Algoritmo Rápido de Seguimiento de Objetos

Trabajo de Grado Desarrollado Por:

John Paul Lemon Urriago

Stephanie Jane Vandenberg Torres

Dirigido Por:

Ing. Francisco Calderón Ph.D.

Pontificia Universidad Javeriana
Facultad De Ingeniería
Departamento de Electrónica
2019-I

Lista de figuras

Lista de tablas

Tabla de contenidos

Abstract: *The objective of the present work is to reduce the computational complexity of an algorithm for detecting, identifying and tracking objects in video sequences. For the fulfillment of this objective, two methods are proposed: particle filter and optical flow. In this document, a contextualization of the technique on which the tracking algorithm of the Bogota Mobility Secretariat is based (YOLO), and a detailed description of the two methods proposed is made. Subsequently, a test protocol is proposed to guarantee the compliance of the proposed objectives and relevant tests are made.*

Resumen: *En el presente trabajo de grado se propone reducir la complejidad computacional de un algoritmo de detección, identificación y seguimiento de objetos en secuencias de video. Para el cumplimiento de este objetivo, se proponen dos métodos: filtrado de partículas y flujo óptico. En este documento, inicialmente se hace una contextualización de la técnica en la que se basa el algoritmo de seguimiento de la Secretaría de Movilidad de Bogotá (YOLO) y se hace una descripción detallada de los dos métodos propuestos. Posteriormente, se propone un protocolo de pruebas que permita garantizar el cumplimiento de los objetivos propuestos y se realizan las pruebas pertinentes.*

1. INTRODUCCIÓN

Los sistemas de CCTV (Circuito Cerrado de Televisión) son los sistemas de monitoreo más utilizados en el mundo. Sin embargo, la supervisión de los mismos se lleva a cabo por operadores humanos, lo cual conlleva a falencias en los resultados esperados. Se llevaron a cabo experimentos en *Sandia National Laboratories* para el Departamento de Energía de Estados Unidos [1]. El objetivo de este experimento era probar la efectividad de una persona cuya tarea era sentarse frente a un monitor de video durante varias horas al día y observar eventos particulares. Estos estudios demostraron que dicha tarea, aun cuando era asignada a una persona dedicada y bien intencionada, no generaba los resultados esperados. Después de sólo 20 minutos de ver y estudiar los videos, la atención de la mayoría de personas se había degenerado significativamente.

En muchos contextos como el mencionado, se identifica que un monitoreo automático para el análisis de videos de CCTV puede repercutir en mayores beneficios para las organizaciones. Un área en la que es útil el monitoreo automático es en control de tráfico. Según la Secretaría de Movilidad de Bogotá (SDM), a 2015 se tiene un total de 2'148.541 vehículos registrados [2], los cuales circulan por las calles de la ciudad, generando un gran flujo a ciertas horas del día. El análisis del flujo de vehículos es un punto de partida para el control del tráfico, y este proceso debe llevarse a cabo de manera eficiente y efectiva [3].

En la ciudad de Bogotá, se ha presentado un incremento anual superior a los 100.000 vehículos, especialmente para los últimos 5 años [2]. Este aumento de vehículos en las vías hace que la gestión del transporte se convierta en un problema cada vez más difícil e importante de gestionar. Un mecanismo que permite mitigar este inconveniente es el control del tráfico [4].

El conteo y flujo vehicular refleja la situación en una ruta determinada [3]. Basándose en la densidad de vehículos en las calles, se hace posible desarrollar un mecanismo que mejore la movilidad. Por ejemplo, sabiendo el número de vehículos que viajan en una ruta en particular, es posible obtener predicciones de embotellamientos, lo que permite tomar medidas, tales como ajustar el tiempo de los semáforos [4]. Además, al tener un sistema que pueda identificar el tipo de vehículo, se facilita el trabajo en campos tales como vigilancia del tráfico, peajes, ingeniería de vías, etc [3].

A lo largo de los años, se han utilizado diferentes métodos para obtener información del tráfico. Sin embargo, los sistemas basados en imágenes han ganado reconocimiento rápidamente sobre sensores electrónicos tales como bucles inductivos debido a su relación costo-efectividad, su portabilidad y su facilidad de mantenimiento [5]. Cuando se usan cámaras de video, la cantidad de datos que se puede obtener aumenta significativamente.

Para poder extraer información de las imágenes obtenidas, es necesario utilizar sistemas sofisticados con alta precisión, los cuales tienen un alto costo computacional [6]. Por lo tanto, su uso en las grandes ciudades se ve obstaculizado porque el costo asociado con la implementación es alto [4]. El objetivo de este trabajo de grado consiste en hacer un sistema eficiente que requiera menos potencia computacional y que brinde una similar certeza en la detección, identificación y seguimiento de vehículos.

El primer paso en este algoritmo consiste en la detección e identificación de objetos de interés, para la cual se utiliza el algoritmo YOLO *(You Only Look Once),* el cual se describe brevemente en el capítulo 3. De igual manera, en esta sección se encuentra una breve contextualización de los métodos utilizados en este trabajo de grado (filtrado de partículas y flujo óptico). En el capítulo 4 se hace una descripción más detallada de cada una de las técnicas implementadas. Una vez explicado el proceso, en el capítulo 5 se propone un protocolo de pruebas, con el cual se pretende garantizar que los resultados del presente trabajo de grado cumplan con los objetivos propuestos. En el capítulo 6 se muestran dichos resultados, así como un análisis de estos. Por último, en el capítulo 7, se mencionan las conclusiones a las que se llegaron, así como posibles trabajos futuros que brinden mejoras al actual proyecto.

2. OBJETIVOS

2.1. Objetivo general:

Reducir la complejidad computacional de un algoritmo de seguimiento de objetos en secuencias de video, a partir de un seguimiento rápido de rectángulos contenedores.

2.2. Objetivos específicos:

1. Generar y almacenar la trayectoria de los objetos en una base de datos de vídeo de la Secretaría de Movilidad.
2. Implementar las técnicas de descriptores de características y filtrado de partículas para acelerar el seguimiento de objetos.
3. Obtener al menos un 30% de mejora en la rapidez de ejecución del algoritmo bajo las mismas condiciones computacionales, permitiendo un error de ±10% en el conteo comparado con un algoritmo de tracking cuadro a cuadro.

A partir del desarrollo e implementación de este trabajo de grado, se fueron evaluando cada uno de los objetivos específicos, los cuales llevan al cumplimiento del objetivo general. Se explicará la solución planteada a cada objetivo para dar un esquema de cómo se llevó a cabo.

2.2.1. Objetivo específico 1.

A partir de un sistema de detección de objetos cuadro a cuadro utilizado por la Secretaría de Movilidad de Bogotá, se identifican objetos de interés en una base de datos de video. Se realizan mediciones tales como el tiempo de ejecución del algoritmo, el conteo de los objetos detectados, utilizando una o varias líneas de conteo y se almacenan las trayectorias de cada objeto. Estas medidas se utilizan como punto de comparación con las técnicas implementadas.

2.2.2. Objetivo específico 2.

Se implementan dos técnicas para acelerar el seguimiento de objetos: filtrado de partículas y una técnica basada en descriptores de características, en este caso se utiliza flujo óptico.
Inicialmente se eligió como segunda técnica CAMShift, pero al ser implementada se identificó que esta técnica era bastante robusta y pesada computacionalmente y no permitía el cumplimiento del objetivo general. Por esta razón, se cambió por la técnica de descriptores de características.
Estas técnicas fueron implementadas en 4 de cada 5 cuadros, en el cuadro restante, se utiliza el algoritmo de detección e identificación actual.

2.2.3. Objetivo específico 3.

Después de implementar las técnicas mencionadas, se debe verificar que se obtenga al menos un 30% de mejora en la rapidez de ejecución del algoritmo bajo las mismas condiciones computacionales, permitiendo un error de ±10% en el conteo comparado con un algoritmo de tracking cuadro a cuadro. Para esto, se utilizaron 8 videos de 15 minutos cada uno de la base de datos obtenida por la Secretaría de Movilidad y se midió y comparó el tiempo de ejecución de cada uno de ellos con las diferentes técnicas. Además, se comparó el conteo de cada técnica utilizando la misma línea de conteo en cada video.

3. MARCO TEÓRICO

3.1. *Opencv software*

Opencv es una librería de software de código abierto (*open-source* software) especializada en visión artificial (*computer vision*) y aprendizaje automático (*machine learning*). Esta librería fue realizada para proporcionar una infraestructura para aplicaciones de *computer vision* y *machine learning*. Además, para promover e impulsar el uso de la percepción de la máquina en productos comerciales. Opencv tiene soporte para lenguajes de programación como C++, Python, Java y MATLAB. Otra de sus particularidades es el soporte multiplataforma en las que se encuentran Windows, Linux, Android y Mac OS. Debido a que es de código abierto, posee muchas colaboraciones, encontrándose así más de 500 funciones. Sus funciones principales pueden servir para identificación de caras, identificación de objetos de interés, clasificar diferentes acciones humanas en video, entre otras. Así como muchas compañías (Google, Microsoft, Intel, IBM) emplean la librería, también existen muchas nuevas empresas y millones de desarrolladores a través de todo el mundo. De las aplicaciones más representativas se encuentran detección de intrusos en videos de vigilancia en Israel, ayudar a robots a navegar y agarrar objetos en Willow Garage y más [7].

3.2. Identificación y seguimiento de objetos.

3.2.1. Identificación

El algoritmo propuesto de seguimiento rápido puede ser implementado con diferentes técnicas de detección. Para este trabajo de grado, se partió del código fuente provisto por la Secretaría de Movilidad de Bogotá (disponible en: www.urban-dataset.com), la mayoría de los algoritmos de identificación y detección de objetos tienen salidas que son compatibles con las técnicas propuestas.

3.2.1.1. YOLO

YOLO (por sus siglas en inglés *You Only Look Once*) es un sistema de detección de objetos del estado del arte. Con esta técnica, se aplica una sola red neuronal a la imagen completa. Esta red divide la imagen en regiones y predice rectángulos de límites y probabilidades para cada región. Estos rectángulos contenedores están ponderados por las probabilidades predichas [6].

Este modelo lee la imagen completa en el momento de la prueba, por lo que sus predicciones se basan en el contexto global de la imagen. También hace predicciones con una red neuronal única, a diferencia de los sistemas como R-CNN, que requieren miles de redes neuronales para una sola imagen [6].

Al ejecutarse este modelo, cada objeto en la imagen está contenido en un rectángulo, el cual incluye 5 elementos: (u, v, x, y). Estos puntos corresponden a la esquina superior izquierda (u, v) al centro del objeto (x, y) respectivamente. También se arroja un puntaje de confianza de cuadro. Este puntaje refleja la

probabilidad de que el rectángulo contenga un objeto y la precisión del rectángulo contenedor, tanto en ubicación como en clasificación [8].

3.2.2. Técnicas de seguimiento de objetos.

3.2.2.1. Filtrado de partículas (Particle filter)

El filtrado de partículas es uno de los métodos de seguimiento de objetos de interés más utilizado en los últimos tiempos, ya que utiliza métodos numéricos para resolver estimaciones en escenarios no lineales [9]. Este método es conocido también como filtro secuencial Monte Carlo. Este tipo de filtros están basados en modelos Bayesianos. Un filtro bayesiano es una herramienta matemática que permite retroalimentar la fórmula que calcula la probabilidad de un acontecimiento, con la experiencia adquirida anteriormente en casos similares. Arulampalam, Maskell, Gordon y Clapp [10], dan una breve explicación del planteamiento de un modelo bayesiano en un filtrado de partículas, como un seguimiento bayesiano no lineal.

En primera instancia se tiene que considerar el cambio del estado de la secuencia de $\{x_k, k \in N\}$ de un objeto de interés, definida como:

$$x_k = f_k(x_{k-1},\ v_{k-1}) \qquad (1)$$

Donde f_k: $R^{n_x} \times R^{n_v} \rightarrow R^{n_x}$ es una posible función del estado de x_{k-1}, y v_{k-1}, las cuales son un conjunto de variables independientes e idénticamente distribuidas (i.i.d.) de secuencia de ruido del proceso. n_x y n_v son dimensiones del estado y vectores de ruido del proceso. El objetivo de esta técnica es obtener una estimación del seguimiento de x_k a través de diferentes procesos o medidas. Este proceso se transforma de la ecuación anterior a:

$$z_k = h_k(x_k,\ n_k) \qquad (2)$$

Donde h_k: $R^{n_x} \times R^{n_n} \rightarrow R^{n_z}$ es una posible función no lineal, n_k es una secuencia de medición de v_{k-1}, n_z y n_n son dimensiones de las medidas de n_x y n_v. Lo deseado es estimar x_k a través de $z_{1:k}$.
Para construir una función de densidad de probabilidad (*pdf* por sus siglas en inglés, *probability density function*) se necesita, desde una perspectiva Bayesiana, calcular el grado de creencia en el estado x_k en el tiempo k, dados los diferentes valores de z_k, creando así la *pfd* $p(x_k|z_{1:k-1})$. Se asume como condición inicial de la *pdf* que $p(x_o)=p(x_o|z_o)$ del vector de estado está disponible, siendo z_o cuando no hay mediciones. A partir de este punto, se obtiene el primer *pdf* a través de la predicción y actualización de estas medias.

Suponiendo que a cierto tiempo *k-1* la *pdf* requerida existe, la predicción de la etapa involucra el sistema de la ecuación (1) se utiliza para obtener el estado anterior del *pdf* en el tiempo *k*, también conocida como la ecuación de Chapman-Kolmogorov:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \qquad (3)$$

La ecuación (1) describe un proceso de Markov de primer orden, es decir que probabilidad de que ocurra un evento depende solamente del evento inmediatamente anterior. Notando que en la ecuación (3), se puede tener que $p(x_k|x_{k-1},z_{1:k-1}) = p(x_k|x_{k-1})$. Por otra parte, la medida z_k ,en el tiempo k se vuelven disponibles, se utiliza para actualizar el anterior estado, a través de la regla de Bayes:

$$p(x_k|z_{1:k}) = p(z_k|x_k)p(x_k|z_{1:k-1}) / p(z_k|z_{1:k-1}) \qquad (4)$$

En la etapa de actualización, ecuación (4), z_k es usado para cambiar la anterior *pdf* y así obtener la nueva *pdf* del estado actual.

La particularidad de ésta técnica consiste en generar unas muestras aleatorias, llamadas partículas, a partir de una función de densidad de probabilidad (*pdf*). Estas partículas tienen un valor o peso, los cuales permiten estimar la futura *pdf* [11]. Se crean n partículas diferentes distribuidas aleatoria y uniformemente a partir de un punto o condición inicial. Una vez hecha esta distribución y posteriormente la asignación de pesos a cada partícula, se determina la validez de cada partícula. Para esto, se utiliza una métrica, la cual compara cada partícula con un determinado umbral. Si esa partícula no cumple con el umbral, será eliminada; aquellas partículas que cumplan el umbral se mantienen.

A pesar de ciertas partículas fueron eliminadas, el número total de partículas no disminuye, sino que se crean nuevas partículas a partir de las que quedaron, generando una nueva *pdf*. A este proceso se le llama remuestreo.
En el seguimiento de objetos, las partículas se crean inicialmente en la ubicación del objeto detectado en un paso previo. Una vez el objeto se mueve a través del tiempo y espacio, ocurre el proceso mencionado anteriormente. La función de densidad de probabilidad va a concentrar las partículas en la posición del objeto. Es por eso de la importancia del remuestreo, para mantener la *pdf* en el lugar que se quiere.

Sin el remuestreo, es decir manteniendo las viejas partículas funcionando, el peso de cada partícula cambia, y posiblemente la concentración de partículas en diferentes puntos que no son de interés aumente. Por otra parte, se altera la *pdf* y se genera una medida errónea, evitando que siga el objeto que se quiere. A este proceso se le llama "agotamiento de partículas" (*particle depletion*), donde las partículas de no interés toman un peso mayor (o una probabilidad mayor) que aquellas que se encuentran en el objeto de interés [12].

a b c

Figura 3.2.2.1-1. Agotamiento de partículas. (a) primeras partículas creadas, (b) segundas partículas creadas junto con las anteriores y (c) Terceras partículas creadas junto con las anteriores.

En la figura 3.2.2.1-1, se muestra cómo las partículas no se actualizan (remuestreo) y se genera agotamiento de partículas. Es decir, en cada cuadro se crean diferentes partículas aleatoriamente. Como se observa en la figura 3.2.2.1-1.b y 3.2.2.1-1.c, se mantienen intactas las partículas creadas en el recuadro anterior (con un color diferente en cada cuadro). Por consecuencia, el objeto a seguir se pierde, ya que la *pdf* creada es errónea debido al no remuestreo de las partículas. A pesar de que en el objeto a seguir existan algunas partículas, la concentración de éstas está en otra parte del cuadro, generando que estas tengan un peso mayor influyendo así en la nueva *pdf* para el siguiente cuadro.

3.2.2.2. Flujo óptico (Optical flow)

El flujo óptico es un modelo del movimiento aparente de un objeto en una escena, causado por el movimiento relativo entre una escena y un observador [13]. En general, el flujo óptico describe un campo vectorial, donde a cada píxel se le asigna un vector de desplazamiento, que apunta a donde se puede encontrar ese píxel en otra imagen. Esta técnica trabaja bajo dos principales suposiciones: las intensidades de píxel de un objeto no cambian entre cuadros consecutivos y los píxeles cercanos tienen movimiento similar [14].

Considere un pixel $I(x,y,t)$, el cual se mueve una distancia $(dx,\ dy)$ entre dos cuadros consecutivos en un tiempo dt. Si se cumplen las suposiciones mencionadas anteriormente, se puede afirmar:

$$I(x,y,t) = I(x+dx,\ y+dy,\ t+dt) \qquad (5)$$

Tomando la serie de Taylor de la parte derecha de la igualdad de (5), se tiene (6):

$$I(x,y,t) = \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \qquad (6)$$

De estas ecuaciones se deduce (7) y (8}):

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \qquad (7)$$

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \qquad (8)$$

Lo cual resulta en (9), la ecuación de flujo óptico:

$$I_x u + I_y v + I_t = 0 \qquad (9)$$

En la anterior ecuación, $I_x = \frac{\partial I}{\partial x},\ I_y = \frac{\partial I}{\partial y},\ I_t = \frac{\partial I}{\partial t}$ son las derivadas de la imagen, las cuales pueden ser calculadas. Sin embargo, u, v (las componentes x, y del flujo óptico de $I(x,y,t)$), son desconocidas. No se puede resolver una ecuación con dos incógnitas, esto es conocido como el problema de apertura. El problema de apertura se refiere al hecho de que el movimiento de una estructura espacial unidimensional, como una barra o un borde, no se puede determinar si se ve a través de una apertura pequeña, de manera

que los extremos no sean visibles [15]. Existen diferentes métodos para resolver este problema. Uno de ellos es Lucas-Kanade.

3.2.2.3. Método Lucas-Kanade

El algoritmo de flujo óptico Lucas-Kanade hace un estimado del movimiento de características interesantes en imágenes sucesivas de una escena [16]. Este algoritmo trabaja bajo las siguientes suposiciones:

- Las dos imágenes están separadas por un incremento de tiempo Δt pequeño, tal que los objetos no se desplazan significativamente de un cuadro a otro.
- Las imágenes a analizar representan una escena que contiene objetos texturizados en escala de grises. Este algoritmo no utiliza información de color.

Este algoritmo soluciona la ecuación de flujo óptico (9}) para los píxeles en el vecindario del píxel de interés, por el criterio de mínimos cuadrados. De esta manera, el algoritmo Lucas-Kanade puede resolver la ambigüedad de la ecuación de flujo óptico.

Inicialmente, se selecciona una ventana de tamaño $n * n$ alrededor del píxel de interés, lo cual resulta en n^2 ecuaciones por píxel, de la manera $A * d = b$, como se muestra en (10).

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \cdots & \cdots \\ I_x(p_{n^2}) & I_y(p_{n^2}) \end{bmatrix} * \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \cdots \\ I_t(p_{n^2}) \end{bmatrix} \tag{10}$$

El sistema mostrado en (10) es un sistema sobre-determinado, porque tiene más ecuaciones que incógnitas. Para solucionarlo, se utiliza el criterio de mínimos cuadrados. Es decir, soluciona el sistema (11).

$$A^T A * d = A^T b \tag{11}$$

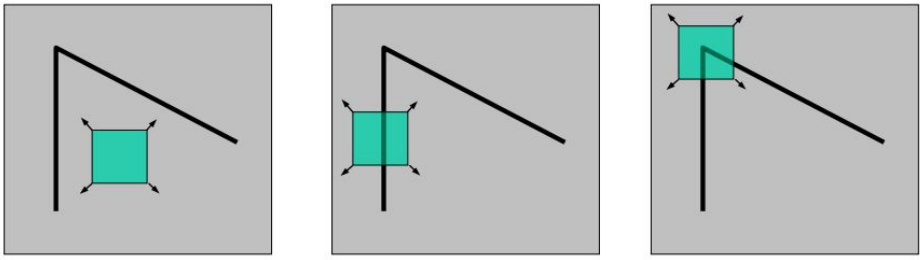
Donde A^T es la matriz transpuesta de A. De esta manera, el sistema mostrado en (10) resulta en (12).

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} * \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{12}$$

3.2.2.3. Estimación de descriptores

La estimación de descriptores de características incluye métodos para abstraer información de una imagen y tomar decisiones en cada punto de la imagen, ya sea que exista o no una característica en ese punto. Se da el nombre de característica a una pieza de información relevante, tales como bordes o esquinas. En general, se buscan elementos en la imagen que sean inusuales [17].

En la figura 3.2.2.3-1 se pueden ver tres diferentes regiones de interés en la imagen. La figura 3.2.2.3-1.a muestra una región plana, en la que no hay cambio en ninguna de las direcciones. La figura 3.2.2.3-1.b muestra un borde, en el que no hay cambio en la dirección del borde (vertical) pero sí hay cambio en la dirección perpendicular al borde (horizontal). La figura 3.2.2.3-1.c muestra una esquina, en la que hay un cambio significativo en todas las direcciones. A partir de este ejemplo se puede deducir que los mejores descriptores de características son las esquinas, o regiones donde las derivadas de segundo orden sean suficientemente altas [26].



a b c

Figura 3.2.2.3-1. Regiones de interés en una imagen [17].

Se dice que los descriptores de características son de buena calidad si la matriz de la ventana de búsqueda está por encima del nivel de ruido de la imagen y está bien acondicionada. El primer requerimiento se satisface si los valores propios de la ventana de búsqueda son valores grandes (mayores que el nivel de ruido de la imagen). La ventana de búsqueda está bien acondicionada si los valores propios no difieren por varios órdenes de magnitud.

4. DESARROLLO

El presente trabajo de grado es desarrollado con el fin de disminuir el costo computacional que requiere el proceso de identificación y seguimiento de objetos utilizada actualmente (*YOLO*) en las cámaras de vigilancia de la Secretaría de Movilidad de la ciudad de Bogotá. Esta institución fue la proveedora de los videos utilizados en el trabajo. Los videos que componen la base de datos fueron obtenidos con una cámara *FULL HD*, con resolución de 1920x1080 píxeles y con una tasa de entre 20 y 30 cuadros por segundo.

En la figura 4-1 se muestra el diagrama en bloques general del sistema que está implementado actualmente, en el cual se utiliza *YOLO* como única técnica para la detección, identificación y seguimiento de objetos de interés en cada cuadro del video.

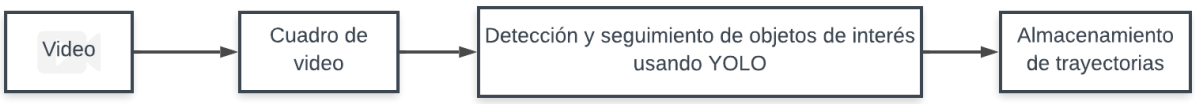


Figura 4-1. Diagrama en bloques del sistema que se utiliza actualmente.

En la figura 4-2 se muestra el diagrama en bloques del sistema que se implementará en el presente trabajo. A diferencia del primero, este sistema solamente utiliza *YOLO* para la detección e identificación de objetos en uno de cada 5 cuadros del video. En los cuatro cuadros restantes, se utiliza una técnica de seguimiento diferente, que requiere menos operaciones y que no actúa sobre la totalidad de la imagen, para así poder aliviar la carga computacional de la ejecución del algoritmo.
Para el seguimiento de los objetos, se implementaron dos técnicas: filtrado de partículas (*particle filter*) y flujo óptico (*optical flow*), las cuales se describirán en detalle en las secciones 4.1 y 4.2 respectivamente.

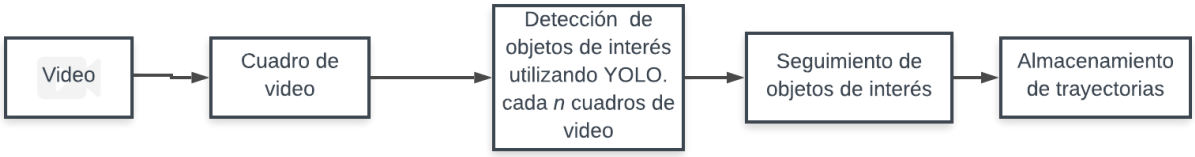


Figura 4-2. Diagrama en bloques del sistema implementado en el trabajo de grado.

4.1. Filtrado de partículas *(Particle filter)*

En la figura 4.1-1 se muestra el pseudocódigo para la implementación del algoritmo de filtrado de partículas. Inicialmente, se hace una depuración del código, es decir, si no se encuentra un cuadro de video (ya sea porque no se encontró el archivo de video o porque el video se acabó), el algoritmo se termina, esta depuración se hará cada vez que haya una lectura de cuadro de video.

```
Tomar cuadro de video
while (hay cuadro de video):
   if (cuadro de video es múltiplo de 5):
      Detección de objetos de interés utilizando YOLO
      Agregar nueva trayectoria o nuevo punto a la trayectoria
      Contador de cuadro de video incrementa

   else:
      Tomar cuadro de video
      if (se detectaron objetos):
         for (cada objeto detectado):

               Generar una distribución normal de partículas alrededor del centroide
               del objeto

               for (cada partícula creada):

                     Generar una plantilla del tamaño del rectángulo contenedor
                     alrededor de cada partícula

                    Aplicar métrica de comparación

               Elegir el mejor rectángulo contenedor
               Agregar nuevo punto a la trayectoria

      else:
          Pasar al siguiente cuadro de video
```

Figura 4.1-1. Pseudocódigo del algoritmo de filtrado de partículas.

En caso de encontrar un cuadro de video, se debe saber si es múltiplo de 5, como se mencionó al principio de este capítulo, sólo se ejecutará *YOLO* en uno de cada 5 cuadros. Si efectivamente el número de cuadro es múltiplo de 5, se ejecuta el algoritmo detección e identificación de objetos y se obtiene la cantidad de objetos detectados, así como las coordenadas de cada rectángulo contenedor.

YOLO es capaz de detectar e identificar 9 clases de objetos diferentes: peatón, particular, taxi, motociclista, bus, camión, minivan, ciclista y tractomula. Además, el tamaño del rectángulo contenedor es diferente para cada objeto detectado, lo cual depende de la lejanía en la que se encuentre el objeto. Una vez obtenidas estas características, se procede a implementar la técnica de seguimiento correspondiente.

El filtrado de partículas es un algoritmo de filtrado bayesiano aproximado basado en la simulación de Monte Carlo [18]. La simulación de Montecarlo es un método estadístico utilizado para resolver problemas matemáticos a través de la generación de variables aleatorias [19]. Por lo tanto, la idea central del filtrado de partículas es generar muestras aleatorias (o partículas) y asignarle un peso a cada partícula mediante la evaluación de una métrica, de la cual se hablará en detalle más adelante.

4.1.1. Creación de partículas

Para la creación de las partículas, retomando la definición de filtrado de partículas de Arulampalam, Maskell, Gordon y Clapp [10], al detectar un objeto de interés, se crea una evolución del estado de secuencia de este, q_k. El objetivo es hacer un seguimiento a partir de estimaciones, z_k, donde se busca estimar q_k basado en todas las medidas posibles $z_{1:k} = \{z_i,\ i = 1, ..., k\}$. Para poder obtener una buena credibilidad del estado en q_k, se necesita crear una función de densidad de probabilidad *(pdf),* $p(q_k|z_{1:k-1})$, para luego crear la distribución normal de una variable aleatoria, de la cual salgan n muestras (partículas) con su respectiva probabilidad (peso). En un principio todas las partículas tienen el mismo peso $1/n$.

Se tiene como entrada la posición del objeto de interés, *q(t)* (x_k), donde q_x y q_y describen la posición del objeto de interés en un sistema de coordenadas cartesianas en 2D:

$$q(t) = \begin{bmatrix} q_x(t) \\ q_y(t) \end{bmatrix}$$

Una distribución normal está definida por (1)

$$\Phi_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(u-\mu)^2}{2\sigma^2}}\, du \qquad (1)$$

Donde μ es la media, σ es la desviación estándar, σ^2 es la varianza. Pero (1) es una distribución para una sola dimensión. Por esta razón, se necesita hacer una distribución normal multivariante, que es n-dimensional, y está descrita por (2)

$$f_X(x_1,\ldots,x_n) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)} \quad (2)$$

Para dos dimensiones estaría definido por (3)

$$f_q(q_x,q_y) = \frac{1}{2\pi|\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)} \quad (3)$$

Donde el vector μ es la esperanza de *q,* Σ es la matriz de covarianza y |Σ| es su determinante, la cual es una matriz no singular, que está definida por (4)

$$\sum_{ij} = E\left[\left(q_i-\mu_i\right)\left(q_j-\mu_j\right)\right] \quad (4)$$

Donde $\mu_i = E(q_i)$, es el valor esperado de la entrada i-ésima del vector *q(t).* Posteriormente se obtiene (5)

$$\sum = \begin{bmatrix} E\left[\left(q_1-\mu_1\right)\left(q_1-\mu_1\right)\right] & E\left[\left(q_1-\mu_1\right)\left(q_2-\mu_2\right)\right] \\ E\left[\left(q_2-\mu_2\right)\left(q_1-\mu_1\right)\right] & E\left[\left(q_2-\mu_2\right)\left(q_2-\mu_2\right)\right] \end{bmatrix} \quad (5)$$

Por propiedades de la covarianza se tiene que es cero si no hay existencia de una relación entre las dos variables estudiadas, es decir, si q_x y q_y son independientes. De esta manera los términos Σ_{12} y Σ_{21} se vuelven 0, obteniendo así (6)

$$\sum = \begin{bmatrix} E\left[\left(q_1-\mu_1\right)\left(q_1-\mu_1\right)\right] & 0 \\ 0 & E\left[\left(q_2-\mu_2\right)\left(q_2-\mu_2\right)\right] \end{bmatrix} \quad (6)$$

Una vez definidas Σ y μ, la distribución normal multivariante puede ser definida como (7)

$$q \sim \mathcal{N}(\mu,\Sigma) \quad (7)$$

De esta manera, se crean las *N* partículas distintas alrededor del origen del objeto, esparcidas cierta distancia a partir de la matriz de correlación y su media o valor esperado. Generando *n* centroides diferentes $f_q = [(x_1, y_1),\ldots,(x_n, y_n)]$.

Después de tener las partículas distribuidas, se crea un rectángulo contenedor, del mismo tamaño del original, para cada par de coordenadas creadas a partir de la distribución. Una vez se tienen los *N* rectángulos contenedores, se procede a aplicar una métrica de comparación que permita determinar cuál de las partículas generadas es la que mejor corresponde al objeto en el cuadro $n+1$. Este proceso se hace a través de una comparación entre dos imágenes de dimensiones iguales.

Inicialmente, se crea una imagen, la cual es una subimagen del cuadro de video, con centro en (x_0, y_0) y dimensiones (w, h). Estos parámetros fueron adquiridos por medio de *YOLO*. Esta imagen, o plantilla,

será la base de la comparación. Después de la creación de las partículas, se tienen N centroides f_q, a partir de los cuales se crean N rectángulos contenedores, las dimensiones de este rectángulo dependen de la métrica que se va a utilizar. Posteriormente se procede a hacer la comparación de cada subimágen creada con la subimagen base. En esta comparación, se evalúa qué tan parecida es la subimágen creada, con respecto a la subimagen base. De esta manera, se puede determinar la posición del objeto detectado en el siguiente cuadro de video, $n + 1$.

4.1.2. Métricas

Una métrica es una medida o conjunto de medidas que se utilizan para conocer características, generalmente para hacer una comparación. En este caso, por medio de las métricas, se estipula cuál de las N partículas será el centroide del objeto en el cuadro $n + 1$.

Al aplicar la métrica, se compara cierta subimagen con la subimagen base, y a partir de dicha comparación, se obtiene un coeficiente, el cual indica el nivel de similitud entre las dos imágenes. Al terminar de hacer las N comparaciones, se ordenan los coeficientes en una lista. Finalmente, se elige el menor o mayor coeficiente (dependiendo de la métrica utilizada), el cual toma todo el peso y se convierte en el centroide del objeto en el cuadro siguiente, y por ende se convertirá en el centroide de la nueva subimagen base. El resto de las partículas pierden su peso, y por lo tanto son eliminadas.

Después de tener el nuevo centroide, se genera una nueva distribución de partículas, y se distribuyen los pesos en las nuevas partículas, generando así una nueva *pdf*. A este proceso se le llama remuestreo (*resampling*). Este ciclo continúa hasta que se acaben los cuadros de video intermedios para regresar a *YOLO*. A continuación, se describirán las métricas que se implementaron en el presente trabajo de grado.

4.1.2.1. *Histograma de color*

Un histograma es una representación gráfica de datos agrupados mediante intervalos. Es una estimación de la distribución de probabilidad de estos datos, que representa la frecuencia con la cual se repite un valor en cada intervalo.

Se define como una función b_i , que cuenta el número de observaciones que caen dentro de cada categoría disjunta (*bins*). Si se define n como el número total de las observaciones y k el número total de los bins. Está definido como (8)

$$n = \sum_{i=1}^{k} b_i \qquad (8)$$

Cuando se utiliza el método de histogramas por color como métrica para el algoritmo, inicialmente se debe convertir el cuadro de video de una representación en RGB (*red, green, blue*) a una representación HSV (*hue, saturation, value*). Al hacer esta conversión, se puede trabajar únicamente con un canal. Se

decide trabajar con el canal que aporte más información de la imagen, en este caso se eligió el canal que representa el matiz (*hue)*.

Luego, se procede a calcular el histograma del rectángulo contenedor original y de cada rectángulo contenedor formado alrededor de las partículas generadas, el cual, en este caso, es del mismo tamaño que el rectángulo contenedor original.

Después de esto, se debe saber cuál histograma es más parecido al histograma del rectángulo contenedor original. Para tal fin, se halla el coeficiente de Bhattacharyya de cada rectángulo contenedor, en relación al original. El coeficiente de Bhattacharyya es una medida aproximada de la cantidad de superposición entre dos muestras estadísticas [20], y está definido como (9)

$$BC(p,q) = \sum_{i-1}^{n} \sqrt{p_i \cdot q_i} \qquad (9)$$

Donde p y q son las muestras, n es la cantidad de *bins* y p_i y q_i son las muestras en la i-ésima partición. Teniendo en cuenta (9), el coeficiente de Bhattacharyya será máximo cuando los histogramas sean iguales. Por lo tanto, se elige el mayor coeficiente calculado, y centro del rectángulo contenedor correspondiente será el centroide del objeto en el siguiente cuadro.

4.1.2.2. Emparejamiento de plantillas (Template Matching)

En el procesamiento de imágenes digitales, el emparejamiento de plantillas es un proceso para determinar la ubicación de la subimagen dentro de una imagen [21]. Para utilizar esta métrica, se necesitan dos componentes principales: la plantilla y la imagen de origen, en la cual se quiere detectar dicha plantilla. En este caso, la plantilla será el rectángulo contenedor detectado y la imagen origen será cada uno de los rectángulos contenedores creados con cada partícula como su centro. Para esta métrica, el tamaño de la imagen de origen será más grande que el de la plantilla.

En esta métrica, se desliza la imagen de plantilla sobre la imagen de origen (como en la convolución 2D), se compara la plantilla y el parche de la imagen de origen debajo de la imagen de la plantilla. A continuación, se mencionan los métodos de comparación implementados, en los cuales R representa el resultado, T la plantilla (o *template*) e I la imagen.

4.1.2.2.1. Diferencia de cuadrados y diferencia de cuadrados normalizada

La diferencia de cuadrados es una medida de coincidencia que se basa en las diferencias de intensidad píxel a píxel entre las dos imágenes. Se calcula como una suma de los cuadrados de las variaciones de la media. Con esta medida de similitud, el punto de coincidencia se puede determinar considerando la

ubicación del valor mínimo en las matrices de la imagen [21]. La diferencia de cuadrados se obtiene mediante (10) y la diferencia de cuadrados normalizada se obtiene mediante (11)

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x', y+y'))^2 \quad (10)$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x', y+y'))^2}{\sqrt{\sum_{x',y'} T(x', y')^2 \cdot \sum_{x',y'} I(x+x', y+y')^2}} \quad (11)$$

4.1.2.2.2. Correlación cruzada y correlación cruzada normalizada

En el procesamiento de señales, la correlación cruzada es una medida de la similitud de dos señales en función de un retardo de tiempo aplicado a una de ellas [22]. También se conoce como producto de punto deslizante o producto interno deslizante. Se calcula simplemente multiplicando y sumando dos series de tiempo juntas. La correlación se obtiene mediante (12) y la correlación normalizada se obtiene mediante (13). Con esta medida de similitud, el punto de coincidencia se puede determinar considerando la ubicación del valor máximo en las matrices de la imagen.

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x', y+y')) \quad (12)$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x', y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x', y+y')^2}} \quad (13)$$

4.1.2.2.3. Coeficiente de correlación y coeficiente de correlación normalizado

El coeficiente de correlación es un método para establecer el grado de asociación que existe entre los movimientos relativos de dos variables aleatorias, o cantidades medidas [23]. Con este método, se evalúa el valor de la intensidad de cada píxel para cada imagen, T e I. Un valor alto representa un mejor emparejamiento que un valor más bajo. R=0, significa que no hubo ningún emparejamiento.

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x', y+y')) \quad (14)$$

En el caso de la normalizada, que se observa en la ecuación (15), el resultado de R puede tomar cualquier valor entre -1 y 1. Siendo R=-1 cuando son anti-correlacionados, es decir que se mueven o cambian en direcciones o valores contrarias. Siendo R=1 cuando son totalmente correlacionadas, es decir se mueven o cambian en direcciones o valores iguales. Siendo R=0 cuando son no correlacionadas.

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',\ y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',\ y+y')^2}} \quad (15)$$

4.2. Flujo óptico (*Optical flow)*

En la figura 4.2-1 se muestra el pseudocódigo para la implementación del algoritmo de flujo óptico. Inicialmente, de la misma manera que para el filtrado de partículas, se tiene una depuración del código, es decir, si no se encuentra un cuadro de video (ya sea porque no se encontró el archivo de video o porque el video se acabó), el algoritmo se termina, esta depuración se hará cada vez que haya una lectura de cuadro de video.

```
Tomar cuadro de video
while(hay cuadro de video):

   if (cuadro de video es múltiplo de 5):
      Detección de objetos de interés utilizando YOLO
      Agregar nueva trayectoria o nuevo punto a la trayectoria

   else:
      if (se detectaron objetos):
         for (cada objeto detectado):
             Hallar descriptores de características en el rectángulo contenedor del
             objeto
             Transformar coordenadas origen del objeto al origen de la imagen
          Crear lista con los descriptores de todos los objetos

      Tomar cuadro de video

          if (hay cuadro de video):
             Calcular flujo óptico Lucas-Kanade de todos los puntos
             Hallar la mediana de desplazamiento para cada objeto
             Agregar nuevo punto a la trayectoria
          else:
             Pasar al siguiente cuadro de video
      else:
           Pasar al siguiente cuadro de video
```

Figura 4.2-1. Pseudocódigo del algoritmo de flujo óptico.

En caso de encontrar un cuadro de video, se debe saber si es múltiplo de 5. Si efectivamente el número de cuadro es múltiplo de 5, se ejecuta el algoritmo detección e identificación de objetos y se obtiene la cantidad de objetos detectados, así como las coordenadas de cada rectángulo contenedor.

Como se mencionó en la sección 4.1, *YOLO* detecta e identifica 9 clases de objetos diferentes: peatón, particular, taxi, motociclista, bus, camión, minivan, ciclista y tractomula. Una vez obtenidas estas características, se procede a implementar la técnica de flujo óptico. El flujo óptico es un modelo del movimiento aparente de un objeto en una escena, causado por el movimiento relativo entre una escena y un observador [13].

El flujo óptico puede ser discreto o denso. El flujo óptico disperso proporciona los vectores de flujo de algunas características interesantes (bordes y/o esquinas), mientras que el flujo óptico denso proporciona los vectores de flujo de todos los píxeles del cuadro de vídeo [24]. El flujo óptico denso tiene una mayor precisión, pero es computacionalmente más costoso. Como el objetivo de este trabajo de grado es reducir el costo computacional de un algoritmo de seguimiento, se utilizará el flujo óptico disperso.

4.2.1. Descriptores

Para el seguimiento de características, hay esencialmente dos pasos importantes. El primero es decidir qué características rastrear, y el segundo es el rastreo en sí mismo. Por lo tanto, inicialmente se deben seleccionar un conjunto de píxeles que pertenezcan a características interesantes para cada objeto detectado, para posteriormente rastrear sus vectores de velocidad (movimiento). Una característica interesante es un punto en una imagen que tiene una posición bien definida y puede ser detectado de forma robusta. Esto significa que una característica interesante puede ser una esquina, pero también puede ser, por ejemplo, un punto aislado de intensidad local máxima o mínima, final de líneas, o un punto en una curva donde la curvatura es localmente máxima. Este procedimiento se hace a través del detector de Harris [24].

Para el detector de Harris [25], se identifican tres diferentes tipos de características: bordes, esquinas y secciones planas. La función para detectar este tipo de características está definida por (1):

$$E(x,y) = \sum_{u,v} w(u,v) \left| I(x+u, y+v) - I(u,v) \right|^2 \tag{1}$$

Por lo tanto (2):

$$E(x,y) = \sum_{u,v} w(u,v) \left[xX + yY + O\left(x^2, y^2\right) \right]^2 \tag{2}$$

Donde I es la intensidad de la imagen, E es el cambio producido por una variación en (x,y), $w(u,v)$ es la ventana que será utilizada, la cual tienen un valor unitario solo en la región específica y fuera de ella es cero. Los operadores X y Y, se definen como gradientes horizontal y vertical, respectivamente. Y están dados por:

$$X = I \otimes (-1, 0, 1) \approx \frac{\partial I}{\partial x}$$

$$Y = I \otimes (-1, 0, 1)^T \approx \frac{\partial I}{\partial y}$$

Como el cambio que sufren los objetos entre un cuadro de video y otro es pequeño, se tiene que (2) puede ser representada como (3)

$$E(x, y) = Ax^2 + 2Cxy + By^2 \qquad (3)$$

Donde,

$$A = X^2 \otimes w$$

$$B = Y^2 \otimes w$$

$$C = (XY) \otimes w$$

Además, se puede evitar que la respuesta tenga ruido, si para la ventana w se utiliza una ventana Gaussiana [25].

$$w(u, v) = e^{(u^2 + v^2)/2\sigma^2}$$

Por otra parte, de (3) se puede simplificar a (4),

$$E(x, y) = [x, y] M [x, y]^T \qquad (4)$$

Donde M es una matriz simétrica de 2x2, definida como:

$$M = \sum_{u, v} w(u, v) \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

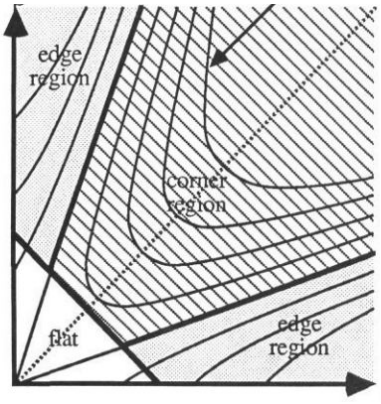


Figura 4.2.1-1. Autocorrelación de la curvatura del espacio. Las líneas gruesas brindan la clasificación. Las líneas delgadas son la respuesta del contorno [25]. El eje horizontal es ρ y el eje vertical es η.

Sean ρ y η los valores propios de la matriz M, estos valores serán proporcionales a las curvaturas de la función de autocorrelación y forman una descripción invariante de rotación de M [25], como se observa en la figura 4.2.1-1. Pueden ocurrir 4 casos:

1. ρ >> η: esto significa que se encontró un borde horizontal, se encuentra en la parte inferior derecha de la figura 4.2.1-1.
2. η >> ρ: esto significa que se encontró un borde vertical, se encuentra en la parte superior izquierda de la figura 4.2.1-1.
3. ρ y η son números bastante pequeños: esto significa que encontró una región plana, se encuentra en la parte inferior izquierda de la figura 4.2.1-1.
4. ρ y η son números bastante grandes: esto significa que encontró una esquina, se encuentra en la parte superior derecha de la figura 4.2.1-1.

Además, hay que obtener la calidad o respuesta de cada esquina o borde que se encuentre. Se define R como la medida de la respuesta de la esquina o borde. La cual es una función que depende de los valores propios de M. Se define R como [25]:

$$R = Det(M) - kTr(M)^2 \qquad (5)$$

Donde k es una constante y que $Det(M)$ y $Tr(M)$ están definidas como:

$$Tr(M) = \rho + \eta = A + B$$
$$Det(M) = \rho\eta = AB - C^2$$

Del valor de R será positivo si la es una región con esquina, será negativo si es una región de borde y será muy pequeño (en magnitud, tendiendo a cero) si es una región plana.

Para el método de Shi-Tomasi [26], se modificó la ecuación (5) obteniendo (6):

$$R = min(\rho, \eta) > \lambda \qquad (6)$$

Donde, λ representa un umbral definido. Es decir, si el mínimo entre los dos valores propios es mayor que el umbral, quiere decir que la región encontrada es una esquina. De esta manera, se eliminan descriptores que no tengan tanta validez o peso, dejando así aquellos que sean importantes para una comparación de movimiento entre imágenes.

Al hacer este proceso en cada uno de los rectángulos contenedores, se asegura que cada objeto tenga sus propias características, además de no hallar características innecesarias en el resto del cuadro de video. Este proceso se muestra en la figura 4.2.1-2.

Figura 4.2.1-2. Descriptores de características hallados en un objeto de interés.

Debido a que las características fueron halladas independientemente para cada rectángulo contenedor, estos puntos tienen como coordenada origen $(0,0)$. Para ubicar las características en el cuadro de video completo, se debe sumar a cada punto la coordenada de origen del rectángulo contenedor. Después de tener las coordenadas correspondientes en el cuadro de video de las características detectadas, se hace una lista con todas ellas, las cuales serán localizadas en el siguiente cuadro de vídeo mediante el cálculo del flujo óptico. Las características extraídas se pasan de cuadro a cuadro para garantizar que se estén siguiendo los mismos puntos.

El siguiente paso es calcular el flujo óptico entre el cuadro anterior I (al que se le extrajeron las características) y el actual J. Se calcula una implementación piramidal del seguidor de características de Lucas-Kanade [27].

4.2.2. Flujo óptico

Sean I y J dos imágenes secuenciales en escala de grises, el objetivo de esta técnica es encontrar la ubicación de los descriptores hallados en el cuadro actual I en el siguiente cuadro J, tal que $I(x,y,t)$ y $J(x+dx,y+dy,t+dt)$ sean similares. El algoritmo Lucas-Kanade soluciona la ecuación de flujo óptico (7) para los píxeles en el vecindario del píxel de interés, por el criterio de mínimos cuadrados. En la ecuación (7) $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, $I_t = \frac{\partial I}{\partial t}$ son las derivadas de la imagen y u,v son las componentes x,y del flujo óptico de $I(x,y,t)$ respectivamente.

$$I_x u + I_y v + I_t = 0 \quad (7).$$

Inicialmente, se selecciona una ventana de tamaño $n * n$ alrededor del píxel de interés, lo cual resulta en n^2 ecuaciones por píxel, de la manera $A^T A * d = A^T b$, como se muestra en (8).

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \cdots & \cdots \\ I_x(p_{n^2}) & I_y(p_{n^2}) \end{bmatrix} * \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \cdots \\ I_t(p_{n^2}) \end{bmatrix} \quad (8)$$

Se define el vector de velocidad de la imagen $d = [u,v]^T$ como el vector que minimiza la función residual ε definida en (9)

$$\varepsilon(dx,dy) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x,y) - J(x+dx,\ y+dy))^2 \quad (9)$$

Se define ω_x y ω_y como los dos enteros que establecen el tamaño de la ventana como $(2\omega_x+1)\times(2\omega_y+1)$. Esta ventana también es conocida como ventana de integración y generalmente toma valores entre 2 y 7. Para elegir un buen valor se deben considerar dos aspectos importantes de

cualquier algoritmo: precisión y robustez. Si se elige un valor pequeño, se aumenta la precisión, ya que considerará solo a los vecinos más cercanos para los cálculos, pero se disminuye la robustez, ya que los movimientos grandes se ignorarán.

Una solución para este problema es una implementación piramidal del algoritmo clásico de Lucas Kanade (figura 4.2-1), el cual realiza un seguimiento desde el nivel más alto de una pirámide de imagen (detalle más bajo) y hacia niveles más bajos (detalle más fino). El seguimiento de las pirámides de imágenes permite capturar grandes movimientos en las ventanas locales.

Sean $L = 0, ..., L_m$ los niveles de la representación piramidal, el algoritmo procede generalmente de la siguiente manera: primero, se calcula el flujo óptico en el nivel de la pirámide más profundo (L_m), es decir, donde la ventana de integración es más pequeña. Luego, el resultado de ese cálculo se propaga al nivel $L_m - 1$ en forma de una estimación inicial del desplazamiento del píxel. Dada la hipótesis inicial, el flujo óptico refinado se calcula en el nivel $L_m - 1$, y el resultado se propaga al nivel $L_m - 2$ y así sucesivamente hasta el nivel 0 (la imagen original).

Al utilizar la implementación piramidal del flujo óptico, se permiten movimientos de píxeles grandes, al tiempo que mantiene el tamaño de la ventana de integración relativamente pequeño.

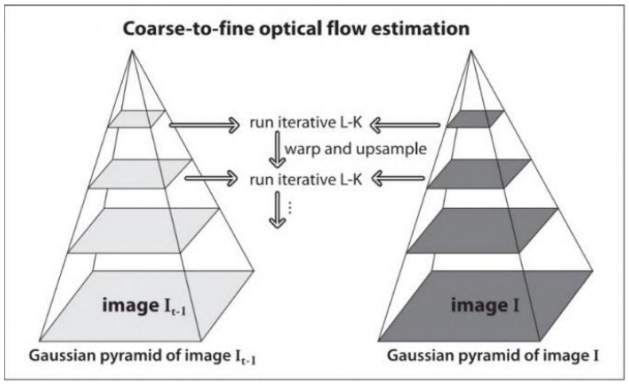


Figura 4.2.2-1. Flujo óptico piramidal Lucas-Kanade [27].

4.2.3. Mediana de desplazamiento

Con el cálculo del flujo óptico, se obtienen las coordenadas de los descriptores de características en el cuadro $n + 1$. Sin embargo, como lo que se quiere seguir es el centroide del objeto de interés, se calcula la mediana del desplazamiento de los descriptores.

La mediana es el valor medio o central de un conjunto de observaciones. Para encontrar la mediana, los valores deben estar ordenados. Si el total de elementos del conjunto de datos es par, la mediana se halla calculando el promedio de los dos números centrales. Si el total de elementos es un número impar entonces la mediana tomará el valor ubicado en el centro de la lista.

Se utiliza esta medida de tendencia central debido a que no es sensible a valores extremos. Así, si se detecta un falso positivo en una coordenada lejana, este valor no tendrá tanto peso sobre la medida como si se calculara la media. Por lo tanto, no se tendrá en cuenta como descriptor y no se detectará esta partícula en el cuadro siguiente.

Hay que tener en cuenta que, en este paso, se debe calcular la media de desplazamiento para cada objeto detectado, es decir que hay que separar la lista de descriptores que se obtuvieron mediante el cálculo del flujo óptico. Como a cada objeto se le asignaron 20 descriptores, se forman n listas de 20 descriptores, siendo n la cantidad de objetos detectados en el cuadro.

Después de calcular la mediana de desplazamiento para cada objeto en el cuadro de video, se agrega este nuevo punto a la trayectoria del objeto, el cual representa su centroide. El proceso mencionado anteriormente se repite hasta que el vídeo finalice.

5. PROTOCOLO DE PRUEBAS

Con el fin de verificar el cumplimiento de los objetivos del presente trabajo de grado, se llevó a cabo el protocolo de pruebas mostrado en el diagrama en bloques de la figura 5-1. Este diagrama en bloques se puede dividir en tres grandes partes: la primera parte está enfocada en el algoritmo que se utiliza actualmente en la Secretaría de Movilidad, la segunda parte se centra en el algoritmo implementado en el presente trabajo de grado y la tercera parte consiste en un conteo manual de los vehículos. Se medirá el tiempo de ejecución de cada algoritmo, así como la cantidad de vehículos contados. Estas medidas serán explicadas en las secciones 5.1 y 5.2 respectivamente.

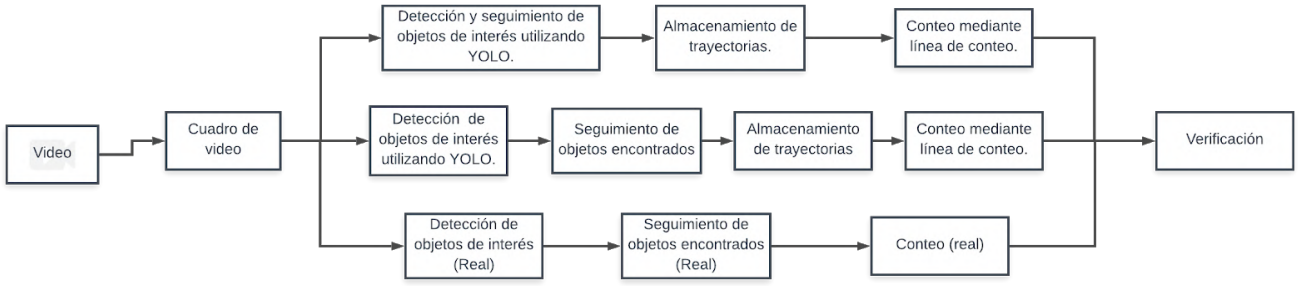


Figura 5-1. Diagrama en bloques del sistema y de evaluación.

Los videos utilizados para hacer las pruebas fueron proporcionados por la Secretaría de Movilidad de Bogotá. Tienen una resolución de 1280x1720 píxeles y una tasa de entre 25 y 30 cuadros por segundo. Se usaron videos tomados desde diversos ángulos y con iluminación distinta, para observar cómo es el comportamiento de cada algoritmo ante las diferentes condiciones.

5.1. Tiempo de ejecución.

Objetivo de la medición: Garantizar que se obtiene al menos un 30% de mejora en la ejecución del algoritmo de detección, identificación y seguimiento propuesto, con respecto al utilizado actualmente por la Secretaría de Movilidad de Bogotá.

Procedimiento: En la figura 5.1-1 se muestra el pseudocódigo que se siguió para medir el tiempo de ejecución de los diferentes algoritmos. Se mide el tiempo al iniciar el algoritmo y al finalizarlo. El tiempo que se demora en ejecutar es la resta de los dos tiempos mencionados.

Se midió el tiempo de ejecución de 8 videos de 15 minutos cada uno con los tres diferentes algoritmos: el algoritmo de la Secretaría de Movilidad de seguimiento cuadro a cuadro que utiliza la Secretaría de Movilidad, el algoritmo implementado utilizando filtrado de partículas y el algoritmo implementado utilizando flujo óptico.

```
Medir tiempo inicial

Ejecución del algoritmo

Medir tiempo final

tiempo ejecución = tiempo final - tiempo inicial
```

Figura 5.1-1. Pseudocódigo de la medición de tiempo de ejecución.

5.2. Conteo.

Objetivo de la medición: Garantizar que se obtiene un error de menos del ±10% en el conteo de objetos de interés, comparado con el utilizado actualmente por la Secretaría de Movilidad de Bogotá. Además, se quiere conocer la certeza de ambos conteos con respecto a un conteo manual.

Procedimiento: Para el conteo de objetos de interés, se utilizaron líneas de conteo, las cuales son configuradas por el usuario al iniciar el video, mediante la interfaz que se muestra en la figura 5.2-1. Se pueden elegir hasta 6 líneas de conteo y ubicarlas en las diferentes zonas del cuadro de video inicial, como se muestra en la figura 5.2-2, en la cual se estableció una línea de conteo.
Para incrementar el contador, se toman los dos registros más recientes de la trayectoria del objeto y se prueba si pasaron la línea de conteo establecida. Cabe resaltar que este conteo tiene en cuenta la identificación de los objetos detectados.

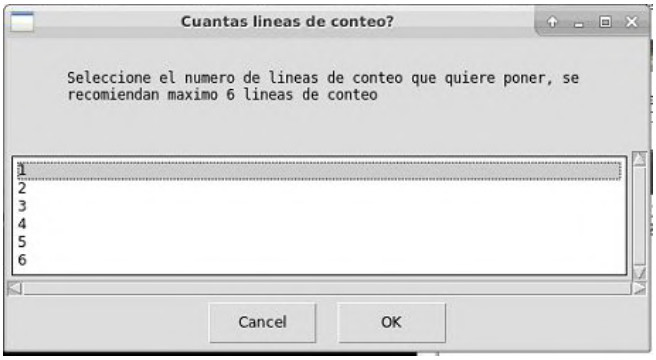
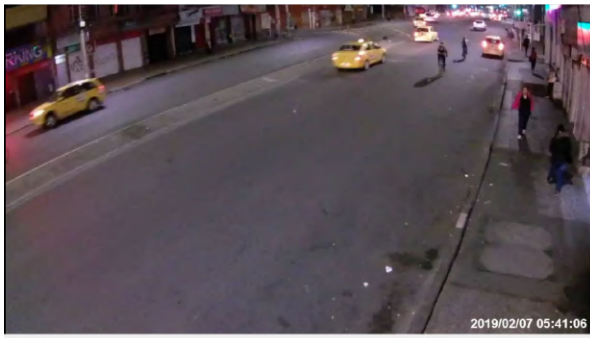


Figura 5.2-1. Interfaz para la elección de las líneas de conteo.

Figura 5.2-2. Línea de conteo configurada.

Al finalizar la ejecución del algoritmo, un archivo .csv es exportado, en el cual se puede visualizar un conteo cronológico de los objetos que cruzaron la línea de conteo, teniendo en cuenta la etiqueta del mismo. Así mismo, se muestra un resumen del conteo de los vehículos, teniendo en cuenta la dirección en la que cruzaron la línea (positiva o negativa).

La base de datos utilizada en este trabajo de grado se dividió en 2 partes: dos videos para diseñar y 8 videos para las pruebas. Para la etapa de diseño, se midió el tiempo de ejecución de cada una de las métricas utilizadas en el filtrado de partículas, teniendo en cuenta de que los ángulos en los que fueron tomados estos videos fueran diferentes, con el fin de analizar los efectos de las diferentes perspectivas en su desempeño. Para la etapa de pruebas, se midió el tiempo de ejecución y el conteo de objetos de interés para cada algoritmo. En la tabla 5-1 se muestran las perspectivas de cada uno de los videos utilizados.

Video	**Perspectiva**
Calle 114 Carrera 7	
Avenida calle 63 Carrera 15	

Carrera 19 Calle 39	
Carrera 19 Calle 39-2	
Calle 37 Carrera 24	
Calle 37 Carrera 24-2	

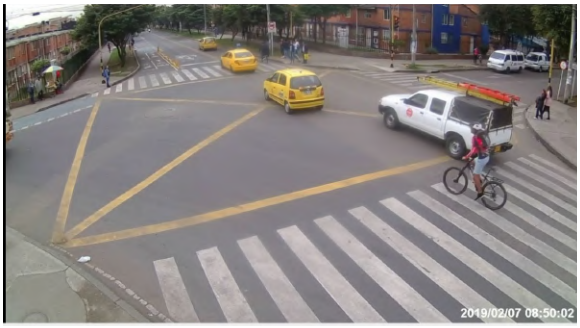
Diagonal 15A Carrera 104	
Diagonal 15A Carrera 104-2	

Tabla 5-1. Perspectivas de cada video utilizado.

6. ANÁLISIS DE RESULTADOS

En este capítulo, se mostrarán los resultados obtenidos siguiendo el protocolo de pruebas descrito en el capítulo 5. Se harán las mediciones para cada video, analizando los efectos que tienen diferentes factores como: el ángulo en el que está ubicado la cámara, la iluminación de la escena y la cantidad de tráfico. Finalmente, se hará un promedio de las mediciones, con el fin de verificar el cumplimiento general de los objetivos planteados en el presente trabajo de grado.

En la sección 6.1 se muestran los resultados de la medición del tiempo de ejecución del algoritmo de filtrado de partículas utilizando las diferentes métricas. En la sección 6.2 se muestran los resultados de la medición del tiempo de ejecución de los diferentes algoritmos y del conteo de objetos de interés para dos diferentes videos, de los cuales se hizo el conteo de objetos de interés manualmente para tener una verdad fundamental o *ground truth.* En la sección 6.3 se muestran los resultados de la medición del tiempo de ejecución de los diferentes algoritmos y del conteo de objetos de interés para 6 diferentes videos.

Los videos utilizados para las pruebas se pueden ver en el anexo 9.1. Cada video tiene una duración de 15 minutos, es decir, de 900 segundos.

6.1. Filtrado de partículas

En la tabla 6.1-1 se puede visualizar el tiempo de ejecución del algoritmo de filtrado de partículas utilizando las diferentes métricas mencionadas en el capítulo 4, en la sección 4.1.2.

En la sección mencionada, se describió un proceso de estimación del centroide del objeto en el cuadro $n+1$ mediante la creación de histogramas y el cálculo del coeficiente de Bhattacharyya. En dicho proceso se estipula que: si se tienen *n* partículas, se crean *n* subimágenes que se comparan con a la original. Para esta comparación, se debe convertir cada píxel de RGB a HSV. Si la dimensión por cada subimagen es de (*w,h*), se necesita realizar esa operación, $w \times h \times d$ veces, siendo d el número de canales de la imagen. Una vez hecha esta conversión, se ordenan los datos en un histograma con *k bins,* obteniendo *n+1* histogramas diferentes. Se debe tener en cuenta que la calidad de seguimiento del objeto depende del número de *bins* del histograma, si se tienen muy pocos *bins*, la estimación no será precisa. Finalmente, al tener todos los histogramas, se deben calcular *n+1* coeficientes de Bhattacharyya y ordenarlos, eligiendo únicamente el mayor, ya que este es el más parecido al coeficiente de la subimagen base.

Esta métrica se descartó, debido a que su costo computacional es demasiado alto. Al implementarlo y probar esta métrica, se concluyó que requiere muchas operaciones en tan solo un cuadro de video. Además, es muy sensible a los cambios de tamaño de ventana, es decir, si se halla un rectángulo contenedor muy grande, el número de comparaciones que se deben hacer aumenta en una relación de $w \times h$. Por ende, no se cumpliria el objetivo específico 3, afectando la finalidad de este trabajo de grado.

A raíz de esto, se probaron las demás métricas de comparación descritas. Se comparó el tiempo de ejecución de cada una de ellas y se obtuvo una mejoría notable con respecto al tiempo de ejecución utilizando el coeficiente de Bhattacharyya.

Video	SQDIFF (s)	SQDIFF_N (s)	CCORR (s)	CCORR_N (s)	CCOEFF (s)	CCOEFF_N (s)
Avenida calle 63 Carrera 15	811.272	898.169	833.400	881.784	863.997	903.728
Calle 37 Carrera 24	1164.630	1247.49	1298.065	1172.705	1164.810	1250.862

Tabla 6.1-1. Tiempo de ejecución de las diferentes métricas en el algoritmo de filtrado de partículas.

Con base en los resultados mostrados en la tabla 6.1-1, se puede ver que la métrica que minimiza el tiempo de ejecución del algoritmo es la diferencia de cuadrados (SQDIFF). Por lo tanto, se utilizará esta métrica para hacer las pruebas que se muestran a continuación.

6.2. Comparación de las técnicas

Las pruebas se hicieron con los videos tomados en la Calle 114 con Carrera 7 y la Avenida Calle 63 con Carrera 15, se muestran las perspectivas de los dos videos que se utilizaron para hacer las pruebas. En el video de la Calle 114 con Carrera 7 hay un flujo vehicular bajo, hay buena iluminación y el ángulo en el que se encuentra ubicada la cámara reduce la oclusión de objetos. Además, desde esta perspectiva, el tamaño de los objetos no es tan grande respecto a los que se encuentran en la Avenida Calle 63 con Carrera 15, lo que resulta en rectángulos contenedores y ventanas de comparación de menor tamaño, acelerando el proceso de seguimiento del objeto.

Por otro lado, en el video de la Avenida Calle 63 con Carrera 15 hay un mayor flujo vehicular y la cámara está ubicada de tal forma que se puede presentar oclusión entre objetos. La probabilidad de esta oclusión aumenta teniendo en cuenta que por la zona pasan vehículos grandes (buses y camiones).

6.2.1. Tiempo de ejecución

Video	Algoritmo de la SDM (s)	Flujo óptico (s)	Filtrado de partículas (s)
Calle 114 Carrera 7	1217.651	944.501	736.835
Avenida calle 63 Carrera 15	1533.087	643.201	811.272

Tabla 6.2.1-1. Tiempo de ejecución de los tres algoritmos a comparar.

Video	Flujo óptico	Filtrado de partículas
Calle 114 Carrera 7	22.43%	39.48%
Avenida calle 63 Carrera 15	46.82%	13.23%

Tabla 6.2.1-2. Mejora del tiempo de ejecución con respecto al algoritmo de la Secretaría de Movilidad.

Como se puede ver en la tabla 6.2.1-2, la técnica de filtrado de partículas presenta una mejora de 39.48% en el primer video y una mejora del 13.23% en el segundo. En este último video, la mejora no es significativa debido a que hay flujo vehicular de carros grandes. Estos vehículos, al estar muy cerca de la cámara, presentan rectángulos contenedores, y por lo tanto ventanas de comparación muy grandes, los cual aumenta las comparaciones en relación a $(w \times h)^2$, haciendo más lento su seguimiento.

La técnica de flujo óptico presenta una mejora en el tiempo de ejecución de 22.43% en el primer video, mientras que en segundo presenta una mejora de 46.82%. Esta técnica, a diferencia del filtrado de partículas, compara únicamente los descriptores de características hallados. Por lo tanto, su velocidad de procesamiento no cambia con el tamaño del rectángulo contenedor.

6.2.2. Conteo

Como se puede observar en la tabla 6.2.2-2, la técnica de flujo óptico presenta error de conteo de objetos de 15.555% en el primer video y para el segundo video un error de 12.916%. La técnica de filtrado de partículas presenta error de conteo de objetos de 2.222% en el primer video y para el segundo video un error de 10%. Estos valores se tomaron con respecto a un conteo manual.

Con respecto al conteo hecho por el algoritmo de la Secretaría de Movilidad, para el primer video se tiene un error de 15.555% para flujo óptico y 2.222% para filtrado de partículas. Para el segundo video estos algoritmos presentan un error de 2.22% para flujo óptico y 7.296% para filtrado de partículas.

A partir de estos datos, se puede ver que se tiene un mayor error de conteo en el video en el que hay un mayor flujo vehicular. Este error en el conteo puede deberse a la oclusión que se presenta entre los objetos. Además, las técnicas detectan aquellos objetos que el algoritmo de la Secretaría de Movilidad identifica en un principio. Es decir que durante la ejecución de cualquiera de las dos técnicas se hace seguimiento de aquellos objetos que el algoritmo de la Secretaría de Movilidad detectó e identificó, pero pueda que aparezcan nuevos objetos durante esos cuadros de video que no fueron detectados.

Video	**Conteo manual**	**Algoritmo de la SDM**	**Flujo óptico**	**Filtrado de partículas**
Calle 114 Carrera 7	45	45	38	44
Avenida calle 63 Carrera 15	240	233	209	216

Tabla 6.2.2-1. Conteo total de objetos de interés manual y en los tres algoritmos a comparar.

Video	**Flujo óptico**	**Filtrado de partículas**
Calle 114 Carrera 7	15.555%	2.222%
Avenida calle 63 Carrera 15	10.3%	7.296%

Tabla 6.2.1-2. Porcentaje de error con respecto al algoritmo de la SDM.

6.3. Resultados generales

En esta sección, se muestran los resultados de los 6 videos restantes utilizados para probar las dos técnicas implementadas en el presente trabajo de grado: flujo óptico y filtrado de partículas. Se hará una comparación del tiempo de ejecución con respecto al del algoritmo utilizado actualmente, así como del conteo de objetos de interés utilizando una línea de conteo. Cabe resaltar que en el conteo no se tienen en cuenta las etiquetas de los objetos, debido a que las técnicas implementadas se enfocan únicamente en el seguimiento y no en la clasificación de objetos de interés.

6.3.1. Tiempo de ejecución

En la tabla 6.3.1-1, se observan los tiempos de ejecución de cada algoritmo.

Video	Algoritmo de la SDM (s)	Flujo óptico (s)	Filtrado de partículas (s)
Carrera 19 Calle 39	1179.135	626.983	1023.099
Carrera 19 Calle 39-2	1173.546	673.854	930.778
Calle 37 Carrera 24	1991.804	950.307	1204.630
Calle 37 Carrera 24-2	1988.216	955.109	1390.630
Diagonal 15A Carrera 104	1135.256	563.419	1293.480
Diagonal 15A Carrera 104-2	1159.661	865.273	1447.235

Tabla 6.3.1-1. Tiempo de ejecución de los tres algoritmos a comparar en videos de duración de 900s.

En la tabla 6.3.1-2, se observan los porcentajes de mejora de los tiempos de ejecución de cada algoritmo, en comparación con el algoritmo utilizado actualmente.

Video	Flujo óptico	Filtrado de partículas
Carrera 19 Calle 39	46.82%	13.23%
Carrera 19 Calle 39-2	42.57%	20.68%
Calle 37 Carrera 24	52.29%	39.52%
Calle 37 Carrera 24-2	42.57%	30.05%
Diagonal 15A Carrera 104	50.37%	-13.93%
Diagonal 15A Carrera 104-2	25.38%	-24.79%

Tabla 6.3.1-2. Mejora del tiempo de ejecución con respecto al algoritmo de la SDM

6.3.2. Conteo

En la tabla 6.3.2-1, se observa el conteo de objetos de interés de cada algoritmo.

Video	Algoritmo de la SDM	Flujo óptico	Filtrado de partículas
Carrera 19 Calle 39	39	38	38
Carrera 19 Calle 39-2	38	37	37
Calle 37 Carrera 24	38	35	35
Calle 37 Carrera 24-2	33	33	34
Diagonal 15A Carrera 104	97	89	90
Diagonal 15A Carrera 104-2	94	85	84

Tabla 6.3.2-1. Conteo en los tres algoritmos a comparar.

En la tabla 6.3.2-2, se observan los porcentajes de error en el conteo en cada algoritmo, con respecto al algoritmo de seguimiento cuadro a cuadro.

Video	Flujo óptico	Filtrado de partículas
Carrera 19 Calle 39	2.56%	2.56%
Carrera 19 Calle 39-2	2.63%	2.63%
Calle 37 Carrera 24	7.89%	7.89%
Calle 37 Carrera 24-2	0%	3.03%
Diagonal 15A Carrera 104	8.24%	7.21%
Diagonal 15A Carrera 104-2	9.57%	10.63%

Tabla 6.3.2-2. Mejora del tiempo de ejecución con respecto al algoritmo de la SDM

Después de analizar en detalle cada vídeo, la tabla 6-1 muestra el promedio de las mejoras en tiempo de ejecución de los algoritmos implementados, así como el error en conteo, en comparación con el algoritmo usado actualmente por la Secretaría de Movilidad.

Prueba	Flujo óptico	Filtrado de partículas
Mejora de tiempo de ejecución	41.15%	14.68%
Error de conteo	7.07%	5.43%

Tabla 6-1. Promedio de la mejora del tiempo y del error en el conteo con respecto al algoritmo de la SDM

Con base en los datos mostrados en la tabla 6-1, se puede ver que utilizando la técnica de flujo óptico se obtiene una mejora del 43.333% con respecto al tiempo de ejecución de un algoritmo de seguimiento

cuadro a cuadro, mientras que con la técnica de filtrado de partículas se obtiene una mejora del 10.793%. Por otra parte, la técnica de flujo óptico presenta un acierto del 94.852% con respecto al porcentaje de acierto del algoritmo de seguimiento actual, mientras que con la técnica de filtrado de partículas se obtiene un acierto del 94.342%.

Esto se debe principalmente a que, con la técnica de filtrado de partículas, la velocidad de procesamiento depende de la ventana de comparación. Cuando se tiene un objeto muy cerca de la cámara, la ventana de comparación es muy grande, ya que es intrínseca a las dimensiones del rectángulo contenedor, y la cantidad de comparaciones que se deben hacer aumenta con el cuadrado de las dimensiones de la ventana. Por otro lado, la técnica de flujo óptico es independiente del tamaño del rectángulo contenedor del objeto, ya que únicamente se busca encontrar los descriptores de características del objeto en el instante $t + \Delta t$.

7. CONCLUSIONES

Para poder controlar el tráfico, se utiliza el análisis de flujo de vehículos. Ya sea manualmente o bien por análisis de videos de CCTV. Para este tipo de análisis se utilizan redes neuronales que sean capaces de identificar, detectar y hacer un seguimiento y conteo de los objetos de interés. En adición, estas redes neuronales pueden utilizar ciertas técnicas de seguimiento para acelerar o mejorar su rendimiento en cuanto al seguimiento de estos objetos. Dos de esas técnicas fueron propuestas en este trabajo de grado, con la finalidad de reducir un 30% el tiempo de ejecución y además en permitir un ±10% de error en el conteo de objetos. De esta manera se reduce la complejidad computacional del algoritmo utilizado por la Secretaría de Movilidad.

Por otra parte, se requería generar y almacenar las trayectorias de los objetos de interés. Este objetivo se logró, pero al haber objetos que tenían trayectorias cortas, irregulares y/o intermitentes; se percató que este objetivo no ayudaba al cumplimiento del objetivo principal, por lo que se enfocó únicamente en el conteo de los objetos y en el aceleramiento del algoritmo.

Inicialmente, las técnicas de seguimiento escogidas fueron: filtrado de partículas y CAMShift. Al ser desarrollados en las primeras etapas de este trabajo de grado, se observó que la segunda técnica era mucho más robusta y con mayor complejidad. Por lo que, al ser implementada para varios objetos de interés en un sistema de video cuadro a cuadro, no iba a permitir el cumplimiento del objetivo general. Esta técnica fue reemplazada por el flujo óptico, permitiendo el aceleramiento del seguimiento de objetos y reduciendo la complejidad computacional. De igual manera, el filtrado de partículas permitió tanto el aceleramiento del seguimiento de objetos, como la reducción de la complejidad computacional

Utilizando la técnica de filtrado de partículas se logró una mejora de tiempo de 14.68% con respecto al algoritmo utilizado por la Secretaría de Movilidad, con un error de 5.43% en el conteo. A pesar de que hubo una mejora con esta técnica, no se alcanzó el objetivo de reducción de tiempo de ejecución debido a que este método depende del tamaño del rectángulo contenedor del objeto. A medida que el rectángulo contenedor aumenta, la cantidad de comparaciones que se deben hacer aumenta a razón de $(w \times h)^2$. Es decir que, el filtrado de partículas es una técnica útil si se tienen tomas aéreas, donde el tamaño del objeto es pequeño, por ende, su rectángulo contenedor no varía. Para una mejora en el tiempo de ejecución, se puede implementar algún tipo de métrica, que sea independiente del tamaño del rectángulo contenedor.

Utilizando la técnica de flujo óptico, se logró una mejora de tiempo de 41.15% con respecto al algoritmo utilizado por la Secretaría de Movilidad, con un error de 7.07% en el conteo. Este método, a diferencia del filtrado de partículas, compara únicamente los descriptores de características hallados en cada objeto detectado. Por lo tanto, su velocidad de procesamiento no depende del tamaño del rectángulo contenedor.

Con respecto al conteo de objetos, la posición de la cámara es significativa. Si la posición de la cámara no es lo suficientemente elevada, los objetos de mayor tamaño ocupan gran parte del cuadro de video, dando lugar a oclusiones. Si se presentan oclusiones, se aumenta el error en el conteo de objetos de interés. Además, no se vio gran diferencia respecto a la diferencia de iluminación presente en los videos, por lo

que se puede afirmar que la luminosidad no es un factor que influya considerablemente en el aceleramiento de tiempo de ejecución o en el conteo de objetos.

Cabe resaltar en este trabajo de grado que con las dos técnicas utilizadas se hizo una tendencia en que, si se quiere más precisión en el conteo de objetos, se debe perder un poco en cuanto a la mejora de tiempo de ejecución. E igualmente, si se quiere más aceleramiento en el tiempo de ejecución, se pierde un poco en cuanto al conteo de objetos de interés. En general, ambas técnicas de seguimiento son parecidas en cuanto al conteo de objetos, pero en cuanto a la reducción de tiempo de ejecución, la técnica de flujo óptico es mucho más rápida por una gran diferencia, a pesar de que ambas técnicas aceleraron el procesamiento y redujeron la complejidad computacional del algoritmo.

8. BIBLIOGRAFÍA

[1] M. W. Green, *The Appropriate and Effective Use of Security Technologies in U.S. Schools: A Guide for Schools and Law Enforcement Agencies*. Albuquerque, New Mexico: Sandia National Laboratories, 1999.

[2] Movilidad en Cifras 2015. Secretaría de movilidad. Alcaldía Mayor de Bogotá. [Online] http://www.movilidadbogota.gov.co/web/datos_abiertos

[3] P. Wang, X. Yan and Z. Gao, "Vehicle counting and traffic flow analysis with UAV by low rank representation," *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Macau, 2017, pp. 1401-1405.

[4] I. S. Farias, B. J. T. Fernandes, E. Q. Albuquerque and B. L. D. Leite, "Tracking and counting of vehicles for flow analysis from urban traffic videos," *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, Arequipa, 2017, pp. 1-6.

[5] M. Kafai and B. Bhanu, "Dynamic Bayesian Networks for Vehicle Classification in Video," in *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 100-109, Feb. 2012.

[6] J. Redmon and A. Farhadi, *YOLO: Real-Time Object Detection*, 2018. [Online]. Available: https://pjreddie.com/darknet/yolo/ [Accedido: 28-May-2019].

[7] G. Bradski and A. Kaehler, *Learning OpenCV*. Capítulo 1. OReilly Media, 2015.

[8] J. Hui, "Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3," *Medium*, 18-Mar-2018. [Online]. Available: https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088 [Accedido: 28-May-2019].

[9] A. Doucet and M. Johansen, "A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later," in *The Oxford Handbook of Nonlinear Filtering*, 1st ed., Oxford: Oxford University Press, 2011, pp. 656–704.

[10] M. S. Arulapalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[11] P. Pan and D. Schonfeld, "Visual Tracking Using High-Order Particle Filtering," in *IEEE Signal Processing Letters*, vol. 18, no. 1, pp. 51-54, Jan. 2011.

[12] K. Hsiao, H. de Plinval-Salgues and J. Miller. Particle Filters and Their Applications. Apr, 2005. Available: http://web.mit.edu/16.412j/www/html/Advanced%20lectures/Slides/Hsaio_plinval_miller_ParticleFiltersPrint.pdf. [Accedido: 28-May-2019].

[13] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence Laboratory. Massachusetts Institute of Technology*. vol. 17, no. 1-3, pp. 185–203, Aug. 1981.

[14] Rojas, R. Lucas-Kanade in a Nutshell. Freie Universität Berlin, Dept. of Computer Science, Arnimallee. [Online] http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/tutorials/Lucas-Kanade2.pdf

[15] "OpenCV documentation index," *OpenCV documentation index*. [Online]. Available: https://docs.opencv.org/ [Accedido: 27-May-2019].

[16] Motion and optical flow. University of Washington Computer Science & Engineering. [Online] https://courses.cs.washington.edu/courses/cse455/16wi/notes/14_Optical_Flow.pdf [Accedido: 28-May-2019].

[17] Features and Image Matching. Paul G. Allen School of Computer Science & Engineering.[Online] https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect6.pdf [Accedido: 28-May-2019].

[18] Y. Xu, K. Xu, J. Wan, Z. Xiong and Y. Li, "Research on Particle Filter Tracking Method Based on Kalman Filter," *2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, Xi'an, 2018, pp. 1564-1568.

[19] J. González. Introducción del Factor Humano al Análisis de Riesgo. Universitat Politècnica de Catalunya, 2015.

[20] Cuadras, C. Distancias Estadísticas. Departamento de Estadística. Universidad de Barcelona.

[21] Hisham, M.B. et al. Template Matching Using Sum of Squared Difference and Normalized Cross Correlation. School of Computer and Communication Engineering, University Malaysia Perlis, Kangar, Perlis, Malaysia.

[22] P. Bourke, "AutoCorrelation -- 2D Pattern Identification," *Cross Correlation*, Aug-1996. [Online]. Available: http://paulbourke.net/miscellaneous/correlate/ [Accedido: 27-May-2019].

[23] A. Kaur, L. Kaur, and S. Gupta, "Image Recognition using Coefficient of Correlation and Structural Similarity Index in Uncontrolled Environment," *International Journal of Computer Applications*, vol. 59, no. 5, pp. 32–39, Dec. 2012.

[24] C.-en Lin, "Introduction to Motion Estimation with Optical Flow," *The Nanonets Blog*, 24-Apr-2019. [Online]. Available: https://blog.nanonets.com/optical-flow/ [Accedido: 27-May-2019].

[25] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proceedings of the Alvey Vision Conference 1988*, pp. 147–152, 1988.

[26] Jianbo Shi and Tomasi, "Good features to track," *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 1994, pp. 593-600.

[27] Tarasenko, V. and Park, D. Detection and Tracking over Image Pyramids using Lucas and Kanade Algorithm. International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 9 (2016) pp 6117-6120

9. ANEXOS

https://drive.google.com/drive/folders/1fJJdMba0XuvhOH-C3AXcX9Ux9uUPisQp?usp=sharing