**PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ**

**FACULTAD DE CIENCIAS E INGENIERÍA**

# ANÁLISIS DE FLUJO PEATONAL EN UNA INTERSECCIÓN DE AVENIDAS UTILIZANDO PROCESAMIENTO DE IMÁGENES

**Tesis para obtener el título profesional de Ingeniero Electrónico**


**AUTOR:**

Rolando Jesús Reátegui Arones (20131076)

**ASESOR:**

Pedro Moisés Crisóstomo Romero


Lima, Agosto 2021

## RESUMEN

En diversos cruces de avenidas de la ciudad de Lima, se observan patrones anormales de comportamiento peatonal, inadecuada semaforización y diseño vial no ideal para los peatones. Estos últimos son los más vulnerables en los accidentes de tránsito y son quienes deberían requerir mayor prioridad en el diseño de intersecciones. Para mejorar los diseños de las intersecciones se requiere datos de flujo peatonal, que se pueden medir de diversas maneras.

Actualmente, el método de conteo de peatones utilizado es manual, el cual es realizado por equipos de personas desde una esquina o mediante una grabación y que tiene una baja eficiencia por error humano, además del costo por hora que ello implica. Por esta razón, la presente tesis expone nuevos métodos de conteo más eficaces ejecutados en una intersección conocida y sus resultados comparados con cifras esperadas.

El primer capítulo de la tesis presenta el marco problemático que justifica la importancia de la presente tesis, los métodos generales usuales para detección de peatones y expone los objetivos de la tesis.

El segundo capítulo describe los fundamentos teóricos sobre procesamiento de imágenes utilizados para el desarrollo de los posteriores capítulos y los métodos específicos para cada etapa del algoritmo.

El tercer capítulo enumera los pasos de la propuesta para el conteo de peatones, las funciones implementadas y librerías utilizadas para cada una de las etapas de esta aplicación.

Por último, el cuarto capítulo revisa los resultados de la propuesta por cada etapa para diferentes videos y hace un análisis de estos para su recomendación de uso en futuras aplicaciones.

## ÍNDICE GENERAL

## ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

Pág.

## Capítulo 1: MARCO PROBLEMÁTICO

### 1.1. Definición problemática

En zonas urbanas, las calles son utilizadas por peatones y vehículos. Es en el cruce de vías donde se genera la interacción entre ellos. En ese momento, se propicia un escenario con riesgo de atropello y accidentalidad. A pesar de ello, los transeúntes evidencian conductas imprudentes. Manifiestan intenciones que muestran una decisión consciente de irrumpir las normas. Además, despliegan un comportamiento riesgoso. Dichas conductas se manifiestan debido a la conveniencia del usuario, el deseo de ahorrar tiempo o el hecho de no percibir algún riesgo al realizar maniobras imprudentes.[1]

Entre estas conductas destacan el cruzar la pista cuando aún no se le ha indicado o en diagonal para ahorrar tiempo, además de no usar los puentes peatonales cuando debería utilizarlos en calles con alto flujo vehicular o abordar el transporte público en medio de la pista.

Diversos gobiernos locales han hecho lo posible para reducir estas conductas imprudentes, ya sea con una mayor cultura peatonal, con una organización del tráfico al temporizar los semáforos o diseñar los cruces peatonales y multas para peatones.

Un ejemplo claro son los llamados bailes de Barnes o cruces diagonales, implementados en ciudades de países desarrollados, los cuales permiten el cruce entre esquinas y cuentan con ventanas de tiempo de 23 segundos para peatones y 90 segundos para automóviles. Los efectos que estos traen a largo plazo son la reducción de los accidentes de tránsito de 19 a 1 por año, además de un aumento de la satisfacción peatonal. Un ejemplo de estos cruces es en el cruce de las avenidas Hollywood Boulevard y Highland Avenue en Los Ángeles, California (Figura 1). Otros diseños se pueden encontrar en Oakland, California y Shibuya, Tokio. [2]

*Figura 1*: El cruce de Hollywood Boulevard antes y después. Aparentemente desorganizado, funciona mejor que el regulado por semáforos tradicionales. Fuente: El diario.es [2]

En el caso de la ciudad de Lima, el 25% de viajes diarios se hace a pie, lo que significa que un aproximado de 4,2 millones de personas caminan para ir a estudiar, trabajar, hacer compras o simplemente salir a pasear, según el Plan Maestro de Transporte Urbano para Lima y Callao [3]. Si bien caminar es el modo más simple y saludable de movilidad, a la vez, es una carrera de obstáculos donde gobierna el automóvil. No hace falta que se lea un documento con un interminable diagnóstico, basta con recorrer a pie cualquier parte de la ciudad para comprobar que no está pensada ni diseñada para el peatón. [3]

Para explicar la anterior postura, se enlistan a continuación los factores que no permiten la movilidad apropiada a pie:

• Hay semaforización peatonal insuficiente y los pocos semáforos existentes no tienen tiempos de verde destinados para los peatones.

• Existe un conflicto de flujos vehiculares con flujos peatonales en el momento de giro a la derecha e izquierda; por ello, no existen tiempos destinados para el cruce de peatones.

• Los puentes peatonales para los cruces de las vías no existen o tienen mala ubicación.

• Hay un escaso o nulo mantenimiento de la señalización preventiva o regulatoria.

• No existe preferencia al peatón en las vías.

• Los vehículos no respetan los cruceros peatonales y tienden a acelerar en los momentos de luz ámbar. [4]

Se darán ejemplos de algunos de estos factores. En primer lugar, la pirámide de jerarquía de movilidad urbana según ITDP (Instituto de Políticas para el Transporte y el Desarrollo, por sus siglas en inglés), que se muestra en la Figura 2, da a notar que los peatones deberían tener la mayor prioridad en la movilidad urbana y los autos privados deben tener la última. [5]

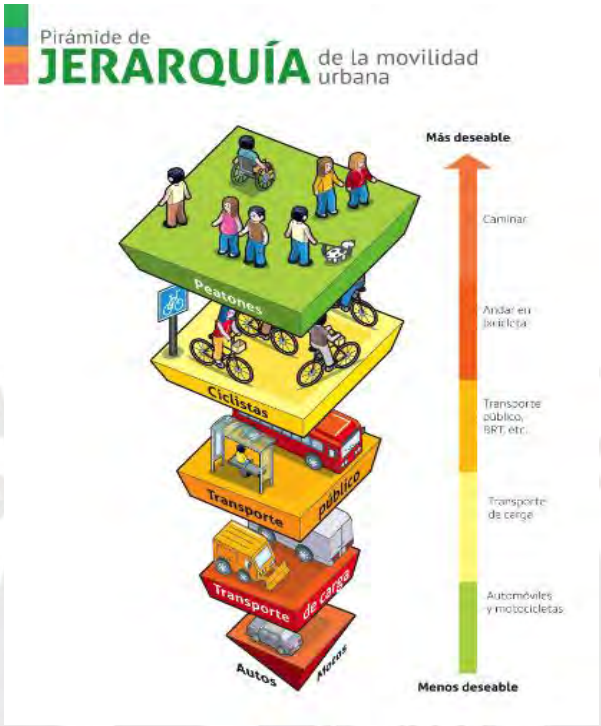



*Figura 2*: Pirámide de jerarquía urbana. Fuente: ITDP [5]

Sin embargo, en la Figura 3, se muestra lo que en realidad sucede. Como se ve claramente, los autos intentar girar al mismo tiempo que los peatones están a tiempo de cruzar y los primeros aún así no toman en cuenta a los últimos. Además, se muestra un ejemplo de falta de semaforización para el peatón en la Figura 4.

*Figura 3*: Inexistencia de prioridad al peatón en Estación "La Cultura". Fuente: propia.

*Figura 4*: Falta de semaforización para el cruce peatonal en Santiago de Surco. Fuente: Informe final de vulnerabilidad de peatones, MTC - Perú [4]

Por otra parte, los ciclos semafóricos están calculados para disminuir la congestión vehicular, mas no para el pase de peatones. Por ejemplo, la Municipalidad de Lima, a través del Proyecto Especial para la Gestión de Tránsito (PROTRÁNSITO), sincronizó los tiempos y ciclos semafóricos en 355 intersecciones para establecer "olas verdes" en importantes vías metropolitanas. Con esta labor, no solo se ha optimizado la gestión del tránsito, sino que también se ha reducido en un 15% los tiempos de viaje, los índices de contaminación sonora y ambiental, lo que ha permitido contribuir a mejorar la calidad de vida de los ciudadanos [6]. Sin embargo, una vez más, se ha priorizado el tránsito vehicular en estas avenidas e intersecciones, mas no se ha mejorado este aspecto para los peatones, los cuales tienen que esperar varios minutos para poder cruzar las avenidas o correr antes de que se aproxime un auto.

Por otro lado, en el ámbito de la seguridad al caminar, se calcula que, de cada 100 personas muertas en accidentes de tránsito en la capital, 78 son peatones, de acuerdo con el estudio "Vulnerabilidad de los peatones en la vialidad del área metropolitana de Lima y Callao", de la Secretaría Técnica del Consejo de Transporte de Lima y Callao. [7] Otro estudio llamado "Supervisión de las condiciones de infraestructura vial en puntos críticos de accidentes de tránsito en los distritos de Lima y Callao", realizado por la defensoría del pueblo, detalló que 46% de cruceros peatonales no están señalizados. [8]

En la Tabla 1, se muestran los puntos de mayor número de atropellos en la zona de Lima Metropolitana que han sucedido en intersecciones:

*Tabla 1*: Ranking de los 10 puntos de ocurrencia de mayor cantidad de atropellos.

| Nº | DISTRITO | ATROPELLOS | PEATONES |
|---|---|---|---|
| 1 | SANTIAGO DE SURCO | 53 | 56 |
| 2 | COMAS | 34 | 37 |
| 3 | INDEPENDENCIA | 34 | 34 |
| 4 | ATE VITARTE | 32 | 31 |
| 5 | SAN JUAN DE LURIGANCHO | 32 | 31 |
| 6 | COMAS | 26 | 28 |
| 7 | ATE VITARTE | 23 | 27 |
| 8 | SAN JUAN DE LURIGANCHO | 23 | 26 |
| 9 | SANTIAGO DE SURCO | 22 | 24 |
| 10 | CERCADO DE LIMA | 24 | 24 |

Fuente: Informe final de vulnerabilidad de peatones, MTC - Perú [4]

¿La imprudencia es la única explicación a todos estos problemas? Diego Vargas Cardoso, responsable del Plan Maestro de Transporte Urbano para Lima y Callao, sostiene que existe otro factor que hace que los transeúntes busquen formas inseguras al momento de cruzar las vías: la inadecuada infraestructura vial para peatones en la ciudad. Por todo lo anterior, se concluye que es innegable que Lima es una ciudad hostil para el caminante, tanto por ser de alto riesgo en temas de seguridad como al no construirse una infraestructura adecuada para ellos [3].

Para cambiar esta situación, se requiere un buen planeamiento, diseño y desarrollo con el fin de garantizar las mejores condiciones de los espacios públicos, con los estándares de calidad y seguridad apropiada para la movilidad de la población y el desarrollo de sus actividades [4].

En 2011, Alex Quistberg, investigador de la Universidad de Washington, dirigió un estudio para evaluar la relación entre los atropellos peatonales y la infraestructura del sistema de transporte público de Lima. Para ello, necesitaban datos como flujo de autos, velocidad máxima permitida y flujo de peatones en algunas avenidas e intersecciones de Lima [7]. Por lo tanto,

conocer el flujo de peatones es de alta importancia al hacer estadísticas o mediciones en tiempo real para tomar decisiones con los resultados obtenidos.

Así, el presente trabajo tiene como objetivo brindar datos numéricos acerca del número de peatones que transitan por una determinada intersección de avenidas de manera automática en determinadas horas.

Se tendrá como campo de estudio al cruce de las avenidas Javier Prado y Aviación, usado continuamente por peatones por estar a la salida de la estación de tren "La Cultura". Este cruce de avenidas tiene 8 cruces peatonales cada uno con cámaras instaladas de manera vertical para los peatones o frontales para los vehículos.

Se ha observado, por simple inspección, conductas imprudentes por parte de los peatones en esta intersección como correr mientras no ven que se acercan vehículos a distancias cortas incluso cuando el semáforo aún no les ha indicado que crucen de frente o en diagonal, por lo que hay altos riesgos de accidentes vehiculares, especialmente en horas punta (7am-10am y 4pm-8pm).

Se mostrarán algunas soluciones disponibles para determinar este número de peatones que se han expuesto en conferencias a lo largo de la historia en el siguiente capítulo.

En la Figura 5 se muestran conductas como las mencionadas anteriormente:

*Figura 5*: Intersección de Javier Prado con Aviación. Arriba: imprudencia del peatón de cruzar cuando no es el momento. Abajo: Necesidad de un cruce en diagonal. Fuente: propia.

## 1.2. Estado del arte

Diversas soluciones se han ideado para la medición del flujo peatonal. En las siguientes líneas, se describirán tres categorías para los métodos de conteo de personas en intersecciones.

### 1.2.1. Técnicas manuales con equipos de personas en las esquinas

El primer método de conteo de personas es el más simple: el conteo manual. Para esto, es necesario que en las intersecciones de avenidas se cuente con equipos preparados para contar a los peatones y ciertas características, tales como género, rango de edad, dirección, etc. Un ejemplo de este tipo de conteo fue hecho en mayo de 2016 por un equipo de la Universidad de Berkeley para comparar métodos de conteo en 10 intersecciones en la ciudad de San Francisco por cuatro días durante cuatro horas por día con descansos de una hora. Dos personas fueron contratadas para esta tarea. Las intersecciones entre sí tenían diferentes flujos de peatones, que variaron entre 12 a 262 peatones cruzando por hora. Para el conteo, se utilizó principalmente tres clases de materiales de ayuda que se explicarán a continuación. [9]

#### 1.2.1.1. Conteo manual usando papel

El observador de campo recibió un documento que contenía tres campos: dirección de movimiento, género y edad (este último era juzgado por el propio observador dentro de siete categorías de edad). Sin embargo, el observador debía concentrarse principalmente en el conteo de personas, incluso si significaba dejar vacíos los campos de género y edad en intersecciones muy congestionadas. Para mejorar el estudio, al observador se le pidió que anotara características que distinguieran a un peatón de otro como color de ropa, color de cabello, maletas o maletines, tiempo exacto, entre otros. Esta información era necesaria para saber si el observador en algún momento no contaba o sobre contaba peatones, además de determinar si los datos manuales estaban sincronizados con el video. [9]

1.2.1.2. Conteo manual usando contador

Durante los dos últimos días del estudio, un observador utilizó un contador (*counter*, en inglés) o *clicker* con el fin de contar a los peatones. El observador avanzaba la cuenta por cada peatón cruzando la intersección, sin importar su dirección. Por cada diez minutos de conteo seguidos, el observador anotaba la cuenta mostrada en el *clicker* en el documento entregado. [9] Los métodos manuales se ilustran en la Figura 6:

*Figura 6*: Ilustración de técnicas manuales. A la izquierda, método de conteo por observador directo y papel. A la derecha, un ejemplo de *clicker* utilizado por el observador para ahorrar tiempo [10] [11]

1.2.1.3. Conteo manual usando grabación de video.

Las intersecciones fueron además grabadas usando una cámara instalada en un parqueo de camiones que miraba a la intersección en estudio. Los investigadores envueltos en el estudio analizaban cuidadosamente las grabaciones con el fin de obtener los resultados más confiables. Ellos trataban de identificar cada peatón contado por el observador. Esta tarea solo era posible en los días en los que el observador anotaba las características individuales de los peatones. Las grabaciones fueron vistas durante un tiempo variable, y, a veces, requerían más de una revisión si es que los resultados estaban en duda. En promedio, una hora de video requería tres horas de análisis. Durante este análisis, los investigadores prestaron atención a si el conteo en campo estaba sincronizado con la grabación y buscaban cualquier discrepancia entre las observaciones de campo y las imágenes en video. [9] Los resultados del conteo manual se muestran la Tabla 2:

*Tabla 2*: resultados de método manual [9]

| Nº | MÉTODO | VT | VP | FN | RFN(%) |
|---|---|---|---|---|---|
| 1 | MANUAL CON PAPEL | 150 | 128 | 22 | 14.67 |
| 2 | MANUAL CON PAPEL | 55 | 49 | 6 | 10.91 |
| 3 | MANUAL CON PAPEL | 521 | 412 | 109 | 20.92 |
| 4 | MANUAL CON PAPEL | 1188 | 1046 | 142 | 11.95 |
| 5 | MANUAL CON PAPEL | 371 | 334 | 37 | 9.97 |
| 6 | MANUAL CON PAPEL | 715 | 651 | 64 | 8.95 |
| 7 | MANUAL CON PAPEL | 772 | 579 | 193 | 25.00 |
| 8 | MANUAL CON PAPEL | 1130 | 994 | 136 | 12.04 |
| 9 | MANUAL CON CLICKER | 175 | 161 | 14 | 8.00 |
| 10 | MANUAL CON CLICKER | 398 | 338 | 60 | 15.08 |

En este caso, el RFN (ratio de falsos negativos o ratio de pérdida) se calcula con la siguiente fórmula 1:

$$RFN\ (\%) = \frac{FN}{FN+VP} * 100\%$$ Fórmula 1

donde

VT: Verdad en tierra, en este caso, el total de peatones que se registró en el video

FN: Falsos negativos, en este caso, los peatones que no se contaron con los métodos manuales

VP: Verdaderos positivos, en este caso, el número de peatones contados en la grabación usando la videocámara

La verdad en tierra, que es el número total de peatones presentes, se calcula utilizando el video ya que no se tenía datos explícitos del número real de peatones. Los problemas presentados al ejecutar cualquier método manual son los siguientes:

- El nivel de atención del observador es difícil de controlar.
- Se requiere planeamiento y organización por parte de la empresa que cuenta.
- El observador debe tener familiaridad con las intersecciones que va a evaluar.
- Se requiere que el observador esté largos periodos atento y que no se distraiga si es que no existe mucho flujo peatonal.

- Factores humanos como distracción o sobreconteo aumentan sistemáticamente el error en el conteo.

Finalmente, cabe recalcar que no hay diferencia relevante entre conteo por papel y conteo con *clickers*, ya que los RFN de ambos casos es similar.

1.2.2. Uso de láser infrarrojo

En 2008, un grupo de investigadores presentaron en Corea del Sur un método para conteo de personas utilizando láseres de luz infrarroja. En este método, la reflexión de láser infrarroja es obtenida utilizando un filtro IR. Por lo tanto, la adquisición de imágenes de objetos moviéndose es extremadamente simple y confiable comparada con otros métodos basados en visión. El número de peatones es contado usando simple análisis de manchas (*blob analysis*) y sus direcciones de movimiento son obtenidas de las inclinaciones de estos segmentos. Este método tiene la ventaja de hacer el proceso más simple ya que solo utiliza un filtro IR que lo hace robusto a cambios en el ambiente como iluminación. Adicionalmente, no molesta al flujo de peatones en absoluto ya que no hace contacto directo con ellos. Además, el láser es invisible, por lo que la gente no se da cuenta del proceso de conteo. [12] La Figura 7 muestra la configuración del método:

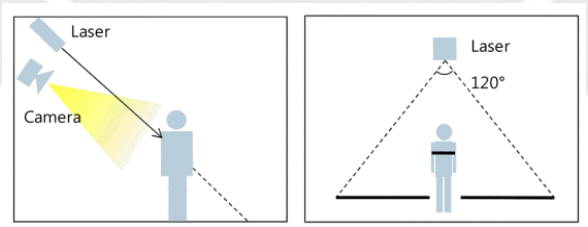



*Figura 7*: a la izquierda, una línea láser infrarroja es proyectada en un área de observación. La reflexión del láser es obtenida y la posición de la reflexión cambia como se muestra a la derecha. Fuente: IEEE [12]

La longitud de onda del módulo láser fue de 855 nm y el ángulo fue de 120º. El filtro IR tiene longitud de onda central de 852.01 nm y ancho de banda de 10.22 nm. [12] El resultado del filtro IR se muestra en la Figura 8:

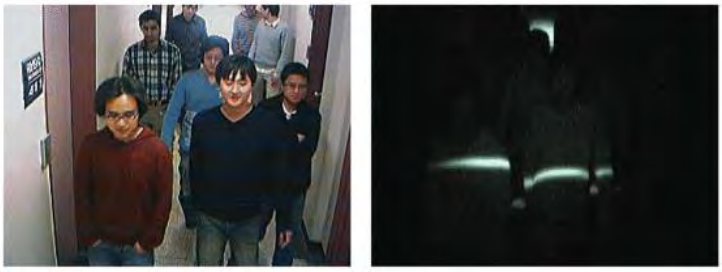


*Figura 8*: a la izquierda, la imagen original; a la derecha, la imagen con filtro IR [12]

Para la evaluación del método, una secuencia de video de una hora obtenida de una escena real es usada. La resolución del video de evaluación fue de 320x240. El total de peatones fue de 277 y 278 para ambas direcciones. [12]. La Tabla 3 resume los resultados obtenidos por este método:

*Tabla 3*: resultados de método con láser IR [12]

| | VT | VP | FP | FN | RFP (%) | RFN (%) |
|---|---|---|---|---|---|---|
| HACIA ABAJO | 277 | 269 | 7 | 15 | 2.54 | 5.28 |
| HACIA ARRIBA | 278 | 271 | 7 | 14 | 2.52 | 4.91 |

En este caso, el RFN (ratio de falsos negativos) se calcula de la misma manera que en la fórmula 1; mientras que el RFP (ratio de falsos positivos), se calcula de manera análoga:

$$RFP\ (\%) = \frac{FP}{FP+VP} * 100\%$$ Fórmula 2

Donde

VT: Verdad en tierra, cantidad total de personas

FP: Falsos positivos, en este caso, segmentos que fueron contados como personas sin que se tratase de ellas

FN: Falsos negativos, en este caso, manchas que no se llegaron a contar como personas

VP: Verdaderos positivos, en este caso, el número total de personas contadas por el algoritmo

Como se muestra en la Tabla 3, el ratio de falsos positivos fue de 2.5% en promedio y de falsos negativos fue de 5% en promedio. El tiempo de procesamiento promedio fue de

aproximadamente 9ms por cuadro en una computadora personal estándar. La mayoría de las detecciones no contadas ocurrieron cuando muchas personas pasaron alrededor del borde del sistema, lo que inducía un pequeño ancho de mancha que fue detectado como candidato para la segmentación. Con estos experimentos, se concluye que efectivamente este método de conteo por láser produce ratios de falsos negativos menor al de los métodos manuales. [12]

1.2.3. Conteo por medio de visión por computadora.

Contar personas con los métodos anteriores resulta laborioso y con alto costo tanto del personal como de los equipos que utilicen para el caso manual o de mantenimiento para el caso IR. Por lo tanto, es importante desarrollar métodos automáticos para contar personas a través de visión por computadora. En años pasados, el efectivo desarrollo de sistemas automáticos para contar personas basados en procesamiento de imágenes digitales ha alcanzado bastante interés de investigación.

Las cámaras que se utilizan para fin de conteo de personas se distribuyen en el mercado con precio entre los 900 y 1300 euros por cadenas como CableMatic o Mirame.net. El costo varía de acuerdo a la tecnología usada y la altura con respecto al suelo [13] [14].

La Figura 9 muestra un ejemplo de estas cámaras y su respectiva visión:

*Figura 9*: ejemplo de cámara IP y el resultado de cuenta personas de Mirame.net. Fuente: Cablematic.com y Mirame.net [13][14]

1.2.3.1. Método de conteo de Bin Li et. al [15]

Este primer método, propuesto en una conferencia en Hong Kong, contempla el conteo de personas en un ambiente cerrado por medio de un video realizado por una cámara vertical

puesta en el techo y usa algoritmos tanto propuestos por los mismos autores como otros usados de otros programadores.

Primeramente, el método usa una cámara vertical para adquirir videos de gente moviéndose con el fin de separar las personas en individuos. Esta tarea puede evitar el solapamiento de personas moviéndose que usualmente ocurre en el conteo basado en detección de cara, hombros o personas. Además, se reduce el ratio de falsa detección a través de la extracción de fondo y el clasificador Adaboost usados, que puede eliminar otros objetos estáticos e irrelevantes en imágenes en análisis. Aún más, el método produce una lista rastreadora de cabezas que se actualiza todo el tiempo, con el fin de almacenar la posición de cada objeto cabeza y asegurar el conteo de personas en la muchedumbre.

Para la extracción de fondo, el primer paso del método, se usa el algoritmo VIBE propuesto por Olivier Barnich et al. por ser eficiente, robusto al ruido y toma poca memoria, por lo que se usará como primera etapa del conteo. El resultado de ello se muestra a continuación:

*Figura 10*: A la izquierda, la imagen inicial de la cámara. A la derecha, la imagen con el fondo extraído usando el algoritmo *ViBe*. Fuente: IEEE. [15]

Luego, se corrigen los posibles hoyos que se encuentre en el fondo extraído, se recorta el contorno de las personas y se identifica la cabeza de cada una. Para descartar si se trata de una cabeza o no, se busca la cabeza similar en una base de datos. Finalmente, se delimita la zona de caminata para verificar si es que la persona está en una zona delimitada para el conteo. [15]

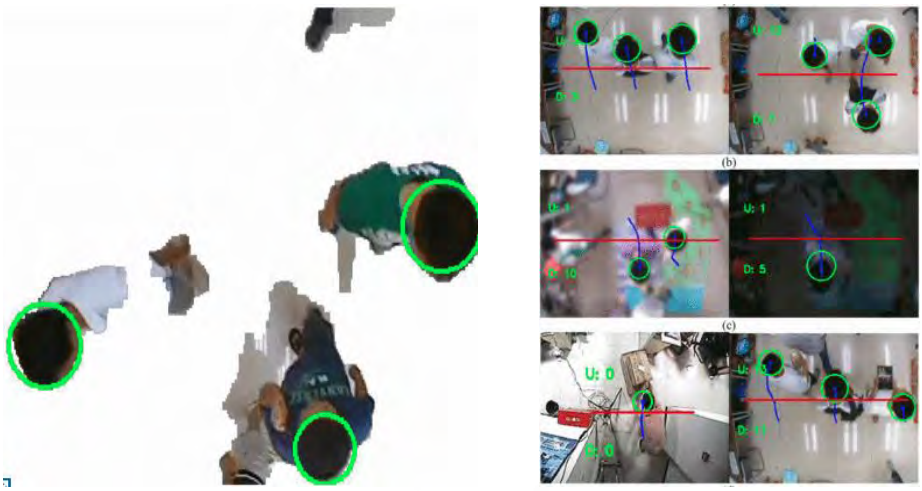


*Figura 11*: A la izquierda, las cabezas de las personas identificadas. A la derecha, el conteo final en zonas restringidas para este. Fuente: IEEE. [15]

Se probó con una cámara de resolución de 352*288, además se usó una PC con 2GB de memoria y un CPU de 2.49 GHz. El tiempo de procesamiento fue de 36.4ms. Los resultados de la prueba se muestran en la Tabla 4.

*Tabla 4*: resultados de método de conteo de cabezas [15]

| Método | Exactitud | RFP (%) | RFN (%) |
|---|---|---|---|
| MÉTODO DE CONTEO DE CABEZAS | 98.1 | 1.1 | 1.9 |

donde

RFP : Ratio de falsos positivos

RFN : Ratio de falsos negativos

Exactitud: se define como

$$Exactitud = \frac{VP+VN}{VP+FP+VN+FN} * 100\%$$ Fórmula 3

VP: verdaderos positivos

FP: falsos positivos

VN: verdaderos negativos

FN: falsos negativos

Por lo indicado en los resultados, este primer método con cámara promete alta exactitud y aceptable velocidad de computación y es ideal para aplicaciones en tiempo real.

1.2.3.2. Un nuevo modelo de conteo de pasajeros de Mukherjee et al.

En este segundo método, se propone un modelo para contar el total número de personas en una estación de tren. Este modelo tiene tres fases: detección de objeto, rastreo de objeto y validación de objeto. [16]

El paso de detección de objeto detecta a una persona mientras esta entra en el rango de la imagen. Luego, el rastreo de objeto sigue a la persona mientras esta se mueve en la imagen y genera como resultado una trayectoria. Las trayectorias son luego analizadas en el paso de validación y el número de trayectorias válidas se graba. Este último número brinda el número de personas. El algoritmo se ejecuta bien tanto en ambientes congestionados o sin muchas personas y funciona con varios tipos de personas: personas con diferentes colores de cabello, usando gorras, chaquetas o llevando bolsas. El modelo propuesto es además inteligente para contar a personas entrando en la imagen desde cualquier dirección. [16]

A continuación, se muestra la detección por parte del algoritmo:

*Figura 12*: El segundo método de conteo para tres diferentes cantidades de personas de menor a mayor (izquierda a derecha). Fuente: IEEE [16]

Para probar este método, se montaron cámaras en los techos de las entradas y salidas en diferentes estaciones de tren de la ciudad durante diferentes horas en un día con tres diferentes densidades: alta, media y baja. El total número de imágenes analizadas fue de 2000. Adicionalmente, el modelo fue probado en un video 5000 imágenes donde las personas se movían en diferentes direcciones. Los resultados de las pruebas se muestran en la Tabla 5.

*Tabla 5*: Resultados del método de conteo de pasajeros [16]

| DENSIDAD | Exactitud (%) | RFN (%) |
|---|---|---|
| BAJA | 83 | 5 |
| MEDIA | 96 | 0 |
| ALTA | 71 | 11 |

donde

RFN : Ratio de falsos negativos

Por lo que se indica, este método no es ideal para la aplicación en estudio ya que varían sus resultados de acuerdo con el número de personas en cuestión a lo largo de las pruebas.

1.2.3.3. Monitoreo de peatones en intersecciones incluyendo comportamiento y conteo de cruces [17]

Esta tercera técnica de visión por computadora presenta un sistema de seguimiento que entrega conteo y análisis del comportamiento de peatones mediante el uso de una infraestructura existente de cámara de tráfico. El sistema propuesto es capaz de detectar peatones tanto estacionarios como en movimiento a través de fusión contextual de apariencia y señales de movimiento. La eficacia del reconocimiento de peatones es mejorada mediante algoritmos cooperativos de seguimiento. La unión de gráficos bipartitos es usada para inicializar nuevos peatones detectados y el flujo óptico es luego utilizado para manejar el seguimiento a través de oclusiones parciales. El análisis del comportamiento peatonal incluye tiempo de espera, velocidad de cruce y parámetros espacio-temporales como longitud de pasos y frecuencia de los mismos.

El diagrama de flujo de esta cuarta técnica se muestra en la Figura 13. Adicionalmente, se da una muestra de video utilizada en este método en la Figura 14.

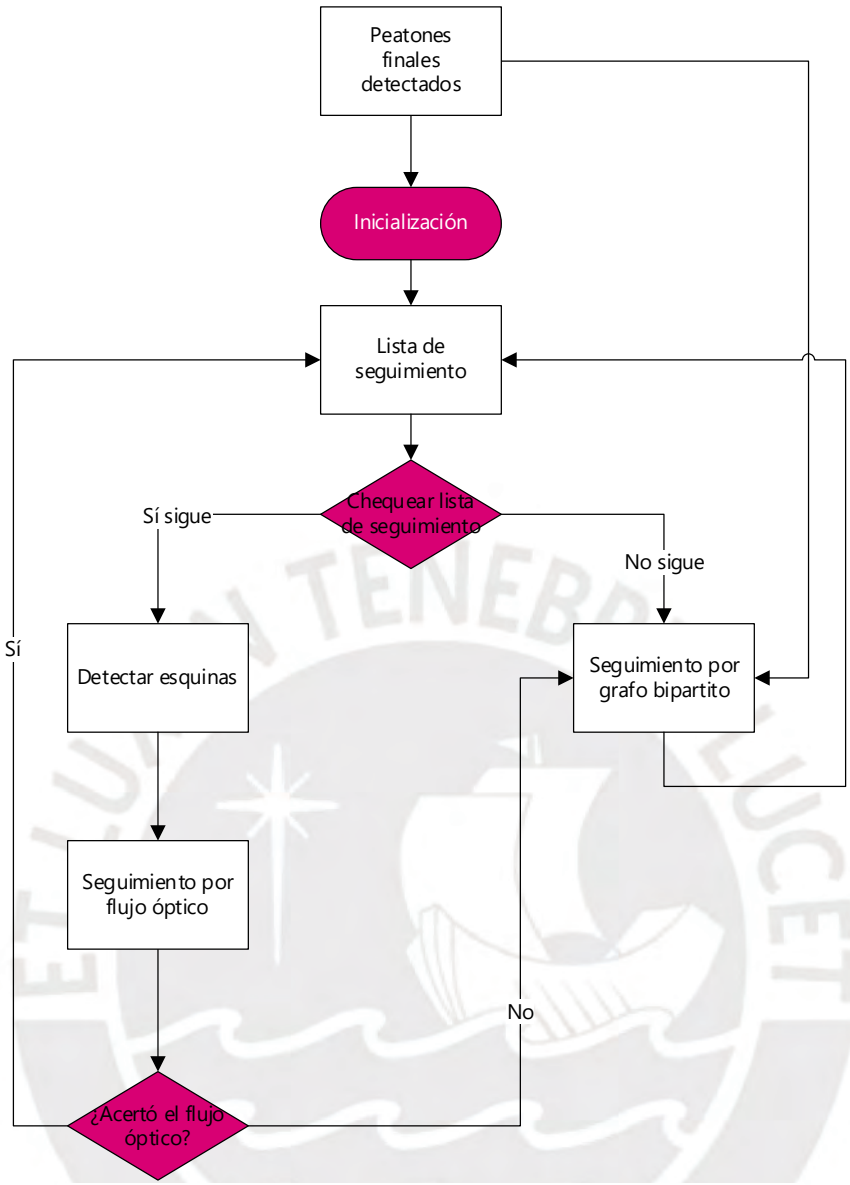



*Figura 13*: Diagrama de flujo de la tercera técnica. Fuente: IEEE [17]


*Figura 14*: Camino trazado usando la cuarta técnica de visión por computadora. Fuente: IEEE [17]

Los resultados de pruebas con este método se muestran a través de dos indicadores en la Tabla 6.

*Tabla 6*: resultados de tercera técnica [17]

| INTERSECCIÓN | RFN (%) | RFP (%) |
|---|---|---|
| INTERSECCIÓN 1 | 62 | 23.64 |
| INTERSECCIÓN 2 | 45 | 38.3 |

Donde:

RFN: Ratio de falsos negativos

RFP: Ratio de falsos positivos

Los valores obtenidos con este método no son muy cercanos al ideal (0% para ambos indicadores).

1.2.3.4. Monitoreo de muchedumbre que preserva la privacidad de Chan et. al.

Se presenta un sistema que preserva la privacidad, es decir se puede implementar con un hardware que no produce registro visual de la gente en escena, para estimar el tamaño de poblaciones no homogéneas compuestas de peatones que se mueven en diferentes direcciones sin usar una explícita segmentación de objetos o seguimiento. [18]

El video es segmentado en regiones de gente moviéndose en diferentes direcciones usando una combinación de texturas dinámicas. Para cada segmento de gente, varias características son extraídas mientras se aplica un mapa de perspectivas para ponderar cada locación de imagen de acuerdo a su tamaño aproximado en escena real. Finalmente, el número de personas por segmento es estimado con la regresión Gaussiana. [18] El diagrama de procesos que se usa en esta técnica se muestra a continuación en la Figura 15:

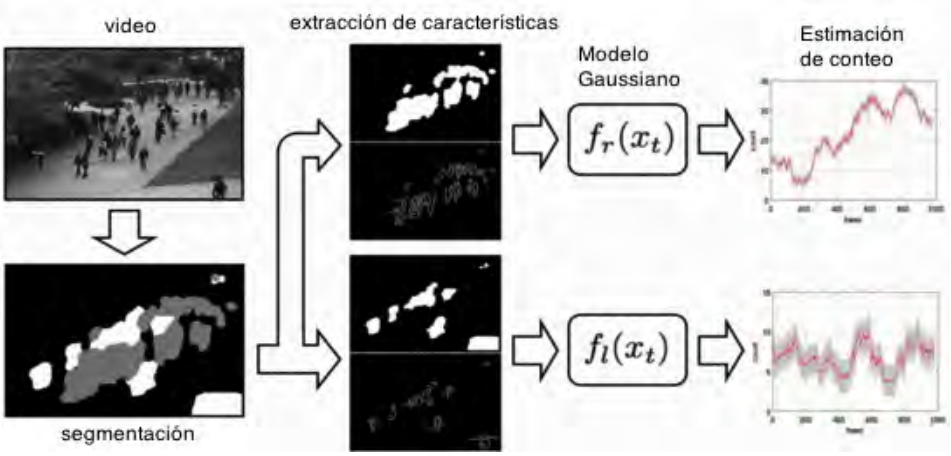



*Figura 15*: diagrama de procesos del quinto método. Fuente: IEEE [18]

La robustez de esta técnica será probada al presentar los resultados de una hora de video con una cámara estacionaria que observa a un camino peatonal. [18]

Los resultados se muestran en la Figura 16, la línea azul indica la cantidad calculada por el algoritmo y la línea gris la real. En el eje horizontal se muestra el número de imagen y en el vertical la cuenta de peatones.[18]

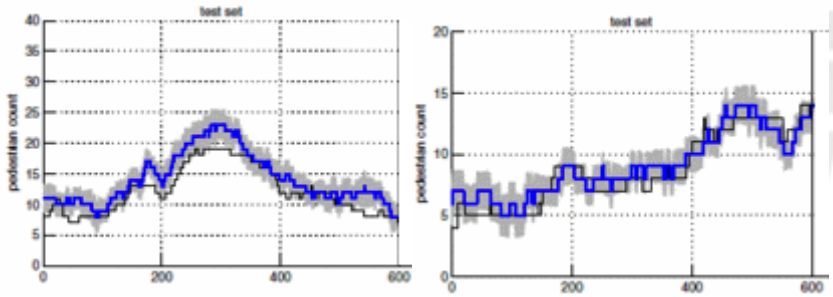



*Figura 16*: Resultados del quinto método. Izquierda: resultados con personas alejándose de la cámara. Derecha: resultados con personas acercándose a la cámara. Fuente: IEEE [18]

### 1.3. Justificación

El conteo de peatones en intersecciones resulta de vital importancia para su uso estadístico que luego proceda a la toma de decisiones en el rediseño de la intersección con prioridad a los peatones. De acuerdo a este número, se puede definir si es que se requiere un cruce en diagonal o mayor semaforización o la construcción de un puente peatonal para uso en cruces con gran afluencia vehicular.

Además, se requiere que este sistema de conteo sea automático y no dependa de mano de obra humana o hardware adicional, para que su uso sea difundido en otras intersecciones, por lo que

se utilizará alguno de los métodos de visión por computadora para agilizar el conteo y disminuir los errores implicados. La elección del método a usar para el conteo depende del hardware que se use y del costo computacional del método. De acuerdo con las técnicas estudiadas, destaca la última ya que no requiere seguimiento a los peatones y es implementada en un ambiente no-homogéneo (parque peatonal) y esto disminuye considerablemente el costo computacional. Adicionalmente, el hardware a emplear en esta técnica es tan solo una cámara estacionaria que se puede instalar fácilmente en cualquier intersección. Se comparará esta cuarta técnica con un modelo de extracción de fondo basado en filtro mediana para verificar los resultados. Estos dos algoritmos se explicarán a fondo y se compararán con otros de extracción de fondo en el capítulo 2.

## 1.4. Objetivos

### 1.4.1. Objetivo principal

Implementar un algoritmo de procesamiento de imágenes para detección y seguimiento de peatones que cruzan por una intersección de avenidas considerando imágenes extraídas de un video grabadas desde una vista superior

### 1.4.2. Objetivos específicos

- Estudiar los algoritmos de procesamiento de imágenes para el conteo y seguimiento de peatones que se mueven
- Crear una base de datos con imágenes extraídas de videos que contienen peatones desplazándose
- Implementar un algoritmo que detecte y siga a peatones que transitan por la intersección de avenidas seleccionada
- Generar un reporte de resultados comparando conteos automáticos con conteos manuales

## Capítulo 2: FUNDAMENTOS TEÓRICOS

### 2.1. Resolución espacial y temporal de imágenes (Tasas de muestreo y dimensiones de imágenes apropiadas para la aplicación)

Dado que el promedio de velocidad de pasos de los peatones es de aproximadamente 1.053 m/s [1] y las imágenes fueron grabadas con una resolución temporal de 30 cuadros por segundo (en adelante cps), se requerirá reducir esta resolución debido a que los pasos de las personas se hacen notar a partir de la imagen número 15 de una segmento de video, por lo que los anteriores cuadros no serían de utilidad para el análisis; por lo tanto, estos serán quitados del análisis y la tasa de muestreo será reducida a solo 5 cps para la aplicación deseada ya que, a través de ensayos, se verificó que los pasos se distinguen bien con esta resolución.

Por otro lado, se usará un tamaño de ventana de 227x785 por ser el tamaño del cuadro necesario para la grabación. La detección a varias escalas es lograda al redimensionar la imagen y recalcular las correspondientes representaciones de características [19].

### 2.2. Análisis de iluminación y contraste

Debido a que el algoritmo usado en la aplicación será probado en la calle durante el día, la iluminación del ambiente es un factor importante al alterar la intensidad de los píxeles y afectar la identificación de peatones.

Medidas estadísticas son ampliamente usadas para reducir los efectos de variaciones de valor de píxeles causados por condiciones de luz. Ejemplos de dichas medidas incluyen la mínima, máxima y más grande diferencia absoluta entre cuadros por cada píxel encontrado durante el periodo de entrenamiento, además de la media o mediana de cada píxel por una distribución similar a una distribución Gaussiana y la varianza de un píxel modelada por una combinación de Gaussianos usando modelos no-paramétricos. Es necesario refrescar periódicamente los píxeles del fondo determinados por estos métodos para reflejar los cambios inducidos por condiciones de luz [19].

En el presente trabajo, para evitar usar las medidas estadísticas anteriormente mencionadas y reducir con ello el costo computacional, se filmarán los videos en horas con poca luminosidad y que no se generen sombras.

### 2.3. Estimación y extracción de fondo (*background extraction*)

El siguiente paso para la extracción de información acerca de peatones en un video es la extracción de fondo.

Detectar objetos moviéndose en una secuencia es usar una simple diferenciación de cuadros entre un modelo de fondo y el cuadro actual, el cual puede dar resultados satisfactorios en la mayoría de las aplicaciones. El principal reto de la extracción de fondo es estimar un fondo robusto de la escena con el fin de lidiar con cambios de iluminación y objetos estáticos [20].

En general, la extracción de fondo se divide en 5 categorías: métodos básicos, métodos estadísticos, métodos borrosos, métodos con redes neuronales y métodos no-paramétricos.[21]

Entre los métodos básicos podemos resaltar los filtros media y mediana. [22] Para explicar estos modelos, utilizaremos las siguientes imágenes de Birgi Tamersoy [22].

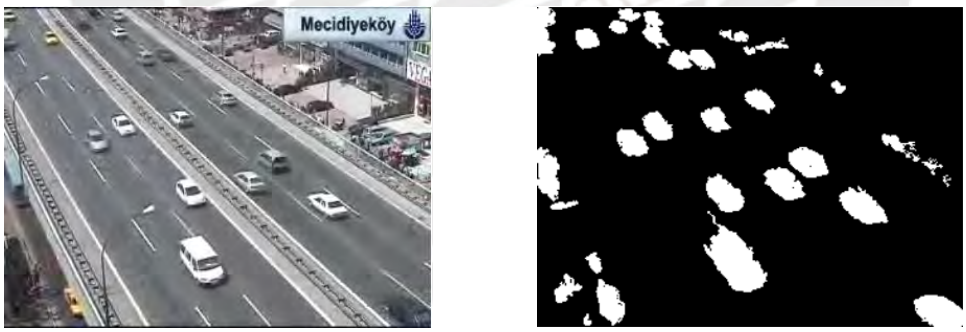



*Figura 17*: A la izquierda, imagen original a que se quitará el fondo; a la derecha, máscara de frente que se obtiene luego de la extracción de fondo. Fuente: Birgi Tamersoy [22]

En ambos modelos, el fondo se aproxima utilizando la media o mediana de las n imágenes previas. A medida que n crece, la aproximación del fondo suele ser más acertada. Para calcular la máscara de frente, que es la imagen en blanco y negro con el fondo ya descartado, se resta la imagen actual con el fondo modelado, ya sea utilizando cualquiera de las dos métricas. Después esta diferencia se compara con un valor umbral. [22]

Para el caso del filtro mediana, las fórmulas y los resultados se muestran a continuación. Se asume que es más probable que aparezca el fondo en una escena, por lo que se puede usar la mediana de los n anteriores cuadros como modelo de fondo:

$$B(x,y,t) = mediana\{I(x,y,t-i)\}$$ Fórmula 4

$$|I(x,y,t) - mediana\{I(x,y,t-i)\}| > Th, \qquad i = \{0,\dots,n-1\}.$$ Fórmula 5

Donde *B(x,y,t)*: Fondo 2-D en el instante t

*I(x,y,t)*: Imagen 2-D en el instante t

*n*: cantidad de imágenes previas

*Th*: Umbral escogido

Se mostrará en la Figura 18 los resultados utilizando este método básico para cada valor de n.

| Para n=10 | |
|---|---|
| Fondo estimado | Máscara de frente |
|  |  |
| Para n=20 | |
| Fondo estimado | Máscara de frente |
|  |  |

| Para n=50 | |
|---|---|
| Fondo estimado | Máscara de frente |
|  |  |

*Figura 18*: Resultados para cada valor de *n* del filtro mediana [22]

Las ventajas de este algoritmo son varias, entre ellas se encuentra su extrema facilidad de implementación y uso, su rapidez de computación, y que los modelos de fondo son dinámicos. Entre las desventajas se enumeran, en primer lugar, la exactitud en la diferenciación de cuadros depende de la velocidad del objeto y el ratio de cuadros (cps); en segundo lugar, el algoritmo de mediana tiene relativamente altos requisitos de memoria; además, el umbral *Th* es uno solo para todos los píxeles de la imagen y no es función del instante de tiempo *t*. [22]

El segundo método a estudiar será el algoritmo de Mezclas de Modelos Gaussianas (GMM). [21] En una Mezcla de Modelos Gaussianos, cada píxel es dividido por su intensidad en el espacio RGB. Cada píxel es computado para hallar la probabilidad de que se trate de un píxel de fondo o de primer plano con la siguiente fórmula: [21]

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta(X_t, \mu_{i,t}, \sum_{i,t}) \quad \text{Fórmula 6}$$

donde:

- $X_t$: píxel actual en el cuadro de tiempo t
- K : número de distribuciones (gaussianos) de la mezcla
- $\omega_{i,t}$: el peso del k-ésimo gaussiano en el cuadro t
- $\mu_{i,t}$: la media del k-ésimo gaussiano en el cuadro t
- $\sum_{i,t}$ : la desviación estándar del k-ésimo gaussiano en el cuadro t

Por otro lado, la función de densidad de probabilidad (fdp) que se halla como término a la derecha de la ecuación 2.3.1 se define de la siguiente manera: [21]

$$\eta(X_t, \mu, \sum\ ) = \frac{1}{(2\pi)^{1\div 2}|\sum\ |^{1\div 2}} exp(-\frac{1}{2}(X_t - \mu)\sum^{-1}(X_t - \mu)) \quad \text{Fórmula 7}$$

En el método de Stauffer y Grimson , cada capa RGB no está correlacionada con otras capas, por lo que se asume que la diferencia en intensidad posee desviación estándar uniforme. La matriz de covarianza asume la siguiente forma: [21]

$$\sum_{i,t}\ = \sigma_{i,t}^2 I \quad \text{Fórmula 8}$$

Para cada gaussiano de mayor valor que un designado umbral *T*, el píxel se clasifica como fondo; si no, se le asigna como primer plano. [21]

$$B = argmin_b(\sum_{i=1}^{b} \omega_{i,t} > T) \quad \text{Fórmula 9}$$

Si un píxel se iguala con uno de los K Gaussianos, entonces los valores de $\omega$,$\mu$ y $\sigma$ son actualizados. [21]

$$\omega_{i,t+1} = (1-\alpha)\omega_{i,t} + \alpha \quad \text{Fórmula 10}$$

$$\mu_{i,t+1} = (1-\rho)\mu_{i,t} + \rho.X_{t+1} \quad \text{Fórmula 11}$$

$$\sigma_{i,t+1}^2 = (1-\rho)\sigma_{i,t}^2 + \rho.(X_{t+1} - \mu_{i,t+1}) \cdot \left(X_{t+1} - \mu_{i,t+1}\right)^T \quad \text{Fórmula 12}$$

donde:

$$\rho = \alpha \times \eta(X_{t+1}, \mu_i, \sum_i\ ) \quad \text{Fórmula 13}$$

Si hay un caso en el que todos los K gaussianos no se igualan con el píxel, entonces solo el parámetro $\omega$ es actualizado: [21]

$$\omega_{j,t+1} = (1-\alpha)\omega_j \quad \text{Fórmula 14}$$

Por lo tanto, si es que se encuentran todos los parámetros requeridos ($\omega, \mu, \sigma$ y $\alpha$), entonces se puede ejecutar la extracción de fondo. [21]

Un tercer algoritmo basado en libros código (*codebooks,* conjunto de *codewords* o palabras código) recomendado por Kim et al. [23] codifica el fondo en una base píxel por píxel. [23] El

algoritmo propuesto adopta un fin de cuantización y agrupamiento. Las muestras de cada píxel son clasificadas en un conjunto de palabras código. Cada píxel tiene un diferente tamaño de la palabra código basado en su variación en el tiempo. Cada entrada de un libro código consiste en dos vectores que representan los valores de color RGB junto con brillo. Durante el entrenamiento, cada muestra es expresada en términos de medidas de brillo y distorsión de color. Los vectores formulados son comparados con las actuales entradas de libro para determinar una actualización de los mismos vectores. Si la diferencia entre la entrada en el libro y la muestra actual está dentro de un cierto límite, la entrada será elegida como la codificación aproximada. No se ejecutan exhaustivas búsquedas para encontrar la entrada con la menor diferencia. [23]

En vez de ello, la primera palabra que satisfaga la condición de umbral es escogida como la aproximación. Una nueva palabra será creada en ausencia de una aproximación en el libro. El mismo procedimiento es continuado para actualizar periódicamente el libro. Después del entrenamiento, los vectores de color y brillo de un píxel son comparados con las entradas del libro. Si es que no se encuentra una similitud, el píxel es considerado como píxel de primer plano. En caso contrario, se determinará que el píxel pertenece al fondo [23]. Este algoritmo, además, es robusto a cambios de luminosidad externos y también diferencia a píxeles de fondo que se mueven con píxeles de primer plano estáticos o móviles [23]

La Figura 19 ilustra el método de Kim et.al.:

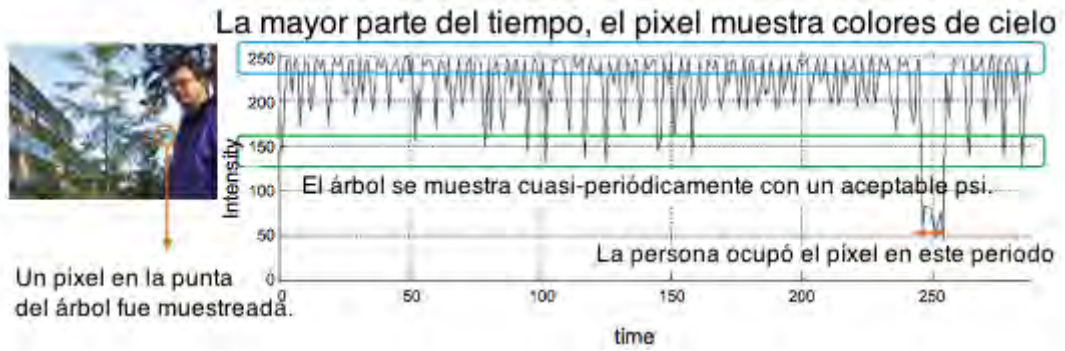



*Figura 19*: Método de Kim et. al. usando el parámetro y la intensidad del píxel a lo largo del tiempo para clasificarlo como píxel de fondo. [23]

Otro método que combina costo computacional y performance fue propuesto por McFarlane et. al. que usa el algoritmo de mediana aproximada para determinar el fondo. Este método considera la mediana 54 de N cuadros almacenados como el fondo. Este fondo calculado es sustraído de los subsecuentes cuadros para extraer objetos en primer plano. El fondo es determinado iterativamente al convertir los cuadros almacenados a imágenes en escala de grises y al encontrar la mediana de los valores de los píxeles en cada locación. [19]

El fondo es inicializado como el primer cuadro y los siguientes cuadros son procesados secuencialmente. Si un píxel en el actual cuadro tiene un valor más alto que su correspondiente píxel en el cuadro de fondo, el valor del píxel de fondo se incrementa en uno. Si el valor del píxel actual es menor que el valor al valor del píxel de fondo, el de fondo es decrementado en uno. Eventualmente, los píxeles convergen la mediana donde la mitad de los píxeles de entrada son mayores que el fondo y la mitad de los valores son menores que el fondo si es que no hay movimiento en la escena. El tiempo requerido para convertir el valor mediano depende del número de cuadros almacenados, la frecuencia de cuadros y la cantidad de movimiento en la escena. [19] El proceso de determinación se resume en la Figura 20:

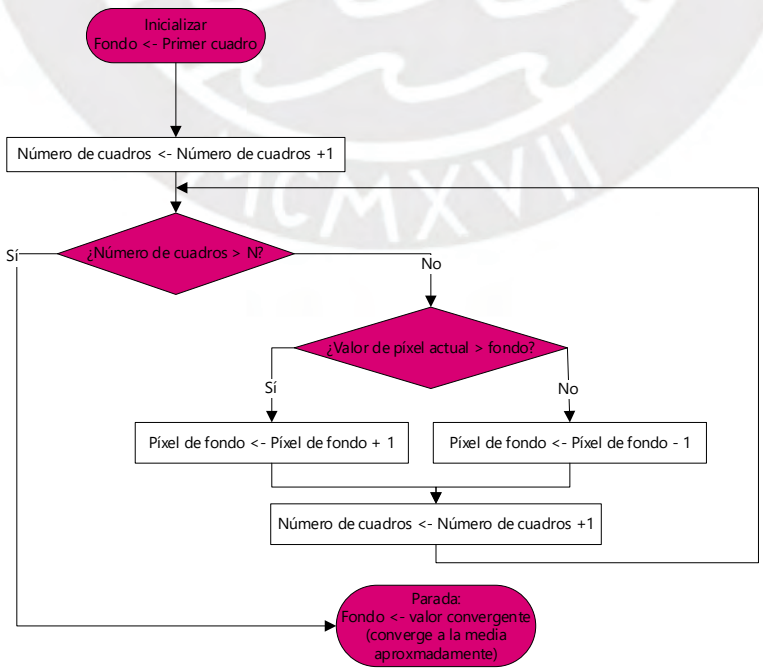



*Figura 20*: Diagrama de flujo del algoritmo presentado por McFarlane et. al. [19]

El número de cuadros almacenados requeridos para determinar el fondo depende de la actividad en la región capturada y el ratio de cuadros. El fondo puede ser periódicamente actualizado para identificar adecuadamente nuevos objetos en primer plano a través de la sustracción. [19] Por otra parte, Bin Li et al. [15] obtiene regiones de objetos moviéndose a través del algoritmo *ViBe* desarrollado y patentado por Olivier Barnich y Marc Van Droogenbroeck. [24]

*ViBe* se desvía de la idea de modelado a través de funciones de densidad probabilística que priorizan las últimas observaciones y, en vez de eso, mantiene una colección de muestras para cada píxel que actualiza continuamente con cierta probabilidad. Luego clasifica a un píxel como parte del fondo siempre que sea similar a solo unas pocas de esas muestras. Los autores argumentan que es más confiable estimar la distribución estadística de un píxel del fondo con un pequeño número de valores de color cercanos al original que con un gran número de muestras dispersas, las cuales son más proclives a ser influenciadas por valores atípicos. El efecto resulta similar a ignorar los extremos de una función gaussiana o considerar sólo su sección central aplicando un umbral. [25]

Formalmente se define $(x)$ como el valor en un espacio de colores euclídeo correspondiente al píxel $x$, y al modelo para ese píxel como

$$M(x) = \{v_1, v_2, v_3\} \qquad \text{Fórmula 15}$$

Donde $vi$ es el valor de la muestra del fondo con índice $i$, y cada píxel del fondo es modelado con $N$ muestras.

$$x\ es\ parte\ del\ fondo \ \leftrightarrow \#\min \leq \#\{S_R(V(x))\} \cap \{v_1, v_2, \ldots\ldots, v_n\} \qquad \text{Fórmula 16}$$

Para clasificar el píxel $x$ de acuerdo a su modelo correspondiente $(x)$, $v(x)$ es comparado con las muestras definiendo una esfera $SR$ $(v(x))$ de radio $R$ y con centro en $v(x)$. El píxel $x$ es clasificado como fondo si la cardinalidad (operación denotada con #) del conjunto intersección entre la esfera con la colección de muestras $(x)$ es mayor o igual a un umbral $\#min$. [25]

El radio $R$ representa una distancia euclídea dentro del mismo espacio de colores, aunque se argumenta que para el caso de RGB el canal azul puede descartarse sin disminuciones significativas en la precisión del algoritmo, acelerando el procesamiento.[25]

En la Figura 21, se muestra una ilustración de los elementos del algoritmo *ViBe* para determinar un píxel de fondo.

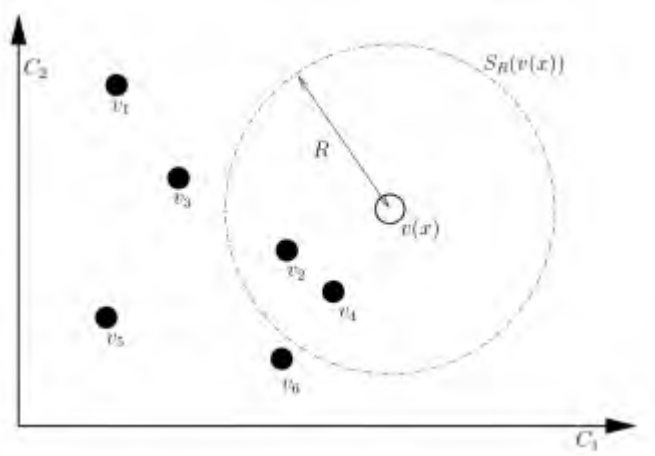



Figura 21: Ilustración de método *ViBe* [25]

Los autores exponen en la misma publicación citada que han determinado experimentalmente que, al menos para imágenes monocromáticas, los parámetros óptimos son $R = 20$ y $\#min = 2$. La selección de $N$ es dejada libre en principio para poder determinar la sensibilidad del algoritmo. Finalmente, por un balance entre el costo, la sensibilidad y la eficiencia, seleccionan $N = 20$, debido a que el porcentaje de clasificación correcta de píxeles sube con la cantidad de muestras, pero se aplana alrededor de las 20, y se requieren exponencialmente más muestras para lograr mejoras apreciables. En general, recomiendan mantener la proporción $\#min / N = 1 / 10$. [25]

### 2.4. Reducción de ruido

Bin Li et al.[15], luego de la sustracción de fondo, las regiones obtenidas contienen mucho ruido, varios hoyos y algunas manchas que pueden no ser parte de objetos de interés moviéndose. Para remover estas manchas, utiliza una fórmula de umbralización para descartar si es que alguna mancha debe ser mantenida o no de acuerdo con el siguiente criterio: [15]

$$B(i,j) = \begin{cases} 0\ ,\ Area(B) < T_{area} \\ 1\ ,\ Area(B) \geq T_{area} \end{cases}$$ Fórmula 17

donde $Area(B)$ es el área de contorno de una mancha y $T_{area}$ es un área umbral determinada por conocimiento previo del tamaño de la cabeza de la persona. [15]. El valor de $T_{area}$ será establecido como 100 y este es adecuado para la cámara que está instalada a 5-6 metros por encima del suelo para la aplicación requerida.

Aún más, se usan operaciones morfológicas que incluyen erosión y dilatación de imagen para eliminar y llenar los hoyos. Bin Li et. al. una operación de cerradura con *template* 7x3 para procesar las regiones y luego una operación de apertura con *template* 7x3 en las mismas. Estas regiones optimizadas después de las operaciones arriba mostradas son mostradas en la Figura 22.[15]

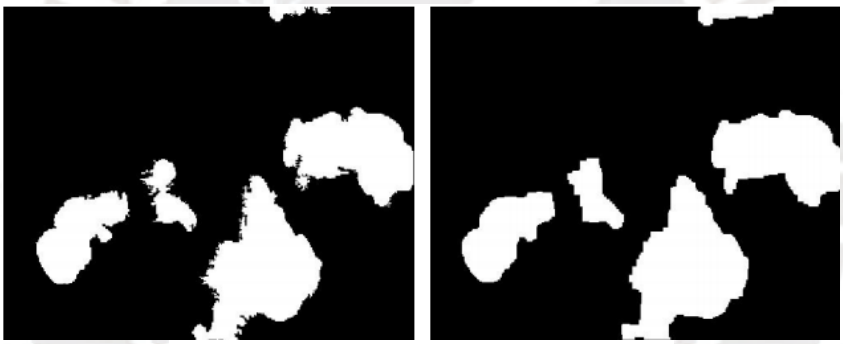


Figura 22: resultado de reducción de ruido [15]

## 2.5. Seguimiento de personas

Según Damien Lefloch et. al.[20], el seguimiento de un objeto en una secuencia de video consiste en encontrar el mismo objeto en diferentes cuadros. Se utilizan, para ello, las diferentes características previamente extraídas en el módulo anterior. Este seguimiento utiliza el filtro Kalman propuesto por Wan para predecir el futuro estado de cada objeto en el siguiente cuadro. El filtro de Kalman modela el movimiento de peatones como una ecuación de la posición actual, la posición previa y el tiempo como parámetros.

El método estima el estado $x$ perteneciente a $\mathbb{R}^n$ de un proceso controlado en tiempo discreto que es gobernado por la ecuación diferencial del tipo [26]

$$x_{k+1} = A_k \,.\, x_k + B \,.\, u_k + w_k$$ Fórmula 18

con una medida $z$ correspondiente a la observación y perteneciente a $\mathbb{R}^m$ que es la siguiente: [26]

$$z_k = H_k x_k + v_k$$ Fórmula 19

Las variables aleatorias $w_k$ y $v_k$ representan el ruido del proceso y de la medida respectivamente y se asume que son independientes y blancos. [26]

Cuando el filtro de Kalman se aplica a la Visión Artificial, el estado $x$ se corresponde con el vector posición del objeto en la imagen determinado por las coordenadas de posición $x_x$ y $x_y$ , y las coordenadas de velocidad $v_x$ y $v_y$ . La observación $z$ en cambio, es únicamente un vector de dos componentes $z_x$ y $z_y$, correspondiente a las coordenadas de la posición observada del objeto de interés. [26]

La matriz $A_{NxM}$ relaciona el estado en tiempo $k$ con el estado en tiempo $k+1$. Esta relación se manifiesta en las siguientes ecuaciones, dando como resultado la matriz $A$.

$$\begin{cases} X_{x_{k+1}} = X_{x_k} + V_x . t \\ X_{y_{k+1}} = X_{y_k} + V_x . t \end{cases} \qquad A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{cases} V_{x_{k+1}} = V_{x_k} \\ V_{y_{k+1}} = V_{y_k} \end{cases}$$ Fórmula 20

La matriz $B_{Nx1}$ relaciona la entrada control $u$ perteneciente a $\mathbb{R}^1$ con el estado $x$. Y la matriz $H_{NxM}$ relaciona el estado con la medida $z_k$ [26].

$$H_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

El filtro de Kalman proporciona una ecuación que computa un estimador del estado a posteriori $\hat{x}_k$ como combinación lineal del estimador a priori $\hat{x}_k^-$ y la diferencia ponderada entre la observación actual $z_k$ y una predicción de medida $H_k \,.\, \hat{x}_k^{\,-}$ [26]

$$\hat{x}_k = \hat{x}_k^{\,-} + K \,.\, (z_k - H_k \,.\, \hat{x}_k^{\,-})$$ Fórmula 21

La matriz $K_{NxM}$ llamada ganancia de Kalman o factor de mezcla establece la cantidad de influencia del error entre nuestra estimación y la medida [26]

$$K_k = P_k^- \,.\, H_k \,.(H_k \,.\, P_k^- \,.\, H_k{}^T)^{-1}$$ Fórmula 22

Siendo $P_k^-$ el estimador de la covarianza del error a priori y $R_k$ la covarianza del error medido. Vemos que si $R_k$ se aproxima a 0, la ganancia ponderará el residuo con mayor peso. Por el contrario, cuando $P_k^-$se aproxime a 0, la ganancia ponderará menos el residuo. [26] El filtro de Kalman estima variables de estado de un proceso con realimentación. Calcula el estado del proceso en algún instante y entonces obtiene información (se realimenta) de la medida. Por tanto, las ecuaciones del filtro se pueden clasificar en dos tipos: actualización del tiempo y actualización de las medidas. Las primeras son responsables de proyectar hacia el futuro los estimadores del estado actual y de la covarianza del error, para obtener los estimadores a priori del siguiente estado. Las ecuaciones de actualización de las medidas son responsables de la realimentación, incorporando una nueva medida a los estimadores a priori para obtener unos estimadores a posteriori mejorados. Las ecuaciones de actualización del tiempo pueden ser interpretadas como ecuaciones de predicción, mientras que las de actualización de la medida pueden pensarse como ecuaciones de corrección. La descripción del filtro de Kalman con sus ecuaciones puede verse en el siguiente diagrama [26].

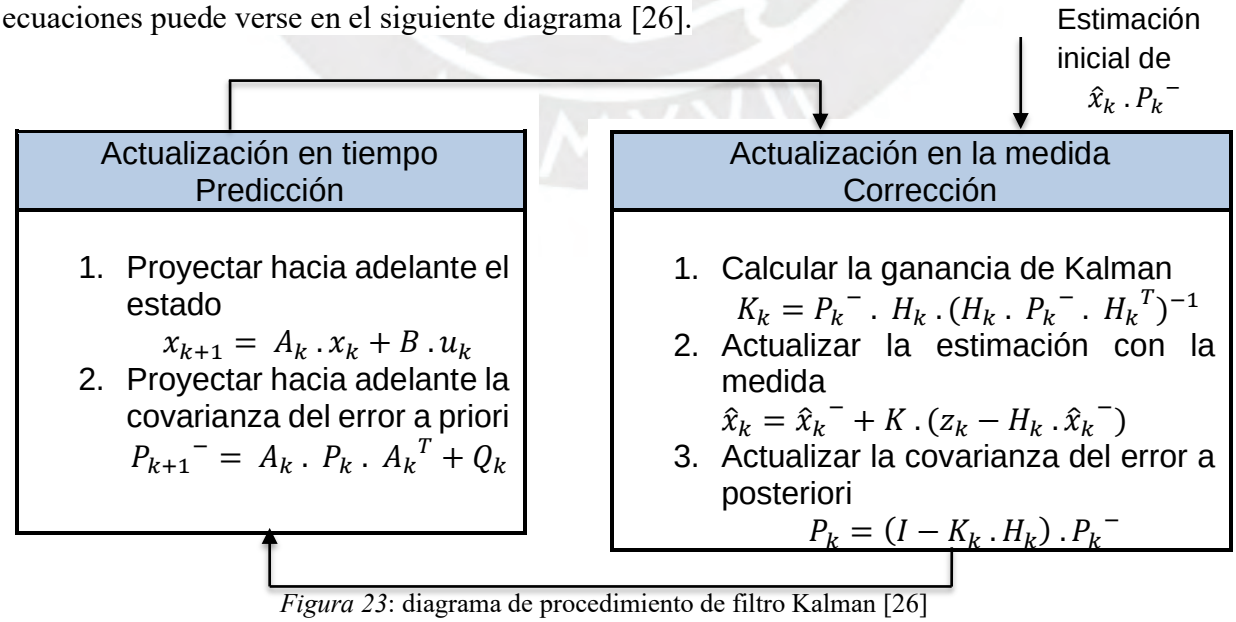



*Figura 23*: diagrama de procedimiento de filtro Kalman [26]

Para aplicar el filtro de Kalman al seguimiento de un objeto, es necesario proporcionar una característica representativa del mismo, que será tomada como la observación del objeto. [26] En nuestro caso, el objeto a seguir será la persona por lo que se plantea seguir a través del centroide de un rectángulo dibujado alrededor de la persona detectada.

Una muestra del seguimiento de filtro Kalman se muestra en la Figura 24. Por otro lado, se cuenta con los resultados de la predicción de Kalman y las posiciones reales respectivas en la Figura 25.

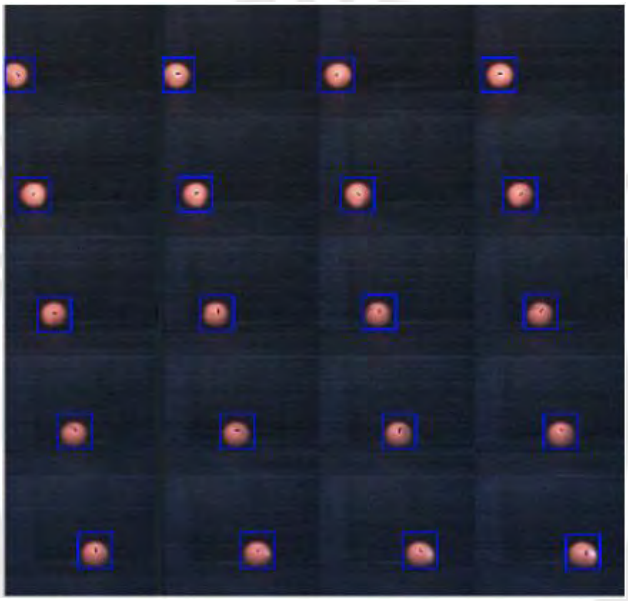


*Figura 24*: Ilustración de seguimiento con filtro Kalman con valores $P_k^- = 100, R_k = 0.02$ *y* $Q = 10^{-3}$ [26]

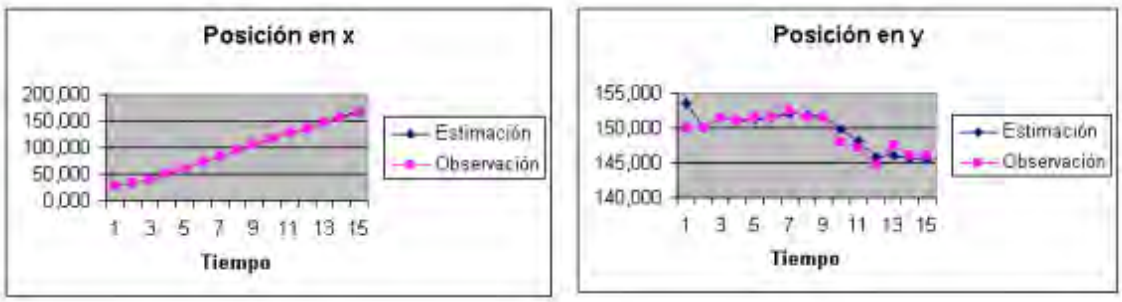



*Figura 25*: Gráficos de las posiciones estimadas y observadas mediante el filtro Kalman con valores $P_k^- = 100$ *y* $R_k = 0.02$ [26]

Gayatri D. Prabhu [19], por otro lado, plantea un seguimiento que envuelve la comparación de ciertas características geométricas o visuales de un peatón detectadas en cada cuadro. Si se dibuja un rectángulo en los límites de la persona, se pueden encontrar características

geométricas como el área del rectángulo, la simetría vertical, la distancia entre centroides del rectángulo y la distancia entre píxeles y su densidad. Las características visuales pueden ser tan simples como el valor en escala de grises promedio o el valor promedio de color de la región. Además, la variabilidad de color debido a movimientos del objeto en seguimiento puede observarse al calcular la métrica de suma de cuadrados de diferencias para las intensidades de color de diferentes píxeles en cuadros consecutivos.[19]

Gayatri et al. menciona que el autor Gonzales et al. recomienda usar una medida de señales geométricas y visuales para propósitos de seguimiento. Los pesos son asignados de acuerdo a la distancia del peatón a la cámara. Por otro lado, Viola et. al. propuso un sistema que utiliza la información de las poses al caminar extraída a través de una estructura en cascada de características Haar y mapas integrales para seguimiento. [19]

Seguir a un objeto también puede ser descrito como el problema de encontrar la posición de un píxel p en el (n+1)-ésimo cuadro dada la posición del mismo píxel en n-ésimo cuadro. Es en este contexto que el parámetro flujo óptico es popular.

El flujo óptico es definido como el vector velocidad de cada píxel en la imagen, estimado al evaluar el gradiente de intensidad generado por el aparente movimiento del píxel. Se debe notar que el flujo óptico se refiere al movimiento aparente causado por el movimiento relativo entre la cámara y el objeto. El cálculo del flujo óptico supone que los píxeles del objeto tienen los mismos valores de intensidad a lo largo del tiempo.[19]

Por último, las etapas de la implementación serán tocadas a profundidad en el capítulo tres aplicando los fundamentos teóricos del presente capítulo dos.

## Capítulo 3: DISEÑO DE LA PROPUESTA

La metodología a seguir para el desarrollo de la aplicación propuesta se muestra en la Figura 26. Para seleccionar los métodos óptimos que se aplicarán en la tercera y cuarta etapa, se utilizará el método más simple y eficiente. Para el caso de la extracción de fondo inicial, se utilizará el filtro mediana y se comparará con el método GMM. Por otro lado, para la reducción de ruido, se utilizará un filtro simple. Finalmente, se usará un Filtro Kalman para el seguimiento y actualización de las siluetas. En el siguiente gráfico, se muestra el procedimiento seguido.

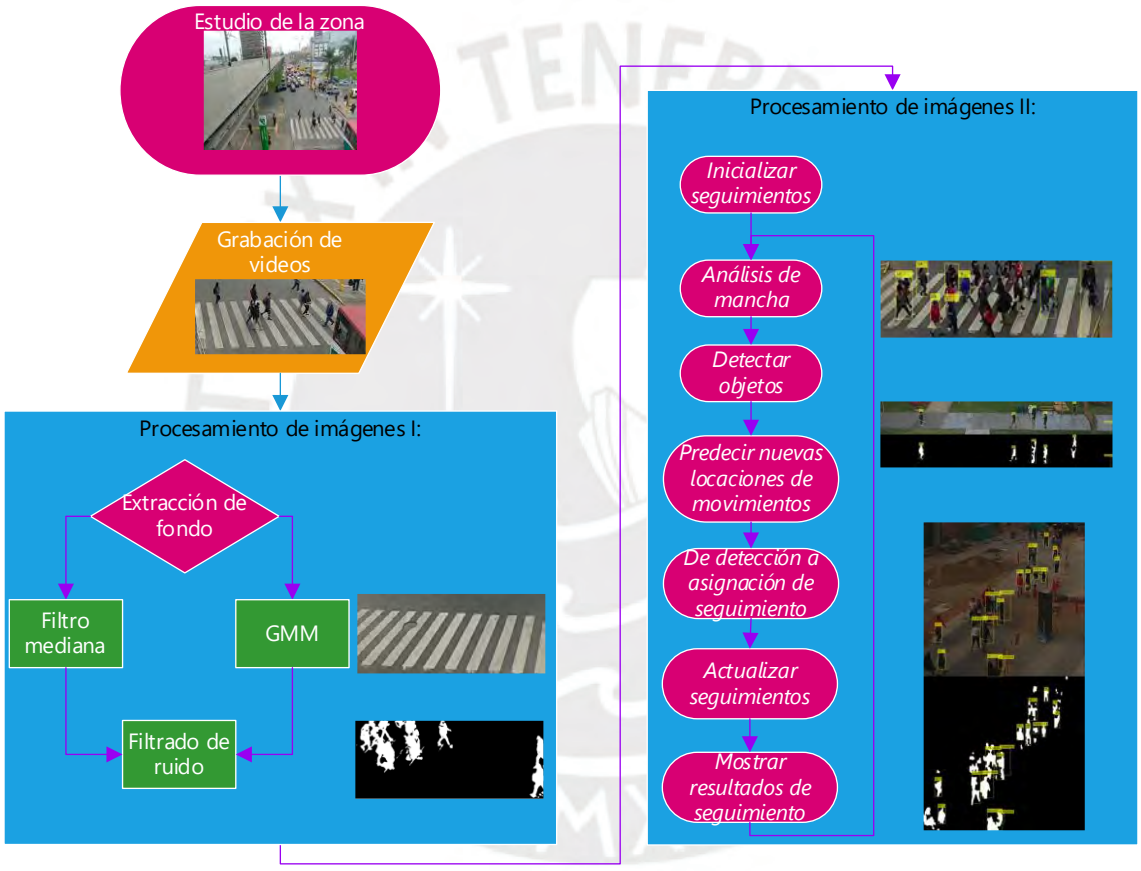



*Figura 26*: metodología para el desarrollo de la tesis

### 3.1. Estudio de la zona

El primer paso será el estudio de la zona y sus cambios debido a horas punta y variaciones de clima. Por un lado, a horas de mayor afluencia de gente y autos se podría disminuir la eficacia del algoritmo, por lo que la grabación se realizará a horas de menor afluencia, tal como entre las 10 y 14 horas. Por otro lado, el clima representa una variable importante en el análisis

debido a que las sombras cambian las condiciones de iluminación y, por consiguiente, afectan la eficacia del algoritmo.

### 3.2. Grabación de videos

El hardware a emplear es tan solo una cámara estacionaria o de teléfono celular que se puede instalar fácilmente en cualquier intersección, debido a la fácil portabilidad de este equipo. El siguiente paso será la grabación de videos respectiva que contará con segmentos de video de aproximadamente un minuto de duración cada uno. La aplicación tomará como dato de entrada el video que corresponde a la grabación de la calle tomada desde vista superior con ángulo 30º con respecto al eje horizontal durante un día nublado para no depender de condiciones de luminosidad y mediante un trípode con el fin de que no haya movimientos relativos para simular correctamente la cámara estacionaria, debido a que la presente grabación se hizo con un smartphone de 16 megapíxeles, 30 cuadros por segundo (cps), a 1080p puesto sobre un trípode a 1 metro del suelo durante 8 minutos aproximadamente, por lo que el total de imágenes generadas fue de 2251 . Además, se procederá a cortar cada cuadro analizado con tal de solo dejar la zona donde se encuentran las personas.

El resultado de la grabación se muestra en la Figura 27:

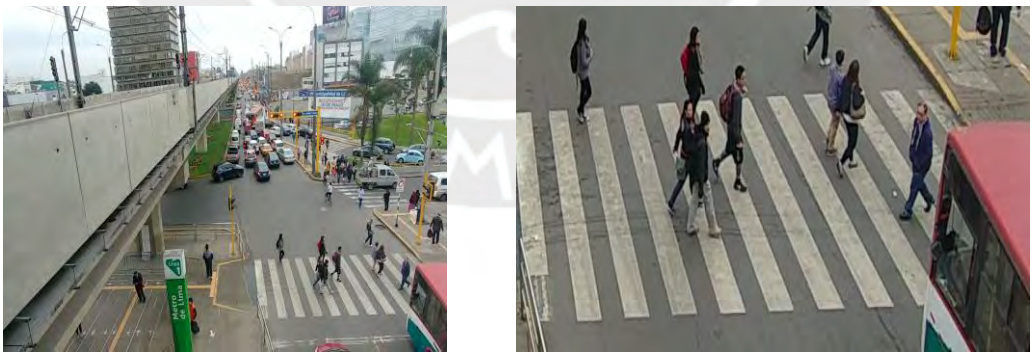


*Figura 27*: Ejemplo de cuadro original del video (izquierda) y área cortada a analizar (derecha). Fuente: propia

La aplicación será implementada en Matlab con el fin de utilizar funciones ya desarrolladas en tal entorno. Estas funciones se describirán en la siguiente subsección.

### 3.3. Procesamiento de imágenes I: Extracción de datos y reducción de ruido

En primer lugar, se extrajo cuadros del video original para ser analizados. El respectivo video se procesará en el formato RGB. Se eliminarán algunos de estos cuadros con tal de disminuir la tasa de cuadros por segundo, lo que resulta en menos cuadros por analizar y menor tiempo de procesamiento. La tasa de cuadros original es de 30 cps y disminuirá a 5 cps por lo que el número de cuadros analizados pasará a aproximadamente la sexta parte del original (de 2251 a 365 cuadros).

En segundo lugar, usando las funciones que Matlab 2015 provee dentro del *Toolbox* de Visión de computadora (*ComputerVision*), se inicializarán estructuras que ayuden al proceso de detección y seguimiento de objetos [27]. Estas estructuras de datos de Matlab (*data structures*) tienen como nombre seguimientos (*tracks*) y permiten obtener datos de los objetos observados en el video como centroides del área seguida, fondos, etc.

Luego de la creación de los seguimientos, se procede a calcular el fondo de la imagen y a segmentar los objetos del plano delantero de los del fondo. Se probarán dos algoritmos diferentes para evaluar su eficacia con los videos de prueba. El primero es el cálculo del fondo a través de un filtro mediana [22] que calcula la mediana de los píxeles para encontrar los píxeles de fondo. En este caso, varía el número de cuadros aleatorios que se utilizan para calcular la mediana respectiva. Se muestra un ejemplo en la Figura 28 del uso de filtro mediana.

*Figura 28*: ejemplo del uso de filtro mediana. Arriba izquierda, derecha y abajo izquierda: cuadros del video. Abajo derecha: mediana de los 3 cuadros. Fuente: propia

Luego del cálculo de la mediana (Subetapa 0), se procede a restar los cuadros del video con el fondo calculado y almacenar su valor en la variable *dif* (Subetapa 1). Después, las tres capas de esta diferencia son elevadas al cuadrado y sumadas para compararlas con un valor umbral (0.04, después de examinar el histograma) para determinar la máscara binaria inicial (Subetapa 2). Esta máscara es filtrada por un filtro simple de tamaño 5*5 para eliminar los píxeles sobrantes del paso de cebra como filtrado adicional (Subetapa 3). Por último, se compara esta máscara binaria inicial filtrada con un segundo umbral (0.4, nuevamente por ser un valor adecuado desde el histograma) para conseguir la máscara binaria final (Subetapa 4).

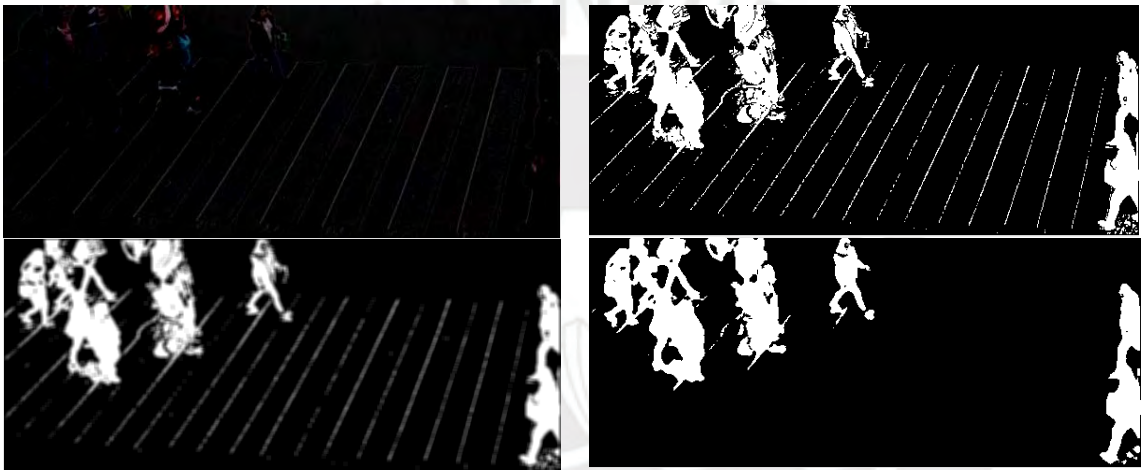


*Figura 29*: Sub-etapas posteriores en el filtro mediana: Arriba izquierda: Subetapa 1. Arriba derecha: Subetapa 2. Abajo izquierda: Subetapa 3. Abajo derecha: Subetapa 4. Fuente: propia

También se empleará el algoritmo *Gaussian Mixture Models* [21] que determina si un píxel es de fondo al analizar si es que durante un tiempo de entrenamiento se ajusta a una campana gaussiana. Sea cual fuere el método, el resultado es una máscara binaria, donde el valor del píxel de 1 corresponde al primer plano y un valor de 0 corresponde al fondo. Los grupos conectados de píxeles de primer plano suelen corresponder a objetos moviéndose. El sistema de análisis de estos grupos de pixeles blancos llamadas manchas (*blobs)* se usa para encontrar dichos grupos y computar sus respectivas características, tales como área, centroide y el rectángulo bordeante. La función "*detectObjects*" se encarga de esta tarea y usando una estructura que guarda el resultado del paso de extracción de fondo y luego realiza operaciones

morfológicas de apertura y cerradura que resultan en una máscara binaria que remueve píxeles con ruido y llena los hoyos en las manchas remanentes. [27] El resultado de dicha etapa se visualiza en la Figura 30:

*Figura 30*: Resultado de la extracción de fondo para diferentes densidades. Arriba: densidad media, Medio: densidad baja, Abajo: densidad alta. Fuente: propia

### 3.4. Procesamiento de imágenes II: Seguimiento y actualización

Después de ello, se utiliza el filtro Kalman para predecir el centroide de cada seguimiento (*track*) en el cuadro actual y actualiza su rectángulo bordeante tal que su centro se desplaza a la locación predicha. [27]

Luego de esta tarea, se asignan las detecciones de objeto del cuadro actual a seguimientos existentes con minimización de costo. El costo se define como la posibilidad de que una detección no corresponda a un seguimiento, el cual es desarrollado en dos etapas: [27]

Primer paso: Se computa el costo de asignar cada detección a cada seguimiento usando el método "*distance*" de "*Vision.Kalman System*". El costo toma en cuenta la distancia entre el centroide predicho y el centroide de la detección. Esto además incluye la confianza en la predicción que se mantiene por el filtro Kalman. Los resultados son guardados en una matriz MxN, donde M es el número de seguimientos y N es el número de detecciones. [27]

Segundo paso: Se resuelve el problema de asignación representado por la matriz de costo usando la función "*assignDetectionsToTracks*". La función toma la matriz costo y el costo de no asignar ninguna detección a un seguimiento. [27]

El valor del costo de no asignar una detección a un seguimiento depende del rango de valores retornados por el método "*distance*" de "*vision.KalmanFilter*". Este valor debe ser concretado experimentalmente. Fijarlo como un valor muy bajo incrementará las posibilidades de crear un nuevo seguimiento, y puede resultar en una fragmentación de seguimientos. Fijarlo con valor muy alto podría resultar en un solo seguimiento correspondiente a una serie de objetos separados moviéndose. Después de varias simulaciones, se determinó que el valor ideal para la aplicación está en el rango de 10 a 20. [27]

La función "*assignDetectionsToTracks*" usa la versión Munkres del algoritmo húngaro para computar una asignación que minimiza el costo total. Retorna una matriz Mx2 que contiene los índices correspondientes a los seguimientos asignados y las detecciones en sus dos columnas. Además, retorna los índices de los seguimientos y detecciones que se mantienen sin asignación. [27]

La función "*updateAssignedTracks*" actualiza cada seguimiento asignado con la detección correspondiente. Esta llama al método "*correct*" de "*vision.KalmanFilter*" para corregir la locación estimada. Luego, almacena el nuevo rectángulo bordeante e incrementa la edad del seguimiento y la cuenta visible total en uno. Finalmente, la función fija el valor de la cuenta invisible a cero. [27]

La función "*updateUnassignedTracks*" marca cada seguimiento no asignado como invisible e incrementa su edad en uno. [27] La siguiente función "*deleteLostTracks*" elimina los seguimientos que han sido invisibles por varios cuadros consecutivos. Además, elimina los seguimientos recientemente creados que han sido invisibles por varios cuadros en general. [27]

Por otro lado, la función "*createNewTracks*" crea seguimientos desde detecciones no asignadas. Se asume que cualquier detección no asignada es el comienzo de un nuevo seguimiento. En la práctica, se puede usar otras señales para eliminar detecciones con ruido, tales como el tamaño, locación o apariencia. [27]

Por último, la función *displayTrackingResults* dibuja un rectángulo bordeante y un ID de capa para cada seguimiento en el cuadro del video y la muestra junto con el cuadro original. Este ejemplo creó un sistema basado en movimiento para detectar y seguir múltiples objetos moviéndose. [27] En la presente tesis, se mejorará y modificará el video utilizando técnicas descritas en el capítulo dos para detectar y seguir a los peatones, ya que con el código inicial y las funciones descritas anteriormente no se logró los resultados de detección y seguimiento deseados, por lo que se optó por modificar los algoritmos de detección de fondo de filtro mediana [22] y GMM [21] y reconocimiento y seguimiento de personas a través de áreas, utilizando diferentes valores umbrales a los ofrecidos por el *toolbox*. El código utilizado se muestra en los anexos.

El resultado de las siguientes etapas se muestra en la Figura 31:

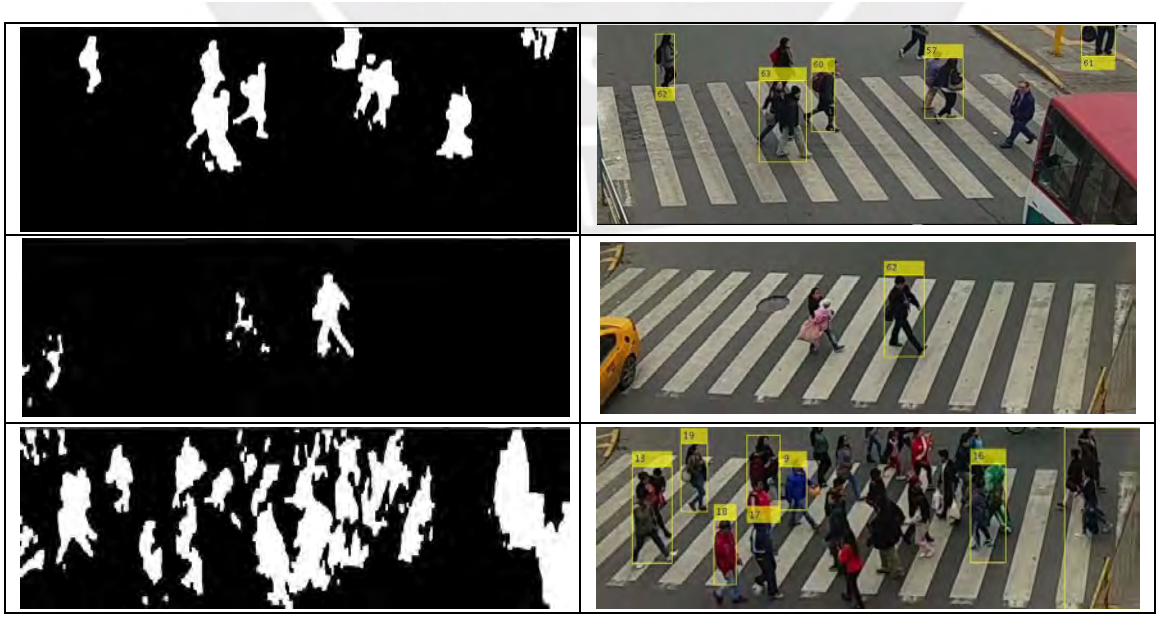


*Figura 31*: Resultado de la cuarta etapa de diseño para diferentes densidades. Fuente: propia

## Capítulo 4: RESULTADOS

Los resultados se obtuvieron utilizando el software Matlab 2015 en una computadora Apple Macbook Pro. El algoritmo fue aplicado a tres videos de aproximadamente un minuto de duración cada uno y un video extra de 5 minutos. Los dos primeros fueron grabados en la estación La Cultura del metro de Lima en dos esquinas con un ángulo de aproximadamente 30º. El tercero fue grabado en la Pontificia Universidad Católica del Perú desde la facultad de Ciencias Sociales y, posteriormente, el último fue grabado en el Centro Comercial Plaza San Miguel. En el presente capítulo, se mostrarán los resultados obtenidos en cada uno de los videos a través de las diversas etapas del mismo.

### 4.1. Resultados de Procesamiento de imágenes I: estimación y extracción de fondo

Se aplicaron dos algoritmos para la estimación y extracción de fondo. En la Figura 32 se muestra los resultados del uso del filtro mediana [22] en el primer video por densidad de personas. Además, la figura 33 muestra el uso del filtro mediana [22] para el segundo video para una alta densidad, y la figura 34 muestra los resultados para el video de más alta densidad.

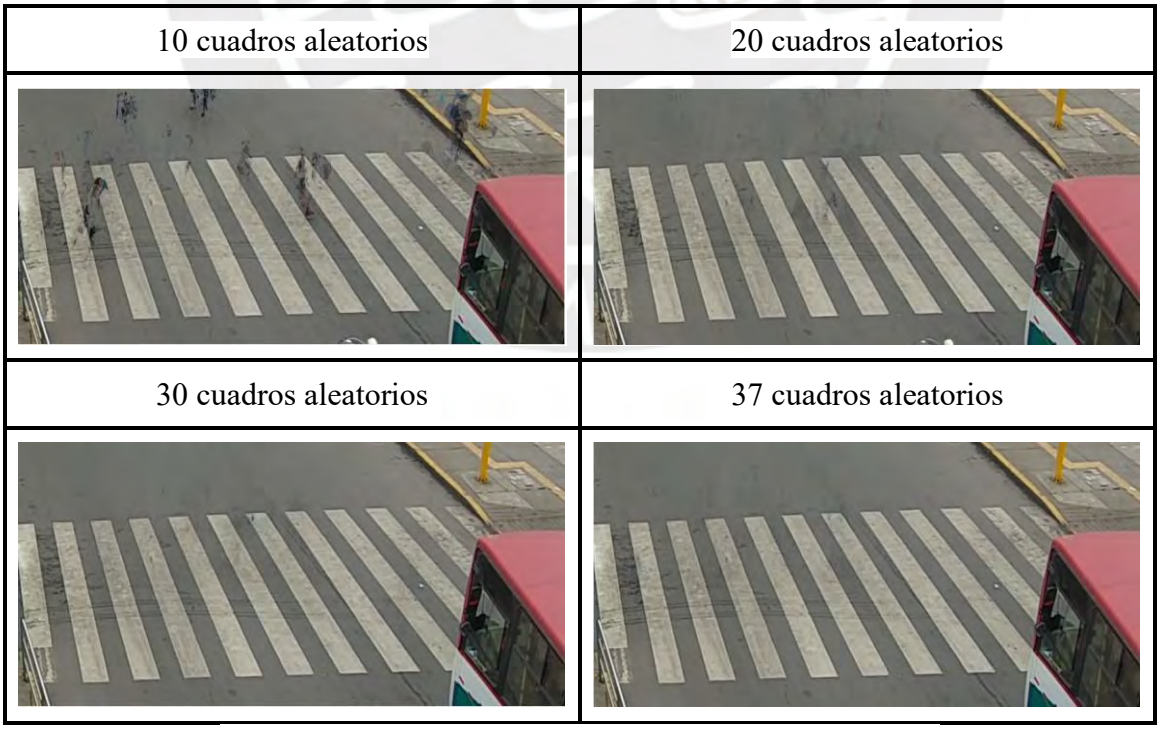



*Figura 32*: Resultados del uso del filtro mediana en el primer video

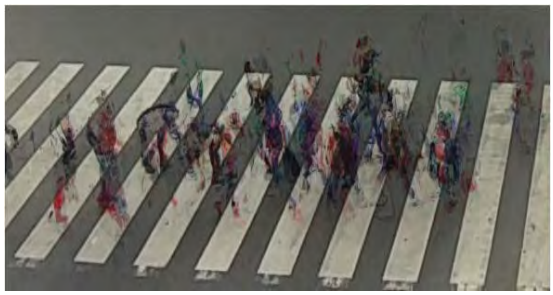
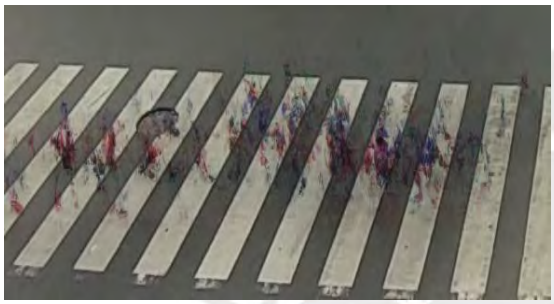
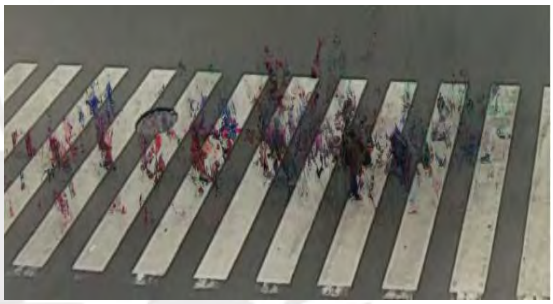
| 10 cuadros aleatorios | 20 cuadros aleatorios |
|---|---|
|  |  |
| 30 cuadros aleatorios | 37 cuadros aleatorios |
|  |  |

*Figura 33*: Resultados del uso del filtro mediana en el segundo video

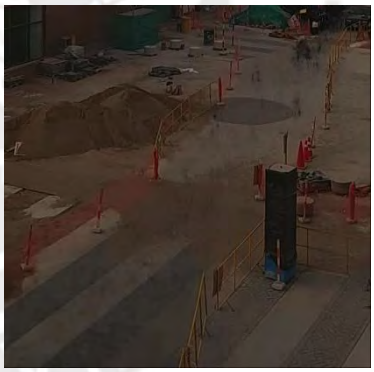
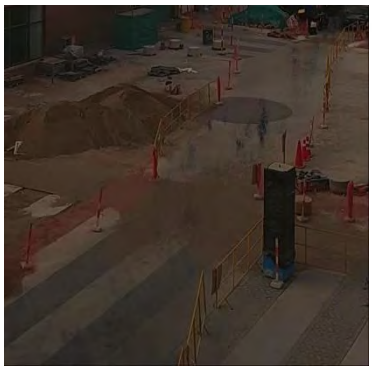
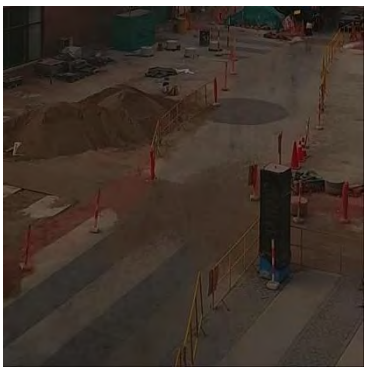
| 10 cuadros aleatorios | 20 cuadros aleatorios |
|---|---|
|  |  |
| 30 cuadros aleatorios | 37 cuadros aleatorios |
|  |  |

*Figura 34*: Resultados de filtro mediana en alta densidad [22]

Como se aprecia, a medida que aumenta el número de cuadros aleatorios, los puntos negros que representan a píxeles de personas se van eliminando. Por lo tanto, se usará el mayor número de cuadros (37 cuadros aleatorios) en este algoritmo para disminuir los errores.

El siguiente algoritmo para estimación de fondo es el de Modelos Mixtos Gaussianos (GMM, por sus siglas en inglés) [21], cuyas especificaciones son las siguientes:

*Tabla 7*: Parámetros del algoritmo GMM

| | |
|---|---|
| Función a utilizar | *ComputerVision . ForegroundDetector* |
| Nº de Gaussianos | 3 (estándar) |
| *MinimumBackgroundRatio* | 0.5 (estándar) |
| *InitialVariance* | (30/255)^2 (estándar) |

Adicionalmente se probó este método variando el número de cuadros de entrenamiento para encontrar el adecuado ante una densidad alta de personas. Dicha prueba se muestra a continuación para el primer video en la Figura 35. Luego, se observan los resultados para el segundo video del método GMM [21] en la Figura 36. Después, en la Figura 37, se da muestra de los resultados para el tercer video.

| 10 cuadros de entrenamiento | 30 cuadros de entrenamiento |
|---|---|
|  |  |
| 40 cuadros de entrenamiento | 50 cuadros de entrenamiento |
|  |  |

*Figura 35*: uso de GMM en el primer video

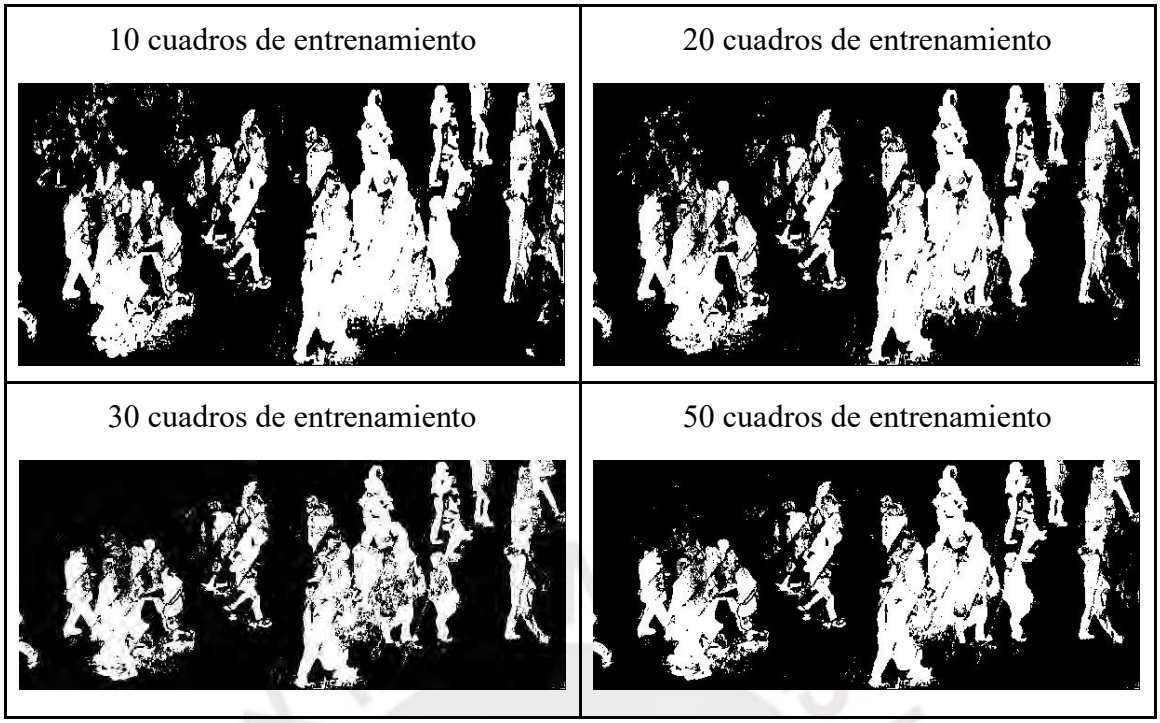



*Figura 36*: uso de GMM en el segundo video

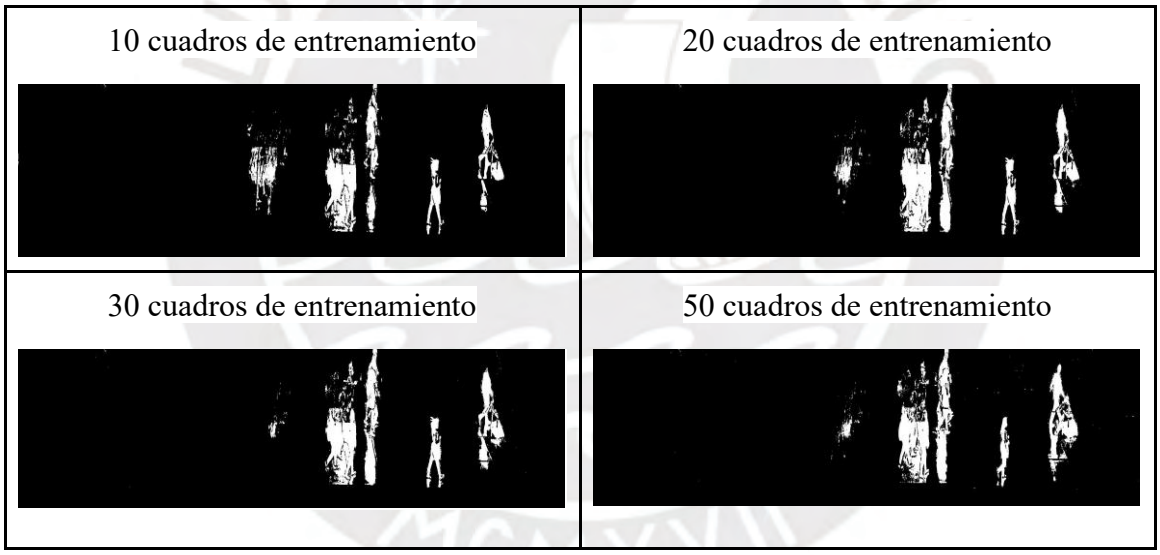



*Figura 37*: uso de GMM en el tercer video

Para este segundo método se optó también por la mayor cantidad de cuadros para el entrenamiento (50), siempre y cuando no se genere mucho ruido en las imágenes de salida, ya que estas luego serán filtradas por operaciones de apertura y cerradura.

### 4.2. Resultados de procesamiento de imágenes II: filtrado y seguimiento

Después de determinar las imágenes binarias, se procedió a filtrarlas con algoritmos simples de apertura y cerradura. Los resultados de dicho filtrado se muestran en la Figura 38:

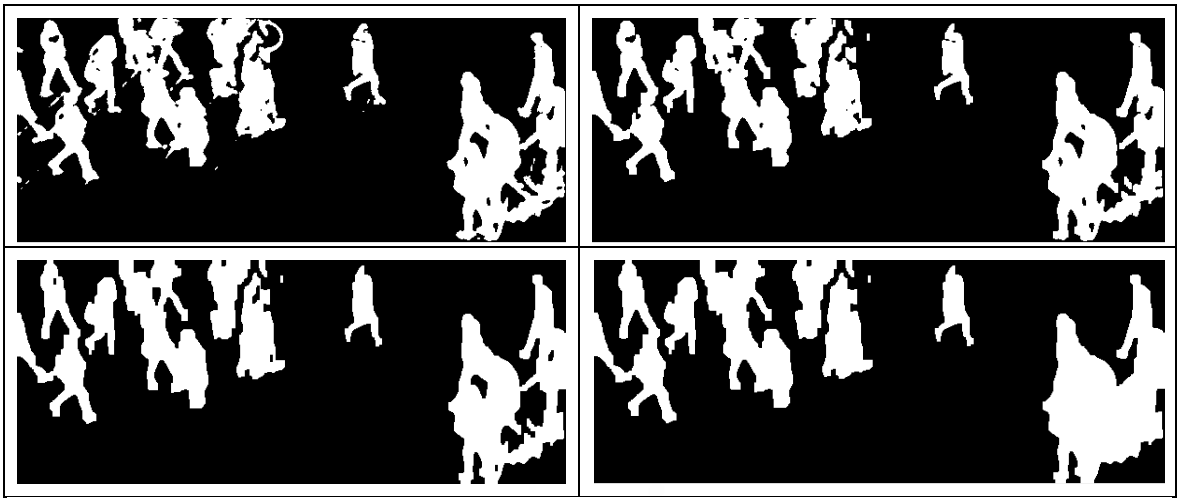


*Figura 38*: resultados de sub-etapas morfológicas. Arriba-izquierda: imagen binaria original. Arriba-derecha: imagen después de apertura. Abajo-izquierda: imagen después de cerradura. Abajo-derecha: imagen después de llenar hoyos.

Por otro lado, a partir del seguimiento utilizando Filtro Kalman y la estructura *tracks* para almacenar los datos de las siluetas seguidas y el algoritmo húngaro [27] para asignación de coste menor, se obtuvieron los resultados mostrados para densidad alta en la Figura 39; para densidades bajas en el tercer video en la Figura 40; y para densidades bajas del cuarto video en la Figura 41.

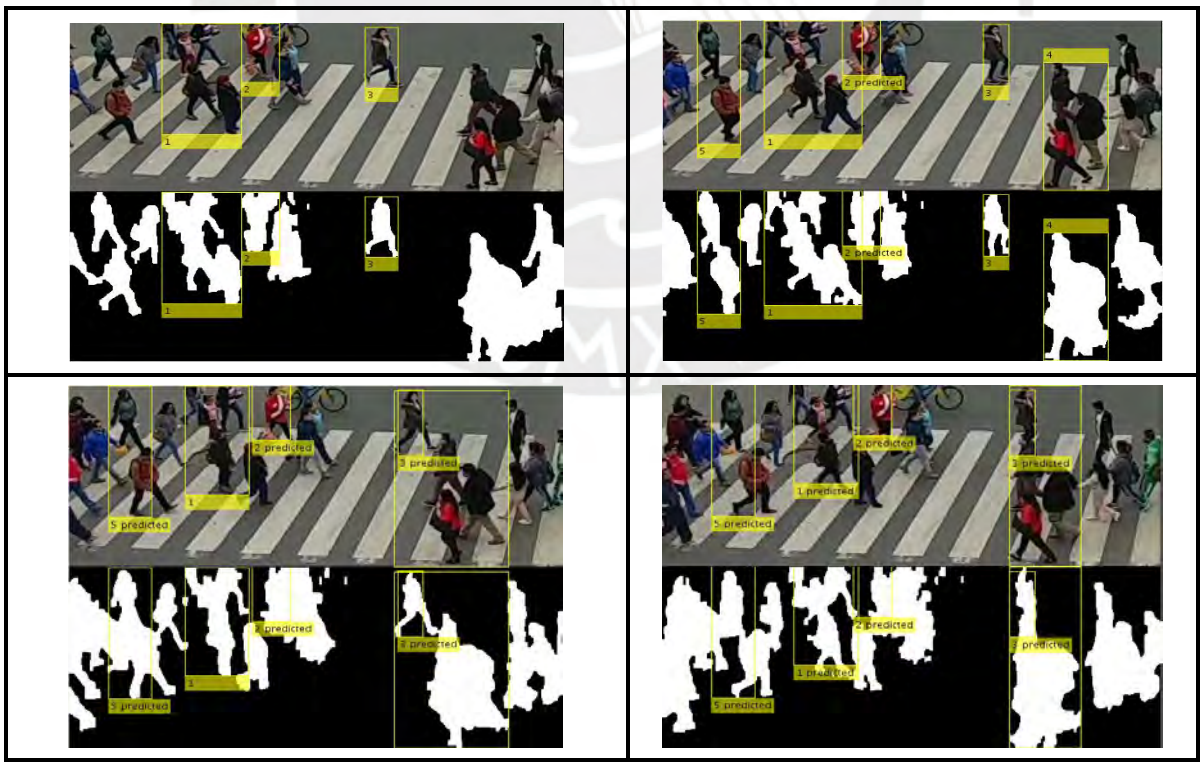


*Figura 39*: Desempeño del algoritmo en altas densidades

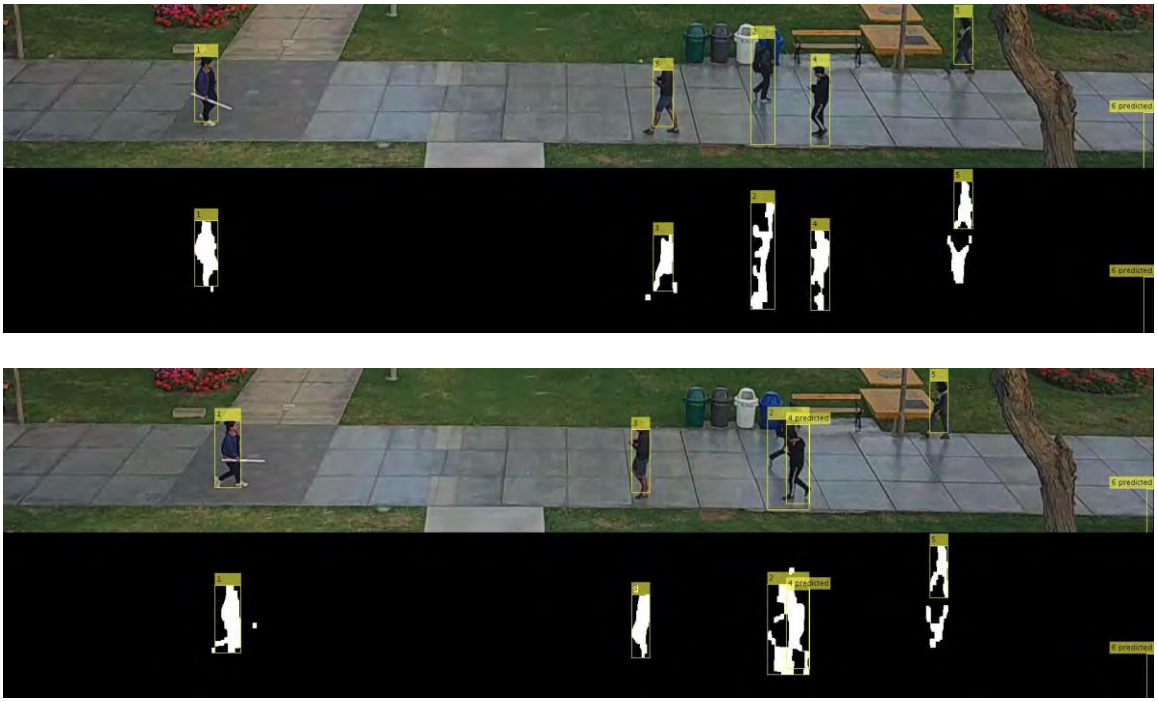


*Figura 40*: desempeño del algoritmo en densidades bajas en el tercer video

Y finalmente en el cuarto video:

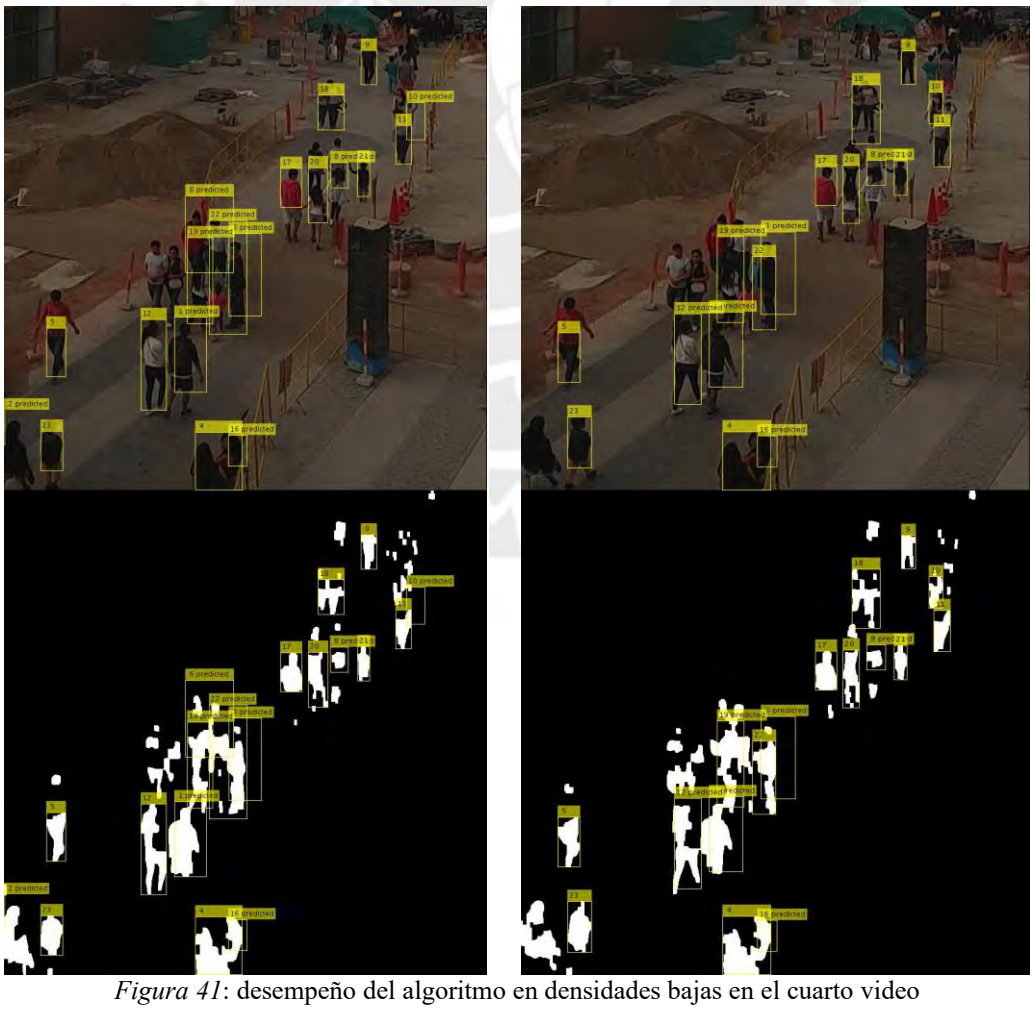


*Figura 41*: desempeño del algoritmo en densidades bajas en el cuarto video

Como se aprecia, en altas densidades de personas el algoritmo omite varias personas agrupadas por oclusión y algunas de ellas se descartan por el límite de área.

El desempeño final del programa entero para detectar peatones utilizando el filtro mediana [22] para la detección de fondo se resume en la Tabla 8.

*Tabla 8*: resultados de aplicación final

| | VP | FP | FN | RFN (%) | RFP (%) |
|---|---|---|---|---|---|
| 1er video | 81 | 0 | 9 | 10 | 0 |
| 2do video | 41 | 13 | 0 | 0 | 24.07 |
| 3er video | 52 | 29 | 15 | 22.39 | 35.80 |
| 4o video | 530 | 110 | 0 | 0 | 17.19 |
| Total | 174 | 42 | 24 | 12.12 | 19.44 |

donde

VP: Verdaderos positivos, en este caso, personas que fueron detectadas

FP: Falsos positivos, en este caso, siluetas que fueron detectadas como personas cuando no correspondían a ellos

FN: Falsos negativos, en este caso, personas faltantes que no fueron detectadas

RFN: Ratio de falsos negativos

RFP: Ratio de falsos positivos

Estos resultados de RFP y RFN [28] son mayores que en el método de conteo de cabezas de Bin Li et. al. [15] pero menores al método de Mukherjee et. al. Con densidad de peatones similar [16].

## CONCLUSIONES

- Aunque los parámetros del algoritmo se hayan ajustado para ejecutar un adecuado filtrado en altas densidades, se encuentra en ocasiones, como en la Figura 35, con no detecciones (falsos negativos) cuando hay mayor oclusión debido al umbral de área máxima en el análisis de manchas (*blobs)* que fue de 115*100 píxeles.
- Para la etapa de extracción de fondo, se halló que el método de filtro mediana tuvo un mejor desempeño con igual número de cuadros aleatorios que el método Gaussiano. Por otro lado, el ángulo de grabación influye considerablemente en los resultados de los ratios de los algoritmos (RFP Y RFN) debido a que se genera menos oclusión a menor ángulo con respecto a la vertical del piso (visión vertical).
- Al clasificar los cuadros por densidad de personas se pudo encontrar un mejor número de cuadros para la extracción de fondo que se desempeña bien para los tres casos de densidad: baja (menor a 8 personas), media (entre 8 y 17 personas) y alta (mayor a 17 personas) ya que se trabaja con el número correspondiente a la densidad más grande.
- Aparecen falsos positivos y negativos en los resultados debido principalmente a la oclusión de personas cuando estas se juntan y forman manchas que las hacen parecer como una única silueta y esto puede ocasionar que el seguimiento de dichas personas se pierda y se contabilicen como una nueva después de que se ellas se separen, ya que el umbral de personas contadas invisibles no alcanza a periodos grandes de tiempos (4 cuadros). Por otro lado, existe un límite para el área máxima de las siluetas, por lo que varias de estas se descartan cuando lo superan, especialmente en densidades altas.
- El porcentaje final de ratios de falsos positivos y negativos (RFP y RFN) en los métodos ejecutados (12.12% y 19.44%) se ajusta a las condiciones de trabajo y resultó mejor que en otros métodos estudiados, pero aún requiere mejoras sustanciales para alcanzar mayor eficacia.

## RECOMENDACIONES Y TRABAJOS FUTUROS

- La filmación se debe realizar en el futuro con una mayor ventana de tiempo de al menos una hora para tener más muestras que procesar y disminuir el error generado. Por otro lado, la vista se debe ejecutar de manera más perpendicular al plano de los peatones para evitar la oclusión.
- Se recomienda realizar la filmación en un espacio único todo el tiempo, ya que el algoritmo trabaja con tamaños de persona establecidos para un determinado escenario, en este caso, en la estación La Cultura.
- Como se probó en el tercer video, la presencia de sombras, aún cuando estas son de baja tonalidad, disminuye la calidad del programa final, por lo que propone utilizar algoritmos que eliminen sombras y favorezcan el conteo.
- Se propone también añadir los tiempos en que vehículos pasan por el paso de cebra al análisis con el fin de descartarlos en el conteo de peatones y, con esto, también aumentar la eficacia del algoritmo, ya que el segundo video tuvo que ser recortado a los segundos en donde no se encuentran autos.
- En una siguiente versión se espera reconocer las direcciones que la gente tome para reconocer si es que cruzaron en la dirección establecida y en tiempos debidos a una velocidad prudente.

**BIBLIOGRAFÍA**


[1] ALFARO, Rubén

2017 *Estudio empírico de comportamiento peatonal en los alrededores del Hospital del Niño, en Lima*. Tesis de licenciatura en Ciencias e Ingeniería con mención en Ingeniería Civil. Lima: Pontificia Universidad Católica del Perú, Facultad de Ciencias e Ingeniería.

[2] CASTRO, Jorge

2016 "El paso de peatones más seguro es el aparentemente más caótico". *El Diario*. Madrid, 3 de junio. Consulta: 12 de abril de 2018. https://www.eldiario.es/edcreativo/prueba/paso-peatones-seguro-parentemente-caotico_6_522907723.html

[3] TORRES, Fabiola

2010 "VIDEO: las calles limeñas no están diseñadas para caminar". *El Comercio.* Lima, 21 de noviembre. Consulta: 12 de abril de 2018. http://archivo.elcomercio.pe/sociedad/lima/video-calles-limenas-no-estan-disenadas-caminar-noticia-672334

[4] MINISTERIO DE TRANSPORTES Y COMUNICACIONES

2009 *La Vulnerabilidad de los Peatones en la Vialidad del Área Metropolitana de Lima y Callao*. Lima

[5] ITDP

*Jerarquía de la movilidad urbana (pirámide).* Consulta: 12 de abril de 2018 https://mexico.itdp.org/multimedia/infografias/jerarquia-de-la-movilidad-urbana-piramide/

[6] MUNICIPALIDAD DE LIMA

2017 "*Municipio de Lima mejora ciclo semafórico en 355 intersecciones viales*". Consulta: 29 de abril de 2018

http://www.munlima.gob.pe/noticias/1-noticias/municipio-de-lima-mejora-ciclo-sema%C3%B3rico-en-355-intersecciones-viales

[7] EL COMERCIO

2015 "Paraderos y riesgo de accidentes". *El Comercio.* Lima, 31 de marzo de 2015. Consulta: 29 de abril de 2018

https://elcomercio.pe/blog/expresiongenetica/2015/03/accidentes-paraderos

[8] EL COMERCIO

2015 "Diez de las zonas críticas de accidentes de tránsito en Lima". *El Comercio.* Lima, 19 de agosto de 2015. Consulta: 29 de abril de 2018

https://elcomercio.pe/lima/diez-zonas-criticas-accidentes-transito-lima-198350

[9] DIOGENES, Mara; GREENE-ROESEL, Ryan; ARNOLD, Lindsay; RAGLAND, David R.

2007 *Pedestrian Counting Methods at Intersections: a Comparative Study*. UC Berkeley. Consulta: 20 de abril de 2018.

https://escholarship.org/uc/item/208349wf

[10] SCHNEIDER, Bob

2010 *Tips for pedestrians and bicycle counting*. Ponencia presentada en *ProWalk/ProBike Conference*. UC Berkeley. Setiembre de 2010.

[11] GOGO

GOGO Tally Counter, Hand Held Counter, 4 Digit Manual Mechanical Click Counter. Consulta: 30 de abril de 2018.

https://www.amazon.ca/GOGO-Tally-Counter-ManualMechanical/dp/ B001KX1VW2

[12] GWANG-GOOK, Lee; HYEONG-KI Kim; JA-YOUNG, Yoon; JAE-JUN Kim; WHOI-YUL Kim

2008 *Pedestrian Counting Using an IR Line Laser*. Ponencia presentada en *Convergence and Hybrid Information Technology* . IEEE. Daejeon, 28-30 de agosto.

[13] CABLEMATIC

Contador de personas IP para instalar en techo de 2.1m de altura. Consulta: 25 de abril de 2018. http://www.cablematic.es/producto/Contador-de-personas-IP-para-instalar-en-techo-de-2_dot_1m-de-altura/SG61/

[14] MIRAME

Contador de personas integrado en cámara IP. Consulta: 25 de abril de 2018. https://www.mirame.net/conteo-personas.html

[15] BIN Li; JIAN Zhang; ZHENG Zhang; YONG Xu

2014 *A people counting method based on head detection and tracking*. Ponencia presentada en *International Conference on Smart Computing* . IEEE. Hong Kong, 3-5 de noviembre.

[16] MUKHERJEE, S; SAHA, B; JAMAL, L; LECLERC, R; RAY, N.

2011 *A novel framework for automatic passenger counting*. Ponencia presentada en *International Conference on Image Processing*. IEEE. Bruselas, 11-14 de septiembre.

[17] SHOKROLAH SHIRAZI, Mohammad; MORRIS, Brendan

2016 *Vision-based pedestrian monitoring at intersections including behavior & crossing count*. Ponencia presentada en *Intelligent Vehicles Symposium*. IEEE. Gothenburg, 19-22 de junio

[18] CHAN, Antoni B.; ZHANG-SHENG, John Liang; NUNO, Vasconcelos

2008 *Privacy preserving crowd monitoring: Counting people without people models or tracking*. Ponencia presentada en *International Conference on Computer Vision and Pattern Recognition*. IEEE. Anchorage, 23-28 de junio

[19] GAYATRI D. Prabhu

2011 *Automated detection and counting of pedestrians on an urban roadside*. Tesis de maestría en Science in electrical and computer engineering. Amherst: University of Massachusetts Amherst, Department of Electrical and computer engineering. Consulta: 16 de mayo de 2018. https://scholarworks.umass.edu/cgi/viewcontent.cgi?referer=https://www.google.com.pe/&httpsredir=1&article=1809&context=theses

[20] LEFLOCH, Damien; ALAYA CHEIKH, Faouzi; HARDEBERG, Jon Yngve; PICOT-CLEMENTE, Romain

2011 *Real-time counting system using a single video-camera*. Presentada en *International society for optical engineering* https://www.researchgate.net/publication/228437591_Real-time_people_counting_system_using_a_single_video_camera

[21] NURHADIYATNA, Adi; JAMITKO, Wisnu; HARDJNO, Benny

2013 *Background Subtraction Using Gaussian Mixture Model Enhanced by Hole Filling Algorithm (GMMHF)*. Ponencia presentada en *International Conference on Systems, Man, and Cybernetics*. IEEE. Manchester, 13-16 de octubre

[22] TAMERSOY, Birgi

2009 *Background Substraction* [diapositiva]. Consulta: 5 de junio de 2018 https://pdfs.semanticscholar.org/4215/f8bdb3bef14fb698e31e56959a0f0 2d4c7f9.pdf

[23] KIM, Kyungnam; CHALIDABHONGSEB Thanarat H.; HARWOODA David; DAVIS, Larry

2005 *Real-time foreground–background segmentation using codebook model*. Presentada en *Computer Vision Lab, Department of Computer Science, University of Maryland.* http://legacydirs.umiacs.umd.edu/~knkim/paper/Kim-RTI2005-FinalPublished.pdf

[24] BARNICH, Olivier; VAN DROOGENBROECK, Marc;

2011 *ViBe: A universal background subtraction algorithm for video sequences*. Publicada en *Transactions on Image Processing*. IEEE.

[25] FERNÁNDEZ ESTEBERENA, Leonardo A.

2017 *Extensión de Algoritmos de Sustracción de Fondo para Cámaras PTZ*. Trabajo final de carrera en Ingeniería de Sistemas. Universidad Nacional del Centro de la Provincia de Buenos Aires, Facultad de Ciencias Exactas. Consulta: 4 de junio de 2018. http://www.ridaa.unicen.edu.ar/xmlui/bitstream/handle/123456789/1353/FErn%C3%A1ndez%20Esteberena%2C%20Leonardo.PDF?sequence=1&isAllowed=y

[26] RODRÍGUEZ MUÑOZ, Patricia.

2003 *Aplicación del filtro de Kalman al seguimiento de objetos en secuencias de imágenes*. Trabajo final de carrera en Ingeniería técnica en informática de sistemas. Universidad Rey Juan Carlos. Consulta: 6 de junio de 2018. http://caporesearch.es/jjpantrigo/PFCs/MemoriaKalmanJun03.pdf

[27] MATHWORKS

Motion-Based Multiple Object Tracking. Consulta: 21 de junio de 2018. https://la.mathworks.com/help/vision/examples/motion-based-multiple-object tracking.html

[28] GOOGLE DEVELOPERS

Clasificación: Precisión y exhaustividad. Consulta: 21 de noviembre de 2018 https://developers.google.com/machine-learning/crash-course/classification/ precision-and-recall?hl=es-419

## ANEXOS

El código utilizado para el programa principal y las funciones respectivas es el siguiente:

```
%%Aplicación con primera entrada
Primer_video='Entrada_1.mp4';%Video de entrada
segundo_inicio=15;%Segundo de inicio y fin de análisis
segundo_fin=126;
rango_x=724:994;%Este rango de x e y indica la zona de análisis del video
rango_y=3:1915;
[Frame_rate_ori,nombre_carpeta_ori,duracion]=videoaframes(Segundo_video,segundo_inicio,segundo_fin,rango_x,rango_y);
nuevo_framerate=5;%Frame rate de cambio
nombre_carpeta_5_fps=cambia_frame_rate(nombre_carpeta_ori,nuevo_framerate,duracion);
%Transforma la carpeta anterior en video para poderlo procesar después
nombre_video_5_fps=framesavideo(6,2700,nombre_carpeta_5_fps,nuevo_framerate);
%Ejecuta el algoritmo de filtro mediana en el video anterior
[obj1,nombre_carpeta_binaria,num_frames]=filtro_mediana(nombre_video_5_fps,50);
%Ejecuta el algoritmo de GMM en el video anterior
[obj2,nombre_carpeta_binaria2,num_frames]=GMM(nombre_video_5_fps,50);
%Procesa las máscaras encontradas con filtro mediana en funciones anteriores y entrega las
%siluetas contadas
[nombre_carpeta_mascara,nombre_carpeta_final]=procesa_mascara(obj1,num_frames,nombre_carpeta_binaria);
%Procesa las máscaras encontradas con GMM en funciones anteriores y entrega las
%siluetas contadas
[nombre_carpeta_mascara_2,nombre_carpeta_final_2]=procesa_mascara_GMM(obj2,num_frames,nombre_carpeta_binaria2);
%Transforma las siluetas contadas y seguidas en video
nombre_video_final=framesavideo(1,num_frames,nombre_carpeta_final,nuevo_framerate);
%Transforma las siluetas contadas y seguidas en video
nombre_video_final_2=framesavideo(1,num_frames,nombre_carpeta_final_2,nuevo_framerate);
%%
%Aplicación con cuarta entrada
Segundo_video='Entrada_4.mp4';%Video de entrada
segundo_inicio=1;%Segundo de inicio y fin de análisis
segundo_fin=300;
rango_x=1063:1907;%Este rango de x e y indica la zona de análisis del video
```

```
rango_y=190:1022
[Frame_rate_ori,nombre_carpeta_ori,duracion]=videoaframes(Segundo_video,segundo_inicio,segundo_fin,rango_x,rango_y);
nuevo_framerate=5;%Frame rate de cambio
nombre_carpeta_5_fps=cambia_frame_rate(nombre_carpeta_ori,nuevo_framerate,duracion);
%Transforma la carpeta anterior en video para poderlo procesar después
nombre_video_5_fps=framesavideo(6,9000,nombre_carpeta_5_fps,nuevo_framerate);
%Ejecuta el algoritmo de filtro mediana en el video anterior
[obj1,nombre_carpeta_binaria,num_frames]=filtro_mediana(nombre_video_5_fps,50);
%Ejecuta el algoritmo de GMM en el video anterior
[obj2,nombre_carpeta_binaria2,num_frames]=GMM(nombre_video_5_fps,50);
%Procesa las máscaras encontradas con filtro mediana en funciones anteriores y entrega las
%siluetas contadas
[nombre_carpeta_mascara,nombre_carpeta_final]=procesa_mascara(obj1,num_frames,nombre_carpeta_binaria);
%Procesa las máscaras encontradas con GMM en funciones anteriores y entrega las
%siluetas contadas
[nombre_carpeta_mascara_2,nombre_carpeta_final_2]=procesa_mascara_GMM(obj2,num_frames,nombre_carpeta_binaria2);
%Transforma las siluetas contadas y seguidas en video
nombre_video_final=framesavideo(1,num_frames,nombre_carpeta_final,nuevo_framerate);
%Transforma las siluetas contadas y seguidas en video
nombre_video_final_2=framesavideo(1,num_frames,nombre_carpeta_final_2,nuevo_framerate);
```

Funciones propias adicionales :

1. Videoaframes.m

```
function [FR,nombre_carpeta,duracion] =
videoaframes(nombre_entrada,segundo_inicio,segundo_fin,rango_x,rango_y)
   video_entrada=VideoReader(nombre_entrada);
   direccion=video_entrada.Path();
   duracion=round(video_entrada.Duration(),0);
   if ((segundo_inicio>=0)&&(segundo_fin>=segundo_inicio)&&(segundo_fin<=duracion))
      FR=round(video_entrada.FrameRate(),0);
      nombre_video=strrep(nombre_entrada,'.mp4','');
      nombre_carpeta=strcat('Frames_de_',nombre_video,'_',num2str(FR),'_fps');
```

```
        mkdir(nombre_carpeta);
      if (segundo_inicio==0)
         Corte_inicio=1;
      else
         Corte_inicio=segundo_inicio*FR;
      end
      Corte_fin=segundo_fin*FR;
      for frame=Corte_inicio:Corte_fin
         frame_actual_input=read(video_entrada,frame);
         cadena_nombre=strcat('Frame Nº',num2str(frame),'.jpg');
         full_direccion=fullfile(direccion,nombre_carpeta,cadena_nombre);
         imwrite(frame_actual_input(rango_x,rango_y,:),full_direccion);
      end
   end
end
```

2. Cambia_frame_rate

```
function [ cadena_nombre_final ] = cambia_frame_rate(
nombre_carpeta_inicial,FRfinal,duracion )
   temporal=strrep(nombre_carpeta_inicial,'Frames_de_','');
   temporal=strrep(temporal,'_fps','');
   pos_numero=strfind(temporal,'_');
   pos_numero=pos_numero(end);
   FRinicial=str2num(temporal(pos_numero+1:end));
   temporal2=strcat('_',num2str(FRinicial));
   nombre_original=strrep(temporal,temporal2,'');

cadena_nombre_final=strcat('Frames_de_',nombre_original,'_',num2str(FRfinal),'_fps');
   div=round(FRinicial/FRfinal,0);
   mkdir(cadena_nombre_final)
   for i=(1:(duracion*FRinicial))
      if (mod(i,div)==0)
         cd(nombre_carpeta_inicial);
         nombre_archivo=strcat('Frame Nº',num2str(i),'.jpg');
         if exist(nombre_archivo)
            imagen=imread(nombre_archivo);
            cd ..;
            cd(cadena_nombre_final);
            imwrite(imagen,nombre_archivo);
         end
         cd ..;
```

```
    end
  end
end
```

3. Framesavideo

```
function [ nombre_video_salida ] = framesavideo(
inicio,fin,nombre_carpeta,nuevo_framerate)
  video_salida=VideoWriter(nombre_carpeta,'MPEG-4');
  video_salida.FrameRate=nuevo_framerate;
  open(video_salida);
  for i=inicio:fin
    cd(nombre_carpeta);
    cad_imagen=strcat('Frame Nº',num2str(i),'.jpg');
    if exist(cad_imagen)
      frame_actual=imread(cad_imagen);
      writeVideo(video_salida,frame_actual);
    end
    cd ..;
  end
  close(video_salida);
  nombre_video_salida=strcat(nombre_carpeta,'.mp4');
```

4. Filtro_mediana

```
function [ obj, nombre_carpeta_binaria,TotalDeFrames] = filtro_mediana(
nombre_video_entrada,num_frames_aleatorios )
    obj.reader = VideoReader(nombre_video_entrada);%para filtro mediana
    TotalDeFrames=obj.reader.NumberOfFrames();
    frames=1:TotalDeFrames;
    for i=1:num_frames_aleatorios
      posicion=randi(length(frames));
      aleatorio=frames(posicion);
      imagen=read(obj.reader,aleatorio);
      if (i==1)
        Vect=size(imagen);
        Fil=Vect(1);
        Col=Vect(2);
        Capas=Vect(3);
```

```
            arreglo=zeros(Fil,Col,Capas,num_frames_aleatorios);
            end
            arreglo(:,:,:,i)=double(imagen(:,:,:))/255;
        end
        fondo=median(arreglo,4);
        nombre_carpeta_binaria=strcat('Frames obtenidos por filtro
mediana_',num2str(num_frames_aleatorios));
        mkdir(nombre_carpeta_binaria);
        cd(nombre_carpeta_binaria);
        imwrite(fondo,strcat('Frame de fondo por filtro
mediana_',num2str(num_frames_aleatorios),'.jpg'));
        for(i=1:TotalDeFrames)
            imagen=read(obj.reader,i);
            dif=double(imagen)/255-fondo;
            mask=sum(dif.^2,3)>0.04;
            k=ones(5)/25;
            im_f=imfilter(double(mask),k);
            mascara=im_f>0.4;
            nombre_mascara=strcat('Frame Nº',num2str(i),'.jpg');
            imwrite(mascara,nombre_mascara);
        end
        cd ..;
end
```

5. GMM

```
function [ obj, nombre_carpeta_binaria,i] = GMM( nombre_video_entrada,numero_TF)
        obj.reader =
vision.VideoFileReader('Filename',nombre_video_entrada,'PlayCount',Inf);

        %TotalDeFrames=obj.reader.NumberOfFrames();
        obj.detector = vision.ForegroundDetector('NumGaussians', 3,'NumTrainingFrames',
numero_TF, 'MinimumBackgroundRatio', 0.3,'InitialVariance',(30/255)^2);
        nombre_carpeta_binaria=strcat('Frames obtenidos por
GMM_',num2str(numero_TF));
        mkdir(nombre_carpeta_binaria);
        cd(nombre_carpeta_binaria);
        i=1;
        while(~isDone(obj.reader))% Para GMM
            frame_actual=obj.reader.step();
            sacado=obj.detector.step(frame_actual);
```

```
cad_salida=strcat('Frame Nº',num2str(i),'.jpg');
        imwrite(sacado,cad_salida);
        i=i+1;
    end
    i=i-1;
    cd ..;
    obj.reader = VideoReader(nombre_video_entrada);
end
```

6. Procesa_mascara

```
function [ nombre_salida_mascara,nombre_salida_resuelta ] = procesa_mascara(
obj,num_frames,nombre_carpeta_binaria)
  tracks = initializeTracks(); % Create an empty array of tracks.
  nextId = 1; % ID of the next track
  nombre_salida_mascara=strcat('Frames_de_máscara_binaria');
  mkdir(nombre_salida_mascara);
  nombre_salida_resuelta=strcat('Frames_de_máscara_resuelta');
  mkdir(nombre_salida_resuelta);
  nframe=353;
  ids_usados=[];
  obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort',
true,'AreaOutputPort', true, 'CentroidOutputPort', true, 'MinimumBlobArea',
800,'MaximumBlobArea',115*100);
  % Detect moving objects, and track them across video frames.
  while (nframe<=num_frames)
    % frame =read(obj.reader,nframe);
    frame=read(obj.reader,nframe);
    cd(nombre_carpeta_binaria);
    binaria=imread(strcat('Frame Nº',num2str(nframe),'.jpg'));
    cd ..;
    [centroids, bboxes, mask] = detectObjects(frame,obj,binaria);
    nombre_mask=strcat('Frame Nº',num2str(nframe),'.jpg');
    cd(nombre_salida_mascara);
    frame_unido=[frame;255*repmat(mask,[ 1, 1, 3])];
    imwrite(frame_unido,nombre_mask);
    cd ..;
    [tracks]=predictNewLocationsOfTracks(tracks);
    [assignments, unassignedTracks, unassignedDetections] =
detectionToTrackAssignment(tracks,centroids);
    [tracks]=updateAssignedTracks(assignments,centroids,bboxes,tracks);
    [tracks]=updateUnassignedTracks(unassignedTracks,tracks);
```

```
        [tracks]=deleteLostTracks(tracks);
        [tracks,nextId] =
createNewTracks(centroids,unassignedDetections,bboxes,tracks,nextId);
         ids_usados =
displayTrackingResults(frame,mask,tracks,nombre_salida_resuelta,nframe,ids_usados);
        nframe=nframe+1;
    end
end
```

7. DetectObjects

```
function [centroids, bboxes, mask] = detectObjects(frame,obj,binaria)
        % Detect foreground.
        mask = (binaria/255)>=0.5;
        % Apply morphological operations to remove noise and fill in holes.
        mask = imopen(mask, strel('rectangle', [9,5]));
        mask = imclose(mask, strel('rectangle', [9,5]));%max 9*5
        mask = imfill(mask, 'holes');

        % Perform blob analysis to find connected components.
        [~, centroids, bboxes] = obj.blobAnalyser.step(mask);
end
```

8. PredictNewLocationsOfTracks

```
function [ nombre_salida_mascara,nombre_salida_resuelta ] = procesa_mascara(
obj,num_frames,nombre_carpeta_binaria)
    tracks = initializeTracks(); % Create an empty array of tracks.
    nextId = 1; % ID of the next track
    nombre_salida_mascara=strcat('Frames_de_máscara_binaria');
    mkdir(nombre_salida_mascara);
    nombre_salida_resuelta=strcat('Frames_de_máscara_resuelta');
    mkdir(nombre_salida_resuelta);
    nframe=1;
    ids_usados=[];
    obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort',
true,'AreaOutputPort', true, 'CentroidOutputPort', true, 'MinimumBlobArea',
800,'MaximumBlobArea',115*100);
    while (nframe<=num_frames)
        frame=read(obj.reader,nframe);
```

```
        cd(nombre_carpeta_binaria);
        binaria=imread(strcat('Frame Nº',num2str(nframe),'.jpg'));
        cd ..;
        [centroids, bboxes, mask] = detectObjects(frame,obj,binaria);
        nombre_mask=strcat('Frame Nº',num2str(nframe),'.jpg');
        cd(nombre_salida_mascara);
        frame_unido=[frame;255*repmat(mask,[ 1, 1, 3])];
        imwrite(frame_unido,nombre_mask);
        cd ..;
        [tracks]=predictNewLocationsOfTracks(tracks);
        [assignments, unassignedTracks, unassignedDetections] =
detectionToTrackAssignment(tracks,centroids);
        [tracks]=updateAssignedTracks(assignments,centroids,bboxes,tracks);
        [tracks]=updateUnassignedTracks(unassignedTracks,tracks);
        [tracks]=deleteLostTracks(tracks);

[tracks,nextId]=createNewTracks(centroids,unassignedDetections,bboxes,tracks,nextId);
ids_usados=displayTrackingResults(frame,mask,tracks,nombre_salida_resuelta,nframe,id
s_usados);
        nframe=nframe+1;
    end
end
```

9. DetectionToTrackAssignment

```
function [assignments, unassignedTracks, unassignedDetections] =
detectionToTrackAssignment(tracks,centroids)
        nTracks = length(tracks);
        nDetections = size(centroids, 1);
        % Compute the cost of assigning each detection to each track.
        cost = zeros(nTracks, nDetections);
        for i = 1:nTracks
            cost(i, :) = distance(tracks(i).kalmanFilter, centroids);
        end
        % Solve the assignment problem.
        costOfNonAssignment = 10;
        [assignments, unassignedTracks, unassignedDetections] =
assignDetectionsToTracks(cost, costOfNonAssignment);
end
```

10. updateAssignedTracks

```
function [tracks]=updateAssignedTracks(assignments,centroids,bboxes,tracks)
    numAssignedTracks = size(assignments, 1);
    for i = 1:numAssignedTracks
        trackIdx = assignments(i, 1);
        detectionIdx = assignments(i, 2);
        centroid = centroids(detectionIdx, :);
        bbox = bboxes(detectionIdx, :);
        % Correct the estimate of the object's location
        % using the new detection.
        correct(tracks(trackIdx).kalmanFilter, centroid);
        % Replace predicted bounding box with detected
        % bounding box.
        tracks(trackIdx).bbox = bbox;
        % Update track's age.
        tracks(trackIdx).age = tracks(trackIdx).age + 1;
        % Update visibility.
        tracks(trackIdx).totalVisibleCount = ...
            tracks(trackIdx).totalVisibleCount + 1;
        tracks(trackIdx).consecutiveInvisibleCount = 0;
    end
    temporal=tracks;
    tracks=temporal;
end
```

11. updateUnassignedTracks

```
function [tracks]=updateUnassignedTracks(unassignedTracks,tracks)
    for i = 1:length(unassignedTracks)
        ind = unassignedTracks(i);
        tracks(ind).age = tracks(ind).age + 1;
        tracks(ind).consecutiveInvisibleCount = tracks(ind).consecutiveInvisibleCount + 1;
    end
    temporal=tracks;
    tracks=temporal;
end
```

12. deleteLostTracks

```
function tracks=deleteLostTracks(tracks)
    if isempty(tracks)
    return;
    end
    invisibleForTooLong = 10;
    ageThreshold = 40;
    % Compute the fraction of the track's age for which it was visible.
    ages = [tracks(:).age];
    totalVisibleCounts = [tracks(:).totalVisibleCount];
    %visibility = totalVisibleCounts ./ ages;
    % Find the indices of 'lost' tracks.
    lostInds = (ages >= ageThreshold)  | [tracks(:).consecutiveInvisibleCount] >=
invisibleForTooLong;
    % Delete lost tracks.
    temporal = tracks(~lostInds);
    tracks=temporal;
end
```

13. CreateNewTracks

```
function [tracks
nextId]=createNewTracks(centroids,unassignedDetections,bboxes,tracks,nextId)
    centroids = centroids(unassignedDetections, :);
    bboxes = bboxes(unassignedDetections, :);
    %areas=areas(unassignedDetections, :);
    temporal=tracks;
    idtemporal=nextId;
    for i = 1:size(centroids, 1)
        centroid = centroids(i,:);
        bbox = bboxes(i, :);
       % area=bbox(3)*bbox(4);
        % Create a Kalman filter object.
        kalmanFilter = configureKalmanFilter('ConstantVelocity',centroid, [200, 50],
[100, 25], 100);
        % Create a new track.
        newTrack = struct('id', idtemporal,'bbox',bbox,'kalmanFilter', kalmanFilter,'age',
1,'totalVisibleCount', 1, 'consecutiveInvisibleCount', 0);
        % Add it to the array of tracks.
        temporal(end + 1) = newTrack;
        % Increment the next id.
        idtemporal = idtemporal + 1;
```

```
    end
    nextId=idtemporal;
    tracks=temporal;
end
```

14. displayTrackingResults

```
function
ids_usados=displayTrackingResults(frame,mask,tracks,nombre_salida,nframe,ids_usados
)
    % Convert the frame and the mask to uint8 RGB.
    frame = im2uint8(frame);
    mask = uint8(repmat(mask, [1, 1, 3])) .* 255;
    minVisibleCount = 4;
    if ~isempty(tracks)
        reliableTrackInds = [tracks(:).totalVisibleCount] > minVisibleCount;
        reliableTracks = tracks(reliableTrackInds);
        % Display the objects. If an object has not been detected
        % in this frame, display its predicted bounding box.
        if ~isempty(reliableTracks)
            % Get bounding boxes.
            [verdaderos_ids,ids_usados]=buscar_ids_usados(reliableTracks,ids_usados);
            bboxes = cat(1, reliableTracks.bbox);
            %areas=bboxes(:,3).*bboxes(:,4);
            %areas=[areas]>1000;
            %bboxes=bboxes(areas,:);
            %reliableTracks=reliableTracks(areas);
            % Get ids.
            ids = int32([verdaderos_ids]);
            % Create labels for objects indicating the ones for
            % which we display the predicted rather than the actual
            % location.
            labels = cellstr(int2str(ids'));
            predictedTrackInds = [reliableTracks(:).consecutiveInvisibleCount] > 0;
            isPredicted = cell(size(labels));
            isPredicted(predictedTrackInds) = {' predicted'};
            labels = strcat(labels, isPredicted);
            % Draw the objects on the frame.
            frame = insertObjectAnnotation(frame, 'rectangle',bboxes, labels);
            % Draw the objects on the mask.
            mask = insertObjectAnnotation(mask, 'rectangle',bboxes, labels);
        end
```

```
        end
        % Display the mask and the frame.
        cd(nombre_salida);
        nombre_unido=strcat('Frame Nº',num2str(nframe),'.jpg');
        frame_unido=[frame;mask];
        imwrite(frame_unido,nombre_unido);
        cd ..;
end
```

15. BuscarIdsUsados

```
function [ ids_reales,ordenado] = buscar_ids_usados( ids_a_buscar,base_de_datos )
        ids_reales=[];
        [~,X]=size(ids_a_buscar);
        ordenado=base_de_datos;
        for i=1:X
            id_a_buscar=ids_a_buscar(i).id;
            ordenado=unique([ordenado id_a_buscar]);
            posicion=find(ordenado==id_a_buscar);
            ids_reales(end+1)=posicion;
        end
end
```