garantizar siempre una entrada válida, debido al problema de desenfoque del área de estudio en el video, y por la alta disponibilidad de recursos que se necesitan para implementar este tipo de métodos, elevando así el costo del sistema, se descarta el modelo, dejando su desarrollo a aplicaciones futuras.

Un algoritmo de prediccion basado en la diferencia de las posiciones de dos puntos previamente obtenidos de manera satisfactoria es el del desplazamiento previo, propuesto por Efros *et al [48]*.Consiste en realizar mediciones constantes de la variación entre los estados previos de un objeto sujeto a un sistema no conocido, para predecir un posible siguiente estado. Posee una ventaja muy apreciable al no necesitar que el sistema que está afectando al objeto sea definido determinística o abstractamente. La Figura 3 muestra un diagrama del funcionamiento de este algoritmo.

.

MODELAMIENTO DEL ALGORITMO DEL FILTRO DE DESPLAZAMIENTO PREVIO

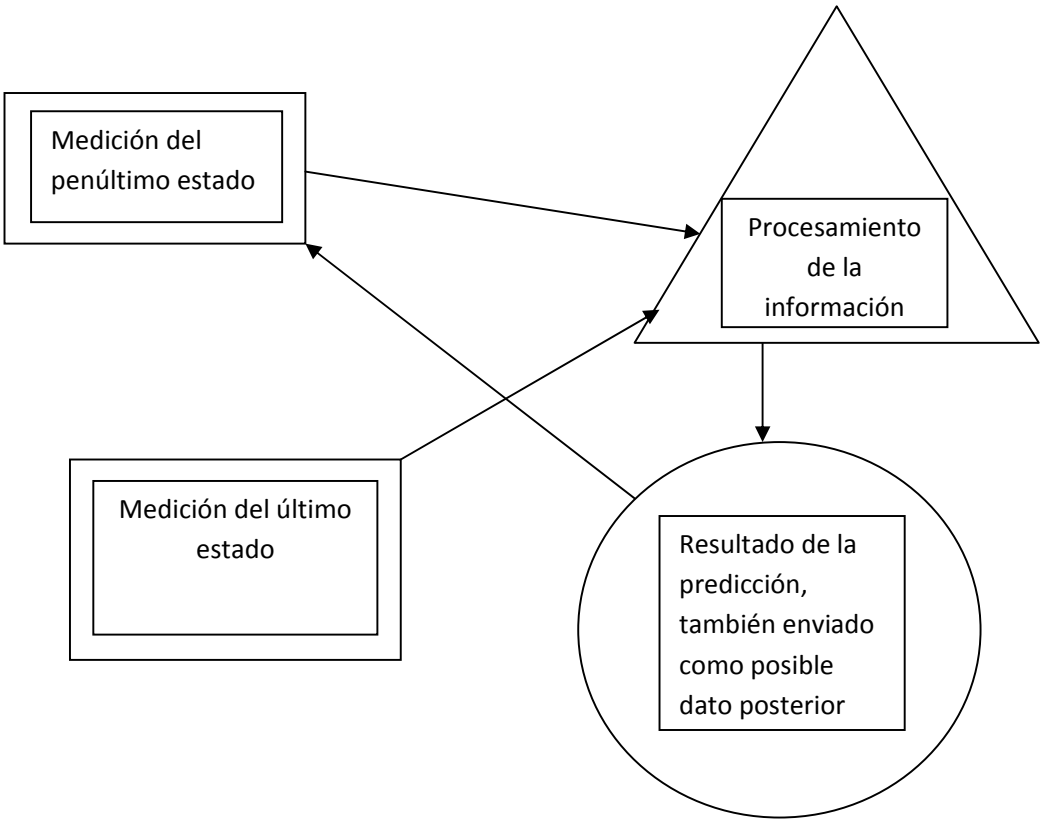

Figura 3 Modelo del filtro de desplazamiento previo

Se expuso previamente su independencia del sistema que está procesando, al concentrarse exclusivamente en los resultados provenientes de estados previos a la predicción. Esta ventaja es muy apreciada para procesamiento real de sistemas de desplazamiento, ya que es sumamente complicado definir un sistema que describa desplazamientos reales de forma eficiente, precisa y robusta.

Por otro lado, la estrategia del algoritmo y métodos matemáticos empleados en el algoritmo de este filtro son sumamente simples y reducidos. Esto hace que su costo computacional sea sumamente atrayente para el investigador.

Sin embargo, por la misma naturaleza del algoritmo que implementa, este filtro posee una importante falla. Esta técnica siempre depende de la veracidad y constancia de los datos proporcionados por los estados.

En caso de que la información recogida por los sensores o medidores de un procesador este afectada por ruido o que en casos aún más extremos, no se provea de nueva información, el filtro de desplazamientos previos arrojará predicciones equivocadas, al tender por naturaleza a promediar los desplazamientos entre un estado y otro, evento que en la vida real es casi imposible que ocurra.

El filtro Kalman, propuesto por Kalman *et al [8]*, es un método de estimación cuyos parámetros se corrigen en cada iteración dependiendo del error de predicción que se haya cometido en la iteración anterior. Es un estimador lineal y óptimo desde el punto de vista de mínimos cuadrados, que ha ganado aceptación en el análisis de series de tiempo.

Rudolf Emil Kalman es un científico húngaro-americano, nacido en Budapest en 1930. La mayor parte de sus investigaciones las hizo en Columbia University y Massachusetts Institute of Technology.

El filtro de Kalman es un conjunto de ecuaciones matemáticas que proveen una solución recursiva eficiente del método de mínimos cuadrados. Esta solución permite calcular un estimador lineal, insesgado y óptimo del estado de un proceso en cada momento del tiempo con base en la información disponible en el momento t-1, y actualizar, con la información adicional disponible en el momento t, dichas estimaciones. Este filtro es el principal algoritmo para estimar sistemas dinámicos especificados en la forma de estado-espacio (State-space).

El filtro tiene su origen en el documento de Kalman (1960) donde describe una solución recursiva para el problema del filtrado lineal de datos discretos. La derivación de Kalman fue dentro de un amplio contexto de modelos estado-espacio, en donde el núcleo es la estimación por medio de mínimos cuadrados recursivos. Desde ese momento, debido en gran parte al avance en el cálculo digital, el filtro de Kalman ha sido objeto de una

extensiva investigación y aplicación, particularmente en el área de la navegación autónoma y asistida, en rastreo de misiles y en economía.

La representación estado-espacio es esencialmente una notación conveniente para la estimación de modelos estocásticos donde se asumen errores en la medición del sistema, lo que permite abordar el manejo de un amplio rango de modelos de series de tiempo.

El filtro es un procedimiento matemático que opera por medio de un mecanismo de predicción y corrección. En esencia este algoritmo pronostica el nuevo estado[2] a partir de su estimación previa añadiendo un término de corrección proporcional al error de predicción, de tal forma que este último es minimizado estadísticamente.

Dentro de la notación estado-espacio, la derivación del filtro de Kalman descansa en el supuesto de normalidad del vector de estado inicial y de las perturbaciones del sistema. De tal forma que es posible calcular la función de verosimilitud sobre el error de predicción con lo cual se lleva a cabo la estimación de los parámetros no conocidos del sistema.

El procedimiento de estimación completo es el siguiente: el modelo es formulado en estado espacio y para un conjunto inicial de parámetros dados, los errores de predicción del modelo son generados por el filtro. Estos son utilizados para evaluar recursivamente la función de verosimilitud hasta maximizarla. La Figura 4 muestra un diagrama típico del acoplamiento del filtro de Kalman a un sistema.

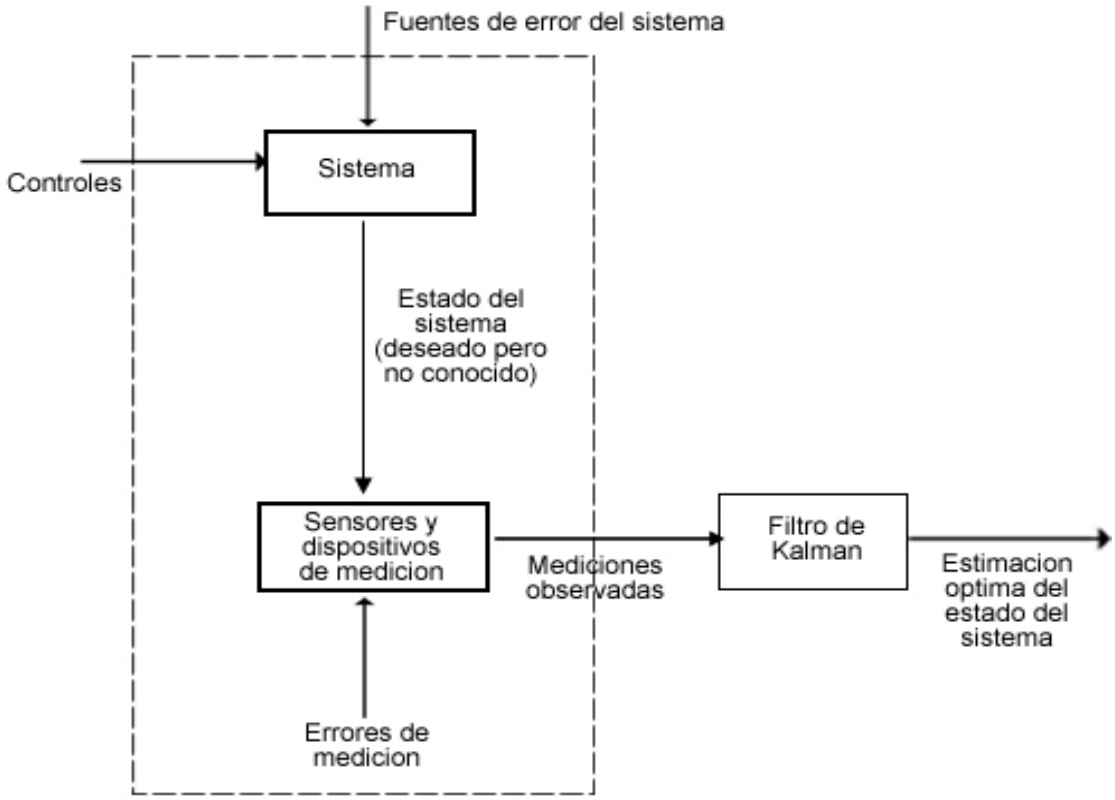

Figura 4 Filtro de Kalman típico

Una de las más grandes fortalezas del filtro de Kalman es que puede manejar los estados pasados, presentes y futuros aún cuando una precisa definición de la naturaleza del sistema modelado no está disponible. En la práctica, las variables estado individuales de

un sistema dinámico no pueden ser exactamente determinadas por una medición directa. Debido a esto, su medición se realiza por medio de procesos estocásticos que involucran algún grado de incertidumbre en la medición.

ESTIMACIÓN RECURSIVA Y EL FILTRO DE KALMAN[1]

El concepto de regresión de mínimos cuadrados se asocia por una parte con Legendre (1752-1833), quien fue el primero en publicar la teoría en 1805 y fue quien acuñó el término mínimos cuadrados7. Sin embargo, fue Gauss (1777-1855) quien desarrolló el método como un instrumento estadístico incorporando los mínimos cuadrados en un contexto en el que se da un tratamiento probabilístico a los errores de observación.

La primera exposición del método de mínimos cuadrados por Gauss, está ligada con la estimación de seis coeficientes para determinar la órbita elíptica de un cuerpo planetario, cuando las observaciones disponibles exceden el número de parámetros.

La segunda exposición fue presentada en una serie de documentos publicados en 1821, 1823 y 1826, recogidos bajo el título Theoria Combinationis Observationum Erroribus Minimis Obnoxiae (1823), donde presenta el famoso teorema que afirma que entre todos los estimadores lineales insesgados, los estimadores de mínimos cuadrados tienen el menor error cuadrático medio. Lo anterior es conocido como el teorema de Gauss-Markov.

La relevancia de la segunda exposición de Gauss para la teoría de estimación de mínimos cuadrados recursivos y para el concepto del filtro de Kalman se encuentra en un breve párrafo donde Gauss muestra que es posible detectar los cambios más probables de un

[1] Tomado de D.S.G. Pollock, The Kalman Filter.

evento desconocido cuando una nueva ecuación es incorporada y determinar los pesos de estas nuevas determinaciones. En efecto, Gauss desarrolló el algoritmo de estimación de mínimos cuadrados recursivos.

El algoritmo de Gauss fue ignorado por alrededor de siglo y medio antes de que fuera redescubierto en dos ocasiones separadas. El primer redescubridor fue Plackett en 1950, lo cual fue antes de la gran revolución observada en la computación y también pasó inadvertido. El segundo redescubridor del algoritmo recursivo fue R.E. Kalman\ en 1960 en el contexto de la teoría del control. A partir del documento de Kalman (1960) y Kalman y R. Bucy (1961), donde describen una solución recursiva para el problema de filtrado lineal de datos discretos, este algoritmo ha presentado una extensa investigación y aplicación.

La exposición de Plackett del algoritmo de mínimos cuadrados recursivos se desarrolla en un esquema algebraico que involucra solamente los conceptos estadísticos de los modelos de regresión lineal clásicos.

La derivación de Kalman fue dentro de un amplio contexto de modelo estado-espacio con parámetros que cambian en el tiempo. Así, el núcleo del filtro de Kalman es todavía el algoritmo de Gauss-Plackett de estimación de mínimos cuadrados recursivos, pero en un contexto donde la extensión y la complejidad del álgebra es más amplio. Kalman basó la construcción del filtro sobre la base de la teoría de probabilidad, y más específicamente, sobre las propiedades de condicionalidad Gaussiana de variables aleatorias. El criterio

que propuso fue minimizar la norma de la matriz de covarianza del vector estado, generando la clásica recursión: la estimación del nuevo estado es deducido desde la estimación previa añadiendo un término de corrección proporcional al error de predicción.

A partir del documento original de Kalman, muchas otras derivaciones han surgido. La mayoría de éstas intentan reducir la terminología a alguna cosa cercana a la teoría ordinaria de regresión de mínimos cuadrados. Otras han surgido desde una función de máxima verosimilitud o desde un punto de vista Bayesiano. Lo cierto es que el filtro de Kalman es un tema complejo. Su derivación, por cualquier método, es extensa y sus ecuaciones difíciles. No obstante, es esta complejidad lo que le da al filtro de Kalman su enorme poder, para resolver un amplio rango de problemas en inferencia estadística.

El filtro de Kalman tiene como objetivo resolver el problema general de estimar el estado $X \in \Re^n$ de un proceso controlado en tiempo discreto, el cual es dominado por una ecuación lineal en diferencia estocástica de la siguiente forma:

13)

$$X_t = AX_{t-1} + w_{t-1}$$

con una medida $Z \in \Re^m$, que sigue un modelo:

14)

$$Z_t = HX_t + v_t$$

Las variables aleatorias w_t y v_t representan el error del proceso y de la medida respectivamente.

Se asume que son independientes entre ellas, que son ruido blanco y con distribución de probabilidad normal:

$$p(w) \cong N(0,Q)$$

$$p(v) \cong N(0, R)$$

En la práctica las matrices de covarianza de la perturbación del proceso, Q, y de la perturbación de la medida, R, podrían cambiar en el tiempo, por simplicidad en general se asumen que son constantes.

La matriz A se asume de una dimensión n x n y relaciona el estado en el periodo previo t-1 con el estado en el momento t. La matriz H de dimensión m x n relaciona el estado con la medición Z_t. Estas matrices pueden cambiar en el tiempo, pero en general se asumen como constantes. La Figura 5 explica el funcionamiento del filtro de Kalman.

Esquema extendido del funcionamiento del filtro de Kalman

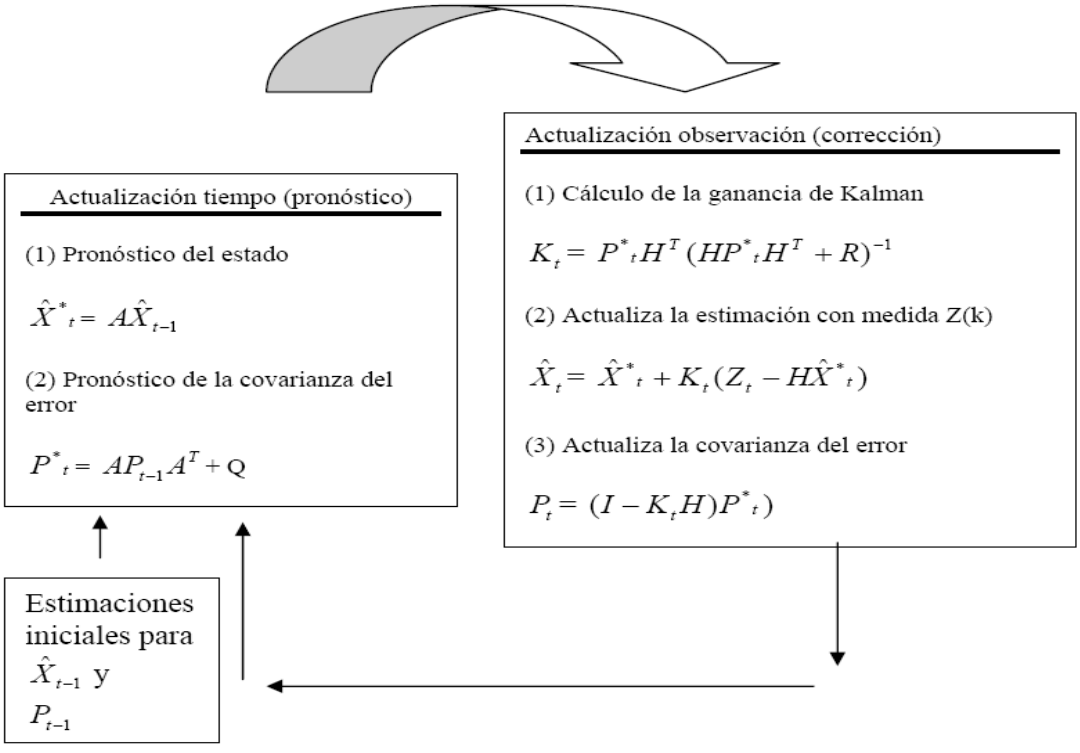

Figura 5 Gráfico explicativo del proceso a seguir en el filtro de Kalman

El filtro de Kalman, al poseer un diseño algorítmico altamente adaptativo, brinda una alta robustez para una amplia gama de casos y circunstancias. El filtro utiliza las mediciones tomadas continuamente en cada estado, desde el sistema real, para poder corregir errores generados en predicciones previas y así acercarse más a una predicción que sea similar al estado real.

El ruido externo es controlado de forma inteligente por esta técnica al reducir la participación de éste, utilizando mínimos cuadrados. Esto brinda una gran ventaja para la investigación, ya que a menudo se invierten muchos recursos y esfuerzos en la eliminación o reducción del ruido.

Una desventaja de este sistema es su mal comportamiento si es que un ruido no detectado llega a introducirse en las medidas. El filtro de Kalman adoptará el estado afectado por el ruido como correcto y a partir de esto, las consecuentes predicciones no podrán recuperarse, a menos que otro estado, afectado por un ruido exactamente inverso al anterior sea introducido, adaptándose de esta forma otra vez al moldeamiento real de la salida del sistema.

2.4 CONCLUSIONES DE TRABAJOS RELACIONADOS

Tomando en cuenta que en nuestro proyecto los objetos en movimiento van a estar representados por vehículos, y que a partir de la información que se obtenga de los módulos de detección y seguimiento se modelará el comportamiento y la trayectoria de dichos vehículos, es necesario considerar las ventajas ofrecidas por las técnicas presentadas previamente para definir la solución más conveniente para nuestro problema.

Para el módulo de detección de vehículos en una secuencia de imágenes utilizaremos el método de *Mezcla Adaptativa de Distribuciones Gaussianas*. Este método, como se mencionó anteriormente, esta basado en substracción de imágenes pero la principal diferencia es que la imagen de background o fondo se adapta a pequeñas variaciones ocurridas durante la adquisición de las imágenes (tales cambios pueden ser debido a la variación de iluminación, viento y cambios afines). La mezcla adaptativa de distribuciones Gaussianas es un método sencillo y robusto, genera un modelo Gaussiano para la imagen de fondo, la imagen de fondo se actualiza o adapta a pequeñas variaciones ocurridas en la escena, las zonas de movimiento son los píxeles que no se asemejan al modelo, proporciona información más completa sobre las características de los datos en movimiento. Este método tiene como desventaja la dependencia de los parámetros de inicialización, lo cual no será problema mayor evitar ya que con el debido estudio e implementación de la solución adecuada para los problemas presentados se obtendrá sin lugar a dudas el adecuado grupo de variables necesarias.

Por otro lado, para el módulo de seguimiento de vehículos en movimiento proponemos utilizar los resultados generados por la detección mediante la mezcla adaptativa de distribuciones Gaussianas. Con estos resultados y empleando una librería de manejo de blobs se pueden obtener resultados excelentes para lo que respecta al seguimiento de automotores. El uso de los blobs nos permitirá segmentar la imagen de movimiento y clasificar a los objetos o vehículos moviéndose de otros posibles resultados que puedan representar ruido. Podemos ver claramente como este es un excelente método para sobrellevar los diferentes cambios de iluminación que se pueden dar a cualquier hora del día, y también algunos otros movimientos secundarios producidos por factores externos. Se observa también lo eficiente que ha sido en tiempo real, lo cual es buena noticia para nosotros, ya que se desarrollará un sistema que no trabaja en tiempo real. Con esto sabremos que no tendremos problemas de rendimiento, y que además, con una sola mezcla de modelos Gaussianos será suficiente.

Con los resultados obtenidos de los blobs es posible extraer, para los objetos o vehículos en movimiento que han sido generados, un conjunto de características básicas tales como: dirección de su movimiento, carril en el que se desplaza, tamaño de la caja englobante que lo contiene, entre otros.

Sin embargo, es necesario asegurarnos que el sistema retorne una respuesta coherente en todas las circunstancias posibles. Es por esto que decidimos incluir métodos adaptativos de predicción de posiciones para el adecuado seguimiento de los vehículos. Los filtros

escogidos a implementarse son el de la diferencia previa y el de Kalman. Ambos métodos probaron ser independientes del sistema que modelan y además tienen la ventaja de ser fácilmente adaptable a recibir el tipo información de entrada que obtenemos en módulos previos.

escogidos a implementarse son el de la diferencia previa y el de Kalman. Ambos métodos probaron ser independientes del sistema que modelan y además tienen la ventaja de ser fácilmente adaptable a recibir el tipo información de entrada que obtenemos en módulos previos.

3. GENERACIÓN DE IMÁGENES Y DETECCIÓN DE VEHICULOS

3.1 INTRODUCCIÓN

La detección de vehículos es parte fundamental del sistema desarrollado en nuestro proyecto. Cabe recalcar que el procesamiento digital de imágenes se hace valer de un sinnúmero de operaciones que se aplican a nivel de píxeles, para los diversos propósitos de la misma. Entre estas operaciones encontramos, entre las más básicas, a las operaciones aritméticas y morfológicas.

La resta es uno de los cuatro operadores aritméticos esenciales de prepocesamiento de imágenes. Generalmente las aplicaciones de la resta de imágenes incluyen el desarrollo de sistemas para detectar movimientos en una secuencia de imágenes, obviamente los sistemas avanzados de detección de movimiento incluyen técnicas más complejas que complementan a este operador para resolver entre otros problemas los problemas derivados de la aplicación de la resta. Un ejemplo que ilustra el uso del operador de resta es mostrado en la Figura 6, en donde la imagen de la izquierda contiene información de una escena sin objetos en movimiento (llamada imagen de fondo o background), la imagen del centro contiene a dos vehículos moviéndose (escena a analizar), y finalmente la imagen de la derecha contiene la información de los objetos que han sido detectados en movimiento (llamada imagen de movimiento o foreground).

Figura 6 (izquierda) Escena original sin objetos en movimiento (imagen de fondo o background), (centro) escena con vehículos en movimiento, (derecha) imagen resultantes con la detección de los objetos en movimiento (imagen de movimiento o foreground).

Por otro lado, las operaciones morfológicas son fundamentalmente dos, a partir de las cuales el resto son elaboradas. La primera es la *erosión*, la cual tiene como propósito disminuir el área de píxeles blancos en una imagen binaria. En segundo lugar está la *dilatación*, que se encarga de lo contrario, es decir, de aumentar el área de píxeles blancos. Un ejemplo del comportamiento de estos operadores es mostrado en la Figura *7*.

Figura 7 (izquierda) Imagen binaria original, (centro) imagen erosionada, (derecha) imagen dilatada

A partir de estas dos operaciones, nacen dos operaciones morfológicas más, que son básicamente la combinación de las anteriores. Tenemos el *operador de cierre*, que no es más que una dilatación seguida de una erosión, y además tenemos el *operador de apertura*, que es una erosión seguida de una dilatación.

Las operaciones anteriormente mencionadas entran en el grupo de las operaciones y filtros espaciales. Existen también operaciones y filtros frecuenciales que son aplicados a imágenes que se encuentran en el dominio frecuencial, las mismas que han sido obtenidas previamente mediante la aplicación de transformadas discretas como por ejemplo la transformada de Fourier.

En nuestro caso solo nos enfocaremos en las operaciones sobre imágenes en el dominio espacial. A partir de las operaciones antes mencionadas se puede conseguir una técnica básica para detectar objetos en movimiento desde una señal de video. Más adelante serán explicadas en detalle las diferentes técnicas que se pueden utilizar, su evolución, y cuál es la que aplicamos en nuestro sistema.

Para el desarrollo del sistema nos valemos de la librería gráfica MFC del framework de desarrollo de Windows, *WinAPI*. Toda la interfaz gráfica, denominada *ImageCapture*, es desarrollada con funciones y métodos de MFC, con los que tenemos la posibilidad de crear ventanas, en las cuáles podemos visualizar los videos y el procesamiento realizado por diferentes algoritmos implementados.

Para realizar la captura en sí, se necesitan funciones de acceso a la cámara y de obtención de frames. Para esto utilizamos la librería de visión por computadora de código abierto, *OpenCV*.

OpenCV (Open source computer vision) es una biblioteca libre de visión artificial por computador originalmente desarrollada por Intel y publicada bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas, en el 2002. Este marco de trabajo facilita en gran manera el aprendizaje e implementación de distintas técnicas de visión por computador, tanto a nivel docente como investigador, ya que aísla al desarrollador de las peculiaridades de los distintos sistemas de visión. La biblioteca es multiplataforma, y puede ser usada en Mac OS X, Windows y Linux. La *OpenCV* es usada ampliamente por los desarrolladores de aplicaciones de visión por computador y debido a esto tiene una comunidad numerosa de adeptos e investigadores.

Algunas de sus ventajas son:

- Robustez, ya que incluye muchas pruebas de integración y todo el código fuente está incluido para su libre modificación bajo una licencia GNU.
- Posee funciones para el manejo desde bajo nivel (matrices, imágenes, video, filtros, DFT, DCT, estructuras de datos, etc.) hasta alto nivel (detección de

esquinas, flujo óptico, detección de objetos, detección de rostros, obtención de contornos, correspondencia estéreo, etc).

- Partes de la librería están paralelizadas, lo que la hace sumamente eficiente en el uso de memoria y recursos del sistema.
- Incluye librerías para implementar filtros predictores de estados, clasificadores de Bayes, árboles de decisiones, redes neuronales (MLP), entre otros métodos sumamente conocidos y poderosos en inteligencia artificial y visión por computador.

Esta librería tiene innumerables recursos en relación a la visión por computador, y es, más que una librería, un framework, una forma de programación, que facilita al máximo la implementación de un sistema como el nuestro. El tipo de dato que utiliza *OpenCV* para una imagen es la *IplImage*.

Los campos fundamentales de la estructura de datos *IplImage* son:

img->depth	profundidad de los píxeles (IPL_DEPTH_8U, IPL_DEPTH_16S, IPL_DEPTH_32F, ...)
img->nChannels	número de canales de la imagen (normalmente 1 cuando es en escala de grises ó 3 con RBG)
img->width	ancho de la imagen (en píxeles)
img->height	alto de la imagen (en píxeles)

Nuestro método de desarrollo se basa en el procesamiento de imágenes consecuentes o videos. Primeramente obtenemos un frame del video y a partir de ahí procedemos a trabajar sobre esta imagen. El procesamiento con este resultado se lo explicará en detalle más adelante. Para el caso de que se maneje un video que se captura directamente desde una cámara, el procedimiento aplicado es el mismo que el descrito arriba. En este caso es ligeramente más complicado, pues en lugar de tener el video en su totalidad, tenemos una señal de video que se genera frame a frame desde la cámara y no se sabe a ciencia cierta el final. Se procede entonces a, dentro de las funciones que manejan los eventos de la cámara con sus distintos estados, llamar a un *callback* que establece los parámetros del frame y los procesa uno a uno para mostrarlos en la interfaz. Esto no es realmente necesario, pues se tienen funciones más básicas con las que se puede hacer exactamente lo mismo que se hace con un archivo de video, pero para esta situación tenemos otro objetivo, el cual es mostrar el video en una única ventana donde se visualicen 2 videos.

Además se manejan eventos tales como cuando la cámara empieza a grabar, cuando la cámara se encuentra grabando, y cuando ha terminado su grabación. Son varios parámetros que se manejan con la MFC, y entonces nos valemos de las mismas funciones para realizar ciertas llamadas dentro de ellas. Luego de realizar esto para un frame se repite el procedimiento una y otra vez hasta que se obtiene el frame actual del video, y se lo envía como parámetro a la función que llama al algoritmo seleccionado para comenzar el procesamiento.

En las siguientes secciones de este capítulo se describirán en detalle cómo han sido implementados el módulo de generación de las imágenes a procesar y el módulo de detección de vehículos en movimiento. El módulo de generación de las imágenes a procesar es el responsable de adquirir las imágenes que van a ser usadas como datos de entrada para el módulo de detección de movimiento, el cual será el encargado de localizar la presencia o no de vehículos moviéndose en dicha secuencia de imágenes.

3.2 GENERACION DE IMÁGENES A PROCESAR

El primer módulo de nuestro sistema es el módulo de generación de las imágenes a procesar. Este módulo tiene como objetivo principal la adquisición o generación de los datos de imágenes que serán usados para su posterior procesamiento y análisis en los consecuentes módulos de detección y seguimiento de vehículos. Se asume que las imágenes generadas desde este módulo contienen información de vehículos circulando por una determinada vía o carretera.

Dado que nuestro sistema no ha sido desarrollado para trabajar en tiempo real de manera óptima, usaremos como datos de entrada archivos de video que han sido previamente obtenidos desde una autopista o carretera. Nuestro sistema no contempla el montaje y calibración de las cámaras responsables de obtener los videos viales desde las autopistas o carreteras. Se asume que esta información ha sido generada previamente desde algún sistema de adquisición de videos viales.

De esta forma, para generar las imágenes a procesar se utiliza una interfaz gráfica que permite cargar un archivo de video ya existente. Esta interfaz además de permitir cargar archivos de video también permite cargar otros 2 tipos de entrada, los cuales son: una simple imagen o capturar una señal de video en vivo. Estos 2 últimos tipos de entrada podrían servir para que en un futuro se puedan realizar avances del sistema propuesto o nuevos proyectos puedan ser considerados y que utilicen las técnicas aquí desarrolladas.

Una vez que el archivo de video ha sido cargado a nuestro módulo, el siguiente paso consiste en capturar cada una de las imágenes o frames que constituyen dicho video. Para esto, se utiliza la función de OpenCV llamada *cvCaptureFromAVI*, la cual recibe como parámetro de entrada el archivo de video y retorna un tipo de dato *CvCapture*.

El *CvCapture* es, a breves rasgos, el video en formato de OpenCV preparado para ser procesado frame a frame. Luego usamos la función *cvQueryFrame* que permite obtener el frame actual, siendo para la primera llamada el primer frame. Esta función es autoincremental, es decir cada vez que se la llama captura el siguiente frame del video.

Esta función se maneja con un *CvCapture* de entrada, en este caso el indicado anteriormente, y retorna una imagen de tipo *IplImage*, que es el tipo de dato sobre el cual se trabajará. Como podemos notar el procesamiento será frame por frame, y no se trabaja sobre un video, sino sobre imágenes. Las Figuras 8 y 9 muestran cuatro imágenes en total que fueron adquiridas desde un archivo de video vial usando el procedimiento previamente explicado. Cada una de estas imágenes son los datos de entrada para el módulo de detección de movimiento en donde serán analizadas para determinar los objetos móviles.

Figura 8 Izq. Imagen correspondiente al frame 45; Der. Imagen correspondiente al frame 194

Figura 9 Izq. Imagen correspondiente al frame 100; Der. Imagen correspondiente al frame 154

Se debe hacer una importante acotación acerca de la generación de imágenes. Aunque nuestro sistema no se encarga de capturar la señal de video, es altamente recomendado que el sistema o los sistemas que se encarguen de hacerlo, consideren ciertas limitaciones con respecto a las imágenes a ser analizadas. Es decir, hay características para las cuales

nuestro prototipo funciona de manera óptima, en lo referente a la captura del video. No cualquier video puede ser correctamente procesado por este sistema. Para el detalle de las características de los videos, revisar en el capítulo 5 la tabla de restricciones y recomendaciones para un óptimo desempeño en la sección de pruebas de campo preliminares.

3.3 DETECCION DE VEHICULOS EN MOVIMIENTO

El módulo de detección de movimiento tiene como propósito único y exclusivo detectar de la mejor manera los objetos en movimiento dentro de una secuencia de imágenes. Una vez realizada la detección inicial de los objetos en movimiento, se obtiene además como resultado una gran cantidad de ruido lo cual está representado por píxeles aislados que no representan ninguna información, y que más bien dificulta a la posterior obtención de características y el seguimiento de los objetos.

El sistema desarrollado maneja la detección en dos partes necesarias para obtener las regiones en movimiento. Estas partes son:

- Generación de la imagen de fondo o background
- Generación de la imagen de movimiento o foreground

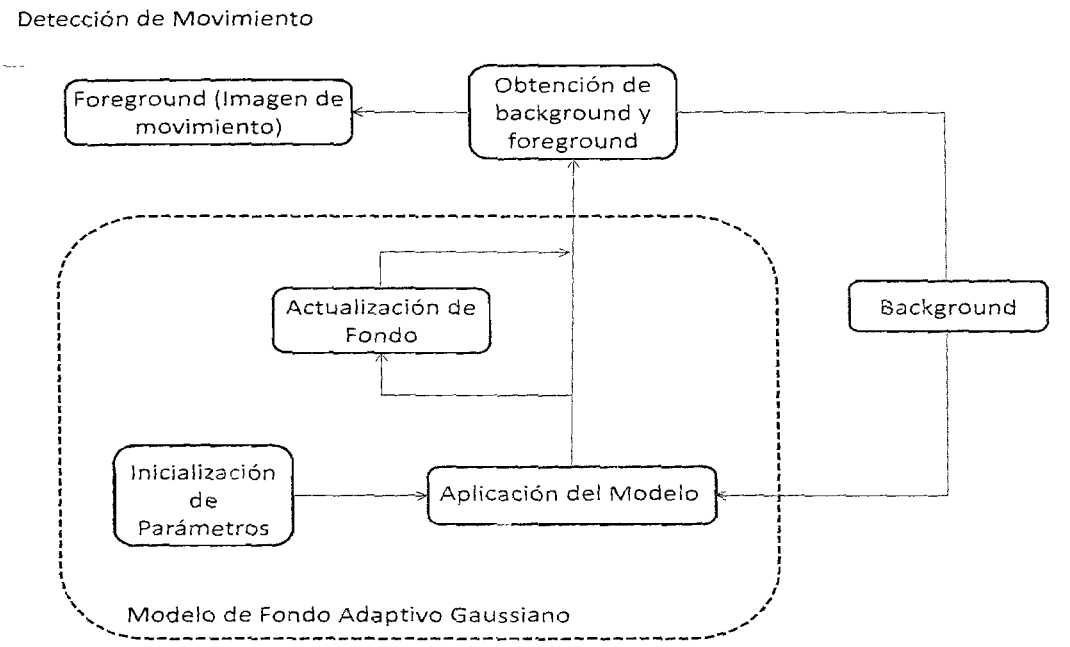

Figura 10 Diagrama de módulo de detección

Antes que realizar cualquier procesamiento, ya sea de detección o seguimiento, se toma a la imagen de entrada, que es resultado del módulo de generación de imágenes, descrito en la sección 3.2, y se la clona, es decir se guarda una copia exactamente igual. Para clonar una imagen se utiliza la función *cvCloneImage*, que recibe una imagen de entrada y devuelve un clon de la misma. De esta manera se trabaja sobre una copia, no con la imagen de entrada original, y es en este clon en donde se plasmarán los resultados.

Se procede entonces a comenzar con la detección de vehículos en movimiento. Para este procesamiento es necesario extraer al fondo o background y la imagen de movimiento o foreground. Para generar las imágenes de background y foreground tomamos ventaja de la

Se procede entonces a comenzar con la detección de vehículos en movimiento. Para este procesamiento es necesario extraer al fondo o background y la imagen de movimiento o foreground. Para generar las imágenes de background y foreground tomamos ventaja de la librería de OpenCV que nos ofrece una función para detectar movimiento en imágenes. Esta función se basa en el uso de un modelo Gaussiano para la detección del background y los objetos en movimiento, para mayor información de cómo trabaja revisar el trabajo de KaewTraKulPong *et al [39]*.

Para hacer uso de esta función inicialmente debemos crear el modelo Gaussiano. Para esto se utiliza la función *cvCreateGaussianBGModel* que recibe como datos de entrada a la primera imagen del video (esta imagen ha sido previamente generada desde la sección 3.2) y además un conjunto de parámetros encerrados dentro de una estructura los cuales sirven para definir el modelo Gausiano. Para la definición de los parámetros usamos un tipo de dato *CvGaussBGStatModelParams*. Los parámetros pueden definirse por default (utilizando valores preestablecidos en OpenCV) o pueden precisarse manualmente. En nuestro caso, los parámetros fueron definidos de manera manual. Se definieron los campos de los parámetros con los siguientes valores:

- win_size (tasa de aprendizaje) =2;
- n_gauss (número de distribuciones Gaussianas en la mezcla) =5;
- bg_threshold (suma de los pesos de los umbrales para la prueba del background o fondo) =0.7;

- std_threshold (suma de los pesos de los umbrales de las desviaciones standard) = 3.5;
- minArea (area mínima de variación para detección de movimiento) =15;
- weight_init (peso inicial) =0.05;
- variance_init (varianza inicial) =30;

Teniendo ya, tanto la primera imagen como el conjunto de parámetros, se procede a crear el modelo Gaussiano con la función *cvCreateGaussianBGModel,* arriba mencionada. En nuestra implementacion, el modelo Gaussiano recibió el nombre de *bgModel*.

El procedimiento de definición del modelo Gaussiano se realiza una sola vez, con la captura del primer frame. Una vez que el modelo Gaussiano, *bgModel*, ha sido creado como se indica arriba se empieza a procesar los siguientes frames del video, con el propósito de detectar las imágenes de background y foreground.

3.3.1 GENERACION DE IMAGEN DE FONDO O BACKGROUND

Para generar una imagen de fondo se deben seguir algunos pasos de manera ordenada en cada frame. Este procedimiento se realizará con todos y cada uno de los frames sin excepción. En primer lugar, se realiza una actualización del modelo Gaussiano con la función *cvUpdateBGStatModel*, que recibe como datos de entrada a la imagen actual y al

modelo Gaussiano previamente definido, el *bgModel*. Esta función actualiza al *bgModel* con la información proveniente de la imagen. Es entonces lógico concluir y dar por hecho que el modelo Gaussiano estará en constante adaptación, ya que se modificará con cada frame que lo actualice mediante la función *cvUpdateBGStatModel.*

Se procede entonces a actualizar y extraer el fondo o background. Se debe mencionar que el modelo Gaussiano, *bgModel*, contiene un campo que incluye esta información. Nos valemos entonces de esta ventaja y clonamos la imagen guardada en el campo de fondo o background de *bgModel*, guardando este clon en una variable *background*. La siguiente línea de código muestra lo realizado:

```
background = cvCloneImage(bgModel->background);
```

Como se puede observar, la generación de una imagen de fondo es sumamente sencilla para nuestro sistema. La imagen de background devuelta por OpenCV al acceder al campo de fondo está en el formato de RGB. La Figura 11 muestra el resultado de este procedimiento.

Figura 11 Imagen de background obtenida por el sistema

De esta manera se establece un fondo o background base, que no será tomado en cuenta al momento de la detección del movimiento y será siempre descartado por el sistema. Como dijimos anteriormente, es necesario manejar una imagen de fondo capaz de adaptarse a muchos factores, siendo los más importantes:

- Cambios en la iluminación: tanto gradual como repentina (nubes)
- Pequeños cambios en el movimiento
- Pequeñas oscilaciones de la cámara

- Cambios en la geometría del fondo fijo: como carros parqueados.
- Objetos con alta frecuencia de aparición en el fondo fijo: como ramas de árboles, olas y cosas similares.

La adaptación a los cambios se garantiza al saber con seguridad que el fondo se adaptará con cada nuevo frame. Estando entonces la imagen de referencia en constante adaptación, se asegura un fondo o background preciso durante todo el tiempo de ejecución. Una vez obtenido el fondo se procede a generar la imagen de movimiento o foreground.

3.3.2 GENERACION DE IMAGEN DE MOVIMIENTO O FOREGROUND

Esta sección de procesamiento es esencial en nuestro sistema ya que define cuáles regiones se encuentran en movimiento. Este procedimiento es muy similar al de la obtención del fondo o background. De manera equivalente, se tiene que realizar un proceso de manera ordenada y debe ser ejecutado en el procesamiento de todos y cada uno de los frames.

Al tener ya definido el modelo Gaussiano, *bgModel*, haberlo actualizado con la función *cvUpdateBGStatModel* y extraído el fondo o background, se procede a encontrar a la imagen de movimiento o foreground tomando ventaja que *bgModel* también contiene en sus campos la información del foreground. De la misma forma que se obtuvo un clon de la imagen de background, se obtiene al foreground, cambiando única y obviamente a la

variable donde se guarda y al campo de *bgModel* que se accede. La siguiente línea muestra lo realizado.

```
motion_img = cvCloneImage(bgModel->foreground);
```

En este caso, la variable que almacena al foreground tiene el nombre de *motion_img*. Es de mencionar que la imagen extraída también se encuentra en RGB. Sin embargo, tiene una apariencia de imagen binaria. Esto se debe a que internamente OpenCV maneja una imagen binaria para la definición de foreground se refiere. Sin embargo, esta imagen binaria es transformada a RGB para cumplir con la definición de la estructura de dato del modelo Gaussiano.

La binarización de una imagen consiste en un proceso de reducción de la información de la misma, en la que sólo persisten dos valores: verdadero y falso. En una imagen digital, estos valores, verdadero y falso, pueden representarse por los valores 0 y 1 o, más frecuentemente, por los colores negro (valor de gris 0) y blanco (valor de gris 255). Para realizar la binarización de una imagen se debe primero pasar la imagen a escala de grises, después fijar un valor de umbral entre 0 y 255 (si la imagen con la cual se está trabajando está definida en un rango de profundidad de 8 bits/píxel), y convertir todos los valores de

la imagen superiores a este umbral a 255 y los menores a 0. Con esto nos queda una imagen en blanco y negro.

Como se puede apreciar en la Figura 12, el resultado que obtenemos es una imagen en blanco y negro, binarizada internamente por OpenCV pero devuelta en RGB para comparecer con las definiciones de las estructuras con las que se trabaja.

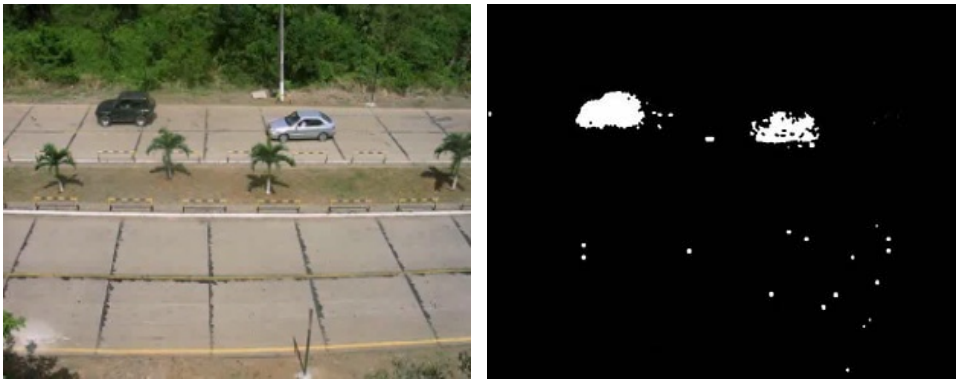

Figura 12 Izq. Imagen original; Der. Imagen de foreground, obtenida por el sistema

3.3.2.1 FILTRADO MORFOLÓGICO DE LA IMAGEN DE MOVIMIENTO

Como se puede apreciar en la Figura 12 (derecha), imagen binaria resultante en la detección de movimiento, al dividir las zonas de movimiento con las zonas inmóviles se ve afectada por la generación de regiones o píxeles con ruido. En muchas ocasiones se obtienen pequeñas zonas de movimiento falsas, es decir, donde realmente no hay movimiento, o el movimiento es mínimo, o no involucra a un vehículo, de manera que se lo considera despreciable. Para solucionar este inconveniente recurrimos a la eliminación de ruido mediante operaciones de filtrado morfológico.

Para realizar una correcta reducción del ruido a la imagen de movimiento es necesario primeramente definir un elemento estructurante, utilizando la función de OpenCV *cvCreateStructuringElementEx*. Este elemento estructurante es en sí un kernel de convolución, necesario para recorrer la matriz de la imagen al aplicársele los filtros de reducción de ruido. La estructura de datos definida por OpenCV para representar al elemento estructurante contiene a los campos de filas, columnas, separación relativa horizontal al punto de referencia, separación relativa vertical al punto de referencia, forma del elemento estructurante y un puntero a los datos del elemento estructurante.

Para el caso del elemento estructurante definido en nuestro sistema, se fijaron los valores de 3 filas e igual número de columnas, separación horizontal y vertical al punto de referencia con valor igual a 1, una forma elíptica y un puntero a NULL a los datos del

elemento, ya que OpenCV no toma en cuenta este campo a menos que la forma escogida sea personalizada. El elemento estructurante recibió el nombre de *element.*

Habiendo definido al elemento estructurante, se procede a la utilización de filtros morfológicos de erosión y dilatación para eliminar el ruido. OpenCV nos brinda implementaciones de los filtros necesarios para este propósito. Se empleó a las funciones *cvErode* para la erosión y *cvDilate* para la dilatación. La erosión se ejecuta en primer lugar seguida de la dilatación. Ambas funciones requieren como datos de entrada a la imagen de origen, la imagen donde se almacenará el resultado, un elemento estructurante para recorrer la imagen y el número de repeticiones que se ejecutará el filtro. Para ambos casos se emplea a la *motion_img,* clon previamente definido del foreground obtenido del modelo Gaussiano, como imagen de entrada e imagen donde se almacenará el resultado. *Element,* es utilizado como elemento estructurante en los dos filtros. Una diferencia yace al definirse 2 repeticiones para la erosión y tan solo 1 para la dilatación. La Figura 13 muestra la imagen resultante obtenida con la aplicación del proceso previo.

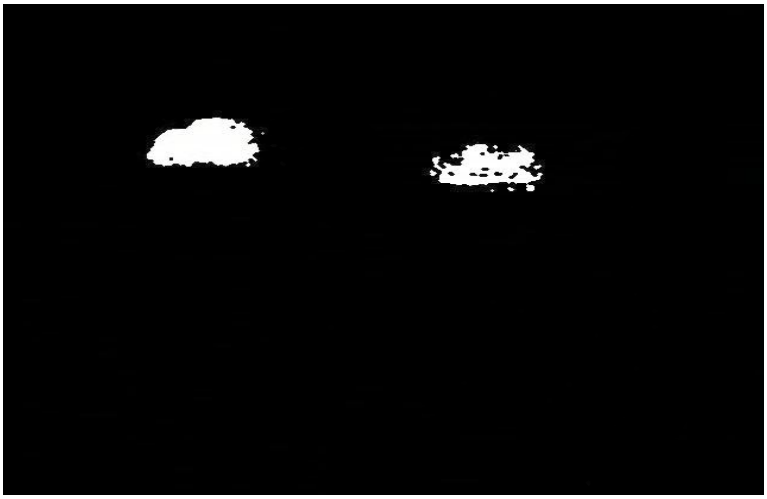

Figura 13 Imagen de movimiento después de la reducción de ruido aplicando filtros morfológicos

Una vez terminada esta reducción de ruido, se prosigue con la segmentación. En general, la segmentación consiste en separar diferentes áreas de la imagen de movimiento y clasificarlas como objetos diferentes, para luego seguir con la extracción de características de objetos independientes. Este procesamiento corresponde al seguimiento y extracción de características básicas de vehículos en movimiento.

4. SEGUIMIENTO Y EXTRACCIÓN DE CARACTERÍSTICAS BÁSICAS DE VEHICULOS EN MOVIMIENTO

4.1 INTRODUCCIÓN

El seguimiento de vehículos en un sistema inteligente de transporte es el encargado de seguir, valga la redundancia, a los vehículos durante todo su recorrido. Esta sección le da sentido a la información devuelta por el módulo de detección, extrayendo y realizando el análisis de características básicas del movimiento de los vehículos.

Este componente dentro de un sistema de visión por computador está compuesto por diferentes submódulos de procesamiento, que en conjunto, son capaces de obtener información precisa y clara acerca del movimiento descrito por un vehículo en particular. Sin embargo, este es un proceso bastante complejo, donde es esencial controlar la calidad y cantidad de información que pasa de un módulo a otro y de esta forma tener un mayor control en el flujo de entradas y salidas, que eventualmente llevan a un resultado de gran calidad y valor. En nuestro sistema, el seguimiento de vehículos está compuesto por los submódulos de:

- Segmentación
- Caja englobante

La segmentación se encarga de darle sentido a la información resultante del proceso de detección. Para el procesamiento de estos datos, el módulo de segmentación se basa en la utilización de una librería para la extracción de *blobs*, desarrollada para OpenCV. Un *blob* es una estructura compuesta por un conjunto de pixeles adyacentes y sus atributos. Este conjunto de píxeles han sido agrupados tomando en cuenta que ellos cumplen con ciertos criterios o parámetros de clasificación y es tomado como un objeto, no como simples píxeles separados y sin sentido. La librería para la extracción de *blobs* tiene el nombre de *cvblobslib* y nos proporciona funciones muy sencillas que nos facilitan de gran manera la segmentación de vehículos.

Cvblobslib es una libreria estática desarrollada en C++, capaz de ejecutar etiquetación de componentes a imágenes binarias. Es un componente algo similar a la librería *regionprops* de Matlab. La *cvblobslib* también provee de funciones para manipular, filtrar y extraer resultados de los blobs generados.

Por otro lado, el módulo de caja englobante se encarga de encerrar a los *blobs* encontrados para hacer posible la adquisición de datos de entrada necesarios para los componentes de extracción de características básicas y predicción de estados futuros. Es importante mencionar que este módulo es el encargado de la obtención del centroide de cada blob. El centroide es el centro de la caja englobante del blob. Es en sí la estructura de datos utilizada como entrada principal en los componentes antes mencionados, por lo que su producción es vital.

Adicional a los 2 módulos anteriores, existe el módulo de extracción de características básicas se desarrolló con el fin de brindarle al usuario una información más completa, describiendo de manera más detallada el movimiento realizado por el vehículo. De esta forma, es posible extraer características tales como: dirección de su movimiento, carril en el que se desplaza, tamaño de la caja englobante que lo contiene, entre otros. Para esta sección también se utiliza a la *cvblobslib*.

Reforzamos el sistema agregándole un módulo de predicción de estados futuros, con lo cual el modelo obtiene mayor robustez al ser capaz de entregar una respuesta incluso cuando los resultados de la detección contienen demasiado ruido, son incongruentes o en el peor de los casos, inexistentes. Este módulo utiliza exclusivamente funciones de OpenCV para su procesamiento. Un ejemplo que resume el comportamiento de este módulo puede ser visualizado en la Figura 14.

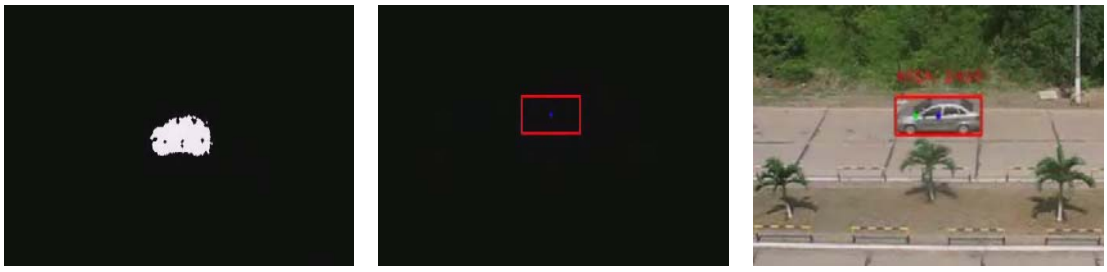

Figura 14 Izq. Imagen resultante de la segmentación; Centro. Imagen resultante de la caja englobante; Der. Imagen resultante del filtro de predicción

Concluimos el sistema con la generación de un video de salida final, el cual contiene los resultados del procesamiento del sistema, presentados de una manera congruente, ordenada y amigable para el observador.

objeto. Así es como se pueden obtener distintas características de diferentes objetos, analizándolos por separado.

A continuación la Figura 15 muestra un diagrama del componente encargado de la segmentación,

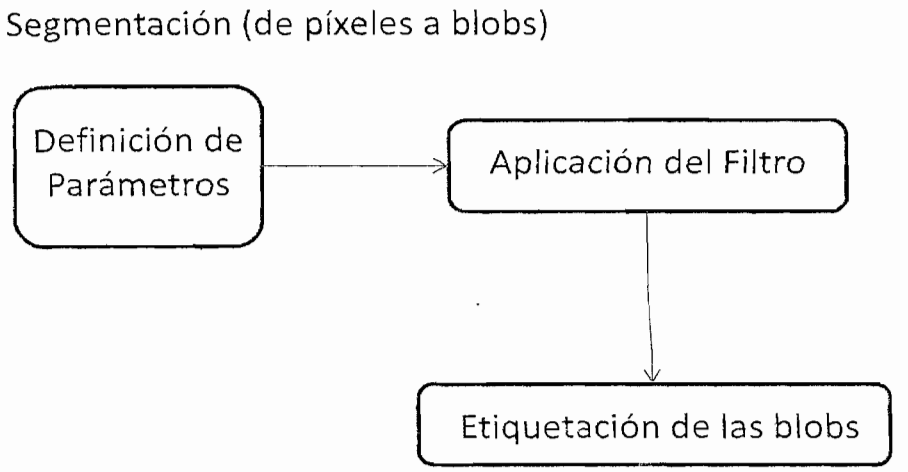

Figura 15 Diagrama del componente encargado de la segmentación

El módulo de segmentación, que agrupa los píxeles en movimiento que se encuentran adyacentes entre sí como una sola región, siguiendo algunos parámetros definidos para el tipo de objeto a ser detectado, hace que los datos devueltos por el módulo de detección tengan coherencia. En este módulo se define ya la forma final del blob que comprende el vehículo en movimiento y se procede a realizar las diferentes tareas de actualización, tanto de fondo o background como en los valores de los indicadores del sistema.

El módulo de segmentación, que agrupa los píxeles en movimiento que se encuentran adyacentes entre sí como una sola región, siguiendo algunos parámetros definidos para el tipo de objeto a ser detectado, hace que los datos devueltos por el módulo de detección tengan coherencia. En este módulo se define ya la forma final del blob que comprende el vehículo en movimiento y se procede a realizar las diferentes tareas de actualización, tanto de fondo o background como en los valores de los indicadores del sistema.

En este punto, se trabaja con una imagen binaria, que consiste, naturalmente, de píxeles negros y blancos, siendo los píxeles blancos aquellos que representan las zonas en movimiento. Debido a la necesidad de agrupar de manera inteligente estos píxeles en diferentes regiones, para de esta forma proseguir con la extracción de características, se maneja a la estructura del blob. Como se dijo anteriormente, para la obtención de un blob se siguen ciertas normas o reglas de clasificación.

En primera instancia se busca agrupar píxeles adyacentes o vecinos. Luego se pretende que sea una región de píxeles que siga una misma tendencia de movimiento, para así cerciorarnos de que los píxeles correspondan a un mismo cuerpo, y no a dos cuerpos diferentes que se encuentran cercanos. Finalmente, y específicamente para la clasificación de vehículos y en muchos casos también para eliminar ruido, se aplica un filtrado de tamaño o área de los blobs. Se procede a limitar la agrupación de píxeles a un tamaño específico, con lo cual se obtienen los diferentes vehículos que se puedan encontrar en movimiento en una escena determinada.

Una vez que cada región o conjunto de píxeles se puede identificar como un objeto separado, se procede a sacar provecho de esto, de esta forma, cajas englobantes son colocadas alrededor de las regiones generadas (vehículo en movimiento en cuestión), y se pintan con colores distintos a las diferentes regiones obtenidas.

En la implementación del sistema llevamos a cabo el procedimiento descrito a continuación. Utilizamos a la estructura de datos *cvBlobResult*, que tiene a los campos de la imagen binaria de movimiento, una imagen de máscara, un valor de umbral o threshold y un valor booleano que especifica si la función debe encontrar los momentos de los blobs. Esta función devuelve un objeto que contiene a todos los blobs de una imagen. Para nuestro sistema, se utilizó a la imagen de movimiento, *motion*_img, como entrada, no se utilizó una imagen de máscara, definiendo entonces este campo como NULL, un valor de threshold de 100 y que un valor FALSE, o falso, para que la función encuentre a los momentos de los blobs.

Nuestro objeto contenedor de todos los blobs de la imagen es en sí un arreglo de estos objetos y recibió el nombre de *blobs*. De esta forma, ahora tenemos un objeto que contiene a todas las regiones en movimiento encontradas en la detección y se las toma como blobs.

Debido a que queremos que los objetos dentro del *cvBlobResult*, *blobs*, solo sean carros, filtramos el contenido, usando el método *Filter*. Esta función recibe como datos de

entrada al objeto donde se guardarán los resultados, si se debe incluir o excluir a los blobs que cumplan con el filtro, el campo que se evaluará de cada blob, una condición para decidir si el resultado de la evaluación debe ser incluido o no y finalmente los limites mínimo y máximo respectivamente.

Nuestro sistema clasifica si un blob corresponde a un vehículo mediante el valor del área de este objeto. Se eliminan las áreas que se consideran o demasiado pequeñas o demasiado grandes para ser un automotor. Por lo tanto, el método *Filter* recibe al objeto *blobs* como objeto donde guardar los resultados, que debe incluir a los blobs que cumplen con este filtro, se envía la función *CBlobGetArea*, que devuelve el área de cada blob para evaluar a cada objeto detectado, considerando a todos los objetos dentro del rango de área entre 750 y 100000 como vehículos. Es de esta manera como se obtiene un arreglo exclusivo de automotores y el antes definido *blobs*, evoluciona a ser un arreglo que contiene solamente blobs correspondientes a vehículos. La Figura 16 muestra a dos blobs de automotores debidamente segmentados.

Figura 16 Imagen con blobs 1 y 2 debidamente segmentados

Concluimos que la segmentación de la imagen de movimiento indirectamente es un complemento del filtro morfológico aplicado a la imagen resultante del módulo de detección, al eliminar blobs pequeños que podrían ser considerados como ruido no eliminados por el filtrado morfológico previo.

4.2.2 CAJA ENGLOBANTE

Continuando con el seguimiento de objetos en movimiento, interviene lo que se conoce en visión por computador como *convex hull* o caja englobante. Fácilmente se puede colocar una caja englobante alrededor de un objeto, o blob obtenido previamente,

mediante operaciones matemáticas que permitan determinar el *centroide* del objeto y su tamaño, para luego posicionar correctamente la caja circundante.

El centroide corresponde al centro de la caja englobante que encierra al blob en cuestión. Este centroide, que no es más que un punto con coordenadas en las abscisas y ordenadas, amplía de gran manera la capacidad de extracción de características. Es decir, a partir de este centroide se pueden realizar una infinidad de operaciones para obtener resultados, tales como la determinación del carril donde circula el vehículo, dirección del movimiento, entre otros.

Este objeto es casi todo lo que se necesita del blob para extraer características de movimiento. Previamente se ha obtenido la masa del automotor, de la cual podemos obtener el área cubierta y la forma. Por otro lado, con el centroide podemos determinar la posición, carril y dirección. Es muy importante también el uso del centroide para los filtros de predicción que se explicarán más adelante.

En nuestro caso el proceso de caja englobante y obtención del centroide está compuesto de una serie de operaciones matemáticas simples. Recorriendo uno a uno al arreglo de blobs, *blobs,* se genera la caja englobante alrededor de cada blob de la imagen de movimiento a partir de la definición de cuatro valores característicos del blob. Estos parámetros son el valor mínimo en X *(Xmin),* el valor mínimo en Y *(Ymin),* el valor máximo en X *(Xmax)* y el valor máximo en Y *(Ymax).* Una vez obtenidos estos

parámetros se arma un rectángulo con la función *cvRectangle*, obviamente utilizando estos valores.

La obtención del centroide se logra simplemente promediando los valores máximos y mínimos de ubicación de cada blob. La Figura 17 ilustra un ejemplo para la caja englobante y centroide generados.

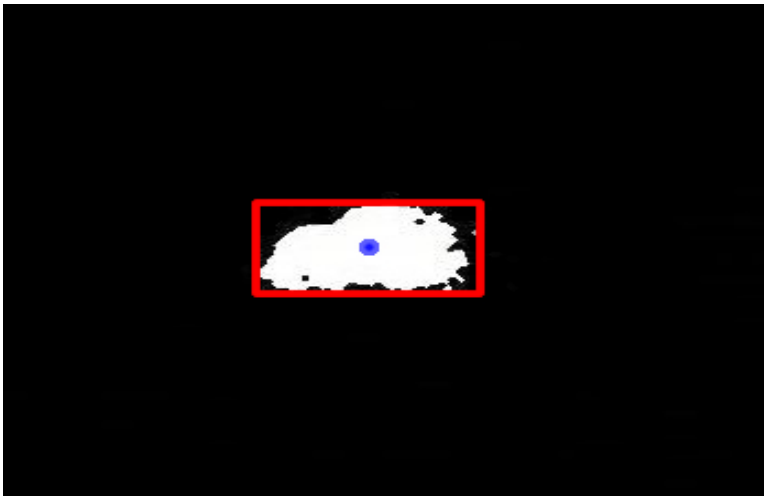

Figura 17 Imagen de un blob encerrado por una caja englobante y su centroide definido

Este proceso se aplica a cada uno de los blobs que hayan sido detectados en la imagen de movimiento.

4.3 EXTRACCIÓN DE CARACTERÍSTICAS BÁSICAS DE VEHÍCULOS EN MOVIMIENTO

Hemos obtenido hasta el momento un vehículo detectado, segmentado y encerrado en una caja englobante, definiendo ya su centroide. Es ahora el momento de extraer información básica con respecto al movimiento que describe el vehículo. Para la implementación de esta importante sección, se analiza a cada centroide. En el centroide se encuentran muchas de las claves de los datos necesarios para extraer la información que requerimos. A partir de este simple punto podemos determinar la mayoría de características descritas por el sistema, tales como la posición, dirección, carril. Por otro lado, el blob se utiliza para determinar el tamaño del vehículo. El arreglo de blobs de vehículos nos da la información para realizar un conteo de carros en un determinado lapso de tiempo.

En primer lugar se determina la posición del centroide, utilizando una estructura de datos de OpenCV denominada *cvPoint*, la cual es una representación de un punto en un plano. Básicamente se crea un punto con coordenadas en las abscisas y ordenadas correspondientes al centroide previamente encontrado. Este procedimiento se realiza para mejorar el acceso y manejo de la información del centroide, introduciendo su información a un objeto estándar de OpenCV. Una vez realizado esto, se procede a extraer la dirección en la cual se está desplazando. Esto se realiza mediante la comparación de la altura del centroide y un valor de límite de carril. En caso de que el

valor de la coordenada en Y del centroide sea mayor que el del límite del carril, el vehículo se encontrará en el carril de arriba, dirigiéndose hacia la izquierda. Por otro lado, si el valor de la coordenada en Y del centroide es menor que el del límite del carril, el vehículo se encontrará en el carril de abajo, dirigiéndose hacia la derecha. Es de mencionar que la dirección se asume por la orientación predefinida del carril. Es decir, nuestro sistema no definirá correctamente la dirección de un vehículo en contravía, siendo esta una limitación de nuestro sistema. Adicionalmente a esta consideración, nuestro sistema también asume que las imágenes de la escena que se esta analizando contienen información de únicamente 2 carriles, un carril que se encuentra en la parte superior de la imagen y otro carril que se encuentra en la parte inferior de la misma, tal como se puede apreciar en la Figura 11.

Para el conteo de vehículos en movimiento, se define a una variable global, *num_carros_global*, que aumenta en uno cada vez que un nuevo blob de vehículo ha sido encontrado mediante un condicional.

Hallamos el valor del área de cada vehículo extrayendo uno por uno a los objetos contenidos en el arreglo *blobs.* A cada objeto se lo enmarca dentro de la estructura de datos *CBlob*, de la librería *cvblobslib.* Una vez definido el *CBlob* con los datos del blob analizado en ese momento, se accede a su campo de área y de esta forma se adquiere el valor de superficie del vehiculo. Esta información es a su vez utilizada para una comparacion con un valor de umbral, definido de manera global, para determinar si se

trata de un vehículo liviano o pesado (considerando para esto que el área de los vehículos livianos será menor al de los pesados). El valor definido en nuestro sistema como umbral para liviano o pesado es de 15000. De esta forma se concluye con la sección de extracción de características.

4.4 PREDICCIÓN DE ESTADOS SUBSECUENTES

Luego de la extracción de características básicas se utiliza un módulo de predicción de movimiento para darle robustez al seguimiento. Este módulo es sumamente importante para la eficiencia del sistema. Es el encargado de predecir, basándose en las coordenadas de los centroides resultantes previamente obtenidos, una siguiente ubicación del objeto con un alto grado de certeza. Para la implementación de este módulo se utilizaron dos métodos muy interesantes. Estas técnicas son:

- Filtro de desplazamiento previo, y
- Filtro de Kalman

Nuestro sistema crea inicialmente un arreglo de *cvPoint*, donde se almacenarán los resultados de las predicciones de centroides futuros. Este arreglo, en el sistema bajo el nombre de *puntosprediccion,* aumentará de tamaño al detectarse la aparición de un nuevo vehículo, introduciendo al centroide generado al final del arreglo y se eliminará al primer centroide cuando un vehículo salga de la pantalla. Esto se monitorea mediante la comparación del valor en X del centroide con el valor máximo de posición en X que se puede tener para seguir estando en el recuadro o ventana de visualización de la imagen.

Para determinar que un vehículo no ha sido detectado correctamente, se compara la cantidad de blobs de vehículos en el arreglo *blobs,* con la cantidad de centroides en el arreglo de *cvPoints, puntosprediccion.*

En caso de que *blobs* tenga menos objetos que *puntosprediccion,* se concluye que un vehículo no ha sido detectado y entonces se envía uno a uno a todos los centroides de *puntosprediccion* a ser comparados con los centroides de los blobs de vehículos que sí han sido encontrados. Se piensa que el centroide enviado por *puntosprediccion* corresponde a un carro correctamente detectado si sus coordenadas están dentro de una diferencia de más o menos 9 con al menos uno de los centroides detectados. Si el centroide enviado por el arreglo de puntos de predicción no cumple esta condición, se concluye que ese es el centroide del vehículo no detectado.

Se entrega entonces este centroide a un procesador especial de extracción de características, utilizado únicamente cuando no se encontró a un vehículo. Este módulo no afecta al conteo de vehículos sino que solo determina su carril y dirección. A su vez, se crea una caja de 30x20 pixeles que tendrá como centroide obviamente al punto con el que se ha trabajado.

Debido a los diferentes casos que pueden aparecer durante el análisis del video y el procesamiento de datos, es necesario definir claramente estrategias de comportamiento para otorgarle robustez al sistema y de esta forma le sea posible ofrecer una respuesta congruente. Los filtros descritos previamente arrojan resultados, a los cuales se les

asignará un peso respectivo. De esta forma, el sistema podrá escoger la mejor opción en caso de verse envuelta en alguna circunstancia especial. A continuación explicamos cada caso:

CASO 1

- El sistema fue capaz de definir por si solo y con exactitud al menos dos instancias previas del movimiento del vehículo. El cálculo realizado por el filtro de predicción brinda entonces un resultado basado totalmente en lecturas reales.

CASO 2

- El sistema no fue capaz de definir al menos los dos estados consecutivos previos del movimiento del vehículo. Se recurre a uno de los resultados generados por ya sea el filtro de Kalman o el filtro del desplazamiento previo. En este caso, uno de los datos utilizados en el caso del filtro del desplazamiento previo será tomado de lo generado por uno de los dos filtros previamente. En el caso de escoger el filtro de Kalman, se generará un dato generado artificialmente, es decir, completamente a partir de predicciones anteriores.

CASO 3

- Para este caso, el sistema tendrá que recurrir enteramente a las predicciones previamente generadas por los filtros a disposición al no tener medidas reales. Estos datos serán entonces información de entrada para el sistema, que arrojará un nuevo resultado, generado artificialmente en su totalidad por los filtros de predicción de movimiento.

El resultado, en caso de obtenerlo el sistema por si solo, siempre tiene más peso que el generado por un filtro, debido a que este es en esencia una medición real del movimiento. Por otro lado, los filtros por más precisos que sean, tienen siempre un porcentaje de error o se ven influenciados por el ruido.

En el caso del filtro del desplazamiento previo, se necesita siempre de dos puntos, y si se toma este resultado muchas veces, arroja una diferencia constante, lo cual muy rara vez ocurre en la vida real.

Por otro lado, el filtro de Kalman al verse afectado por un pico de la señal correspondiente al ruido, no podrá recuperarse jamás y predecir exactamente un punto de la linea real, mas estará sujeto de todas formas a una trayectoria parecida a la del movimiento real. Sin embargo, esto no ha ocurrido hasta ahora gracias al buen manejo y control de los datos de entrada y su procesamiento.

4.4.1 FILTRO DE DESPLAZAMIENTO PREVIO

El filtro del desplazamiento previo es en sí un promedio de las variaciones de posición entre estados previos o consecutivos para así conseguir una distancia que separará a un futuro punto del punto actual. De esta forma al extraer la distancia entre dos centroides consecuentes, se halla una distancia aproximada, que sumada a la posición del centroide actual daría la ubicación del siguiente centroide de ese vehículo.

Este filtro moldea linealmente el movimiento descrito por el vehículo en centroides representativos de la posición de la caja englobante que encierra al automotor. Para el adecuado funcionamiento de este filtro, se debe esperar por lo menos dos apariciones correctamente detectadas y consecuentes de un vehículo, para de esta forma tener al menos una distancia aproximada entre puntos consecutivos.

Para encontrar la distancia, simplemente se recepta a un centroide y se lo almacena en un *cvPoint* hasta la próxima ejecución del seguimiento, donde con la llegada de un nuevo centroide, se comparan las coordenadas tanto en X como en Y y se extrae un valor de diferencia entre los dos puntos. El resultado obtenido será sumado, manteniendo su signo, al centroide más reciente y se enviará este nuevo punto al arreglo *puntosprediccion,* reemplazando al punto correspondiente al del centroide del vehículo en ese momento analizado, en caso de que *puntosprediccion* y *blobs* tengan el mismo número de objetos, o agregándolo al final, en caso de que *blobs* tenga un número mayor

de elementos que *puntosprediccion* y ya se haya alcanzado una posición más baja que la correspondiente al último de los *cvPoint* de *puntosprediccion*.

4.4.2 FILTRO DE KALMAN

Por otro lado encontramos al filtro de Kalman. Este filtro es extremadamente poderoso y popular, es utilizado por nuestro sistema especificando un centroide inicial para cualquier vehículo detectado como en movimiento. Usamos la funcionalidad de OpenCV para definir un filtro predictor de Kalman con el método *cvCreateKalman,* que recibe como entradas las dimensiones del vector de parámetros dinámicos, las dimensiones del vector de parámetros de medida y las dimensiones del vector de parámetros de control. Para nuestro sistema se definieron valores de dos dimensiones de parámetros dinámicos, al tener un punto valores en X y Y, una dimensión de parámetros de medida, y ningún parámetro de control.

Se procede entonces a definir la identidad del filtro de Kalman estableciendo que su matriz de mediciones tendrá datos escalares reales, una covarianza del proceso igual a 1^{-5} (siendo este un valor recomendado por OpenCV para la predicción) y un valor de la covarianza de las medidas de 1^{-1} también sugerido por OpenCV. Es necesario definir también una matriz que sea capaz de generar por si sola todas las combinaciones posibles

de predicciones. En nuestro sistema, esta matriz recibió el nombre de A y tuvo los siguientes valores:

$$A = \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix}$$

Definimos finalmente un *cvPoint curFilt*, donde se almacenarán los resultados obtenidos por el filtro.

Una vez realizado este ajuste inicial, se procede a receptar las coordenadas del centroide del punto actual. Cuando se recibe la información de las posiciones de los centroides, se utilizan estos valores para ingresar medidas de ajuste al filtro predictor, en este caso la diferencia entre el centroide anterior y el actual por un lado y las coordenadas del punto actual en otro.

El filtro de Kalman estima el proceso utilizando una especie de control de retroalimentación, esto es, estima el proceso a un momento posterior en el tiempo mediante la función *kalmanPredict*, que recibe como entradas al filtro arriba definido y a un vector de control, que en nuestro sistema no tiene valor. Este valor es almacenado en *curFilt*. Este punto es enviado a *puntosprediccion* para un manejo similar al de los puntos generados por el filtro del desplazamiento previo para determinar si es un valor que debe reemplazar a otro o ser agregado al final.

Mediante esta predicción se obtiene retroalimentación comparando la estimación con datos reales observados. Con esta información de corrección se modifica el valor del filtro de predicción por uno que toma en consideración el nuevo valor de ajuste. Para este proceso de adaptación se usa al método definido en OpenCV como *kalmanCorrect*, que recibe como entradas al filtro de Kalman y a la medida de corrección, en nuestro caso, el valor de la coordenada en X del centroide del vehículo analizado en ese momento.

En sí, el primer paso consiste en generar un pronóstico del estado futuro en el tiempo tomando en cuenta toda la información disponible en ese momento y procesada internamente por el filtro. En un segundo paso, se modifica al filtro con la introducción de una medida real, de tal manera que se asemeje más al proceso a estimar y consecuentemente el error es minimizado estadísticamente. Finalmente, se copia al arreglo *puntosprediccion* para que en el siguiente frame se actualice primeramente con estos valores y de ahí se modifique con el valor actual para ese frame. Este doble procesamiento brinda mejores resultados, ya que se genera una curva estimada mucho más suavizada.

La Figura 18 muestra al filtro de Kalman operando sobre una función en la presencia de ruido.

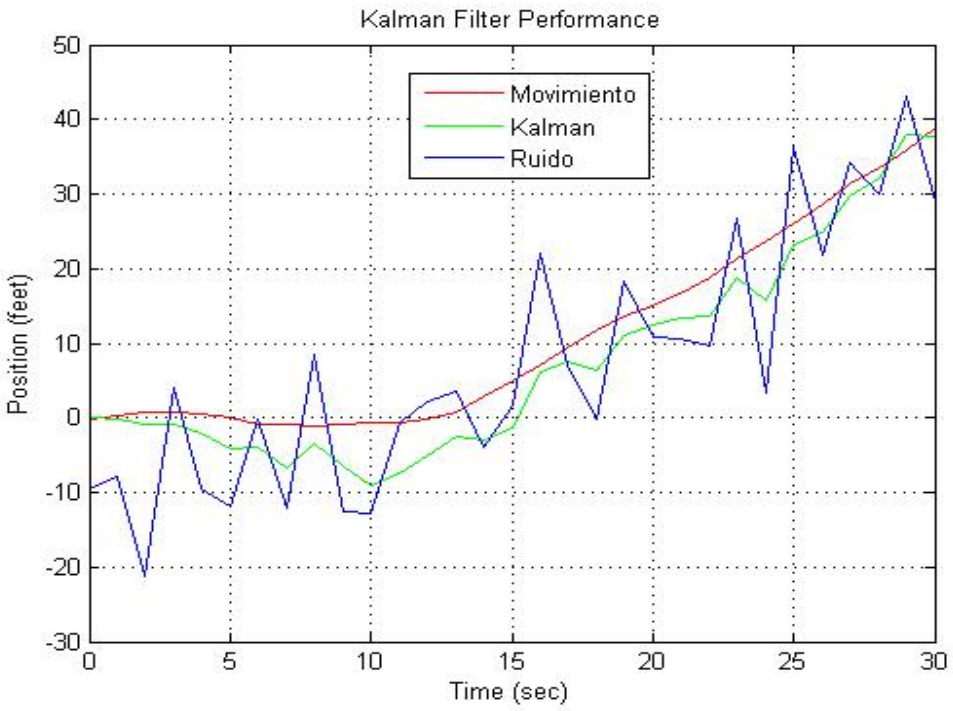

Figura 18 Gráfico comparativo de la predicción del filtro de Kalman con la presencia de ruido

Efectivamente, el algoritmo de Kalman puede definirse como un algoritmo de pronóstico-corrección para resolver numerosos problemas. Así, el filtro de Kalman funciona por medio de un mecanismo de proyección y corrección al pronosticar el nuevo estado y su incertidumbre y corregir la proyección con la nueva medida.

Después de cada par de actualizaciones, tanto del tiempo como de la medida, el proceso es repetido tomando como punto de partida las nuevas estimaciones del estado y de la covarianza del error. Esta naturaleza recursiva es una de las características llamativas y poderosas del filtro de Kalman. Al ser un algoritmo recursivo, su facilidad de

implementación computacional es alta. Por otro lado, ofrece adaptación al ruido al que es expuesto, ya que parte del valor de ajuste que genera contiene la varianza de los ruidos a los que el sistema está expuesto, la Figura 18 ilustra el comportamiento del filtro de predicción de Kalman con la presencia de ruido.

Finalmente llegamos a un pequeño pero muy importante submódulo. Esta parte, la cual en realidad se ejecuta durante toda la sesión del sistema, se encarga de grabar el video procesado, y tiene el objetivo de entregarnos un video con los resultados obtenidos durante la detección y el seguimiento, para poder analizar repetidas veces una misma ejecución del programa, y de esta forma poder sacar varias conclusiones.

Este submódulo consta de dos partes. La primera es la que se encarga de mostrar los resultados en el momento de la ejecución, que en realidad solo sirve para el propósito de retroalimentación para el desarrollador o administrador del sistema, lo que es evidente cuando nuestra interfaz muestra dos videos; el de entrada, y el de salida. La segunda parte, la cual ofrece información realmente importante, es la que se encarga de hacer un grabado de un video de salida para un futuro análisis de resultados de forma eficiente.

El proceso de generación del video de salida resultante, o recopilación de imágenes, es sencillo. Inicialmente, para la captura de frames de un video desde un archivo de video, se utiliza la función *cvCaptureFromAVI*, la cual recibe como parámetro de entrada el archivo de video, y retorna un tipo de dato *CvCapture*.

Básicamente, se empleó a *ImageCaptureDlg.cpp*, un archivo que recoge la entrada en formato de *CvCapture* y la muestra en el lado izquierdo de la ventana de presentación. A su vez, este archivo se encarga de enviar todos y cada uno de los frames a nuestro algoritmo para ser debidamente procesado. Una vez que recibe el resultado desde nuestra técnica, lo muestra en la parte derecha de la ventana de presentación.

Para capturar el video resultante se crea un archivo de video al inicio de cada corrida, definido como una variable global llamada *outputmovie*. En cada iteración se procede a recolectar el frame en esa instancia procesado y se lo anexa al archivo. Finalmente, se termina este proceso cerrando el video. La creación del archivo se realiza utilizando la función *cvCreateVideoWriter,* que recibe el nombre y extensión del archivo de video a retornar, un código de 4 caracteres que identifica al códec que se utilizará para comprimir el video, la cantidad de frames por segundo (fps) que tendrá el archivo, el tamaño de los frames y finalmente si el resultado debe ser a color o en escala de grises, siendo cualquier valor diferente de cero para un video a color. A continuación mostramos la línea de código definida para crear el archivo de video:

```
CvVideoWriter* outputMovie = cvCreateVideoWriter("video_out.avi",
CV_FOURCC('D', 'I', 'V', 'X'), 29.97, cvSize(width, height), 1);
```

Para anexar cada frame a la secuencia de imágenes, se utiliza el método *cvWriteFrame*, que recibe al video donde se guardará y al frame donde se debe anexar al final. Para

nuestro caso, el video es *outputMovie* y el frame es la imagen que devuelve el algoritmo, definido por nosotros como *PresentImage*. Mostramos la línea encargada de guardar cada nuevo frame en el video de salida.

```
frame:cvWriteFrame(outputMovie,PresentImage);
```

Finalmente, para cerrar el archivo se utiliza una función llamada cvReleaseVideoWriter, que libera al archivo de video. Este método recibe como entrada un puntero al archivo de video utilizado. Es de mencionar que si no se cierra el video de salida, este queda inutilizable, por lo que su correcto manejo es fundamental.

El video de salida es el resultado final de todo el sistema. La Figura 19 muestra el resultado final devuelto por nuestro sistema.

Luego de varios análisis se llegó a la conclusión de que era sumamente necesario un módulo de pregrabado. Su importancia se debe a que el sistema está hecho para correr por largas horas seguidas durante el día, y cuadro a cuadro se realizan grandes cantidades de operaciones de procesamiento de imágenes, que muchas veces son innecesarias, como por ejemplo, en la ausencia de vehículos.

Hemos podido observar que por varios lapsos de tiempo, todo lo que ha sido capturado en los videos de entrada, es una carretera vacía. No tiene caso realizar todo el procesamiento sobre estos frames, sino solo lo estrictamente necesario para detectar la existencia de movimiento de lo que pudiese ser un vehículo. Es entonces que se usa este

módulo de pregrabado, que en realidad es la reutilización de una porción de código del módulo de grabación de salida junto con una porción del código de la detección, pero utilizado para proporcionar un dato de entrada reducido al sistema.

Se reduce en un gran porcentaje la cantidad de procesamiento, pues luego de pruebas y cálculos, se llegó a la conclusión de que si tuviéramos el sistema monitoreando una carretera por 24 horas, se tendría en promedio 4 horas acumuladas de tráfico en movimiento atravesando el marco de visión definido. En realidad, sí ocurre un doble procesamiento con ciertas operaciones, pero luego se procesa aproximadamente un 16,6% del video original. Podemos afirmar que es un submódulo sumamente necesario, que incrementa la eficiencia del sistema y queda por demás justificado. La Figura 19 muestra el resultado final devuelto por el sistema.

Figura 19 Resultado final

Cada módulo tiene su propia importancia, pero todos trabajan juntos por un mismo objetivo, por lo cual todos merecen ser estudiados con el mismo interés. Se ha analizado a breves rasgos, aunque en toda su extensión, la funcionalidad del sistema.

5. RESULTADOS EXPERIMENTALES Y ANÁLISIS DE RESULTADOS

5.1 INTRODUCCION

Antes de realizar cualquier prueba, se necesitaba desarrollar un prototipo. Para este fin, nos valimos de un video simple de carros en el tráfico urbano, obtenido en el Internet, para desarrollar de manera creciente el sistema. Se procedió de esta manera hasta llegar a una versión beta del prototipo.

Una vez que el prototipo entró en fase beta, se empezaron a realizar pruebas preliminares para determinar las capacidades reales y limitaciones del sistema. Un sistema entra en fase beta cuando se encuentra totalmente funcional, pero todavía necesita ciertos cambios o ajustes que solo se pueden determinar luego de que el sistema sea utilizado y probado para obtener la retroalimentación que necesita el desarrollador para realizar estos cambios o ajustes. Esto sucede siempre en cualquier sistema o proyecto en general. Un proyecto puede estar en teoría terminado, pero las cosas son diferentes en la práctica, y resultados inesperados o inconvenientes inadvertidos previamente pueden suceder.

5.2 PRUEBAS DE CAMPO PRELIMINARES

En esta sección se describen los trabajos iniciales que se realizaron para generar los archivos de videos viales. Aunque estos trabajos fueron realizados, hay que recordar que nuestro sistema no contempla el montaje y calibración de las cámaras responsables de obtener los videos viales desde las autopistas o carreteras. Se asume que esta información ha sido generada previamente desde algún sistema de adquisición de videos viales.

Las primeras pruebas preliminares fueron realizadas con videos capturados en el área de tecnologías de la ESPOL, con una cámara fotográfica comercial cuyas características no eran las adecuadas. Además la ubicación en la que nos colocamos no era lo más remotamente cercano a lo que se requería. Sin embargo estos videos cumplieron un propósito, al servir de referencia para ajustar parámetros en el código. El avance fue sustancial pero se necesitaban más pruebas que se acerquen más a lo que se quiere lograr con el sistema.

Las segundas pruebas, presentadas en este documento, fueron realizadas con videos obtenidos en el área de ingenierías de la ESPOL. La cámara fue colocada en el techo de la biblioteca central de ingenierías, de donde se obtuvieron instancias o grabaciones largas de video con una cámara digital. Estas capturas fueron mucho más nítidas y menos ruidosas que las anteriores. Se pudo experimentar con mayor capacidad de análisis y mayores conclusiones fueron sacadas. Se había obtenido, en otras palabras, mucho más

material con el cual trabajar. Las características de la cámara digital usadas son las que se describen en la Tabla 1.

Tabla 1 Características de cámara Canon PowerShot A610

Resolución	5.0 megapíxeles
Lente	Desde 7.3mm hasta 29.2mm
Apertura del lente	Desde f/2.8 a f/8.0
Retraso del obturador	Desde 1/2500 a 15 seg
Zoom del lente	4x
Zoom digital	4x

Para el proceso de adquisición de los videos es necesario tomar en cuenta ciertos parámetros muy importantes en cuanto a las características de la cámara, altura a la que se la debe colocar, distancia a la cual debe de estar del objeto de interés (distancia de enfoque) y el ángulo en el que se logra una mejor captura. Las medidas óptimas para la adquisición se muestran en la Tabla 2

Tabla 2 Restricciones mínimas y configuración para funcionalidad óptima del sistema

Altura a la que se colocó la cámara	14,7 m.
Distancia de enfoque	10,8 m
Ángulo de inclinación de la cámara con respecto al eje horizontal	-42°
Resolución de la cámara	640 x 480
Frames por segundo	30

Una vez que los videos fueron adquiridos, se procedió a analizar la efectividad del sistema. En otras palabras, qué tanto cumple con el objetivo principal de detectar vehículos en movimiento. Los resultados, en este punto, fueron más que satisfactorios.

Por otro lado, nos enfocamos en calificar la eficiencia del sistema, es decir, en qué tan rápido lo hace y a qué costo. Básicamente, hacer los cambios necesarios para optimizar el costo computacional del sistema. En este ámbito se sacó la mayor ventaja posible debido a un conocimiento profundo del manejo de recursos por parte del sistema, como liberación de memoria y evitar el doble procesamiento en partes donde no era necesario.

Fue a partir de este análisis que nos dimos cuenta del exceso de procesamiento que estaba realizando el sistema en lo que hemos denominado como *tiempo muerto*. Pudiéndose entender por tiempo muerto aquellas instancias del video donde no se registraba ningún

movimiento, ya que durante cierto tiempo las imágenes permanecían constantes y ningún vehículo circulaba.

En el caso de nuestro prototipo se realiza el mismo tipo de procesamiento para todos los frames del video, es decir instancias con y sin presencia de vehículos, lo cual para los casos en los que ocurrían las instancias de tiempo muerto era un desperdicio computacional. Fue entonces de donde nació la iniciativa de utilizar un módulo de pregrabado, descrito al final del capítulo anterior.

Encontramos también varios detalles en lo referente a casos especiales. Entre los casos especiales más destacados tenemos la situación de varios vehículos viajando en conjunto, de tal manera que cuando existe solapamiento entre ellos se pueden confundir como un solo objeto, o el posible suceso de varias personas caminando en grupo, de tal manera que forman un único objeto en movimiento que puede confundirse con un automóvil, o también la situación de una carreta u otro tipo de vehículo que no sea un automóvil que pueda ser confundido por uno.

Hayamos entonces una gran ayuda proporcionada por el módulo de predicción. Para estos casos especiales, hemos definido que la predicción de este módulo sea la información tomada en cuenta por el sistema.

Para el análisis de los resultados se hace uso del módulo de pregrabado, definiendo como entrada el video procesado. De esta forma conseguimos un video de salida con la

detección y el seguimiento integrados. Se obtuvieron varios videos, y fue a base de estos que se obtuvieron las conclusiones y se realizaron las observaciones anteriormente indicadas.

5.3 RESULTADOS EXPERIMENTALES

Los siguientes gráficos y tablas muestran los resultados experimentales obtenidos de las tres fuentes monitoreadas, a diferentes horas del día, de observaciones pertenecientes a un vehículo elegido de manera aleatoria:

- Movimiento detectado
- Filtro del desplazamiento previo
- Filtro de Kalman

Desde 08h30 hasta 09h05 am

Tabla 3 Vehículos en movimiento desde 08h30 hasta 09h05 am.

Vehículos que aparecen en el video	180
Vehículos que detecta el sistema	175
Porcentaje de vehículos detectados	97%
Objetos en movimiento en el video (personas, animales, etc)	11
Objetos en movimiento detectados por el sistema	0

Para cada una de las Tablas de Medición de Desempeño (4, 6 y 8) de los filtros de predicción se escogió a un vehículo del conjunto que recorrió el tramo bajo monitoreo de manera aleatoria. Este vehículo sirvió de muestra para representar la veracidad de los filtros en un determinado horario.

Como se mencionó en la Tabla 2, el sistema recibe videos grabados a 30 frames por segundo. En las Tablas 4, 6 y 8 se muestran las posiciones en X detectada del vehículo durante 35 frames de su movimiento. Generalmente a un vehículo le toma entre 1.25 y 2.1 segundos atravesar completamente el área de monitoreo, por lo que se analizó al automotor durante 1.18 segundos de su recorrido estando ya correctamente detectado y antes de que se salga del área de detección para de esta forma asegurarnos que todos los vehículos analizados se encuentren dentro del área de detección y siendo detectados por el sistema.

Tanto la posición en X detectada, como los valores del filtro de desplazamiento previo y filtro de Kalman vienen dados en píxeles. Recordemos que el área de monitoreo tiene 640 píxeles de ancho.

Tabla 4 Medición de desempeño desde las 08h30 am hasta las 09h05 am

Frame	Posición en X (píxel)	Desplazamiento previo (píxel)	Filtro de Kalman (píxel)
1	2	2	5
2	6	6	5
3	20	10	20
4	35	35	35
5	55	50	45
6	65	75	60
7	80	75	80
8	85	95	90
9	105	90	105

10	120	125	110
11	135	135	130
12	160	150	150
13	175	185	160
14	185	190	185
15	190	195	200
16	210	195	210
17	235	230	215
18	250	260	235
19	270	265	260
20	280	290	275
21	295	290	295
22	310	310	305
23	320	325	320
24	335	330	335

25	350	350	345
26	370	365	360
27	380	390	375
28	395	390	395
29	415	410	405
30	435	435	420
31	445	455	440
32	450	455	460
33	460	455	470
34	475	470	475
35	495	490	485

La Figura 20 muestra una imagen obtenida en el horario de la mañana.

Figura 20 Imagen obtenida entre 08h30 y 09h05 am.

La Figura 21 muestra un gráfico comparativo de las posiciones en X detectadas, predicción del filtro de desplazamiento previo y del filtro de Kalman. Las abscisas corresponden al número de frame y las ordenadas al píxel en X.

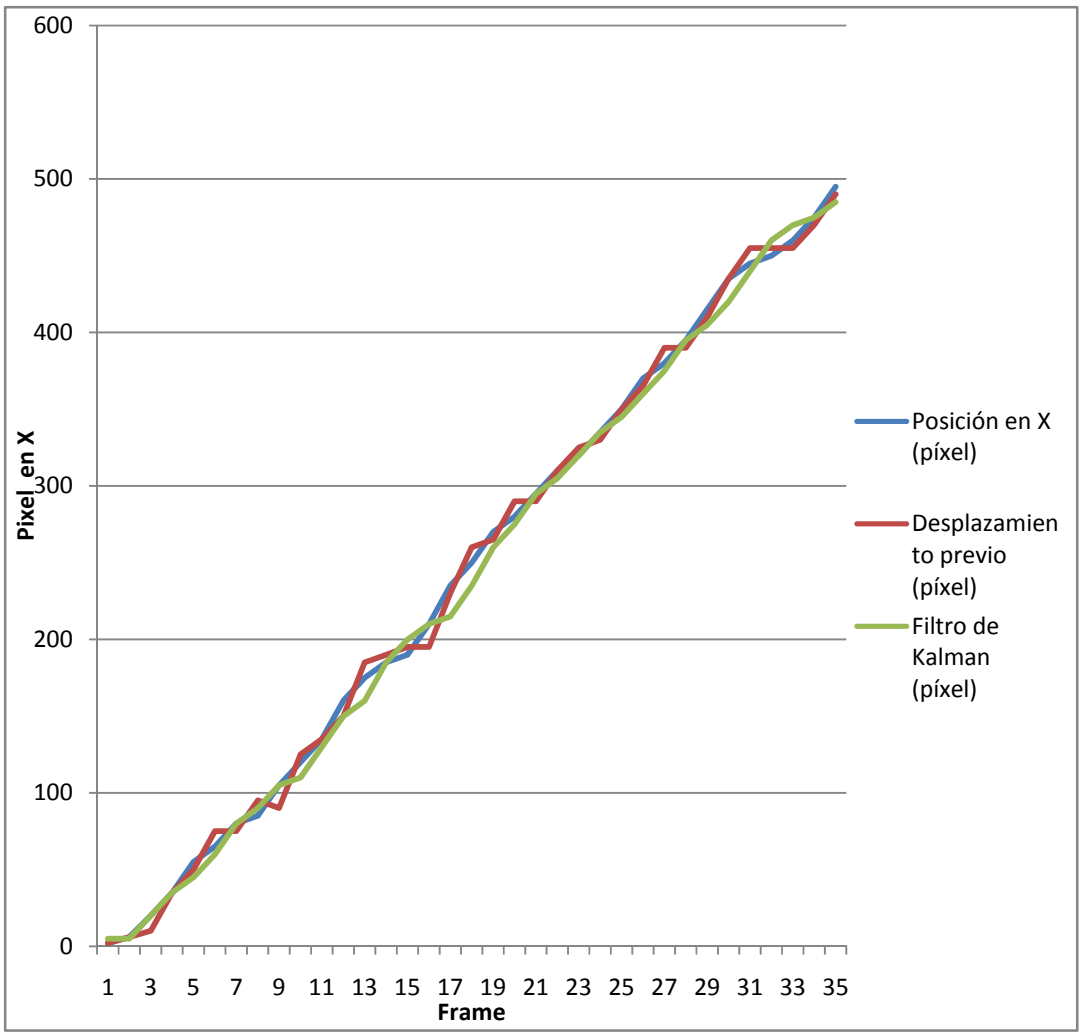

Figura 21 Gráfico comparativo con datos reales en la mañana

Desde 12h30 hasta 13h40 pm

Tabla 5 Vehículos en movimiento desde 12h30 hasta 13h40 pm

Vehículos que aparecen en el video	201
Vehículos que detecta el sistema	200
Porcentaje de vehículos detectados	99.5%
Objetos en movimiento en el video (personas, animales, etc)	18
Objetos en movimiento detectados por el sistema	0

Tabla 6 Medición de desempeño desde las 12h30 hasta las 13h40 pm

Frame	Posición en X (píxel)	Desplazamiento previo (píxel)	Filtro de Kalman (píxel)
1	4	4	0
2	5	5	8
3	8	6	8
4	12	11	16
5	20	16	32
6	52	28	48

7	68	84	72
8	80	84	80
9	88	92	88
10	116	96	120
11	128	144	128
12	140	140	136
13	156	152	152
14	168	172	176
15	196	180	192
16	208	224	200
17	220	220	216
18	248	232	248
19	252	276	264
20	276	256	280
21	292	300	288

22	300	308	304
23	316	308	320
24	320	332	328
25	336	324	344
26	360	352	360
27	372	384	376
28	392	384	392
29	396	412	400
30	420	400	416
31	432	444	432
32	440	444	440
33	452	448	456
34	468	464	464
35	480	484	480

La Figura 21 muestra una imagen tomada al mediodía.

Figura 22 Imagen obtenida entre 12h30 y 13h40 am.

La Figura 23 muestra un gráfico comparativo de las posiciones en X detectadas, predicción del filtro de desplazamiento previo y del filtro de Kalman. Las abscisas corresponden al número de frame y las ordenadas al píxel en X.

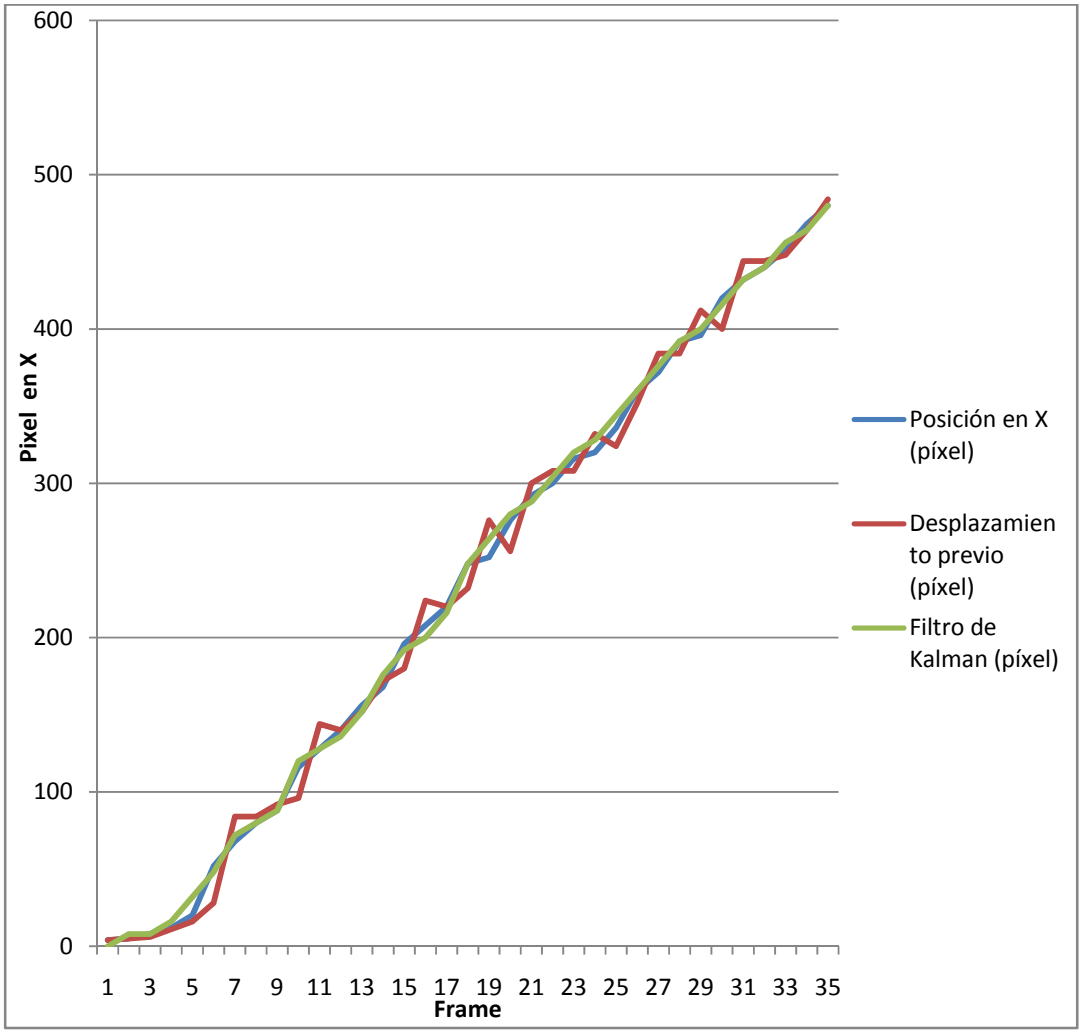

Figura 23 Gráfico comparativo con datos reales al mediodía

Desde 18h30 hasta 19h00 pm

Tabla 7 Vehículos en movimiento desde 18h30 hasta 19h00 pm

Vehículos que aparecen en el video	56
Vehículos que detecta el sistema	37
Porcentaje de vehículos detectados	66%
Objetos en movimiento en el video (personas, animales, etc)	3
Objetos en movimiento detectados por el sistema	0

Tabla 8 Medición de desempeño desde las 18h30 pm hasta las 19h00 pm

Frame	Posición en X (píxel)	Desplazamiento previo (píxeles)	Filtro de Kalman (píxeles)
1	12	6	3
2	15	9	3
3	21	18	9
4	24	24	24
5	33	42	45
6	39	60	42

7	57	66	51
8	69	84	75
9	75	90	105
10	84	99	99
11	99	114	123
12	105	129	114
13	108	135	168
14	114	141	132
15	135	162	135
16	144	171	165
17	150	177	135
18	159	186	189
19	165	192	204
20	183	210	177
21	195	222	210

22	201	228	195
23	204	231	240
24	219	246	213
25	234	261	207
26	252	279	243
27	264	291	252
28	279	306	291
29	288	315	330
30	297	261	339

La Figura 22 muestra una imagen obtenida en la noche, donde el ruido y la falta de iluminación se hacen presentes.

Figura 24 Imagen obtenida entre 18h30 y 19h00 pm.

La Figura 25 muestra un gráfico comparativo de las posiciones en X detectadas, filtro de desplazamiento previo y del filtro de Kalman. Las abscisas corresponden al número de frame y las ordenadas al píxel en X.

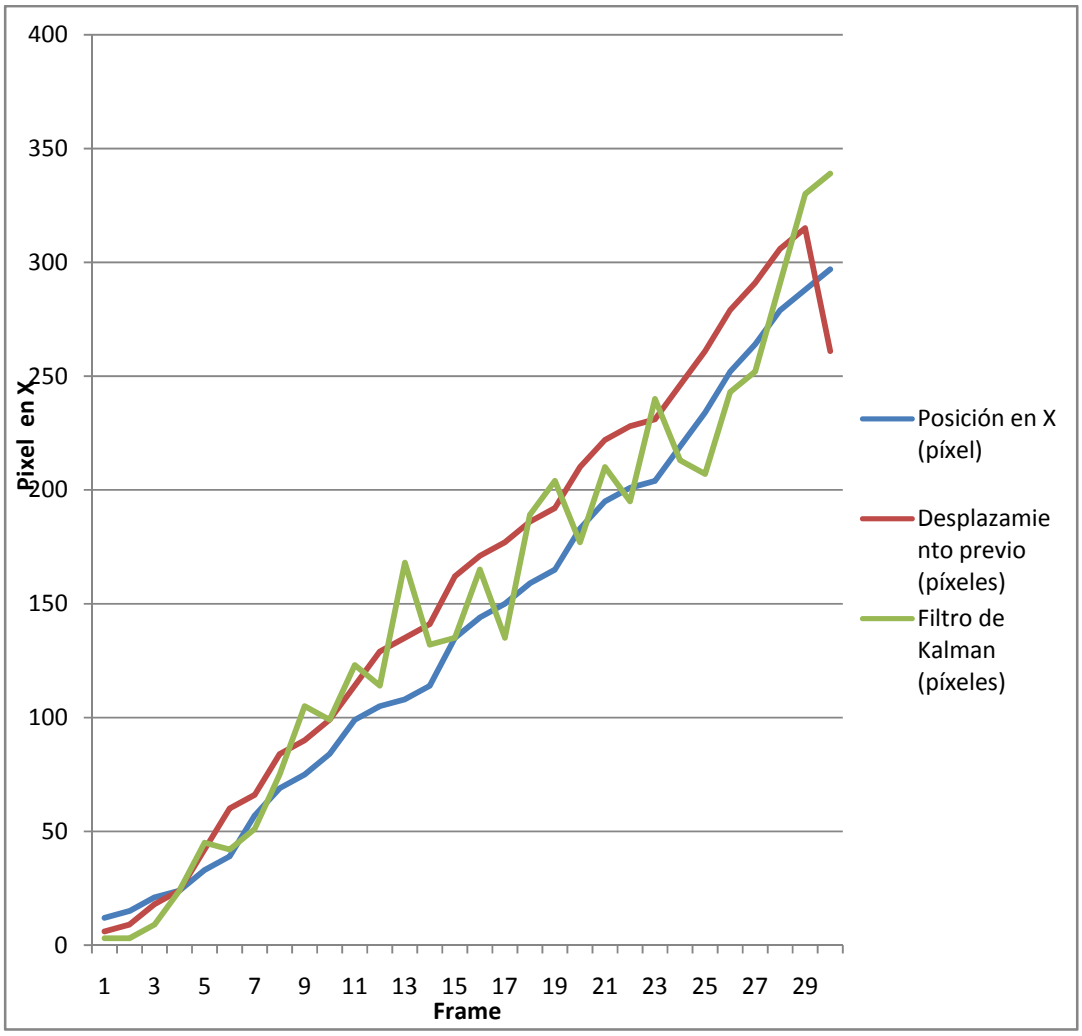

Figura 25 Gráfico comparativo con datos reales en la tarde

Se implementaron funciones en Matlab que aplican el filtro de Kalman y el de desplazamiento previo recibiendo una muestra de movimiento. Se generó una muestra aleatoria y se la introdujo en cada funcion, siendo monitoreada la ocupacion de recursos de sistema asignados a Matlab por cada uno de ellos. La Figura 26 muestra el resultado de la observación, donde las abscisas muestran el tiempo en centisegundos y las ordenadas el porcentaje de ocupación de recursos.

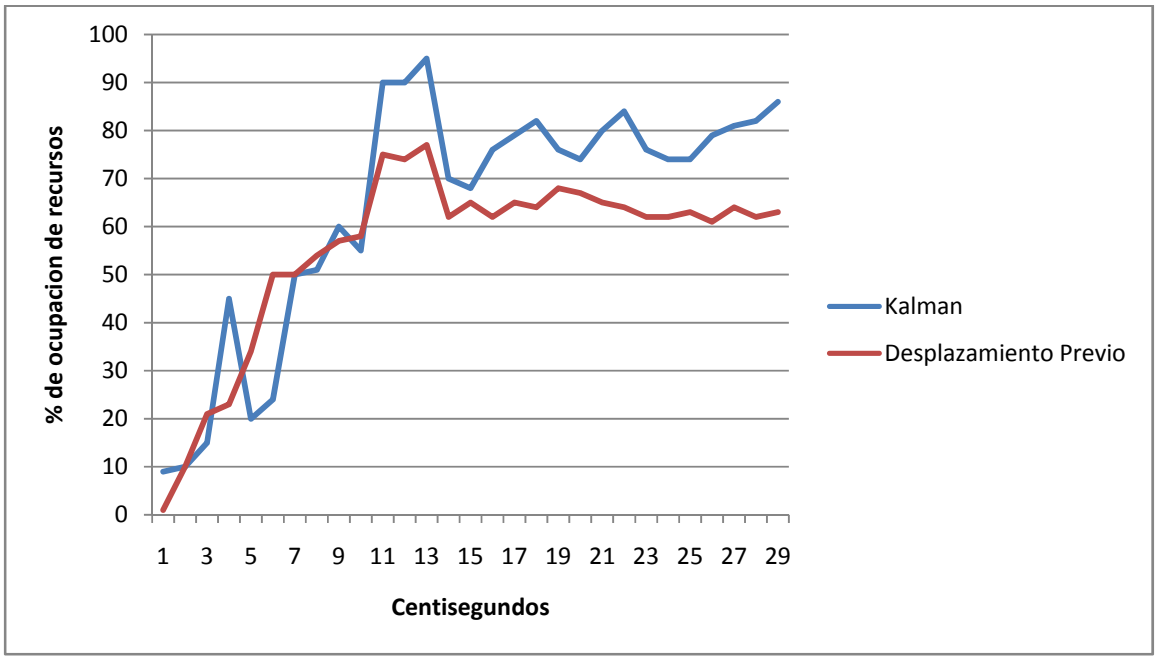

Figura 26 Gráfico comparativo de uso computacional: Kalman y Desplazamiento previos

5.4 ANÁLISIS DE RESULTADOS

Después de realizar las diferentes pruebas pudimos observar que la detección más cercana al resultado real ocurría en el horario del medio día (200 de 201 detectados, correspondiente al 99.5%), por lo que a esa hora la visibilidad fue la mejor de los tres casos. Como se aprecia también, el peor desempeño del sistema ocurrió en el horario de 18h30 a 19h00 con tan solo un 66% de vehículos existentes siendo detectados. Esto se debe a la mala iluminación y por ende la enorme cantidad de ruido presente y falta de información de calidad en los datos de entrada. Sin embargo, este problema podría ser solucionado si una cámara con iluminación infrarrojo es usada, ya que de esta forma la iluminación podría mejorar considerablemente.

Se pudo determinar casos especiales donde el sistema es incapaz de detectar un vehículo si es demasiado grande, como el caso de un autobús, ya que en este caso el automotor cubre el área de detección de manera absoluta, por lo que el sistema se ve imposibilitado de encerrarlo en una caja englobante.

Por otro lado, el movimiento de ramas y pequeños objetos, los cuales generan ruido, fue manejado de gran manera por el módulo de segmentación, que descartó adecuadamente ese movimiento sin afectar el correcto seguimiento de vehículos. La Figura 27 muestra una secuencia de imágenes con movimiento de ramas. Ponga especial atención a la rama

ubicada en la parte inferior izquierda del frame (encerrada en el círculo azul), ya que esta se mueve.

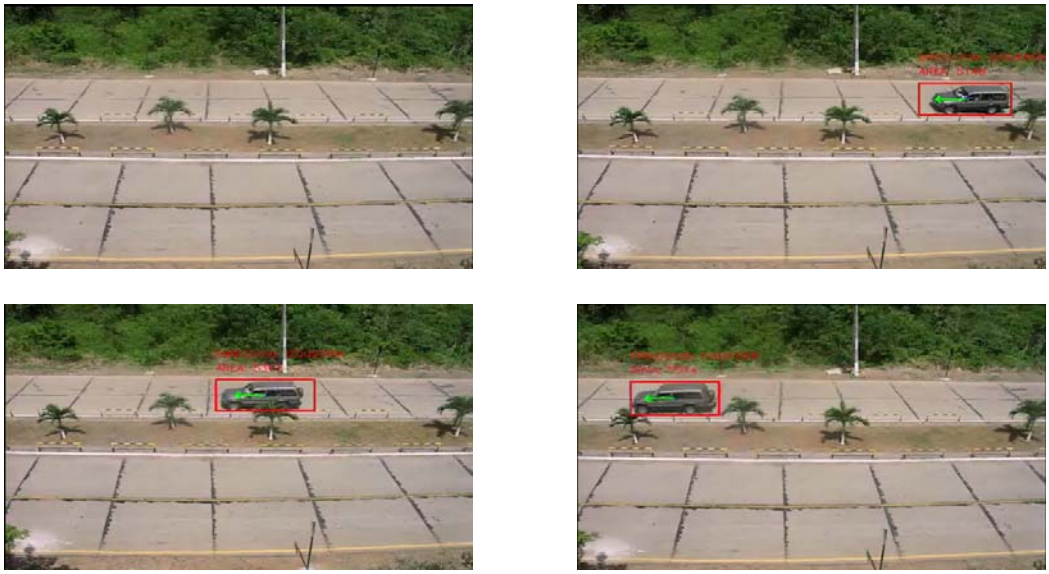

Figura 27 Secuencia de imágenes con rama en movimiento (encerrada dentro de círculo azul)

A su vez, se pudo constatar que el sistema detecta exclusivamente a vehículos. La presencia de otros objetos, como personas o animales es obviada con gran eficacia. La Figura 28 muestra a una moto (encerrada en círculo azul) y a un vehículo y como es ignorada la motocicleta por el sistema, mientras que si detecta al vehículo.

Figura 28 Una moto cruzando y auto, siendo solo el automotor detectado

La Figura 29 muestra a un perro (encerrado en un círculo azul) entrando en el área de deteccion mientras un vehiculo está a punto de terminar su paso. Se destaca que solamente el vehículo es detectado.

Figura 29 Perro no siendo tomado en cuenta por el sistema sin afectar el seguimiento de un vehículo

En los resultados mostrados se pudo apreciar que el estado pronosticado por el filtro de Kalman coincide en muchos casos con la medida real obtenida. La técnica del desplazamiento previo provee también una aproximación muy buena, pero falla en ciertos puntos, desviándose de la trayectoria real.

Por otro lado, la alta independencia y robustez del filtro de Kalman, comparado con el filtro del desplazamiento previo le otorgan ventaja a la hora de seleccionar un filtro predictor adaptativo.

A favor del filtro adaptativo podemos indicar que los recursos empleados por éste son menores que los empleados por la técnica del filtro adaptativo de Kalman. Por lo que su

ejecución e implementación demandarían una inversión menor por parte del desarrollador.

Ambos comparten la ventaja de ser totalmente indiferentes al sistema que se está moldeando, por lo que la definición del sistema, muchas veces extremadamente complicada o costosa, no es necesaria.

Podemos encontrar varios puntos de inflexión entre la curva real del movimiento y la predicha por el filtro de Kalman. Aunque el segundo filtro también acierta en algunas de las predicciones, el filtro de Kalman predice un mayor número de puntos congruentes exactamente con los del movimiento real.

6. CONCLUSIONES

Mediante análisis detenido conclusiones importantes pueden determinarse para este estudio.

- Los resultados obtenidos con el sistema desarrollado muestran altos porcentajes de efectividad en diferentes horarios del día, sobre todo en los horarios de la mañana y tarde.
- Es importante mencionar que el filtro de Kalman fue seleccionado como predictor final. Muchas características sobresalientes, como la alta precisión de sus aproximaciones, el relativo bajo costo computacional que ofrece, la independencia completa con respecto al sistema al cual trata de modelar y su increíble flexibilidad y adaptación, lo hacen un algoritmo extremadamente útil y versátil.
- Muchos algoritmos de extracción de background que no han sido discutidos aquí asumen que el background no varía y entonces puede ser capturado previamente. Esto limita enormemente su utilidad en la implementación práctica. Muy pocos documentos describen y discuten su algoritmo con el suficiente detalle para que sea fácil su reimplementación.
- Aunque el sistema es bastante económico en lo que a recursos computacionales se refiere, es necesario que corra bajo ciertos parámetros estrictos de exclusividad de uso sobre un procesador de alto rendimiento. Sin embargo, esto no debería ser un

gran inconveniente, ya que las compañías encargadas de construcciones viales y la empresa pública manejan cifras mucho más altas para este tipo de estudios.

- Por otro lado, los resultados obtenidos muestran una alta aproximación a lo real, con lo que el sistema queda justificado. Es decir, se ha podido obtener un sistema robusto y efectivo, que podría brindar asistencia para el control de carreteras rurales o estudio del comportamiento vehicular de las mismas.
- Es importante resaltar que debido a factores ajenos a la programación y algoritmos empleados en este sistema, el proyecto no responderá de manera aceptable en ambientes donde no existe una buena iluminación. Esto se observó claramente al responder el sistema deficientemente en el horario nocturno. Sin embargo, también se debe recalcar que este problema podría ser solucionado si una cámara con iluminación infrarroja es usada, ya que de esta forma se mejoraría la iluminación de la noche. Esto podría ser una propuesta de mejora para un trabajo futuro.
- Por otro lado, el sistema tampoco podrá actuar de buena forma si se encuentra bajo una alta concentración de ruido externo, tal es el caso de lluvia o tormentas de polvo, ya que se agruparían como regiones grandes en movimiento debido a la cercanía de las gotas o partículas de polvo.
- Dentro de un ambiente medianamente controlado, con poco ruido externo, el sistema brinda su mejor actuación. Debido a este factor concluimos que es mejor

que se utilicen videos grabados y de ser posible con algún tipo de filtrado de ruido. De esta forma se garantizaría con un mayor grado de certeza el mejor desempeño posible.

- El sistema es capaz de ignorar de excelente manera a objetos pequeños en movimiento, como perros, personas y ramas mediante el umbral de áreas mínimas y máximas. Gracias a esto se pudo determinar con exactitud si un objeto era un vehículo o no.
- Los cambios de iluminación repentinos no afectaron mayormente el desempeño del sistema. Esto se debe a la gran ventaja que nos brinda la mezcla de distribuciones Gaussinas. Al establecer un fondo adaptativo en constante modificación, los cambios son imperceptibles para el procesamiento al resultar siempre un cambio gradual y controlable.

7. RECOMENDACIONES

Extrayendo experiencias adquiridas antes, durante y después del desarrollo de este proyecto podemos recomendar lo siguiente:

- Es imperativo proporcionar siempre iluminación adecuada para que el sistema funcione de la mejor manera posible. En caso contrario, se puede obtener un video con una gran cantidad de ruido externo, factor que afectaría de manera terrible los resultados.
- Sería una excelente mejora al sistema la adaptación correcta a procesamiento en tiempo real, haciendo que de esta forma el sistema sea más versátil.
- Una mayor resolución en las características de la cámara sería ideal para un mejor detalle en el contraste de la imagen del video. De esta característica dependen mucho la binarización y extracción de ruido. Sin embargo hay que considerar que esto incrementaría el costo computacional, pues el tamaño de la imagen se incrementaría teniéndose con esto que procesar mayor cantidad de datos en cada imagen.
- Un dispositivo de preprocesamiento capaz de aislar las imágenes que serán utilizadas por el sistema facilitaría enormemente el trabajo de procesamiento, ya que el porcentaje de recursos computacionales utilizados por el sistema se vería altamente reducido, mejorando entonces el rendimiento final del proyecto.

- Recomendamos tener en consideración que el filtro de segmentación es directamente dependiente del tamaño del frame. Es por esto que cuando se trabaja con diferentes cámaras, es necesario realizar pequeños ajustes en los parámetros.
- Es muy recomendable definir un vector de estado inicial lo más completo y robusto posible para el filtro de Kalman. Es decir, uno que pueda en sus diferentes combinaciones describir de la forma más exacta las posiciones y movimientos que la señal realiza. A mayor robustez, el filtro responderá de una mucha mejor forma a las posteriores correcciones realizadas, arrojando recursivamente mejores predicciones futuras.
- Para una óptima actualización del fondo, se debe buscar la combinación perfecta de los parámetros que manejan el filtro adaptativo Gaussiano. Es común encontrar parámetros estándar en Internet, pero es mucho mejor definir personalmente las entradas, para así asegurar un resultado eficiente y válido.
- Al momento de hacer uso de la aplicación, es necesario mantenerse dentro de las restricciones del sistema, considerando en gran parte la distancia de enfoque, altura y ángulo de inclinación de la cámara.
- La mayoría de los tipos de datos utilizados en OpenCV son punteros, y tenemos que recordar hacer la liberación de memoria respectiva, para que no se sobrecargue la memoria principal y la memoria virtual. De hecho, este fue uno de los problemas que se suscitó en el desarrollo del prototipo, pues no se tomó las

debidas consideraciones del caso, y luego se tuvo que regresar a revisar meticulosamente el código. Tiempo se perdió, por lo que siempre es mejor prever. Cabe también mencionar que la memoria se acabaría rápidamente, pues trabajamos con secuencias de imágenes, en caso de no liberar memoria.

- Finalmente, también quisiéramos dejar planteado como recomendación el uso de este sistema con más frecuencia sobre videos reales, ya que de esta forma futuros problemas no considerados podrían aparecer, lo cual serviría para continuar mejorando el sistema aquí implementado.

BIBLIOGRAFIA

1. N. Siebel, **"The Reading people tracker", proyecto en ejecución, para más información visite: http://www.siebelresearch.de/people_tracking/reading_people_tracker/**
2. D.Hall, J. Nascimento, P. Ribeiro, E. Andrade, P. Moreno, S. Pesnel, T. List, R. Emonet, R.B. Fisher, J. Santos Victor y J.L. Crowley, **"Comparison of target detection algorithms using adaptive background models"** Proceedings on 2nd Joint IEEE International Workshop on Visual Surveillance and tracking, 2005, pg. 113-120.
3. M. Piccardi, **"Background subtraction techniques: a review"** Proceedings of 2004 Systems, man and cybernetics, volume 4, 2004, pg. 3099-3104.
4. A. Efros, **"Rendering and Image Processing"** tomado de la biblioteca de Ciencias de la Computacion, de la facultad de Computacion de la universidad de Carnegie Mellon, última versión en http://www.cs.cmu.edu/~efros/
5. D. Simon, **"Kalman filtering"** página web de investigacion Scribd, última version en http://www.scribd.com
6. D. Bundhwar, S. Sandhu, A. Karna **"Target tracking using Kalman filter"** página web de investigacion Scribd, última version en http://www.scribd.com

7. D. Peña, **“Sobre la robustificación interna del algoritmo de Plackett-Kalman para la estimación recursiva del modelo de regresión lineal”** Trabajos de estadística e investigación operativa, Springer Berlin / Heidelberg, Berlín, Alemania, Julio 1985, pg. 93-106.
8. R. Kalman, “**A New Approach to Linear Filtering and Prediction Problems**” Transactions of the ASME – Journal of Basic Engineering, revista. 82, 1960, pg. 35-45
9. J. Perrin, S. Volino, **“Gaussian model for localized translational motion: Application to incoherent neutron scattering”** The European Physical Journal: Special topics, Springer Berlin / Heidelberg, Berlín, Alemania, Julio 2007, pg. 57-60.
10. A. Calway, R. Wilson, “**Multiresolution Gaussian mixture models for visual motion estimation**” Proceedings 2001 International Conference on Image Processing, volumen 2, 2001, pg. 921-924.
11. D.S.G. Pollock, **“The Kalman Filter”** Computational Statistics and Data analisis, volumen 50, 1 May 2006, pg. 2137-2145.
12. W. Zhang, X. Fang, W. Lin **“Moving vehicles segmentation based on Gaussian motion mode”** Visual Communications and Image Processing 2005, volumen 5960, 2005, pg. 141-148.

13. G. Halevy, D. Weinshall **"Motion of disturbances: detection and tracking of multibody non-rigid motion"** 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volumen 17, 1997, pg.897-902.
14. C. Wren **"Pfinder: Real time tracking of the human body"** IEEE Transactions on Pattern Analysis and Machine Intelligence, volumen 19, 1997, pg. 780-785.
15. Y. Huang **"A vision based vehicle to vehicle detection and tracking system"** ICPR 2006. 18th International Conference on Pattern Recognition, volumen 2, 2006, pg. 1070-1073.
16. N. Kanhere, S. Pundlik, S. Birchfield **"Vehicle segmentation and tracking from a low able off axis camera"** Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volumen 2, 2005, pg. 1152-1157.
17. H. Fujiyoshi, A. Lipton, T. Kanade **"Real time human motion analysis by image skeletonization"** 4th IEEE Workshop on application on Computer Vision, 1998, pg. 15-21.
18. M. Bertozzi, A. Broggi, S. Castelluccio, **"A real-time oriented system for vehicle detection"** Special quintuple issue: Euromicro 1995 short contributions, volume 43, 1997, pg.317 -325.
19. Y. Du, N.P. Papanikolopoulos, **"Real-time vehicle following through a novel symmetry-based approach"** Proceedings on IEEE International Conference on Robotics and Automation, volumen 4, 1997, pg. 3160-3165.

20. C. Hoffmann, T. Dang, C. Stiller, **“Vehicle detection fusing 2D visual features”** 2004 IEEE Intelligent Vehicles Symposium, Volumen 14, junio 2004, pg. 280-285.
21. Hu, Z., and K. Uchimura, **“Tracking cycle: a new concept for simultaneous tracking of multiple moving objects in a typical traffic scene”** EURASIP Journal on Applied Signal Processing, 2005, pg. 2322-2329.
22. Huang, S.-S., C.-J. Chen, P.-Y. Hsiao, L.-C. Fu, **“On-board vision system for lane recognition and front-vehicle detection to enhance driver's awareness”** 2004 Proceedings of ICRA IEEE International Conference on Robotics and Automation, volumen 3, 2004 pg. 2456-2461.
23. Wu, J. and X. Zhang, **“A PCA classifier and its application in vehicle detection”** 2001 Proceedings of International Joint Conference on Neural Networks, IJCNN, volumen 1, 2001, pg. 600-604.
24. S.-W. Seol, J.-H. Jang, H.-S. Kim, C.-H. Lee, and K.-G. Nam, **“An automatic detection and tracking system of moving objects using double difference based motion estimation”** Proceedings of International Technical Conference on Circuits/Systems, Computers and Communications, 2003, pg. 260-263.
25. J. Suhr, J. Hong, **“Automatic free parking space detection by using motion stereo-based 3D reconstruction”** Machine Vision and Applications, Springer Berlin / Heidelberg, Berlín, Alemania, Julio 2008.

26. A.Argyros, S. Orphanoudakis, "**Independent 3D Motion Detection Based on depth elimination in normal flow fields"** IEEE Computer Society Conference Proceedings of Computer Vision and Pattern Recognition, Revista 17, Jun 1997 pg. 672-677.
27. W. Grimson, C. Stauffer, R. Romano, **"Using adaptative tracking to classify and monitor activities in a site"** IEEE Computer Society Conference Proceedings of Computer Vision and Pattern Recognition, Revista 18, Jun 1998 pg. 22-29.
28. D. Beymer, P. McLauchlan, B. Coifman, J. Malik **"A real time computer vision system for measuring traffic parameters"** IEEE Computer Society Conference Proceedings of Computer Vision and Pattern Recognition, Revista 17, Jun 1997 pg. 495-501.
29. T. Svoboda **"Kanade Lucas Tomasi tracker"** Proceedings of 13th International Conference on Computer Communications and Networks ICCCN, Revista 13, Agosto. 2004 pg.904-909
30. M. Shih, Y. Chang, B Fu, C Huang "**Motion-based Background Modeling for Moving Object Detection on Moving Platforms"** Proceedings of 16th International Conference on Computer Communications and Networks ICCCN, Revista 16, Aug. 2007 pg.1178-1182

31. A. McIvor **“Background Subtraction Techniques”** Proceedings of Image and Vision computing, Reveal Limited, Auckland Nueva Zelanda, 2000.
32. C. Stauffer, W. Grimson **“Adaptive background mixture models for real-time tracking”** 1999. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999, Vol. 2.
33. D. Koller, K. Daniilidis, H.-H. Nagel **“Model-Based Object Tracking in Road Traffic Scenes”** International Journal of Computer Vision 10:3, 1993, pg. 257-281.
34. J. Boreczky, L. Wilcox **“A hidden Markov model framework for video segmentation using audio and image”** Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, 1998, pg. 3741-3744.
35. C. Anderson, P. Burt, G. van del Wal **“Change detection and tracking using pyramid transformation techniques”** SPIE – Intelligent robots and computer vision, vol. 579, 1985, pg. 72-78.
36. J. Barron, D. Fleet, S. Beasuchemin **“Performance of optical flow techniques”** Int. J. Computer Vision, vol. 12, enero 1994, pg.42-77.
37. I. Haritauglu, D. Harwood, L.S. Davis **“W^4 Who? Where? When? What? A real time system for detecting and tracking people”** FGR98, 1998, pg. 222-227.

38. G. Cohen, G. Medioni **“Multi-views tracking within and across uncalibrated camera streams”** UC Berkeley First ACM SIGMM international workshop on Video surveillance, 2003, pg. 21-23

39. P. KaewTraKulPong, R. Bowden **“An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection”** Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01, 2001, pg. 25-28.