DETECCIÓN DE OBJETOS MÓVILES EN UNA ESCENA UTILIZANDO FLUJO ÓPTICO

DAVID LEONARDO MORA MARIÑO
ANDRÉS ORLANDO PÁEZ MELO

TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE INGENIERO ELECTRÓNICO

DIRECTOR
ING. JULIÁN QUIROGA SEPÚLVEDA, M.Sc.

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA
BOGOTÁ D.C.
JUNIO 2010

RECTOR MAGNÍFICO:	**JOAQUÍN SÁNCHEZ GARCÍA S. J.**
DECANO ACADÉMICO:	**Ing. FRANCISCO J. REBOLLEDO M.**
DECANO DEL MEDIO UNIVERSITARIO:	**SERGIO BERNAL RESTREPO S. J.**
DIRECTOR DE CARRERA:	**Ing. JUAN MANUEL CRUZ**
DIRECTOR DEL TRABAJO DE GRADO:	**Ing. JULIÁN QUIROGA SEPÚLVEDA**

ARTÍCULO 23 DE LA RESOLUCIÓN N° 13 DE JUNIO DE 1946

"La universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus trabajos de grado. Solo velara porque no se publique nada contrario al dogma y la moral católica y porque los trabajos no contengan ataques o polémicas puramente personales. Antes bien que se vea en ellos el anhelo de buscar la verdad y la justicia"

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

La investigación y desarrollo de métodos de detección de objetos mediante el procesamiento de imágenes se ha incrementando en los últimos años, debido a su aplicación en el campo de la robótica y en el control de tráfico, además del mejoramiento de los sistemas de computación. Las investigaciones en el estudio de tráfico, se han enfocado en el área de análisis automático del flujo vehicular y peatonal por medio del desarrollo de algoritmos de visión por computador, motivado por el incremento acelerado del tráfico vehicular y de los altos índices de accidentalidad en las principales capitales del mundo. Por ejemplo, en el caso de Bogotá, la cantidad de vehículos matriculados en la ciudad se incrementó en un 55% en los últimos seis años, mientras que las motocicletas casi triplicaron su número entre 2005 y 2007, según el Observatorio de movilidad de la CCB [15]. El uso de visión artificial en sistemas de monitoreo de trafico puede ofrecer muchas ventajas respecto a otras tecnologías [1].

Actualmente, el grupo de investigación en Sistemas Inteligentes, Robótica y Percepción (*SIRP*) del Departamento de Ingeniería Electrónica de la Pontificia Universidad Javeriana está llevando a cabo investigaciones en el área de análisis automático de variables de tráfico por medio del desarrollo de algoritmos de visión por computador. Hasta el momento, los trabajos se han basado en la estimación o modelado del fondo de la escena para detectar los objetos de interés como vehículos en [2] o peatones en [3]. En [2] el primer plano es estimado a partir de un modelo del fondo obtenido utilizando el método *FGDStatmodel* [4], el cual realiza una clasificación Bayesiana de los píxeles de la imagen, como pertenecientes al fondo o al primer plano, en función de su comportamiento temporal. En [3] el fondo es modelado a partir de un segmento de video en donde para cada píxel del fondo es elaborado un conjunto de posibles valores según su frecuencia de aparición, de acuerdo a [5].

La estimación del primer plano basada en fondo presenta algunos problemas, entre los cuales cabe destacar el alto costo computacional, debido a que el fondo debe ser actualizado o frecuentemente estimado, la sensibilidad a los cambios de iluminación, y la dificultad para estimar el fondo ante la presencia de objetos de interés estáticos [6].

Con el fin de superar estos inconvenientes, en los últimos años se han venido desarrollando diferentes métodos para la detección y el seguimiento de objetos móviles en los cuales no se realiza un modelado del fondo. Por lo general estos métodos se basan en la extracción y correspondencia de puntos característicos sobre un conjunto de imágenes para determinar desplazamiento aparente de los objetos [7], o en la estimación del flujo óptico de la imagen [8].

Los puntos característicos pueden ser elegidos de diferentes formas siguiendo un compromiso entre tiempo y robustez. Las características invariantes a escala (SIFT) propuestas por Lowe en [9] son comúnmente utilizados para este fin, sin embargo, recientemente, características diferentes basadas en superficies deformables han probado ser más robustas [7]. La correspondencia de SIFT, entre un par de cuadros de video, permite determinar con exactitud el desplazamiento relativo de un objeto en la escena, sin embargo la extracción y posterior comparación de estas características requiere de un alto costo computacional, el cual se incrementa rápidamente con el número de objetos móviles.

En este trabajo se presenta un algoritmo que permite detectar objetos móviles a partir de la estimación del flujo óptico e información a priori del tipo de objeto. La estimación del movimiento aparente es realizada utilizando una versión piramidal del algoritmo de Lucas y Kanade [10], lo que permite un algoritmo más rápido respecto a las estrategias basadas en SIFT, pero a su vez menos robusto a los cambios de iluminación y de escala. Con el fin de incrementar la velocidad del algoritmo, el flujo óptico es estimado sólo sobre un conjunto de puntos seleccionados de acuerdo al método propuesto por Shi y Tomassi [13]. Los objetos móviles son detectados a partir de las regiones de la imagen con flujo diferente de cero y agrupados de acuerdo a la velocidad de movimiento, y la distancia de Mahalanobis. Para la segmentación

de los objetos detectados, se presentaron 2 soluciones, una basada en el algoritmo de *k-means* y minimización de variables dependientes de la velocidad y la otra basada en la distancia de Chebychev y un factor de similitud.

Para la evaluación del algoritmo, se utilizaron videos que contienen flujo vehicular y videos que contienen flujo peatonal, en los cuales se cumple que los objetos presentan un buen contraste respecto la vía y el ángulo de captura minimice el traslape entre objetos de interés. El algoritmo se implementó en el lenguaje de programación C++ y se utilizaron las librerías de OpenCV versión 2.0 [14].

1. MARCO TEÓRICO

1.1 FLUJO ÓPTICO

El flujo óptico puede ser definido como el movimiento aparente de los patrones de intensidad en una imagen. La palabra aparente indica que el movimiento espacial de los objetos (campo de movimiento) puede coincidir o no con el flujo estimado. No obstante, en situaciones en las cuales el movimiento de los objetos implica un movimiento de sus patrones de intensidad en el plano de la imagen, el flujo óptico puede relacionarse directamente con el movimiento de los objetos en la escena.

La mayoría de las técnicas existentes para la estimación del flujo óptico se puede clasificar en 4 categorías: las basadas en gradientes espacio-temporales, las basadas en comparación de regiones, las basadas en fase y las basadas en energía.

(1)

En todas las estrategias de estimación de flujo óptico se parte de la hipótesis de que los niveles de gris permanecen constantes ante movimientos espaciales en un tiempo dado. Dicha hipótesis da lugar a la ecuación general de flujo óptico (1), donde corresponde a la intensidad en niveles de gris del píxel en la posición de la imagen en el tiempo .

Expandiendo (1) en series de Taylor sobre el punto y el tiempo , se obtiene:

— — —

donde contiene la información de las derivadas de orden superior. Si se asume despreciable, la ecuación de flujo óptico puede reescribirse como:

(2)

donde , con — y — corresponde al vector de flujo óptico. e son las derivadas parciales horizontal y vertical de la imagen, respectivamente. Para cada píxel de la imagen, en el tiempo *t*, puede plantearse la ecuación (2), sin embargo no existe una única solución para esta ecuación. Diferentes restricciones pueden emplearse para estimar el flujo óptico en la imagen.

1.2 MÉTODO DE LUCAS Y KANADE

Este método asume que el flujo óptico es constante sobre una región *R*. Sea *R* una región de la imagen, y su vector de flujo óptico asociado, entonces (2) se cumple para cada píxel de la región *R*, es decir:

Organizando el conjunto de ecuaciones en forma matricial se tiene:

(3)

donde la matriz *A* contiene las derivadas espaciales de la imagen, el vector *d* corresponde al vector de flujo óptico y el vector *b* contiene las derivadas temporales de la imagen. Pre-multiplicando (3) por la transpuesta de *A* se tiene:

donde el vector de flujo óptico es encontrado como:

El cálculo del flujo óptico implica la inversión de la matriz

(4)

por lo cual la solución existe si la matriz A^TA es invertible y bien condicionada.

1.2.1 Lucas y Kanade piramidal

Sean *I, J* dos imágenes bidimensionales en escala de grises, y sea la ubicación de un píxel en el plano, entonces:

La imagen será referenciada como la primera imagen y como la segunda imagen. Considere un punto específico en la primera imagen. La idea del seguimiento de características por medio

de la estimación del flujo óptico es encontrar la posición , en la siguiente imagen tal que y sean casi iguales. También es importante definir la similitud del brillo de puntos vecinos de entre las dos imágenes debido al problema de apertura [14]. Estos puntos vecinos se denominan ventana de integración.

La elección del tamaño de dicha ventana es muy importante. Si se escoge una ventana muy pequeña, es posible que no se puedan manejar grandes rangos de movimiento, y si se escoge una ventana muy grande, es posible que se presenten seguimientos incorrectos debido a la presencia de múltiples movimientos [16]. Con el fin de solucionar este problema, se realiza una implementación piramidal del algoritmo de Lucas y Kanade. La idea central de utilizar una representación piramidal es poder manejar grandes movimientos de los píxeles dentro de una secuencia de imágenes, y facilitar su seguimiento mediante una ventana fija y evitar inconsistencias de movimiento.

En la implementación piramidal de Lucas y Kanade (LK), el flujo óptico es estimado por primera vez sobre la imagen de menor resolución, luego, dicha imagen va creciendo en un factor de 2 hasta llegar a su tamaño de adquisición, es decir con la mayor resolución posible.

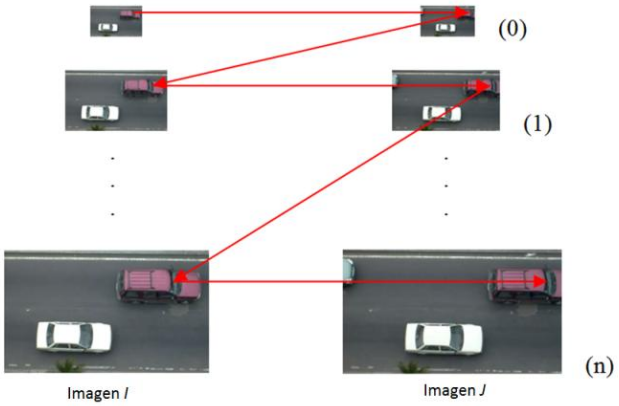


Figura 1. Forma piramidal del algoritmo de Lucas y Kanade.

El algoritmo piramidal estima primero el flujo óptico sobre el nivel 0 (ver Figura 1), entre las imágenes *I* y *J*, encontrando el desplazamiento d_0 de un punto específico sobre una imagen de baja resolución y una ventana pequeña. Luego ubica en la imagen *I* del nivel 1, el punto encontrado anteriormente, y estima de nuevo el flujo óptico entre *I* y *J* para el nivel 1 encontrado el desplazamiento d_1 sobre la misma ventana. Estos pasos se repiten hasta alcanzar el nivel *n* obteniendo el desplazamiento total del punto como:

1.3 ALGORITMO DE SHI Y TOMMASI

Shi y Tomassi definen en [13] un conjunto de puntos con características singulares que se basan principalmente en propiedades específicas de los bordes y texturas como lo es la intensidad espacial.
Para determinar un conjunto de "buenos puntos para seguir", se parte del desplazamiento de dichos puntos dentro en una ventana de tamaño fijo. Sea el vector el centro de la ventana en la imagen *I*. En la siguiente imagen *J*, el centro se encontrará en $A + d$, donde *d* es el desplazamiento del centro de la

ventana, y *A*, una variable que depende de la matriz de deformación.

La idea es obtener buenos puntos cuando se minimiza el error haciendo:

donde *w*( ) es una función de peso para el error.

Estos puntos resultan ser los adecuados para estimar el flujo óptico de una manera más confiable, debido a que los puntos encontrados por el algoritmo de Shi y Tommasi dan solución a la matriz A^TA.
De igual forma se obtienen las características que deben cumplir dichos puntos. Sean y los valores propios de la matriz A^TA para cierta región *R* de la imagen, entonces para ser un buen punto para seguir, se debe cumplir que:

Textura:

- , donde es un valor fijo establecido, lo cual garantiza que A^TA es invertible y la región no es ruidosa.

Bordes:

- — , lo que garantiza que A^TA está bien condicionada y no se presenta bordes en una sola dirección [13].

1.4 COVARIANZA

La covarianza es una medida de dispersión conjunta de dos variables estadísticas, también indica la medida del grado en que dos variables aleatorias se mueven hacia la misma dirección o en direcciones opuestas una con respecto a la otra. En otras palabras, si dos variables aleatorias generalmente se mueven hacia la misma dirección, se dirá que tienen una covarianza positiva. Si tienden a moverse en direcciones opuestas, se dirá que tienen una covarianza negativa. De igual forma, la covarianza se define como el valor que se espera de la media aritmética de los productos de las desviaciones de cada una de las variables respecto a sus medias respectivas, o también como el producto de dos variables aleatorias con respecto a sus valores medios. Matemáticamente, la covarianza está definida por la varianza como:

– –

donde es la media de x y es la media de *y* para las *n* muestras. A partir del cálculo de la covarianza por medio de la ecuación anterior es posible interpretarlos de la siguiente forma:

1. *cov* (*x*,*y*) > 0, implica que a grandes valores de x corresponden grandes valores de y.
2. *cov* (*x*,*y*) = 0, se interpreta como la no existencia de una relación entre las dos variables.
3. *cov* (*x*,*y*) < 0, implica que a grandes valores de x corresponden pequeños valores de y.

1.4.1 Matriz de covarianza

Es la matriz que representa la covarianza entre los datos de variables aleatorias. Esta matriz contiene la covarianza entre los elementos de dos vectores y es representada como .

Donde:

Y se representa de forma matricial como:

La matriz de covarianza cumple con las siguientes propiedades:

- , lo que implica que la diagonal de la matriz , son las varianzas.
- , lo que implica que la matriz es simétrica.
- Para *i* diferente de *j* , si y no están correlacionadas, implica que es una matriz diagonal y sus ejes son independientes en dicho caso.
- Es definida positiva.

La independencia de ejes implica que si *x* crece o decrece no depende de *y,* y viceversa.

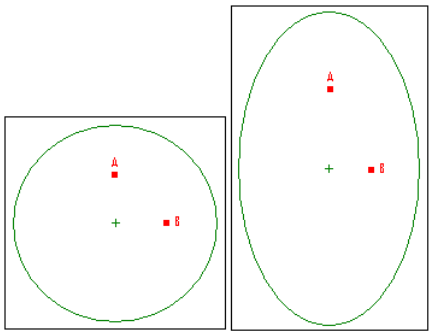
1.5 DISTANCIA DE MAHALANOBIS

La distancia de Mahalanobis es una medida de distancia que encuentra la similitud entre dos variables aleatorias multidimensionales y que responde a la covarianza o estiramiento del espacio de los datos. Debido a esto la distancia de Mahalanobis hace referencia a la dependencia de los ejes y se diferencia de la distancia Euclidiana en que esta sí tiene en cuenta la correlación entre las variables aleatorias. Esta distancia se encuentra definida con la siguiente ecuación:

(5)

En la Figura 2 se ilustra la distancia de Mahalanobis mediante las elipses de color verde según la varianza los ejes. En la Figura 2.a, la varianza para los dos ejes es la misma y no poseen correlación. En este caso la

distancia de Mahalanobis coincide con la distancia Euclidiana. En la Figura 2.b, se observa que para el eje *y*, donde se encuentra el punto A, la varianza es mayor por lo que la distancia de A es menor que la de B aunque la distancia Euclidiana de A y B sean iguales.



(a) (b)

Figura 2. Distancia de Mahalanobis. (a) Imagen con varianza igual para los dos ejes, (b)Imagen con varianza mayor en el eje *y*.

1.6 AGRUPAMIENTO O *CLUSTERING*

El agrupamiento o *clustering* es la clasificación de objetos dentro de diferentes grupos. Más precisamente es la partición de un conjunto de datos dentro de subgrupos (*clusters*), de tal forma que dentro de cada grupo (idealmente) se compartan rasgos comunes como por ejemplo la proximidad de acuerdo con una medida de distancia [16]. En la Figura 3.a, se muestran un conjunto de puntos en el plano *x*, *y*. En la Figura 3.b, se agrupan los puntos según su cercanía espacial dentro de 2 conjuntos.

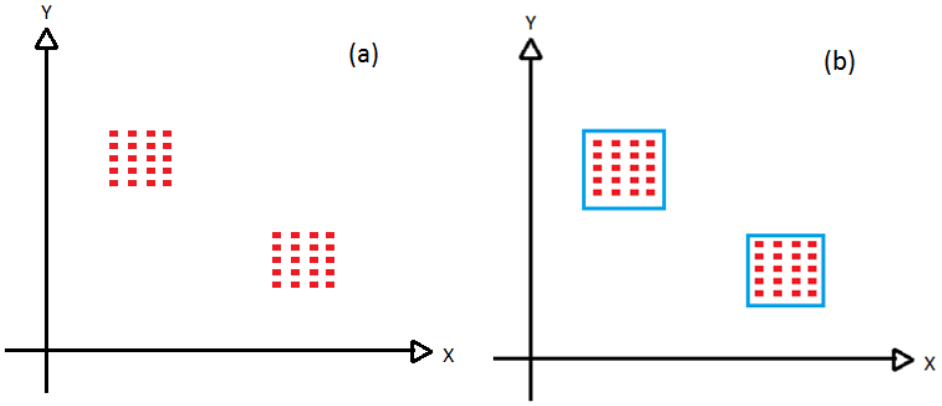


Figura 3. Agrupamiento espacial. (a) Puntos en el plano x,y, (b) Agrupación según su cercanía espacial.

1.6.1 *K-means*

K-means es un algoritmo creado por Hugo Steinhaus, el cual intenta encontrar la agrupación natural de un conjunto de datos. En este algoritmo se debe determinar primero el número de grupos que van a ser encontrados. *K-means* encuentra rápidamente los centros de los grupos de datos cercanos espacialmente.

Esta técnica es utilizada con frecuencia en agrupación de datos ya que es una versión mejorada de Gauss Mixture y es muy similar al algoritmo de Mean-Shift [14]. Este algoritmo no tiene en cuenta la covarianza, por esto se suelen normalizar los datos según la distancia requerida antes de ingresarlos.

El algoritmo se ejecuta de la siguiente forma:

1) Se ingresan los datos y número de grupos entre los cuales se van a dividir los datos.
2) Se asignan centros al azar, uno para cada grupo.
3) Se asocia a cada centro los puntos más cercanos.
4) Se mueven o reubican los centros según la media de la distancia del centro respecto al los puntos encontrados.
5) Se regresa al paso 3 hasta que converjan los centros (no se muevan los centros).

En la Figura 4, se muestra la forma en que trabaja el algoritmo de *k-means* siguiendo los pasos anteriores.

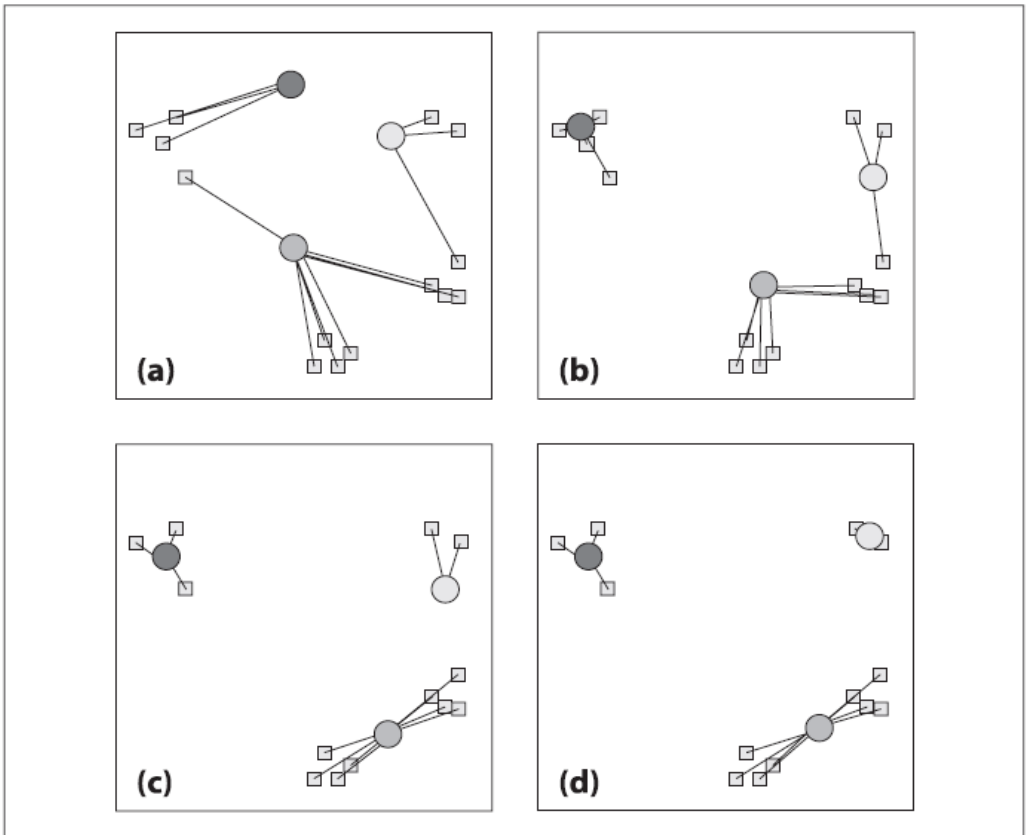


Figura 4. Algoritmo de *k-means* con dos iteraciones. (a) Se asocian a cada centro los puntos más cercanos, (b) Se desplazan los centros según la media de la distancia respecto al centro de los puntos encontrados, (c) Se repite el movimiento anterior, (d) Localización final de los centros. Imagen tomada de [14].

1.7 CORRECCIÓN DE PERSPECTIVA

Al tomar una imagen en 2-D con una cámara sobre una escena 3-D, se presenta una deformación geométrica conocida como efecto de perspectiva. Por ejemplo, si las formas que conocemos de un objeto son cuadriláteros rectangulares, debido a la distorsión del efecto de perspectiva, se presentarían como cuadriláteros deformados, (ver Figura 5). De igual forma, por el efecto de perspectiva se afecta la relación

entre tamaños de los objetos presentes a diferentes distancias de la cámara, los objetos cercanos a la cámara tienden a parecer grandes en tamaño con respecto a los objetos lejanos.

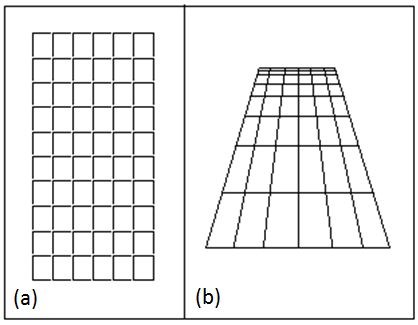


Figura 5. Deformación por perspectiva. (a) Plano mundo. (b) Plano real o de la imagen.

Con el fin de corregir estos problemas de perspectiva, se utiliza una transformación proyectiva. Si se tiene un vector de tres dimensiones dado por que representan un punto en el plano, las coordenadas de este punto en el plano están definidas como , donde y reciben el nombre de coordenadas homogéneas.

Ahora definimos proyectividad como una transformación invertible dada por h: $R^2 \rightarrow R^2$ de manera tal que una línea recta es transformada como otra línea recta.

donde H es una matriz de 3x3 no singular.

A partir de lo anterior definimos una trasformación proyectiva entre planos como una transformación lineal sobre 3 vectores homogéneos , representada por una matriz H.

La finalidad de la corrección de perspectiva es encontrar H tal que

(6)

donde y .

Dado que la matriz H y la matriz kH, con **,** son equivalentes (representan la misma transformación), se ajusta para tener un total de 8 incógnitas en la matriz H.

Por tal motivo:

Las dos ecuaciones se pueden reescribir de la siguiente manera:

Para resolver las 8 incógnitas se requiere la correspondencia entre 4 pares de puntos. Sean y conjuntos de puntos del plano de la imagen y el plano del plano de la escena, entonces se obtiene el sistema:

de donde se puede despejar *h*:

(7)

A partir del vector *h* se encuentra la matriz *H*, con la cual se transforman los puntos deseados [20].

1.8 DISTANCIA DE CHEBYSHEV

En matemáticas existen muchas maneras de definir la distancia y normas, las cuales asignan una longitud o tamaño a cada uno de los vectores en un espacio vectorial. La más común es la distancia Euclidiana o norma L2. Si es el origen, la distancia L2 se define como:

La norma *n* para un vector con origen en 0, se define como:

Cuando se conoce como la norma L infinito y se puede demostrar que:

Esta distancia se conoce como la distancia de Chebyshev ó Tchebychev. También se le dice distancia del tablero de ajedrez ya que cuando , con *k* constante, se forma un cuadrado de lado 2*k* en el espacio *x,y,* tal como son los movimientos permitidos para el rey en un tablero de ajedrez. Por otro lado, para la distancia Euclidiana, cuanto se tiene se forma un círculo de diámetro , (ver Figura 6). Estas normas se utilizan según criterios y propiedades que se estén utilizando.

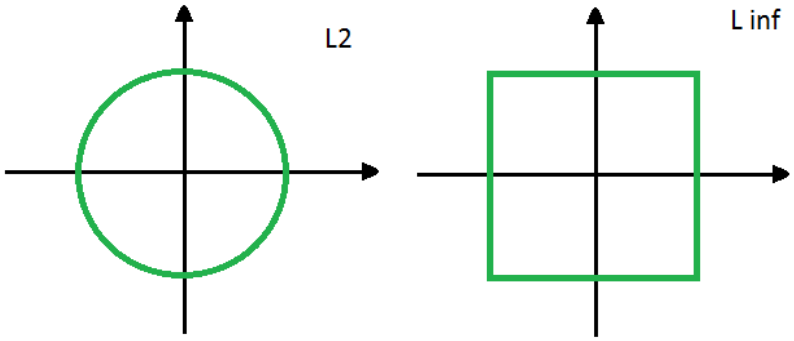


Figura 6. Norma L2 y L infinito.

1.9 HERRAMIENTAS TEÓRICAS

1.9.1 Factor de similitud

La probabilidad mide según su definición la frecuencia con la cual ocurre uno o varios eventos, al llevar a cabo un experimento aleatorio. Existen varios tipos de probabilidad tales como: la intuición, frecuencia de ocurrencia y probabilidad clásica. En el algoritmo se utilizó un factor de similitud que trata de medir que tanto se parecen las velocidades asociadas a los puntos característicos para concluir si un punto pertenece o no a un objeto de interés. Esto se realiza mediante la siguiente función:

(8)

donde es un factor que representa la desviación de los datos o que tan selectivo es la función para escoger los puntos de un objeto, y

La restricción dada por se realiza para poder implementar el algoritmo, teniendo en cuenta que si supera el número máximo de un flotante, se genera error. También se escogió ya que las pruebas para encontrar los centros reales no presentaban cambios significativos después de este valor.

La idea del factor es dar un peso según la velocidad y la desviación estándar a un conjunto de datos para encontrar su centro, o encontrar un umbral para seleccionar puntos que pertenezcan a un objeto de interés. Después de realizar varias pruebas con este factor, se encontró que la mejor solución es cuando dicho factor es mayor que uno. Esta solución se escogió evitando incluir puntos que no pertenecieran al objeto de interés y que tampoco eliminara los que si pertenecen.

2. DESCRIPCIÓN GENERAL Y ESPECIFICACIONES

En este trabajo de grado se presenta un algoritmo basado en la estimación del flujo óptico, que permite encontrar y segmentar objetos en movimiento a partir de una secuencia de imágenes. Este algoritmo fue implementado en el lenguaje de programación C++ utilizando las librerías de OpenCv 2.0.

El algoritmo está enfocado a la estimación de flujo de tráfico y fue evaluado mediante videos que presentan flujo vehicular o flujo de peatones que presentan flujo vehicular o flujo de peatones que cumplen con un conjunto de condiciones de contraste y de traslape de los objetos de interés. de traslape de los objetos de interés. Adicionalmente, el algoritmo estima la velocidad instantánea de los objetos detectados en función del flujo objetos detectados en función del flujo óptico estimado. Para los videos sobre los cuales se desea estimar la velocidad instantánea de cada objeto, la velocidad instantánea de cada objeto, es necesario contar con acceso a la escena para realizar mediciones de la misma con el fin de realizar la mediciones de la misma con el fin de realizar la corrección de perspectiva. En la

Figura 7, se presenta el diagrama de bloques del sistema.

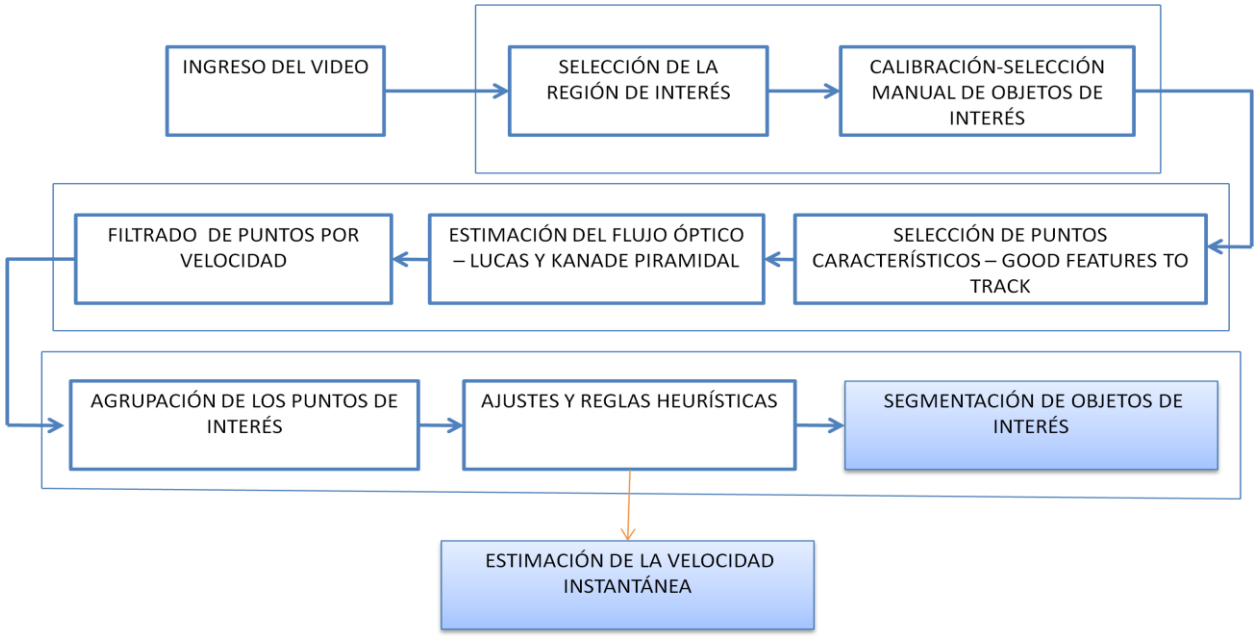


Figura 7. Diagrama de bloques general del sistema.

En este trabajo de grado, se utilizan videos tomados por el grupo de investigación *SIRP* del Departamento de Ingeniería Electrónica de la Pontificia Universidad Javeriana. Los videos utilizados se encuentran en el formato de archivo *AVI* y *WMV* y formato de compresión *Xvid*, *DivX* o *MPEG*. Estos videos fueron tomados entre las 9:00 am y las 5:00 pm, en una locación elevada que limita con la vía a ser estudiada. La cámara fue orientada de manera paralela a la vía tratando de minimizar los problemas de perspectiva de los objetos de interés [2] (ver Figura 8).

Figura 8. Videos tomados con la cámara de forma paralela a la vía.

Con el fin de estimar el flujo óptico sólo sobre una región que enmarque el movimiento de los objetos de interés, el usuario tiene la posibilidad de seleccionar manualmente la región sobre la cual se va a ejecutar el algoritmo, evitando procesar partes de la escena que no presentan información útil acerca del movimiento de los objetos, como es el caso de los andenes para videos con flujo vehicular. De igual forma se cuenta con información a priori de la forma de los objetos para su detección y segmentación, información adquirida durante la inicialización en cada ejecución del algoritmo.

La estimación del flujo óptico se realiza por medio de la versión piramidal del algoritmo de Lucas y Kanade [10], la cual estima el flujo óptico por medio de pirámides Gaussianas de una manera iterativa, para obtener una implementación más eficiente. Con el fin de trabajar con el algoritmo de flujo óptico, es necesario que los objetos de interés, cumplan con las siguientes condiciones:

- Presenten buen contraste respecto a la vía.
- No se traslapen entre sí.
- El brillo de un píxel perteneciente a un objeto de interés no cambie mientras es seguido *frame* a *frame*.
- Los objetos se mueven pocos píxeles de *frame* a *frame*.
- El movimiento de los píxeles que pertenecen a un objeto esté directamente relacionado con el movimiento del objeto en la escena.

En este algoritmo, el flujo óptico es estimado solo sobre puntos que sean buenos para seguir y que ofrezcan información útil de los objetos en movimiento. Para esto, se hace una selección de puntos característicos que permitan ser encontrados fácilmente entre dos *frames* consecutivos, en una secuencia de video mediante el algoritmo de Shi y Tomassi: *Good Features to Track* [13].

A partir de la estimación del flujo óptico, se eliminan los puntos que no se encuentran dentro de un rango de desplazamiento dado, ignorando de esta forma, puntos que representan ruido para nuestro sistema, como son los puntos asociados a las vías o a otros objetos que no son de interés.

La estimación de los centros de los objetos de interés se realiza utilizando el algoritmo de *k-means* y mediante la minimización de variables asociadas a la velocidad de dichos puntos.

Una vez los centros son encontrados, se realiza una clasificación de puntos, y se determinan cuales pertenecen al objeto de interés, mediante 2 algoritmos: uno basado en *k-means* y minimización de variables dependientes de la velocidad, y el otro algoritmo basada en la distancia de Chebychev y un factor de similitud.

Finalmente el algoritmo segmenta los objetos detectados en cada par de *frames* consecutivos mediante un rectángulo contenedor de los puntos asociados a cada objeto y algunas reglas heurísticas.

3. DESARROLLO TEÓRICO

En este capítulo se describe de forma detallada el desarrollo del algoritmo de detección de objetos móviles a partir del flujo óptico, y los parámetros y suposiciones utilizadas. Asimismo, se muestra de forma general la implementación por medio de las funciones de OpenCv. En la Figura 9, se presenta el diagrama de flujo del sistema.

El algoritmo consta de 3 grandes etapas: detección de movimiento en la escena, segmentación de objetos en movimiento, y ajustes y reglas heurísticas. Estas etapas se describen a continuación.

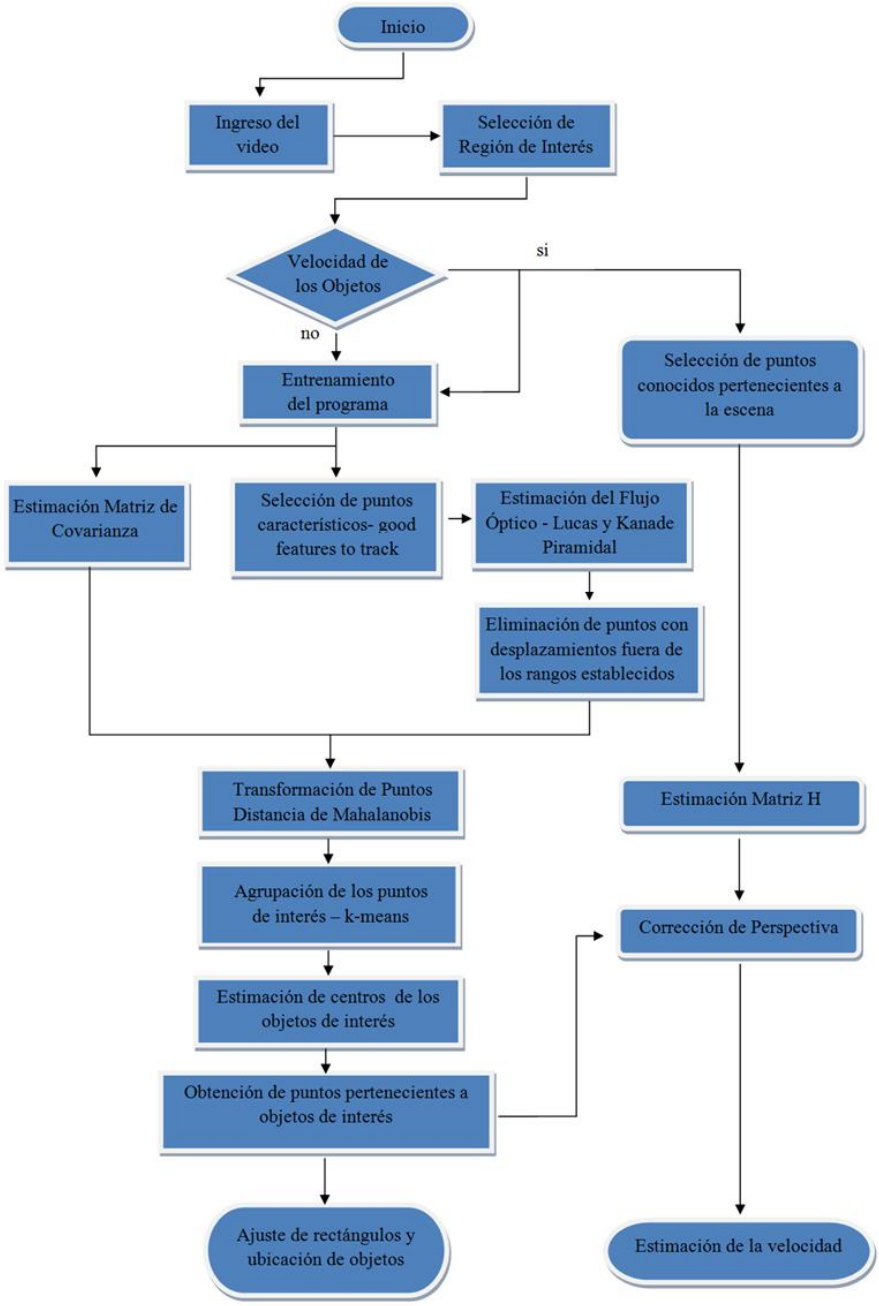


Figura 9. Diagrama de flujo del sistema.

3.1 DETECCIÓN DE MOVIMIENTO EN LA ESCENA

En este trabajo de grado se utiliza como herramienta de detección de objetos móviles dentro de una escena, el flujo óptico estimado en una secuencia de video, como una alternativa a la detección de objetos a partir del fondo de la escena. Esta alternativa, parte de la relación que se puede hacer entre el flujo óptico y el movimiento presente en una secuencia de imágenes bajo ciertas condiciones, como las planteadas en la Sección 2.

El algoritmo utiliza una implementación piramidal de Lucas y Kanade, la cual estima el flujo óptico sobre un conjunto de puntos dado. El algoritmo comienza con la selección de una región de interés (ROI) en el video sobre la cual se desea hacer la detección. El usuario tiene la posibilidad de seleccionar mediante el *mouse* la ROI sobre la cual se realizará la detección de los objetos, con el fin de evitar regiones de la imagen que no aporten información acerca del movimiento de los objetos de interés, o regiones que simplemente no sean de interés.

En la Figura 10 se puede apreciar un cuadro del video en el cual se selecciona una (ROI), región enmarcada por un rectángulo azul, sobre una parte significativa de la vía vehicular omitiendo regiones que no poseen información útil para la detección de los vehículos.

Figura 10. Selección de la región de interés por parte del usuario.

3.1.1 Estimación del flujo óptico

Mediante la estimación del flujo óptico, se puede asociar una velocidad a cada píxel en el *frame*, o que es lo mismo, el desplazamiento de un píxel entre un par de *frames*. Sin embargo, el método de estimación del flujo óptico propuesto por Lucas y Kanade calcula el flujo óptico sólo sobre un conjunto de puntos de interés. Este método asume que cada punto de interés se encuentra en una región *R*, que presenta un flujo óptico constante, y no sobre todos los píxeles pertenecientes a la imagen, como se indica en la Sección 1.2.

A partir de lo anterior, se busca estimar el flujo óptico sobre un conjunto de regiones en la imagen que aporten información del movimiento dentro de la escena. Para nuestro caso, cada región tiene unas

dimensiones fijas y es representada por un punto de interés ubicado en un píxel específico.

3.1.1.1 Puntos de interés sobre una malla uniforme

Buscando calcular el flujo óptico sobre puntos que ofrezcan información útil acerca de los movimientos presentes en la escena, sin que esto implique estimar el flujo óptico sobre una gran cantidad de píxeles, se computa el flujo óptico sobre una malla de puntos de interés, como se ilustra en la Figura 11(a). Esta malla posee un espaciamiento fijo entre los puntos y es actualizada cada cierto número de *frames*.

El flujo óptico estimado sobre cada uno de los puntos de la grilla ofrece poca información acerca del movimiento presente en la escena. Esto se debe a que muchos de los puntos sobre los cuales se realiza el seguimiento resultan difíciles de seguir como, por ejemplo, los que se encuentran sobre la carretera o en medio de una superficie que posee la misma textura y color. Para estos puntos, existen puntos cercanos que poseen valores similares de intensidad de gris, por lo que al buscarlos en el siguiente *frame*, es posible asociarlos a cualquier punto que cumpla con las mismas características, dando información errónea acerca del movimiento de los puntos, (ver Figura 11.(b)).

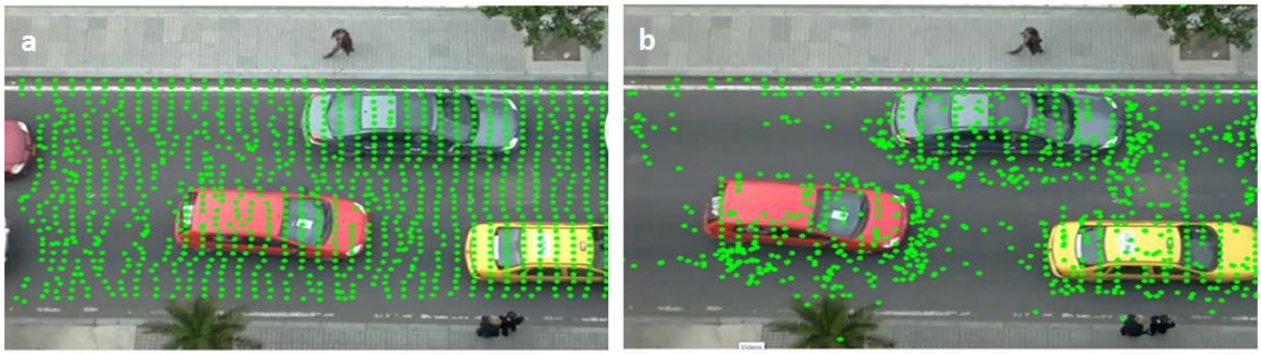


Figura 11. (a) Malla de puntos sobre la vía. (b) Flujo óptico sobre malla de puntos.

A partir de lo anterior se puede decir que, para obtener información útil acerca del movimiento dentro de una escena, se debe evitar estimar el flujo óptico sobre puntos que no tengan características discriminativas que permitan su seguimiento. Diferentes criterios se han propuesto para la selección de buenos puntos para seguir como los que se mencionan a continuación.

3.1.1.2 Selección de puntos de interés utilizando el algoritmo de Shi y Tommasi

Según lo anterior, el seguimiento de todos los puntos entre un *frame* y otro, por medio de LK, no resulta confiable. Como solución se utiliza el criterio de Shi y Tomasi para la selección de buenos puntos para seguir, el cual encuentra regiones que contienen bordes y esquinas dentro de la imagen, las cuales resultan fáciles de seguir. Esto ocurre cuando la matriz A^TA de la ecuación (4), posee valores propios grandes.

La implementación del criterio de Shi y Tommasi, se realiza por medio de la función `cvGoodFeaturesToTrack` en OpenCv 2.0. Esta función calcula primero el valor propio mínimo para cada píxel de la imagen de origen, usando la función `cvCornerMinEigenVal` y almacena en la variable `eig_image`. Después, son rechazadas las esquinas con valor propio menor a `quality_level•max(eig_image(x,y))`. Finalmente, la función asegura que todas las esquinas

detectadas se encuentran separadas unas de otras por una distancia mayor a `min_distance`, eliminando las que se encuentran muy cerca y dándole prioridad a las esquinas más fuertes. Los mejores resultados se obtuvieron utilizando `quality_level` = 0.01 y `min_distance` = 10.

La ejecución de la función `cvGoodFeaturesToTrack`, da como resultado las coordenadas de un conjunto de puntos denominados puntos de interés, los cuales, en su mayoría pertenecen a esquinas y contornos de los objetos presentes en la escena como se aprecia en la Figura 12. La cantidad de puntos seleccionados por la función no puede superar a MAX_COUNT, variable que se deja fija en nuestro algoritmo y es igual a 500.

Figura 12. Puntos característicos seleccionados de acuerdo a [13]

3.1.1.3 Estimación del flujo óptico utilizando el algoritmo de Lucas y Kanade piramidal

Como se mencionó anteriormente, el flujo óptico es estimado para los puntos encontrados mediante del criterio de Shi y Tommasi, utilizando una versión piramidal de Lucas y Kanade. Para la implementación, se utiliza la función `cvCalcOpticalFlowPyrLK` de Opencv 2.0, que ejecuta de forma iterativa la versión piramidal de LK. Esta función estima el flujo óptico sobre los puntos de interés, encontrados por medio de `cvGoodFeaturesToTrack` para un *frame n*-1, y da como resultado las coordenadas finales del flujo óptico para dichos puntos en el *frame n* del video. El algoritmo utiliza pirámides de máximo 3 niveles, ventanas fijas de 10 x 10 píxeles y máximo 20 iteraciones.

Al ejecutar el algoritmo sobre los puntos de interés, la mayoría de los desplazamientos obtenidos (flujo óptico diferente de cero), se pueden asociar al movimiento de objetos dentro de la escena, como se muestra en la Figura 13.

Figura 13. Implementación del algoritmo de Lucas y Kanade piramidal.

Los puntos de interés, se actualizan cada 10 *frames,* por lo que durante ese tiempo, se estima el flujo óptico sobre los mismo puntos.

3.2 SEGMENTACIÓN DE OBJETOS EN MOVIMIENTO

Como resultado del proceso anterior, se obtienen puntos distribuidos sobre toda la imagen, en su mayoría, pertenecientes a objetos en movimiento. Estos puntos están distribuidos dentro de nubes, como se ilustra en la Figura 14. La forma de las nubes depende del objeto en particular, para vehículos se espera una forma diferente que para peatones. Esta información se puede determinar a partir de la matriz de covarianza de los puntos pertenecientes al objeto de interés.

Partiendo de lo anterior, para realizar la segmentación, se utiliza información a priori de los objetos de interés. Por este motivo, se realiza un periodo de entrenamiento en donde se determinan ciertas características asociadas a la distribución de los puntos pertenecientes a los objetos. Para la segmentación se utiliza un algoritmo de agrupamiento de puntos mediante la distancia espacial entre ellos debido a su correspondencia con los objetos de interés.

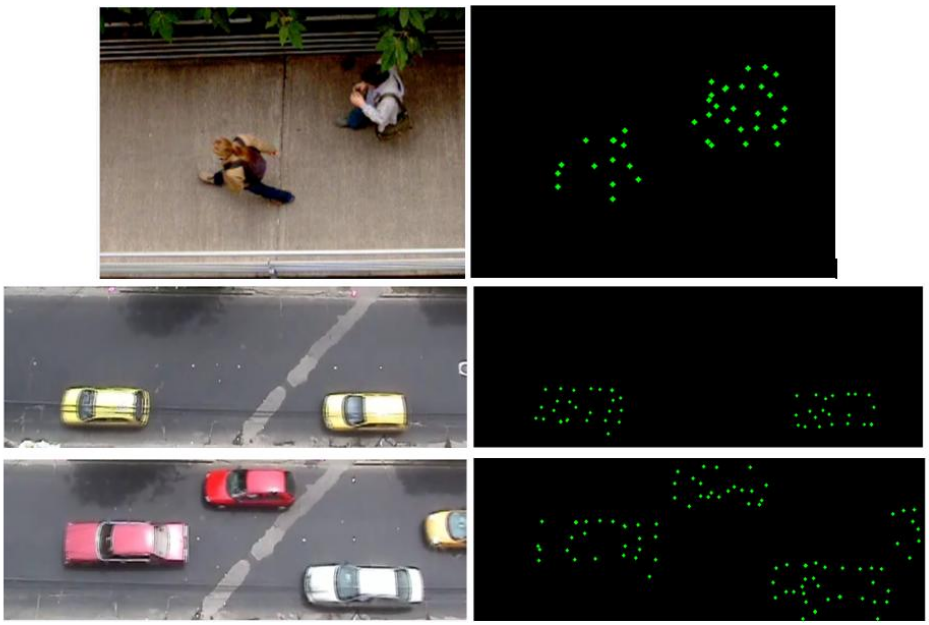


Figura 14. Puntos de interés en movimiento.

3.2.1 Inicialización

Al inicio del programa, una vez delimitada la ROI, se realiza el entrenamiento del algoritmo a partir de la selección de un número de objetos de interés presentes en la escena. Los objetos de ejemplos deben estar en movimiento y son segmentados por el usuario, como se ilustra en la Figura 15. A partir de estos, se obtiene información a priori de los objetos de interés presentes en el video, como los valores esperados del tamaño y la forma de los objetos, el promedio y la desviación estándar de la velocidad y el número de puntos asociados a los objetos. Para esto, el usuario enmarca mediante un rectángulo los objetos de interés, y a partir de este, se toman los puntos contenidos para obtener la información requerida.

Figura 15. Selección de objetos de interés para inicialización.

3.2.2 Filtro de puntos de interés por velocidad

A partir de los vectores de desplazamiento estimados con el algoritmo de Lucas y Kanade piramidal, se eliminan los puntos que tienen desplazamientos asociados fuera de los rangos establecidos, determinados por la velocidad promedio y la desviación estándar computados en la inicialización, utilizando el siguiente criterio:

donde es la magnitud del desplazamiento para cada uno de los puntos de interés, es la velocidad o desplazamiento promedio estimado en la inicialización y su desviación estándar, suponiendo que las velocidades asociadas a los puntos de interés presentan una distribución normal.

Este filtro puede eliminar el ruido de puntos pertenecientes a la vía, de algunos objetos que no son de interés y sus sombras, y de puntos que se desplazan mucho entre dos cuadros debido a errores en la estimación del flujo óptico. En la Figura 16, se muestra una imagen en la cual se ha aplicado la

eliminación de ruido. La gran mayoría de los puntos restantes pertenecen a los objetos de interés dentro de la escena.

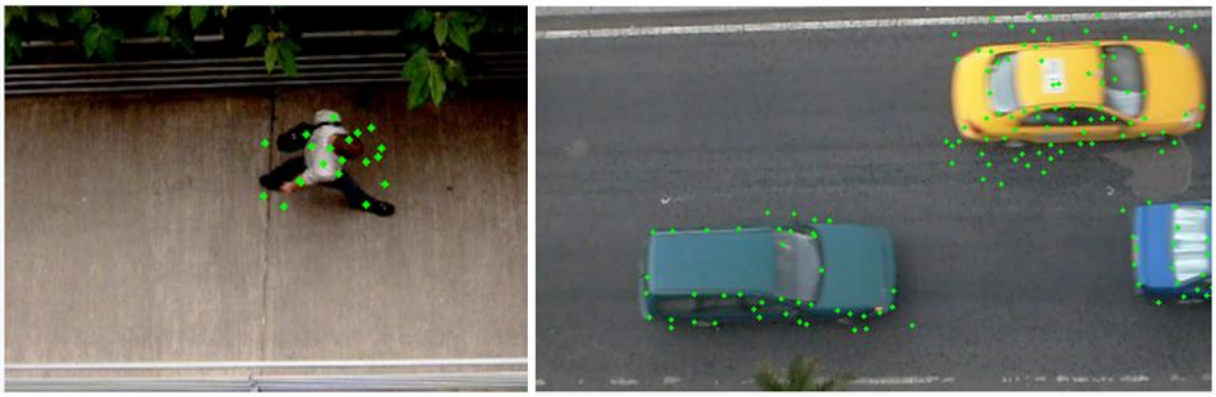


Figura 16. Puntos característicos de objetos en movimiento a partir de la eliminación de ruido.

3.2.3 Agrupación de puntos de interés por medio del algoritmo de *k-means*

Los puntos de interés pertenecientes a un mismo objeto, se encuentran cercanos entre sí, motivo por el cual la ubicación relativa de esta, puede ser utilizada para la segmentación de los objetos.

El algoritmo *k-means* permite agrupar puntos utilizando como criterio una medida de distancia apropiada. En la Figura 17, se muestra el diagrama de bloques de la agrupación de puntos de interés por medio del algoritmo de *k-means*.

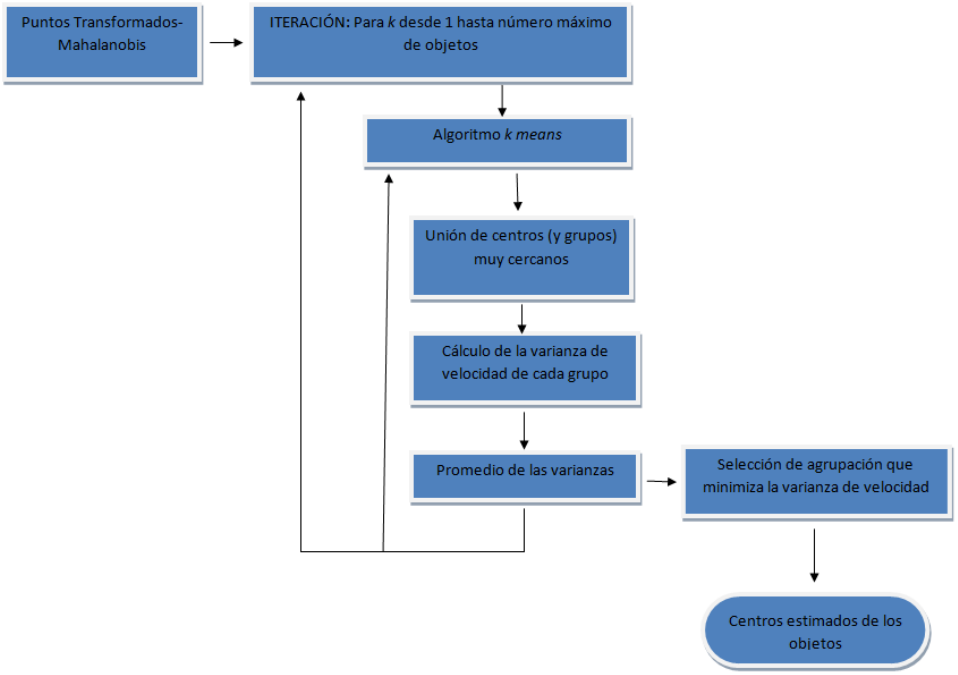


Figura 17. Diagrama de bloques: Agrupación de objetos característicos por medio del algoritmo de *k-means*.

La distancia de Mahalanobis, en nuestro caso, se utiliza como una transformación con el fin de tener en cuenta la forma de los objetos.
Esta transformación permite la agrupación basada en distancia Euclidiana para conjuntos de puntos que presentan una covarianza correlacionada y una varianza diferente en sus ejes, es decir que no crezcan de la misma manera para cada uno de sus ejes.

Por ejemplo para el caso de puntos que representan vehículos (ver Figura 14), al aplicar la transformación de puntos utilizando la distancia de Mahalanobis, se obtienen puntos distribuidos de igual manera tanto horizontal como verticalmente (ver Figura 18).

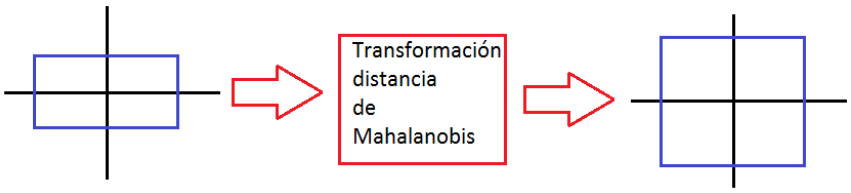


Figura 18. Transformación utilizando la distancia de Mahalanobis.

Debido a que la matriz de covarianza es singular, ya que es un caso especial de la matriz hermitiana o hermítica, se puede demostrar que a partir de la ecuación (5) es posible realizar la transformación de Mahalanobis mediante la siguiente ecuación para un punto (x,y):

donde y son los puntos transformados.

Para estimar se tiene en cuenta que una matriz hermitiana tiene sus valores propios reales. Además, según el teorema espectral de dimensión finita esta matriz se puede diagonalizar. Así mismo, es fácil verificar que si:

entonces:

Teniendo en cuenta lo anterior se diagonaliza por medio de la transformación lineal:

donde es la matriz de vectores propios y es su respectiva matriz diagonal. Despues se hace

con k = - –, y finalmente se trasforma a su espacio inicial haciendo . [19]

Los puntos transformados mediante la distancia de Mahalanobis son agrupados utilizando el algoritmo de *k-means* con distancia Euclidiana, por medio de la función `cvKMeans2` de OpenCv 2.0. Una de las ventajas de utilizar este algoritmo es su simplicidad y velocidad que permite correrlo sobre un conjunto grande de datos. Este algoritmo recibe el número de grupos (k), además de los puntos sobre los cuales se realizará la agrupación, y da como resultado las etiquetas para cada punto según el grupo al que pertenecen, y los centros encontrados.

Debido a que no se conocen cuantos objetos de interés se encuentran en la escena, se realizan diferentes iteraciones con k variando desde cero hasta `MAX_OBJ`. En la implementación del algoritmo se utilizo `MAX_OBJ`=10.

Muchas de las soluciones obtenidas, presentan dos o más grupos contenidos en un objeto de interés debido a que se k es mayor al número de objetos de interés presentes en la escena (ver Figura 19).

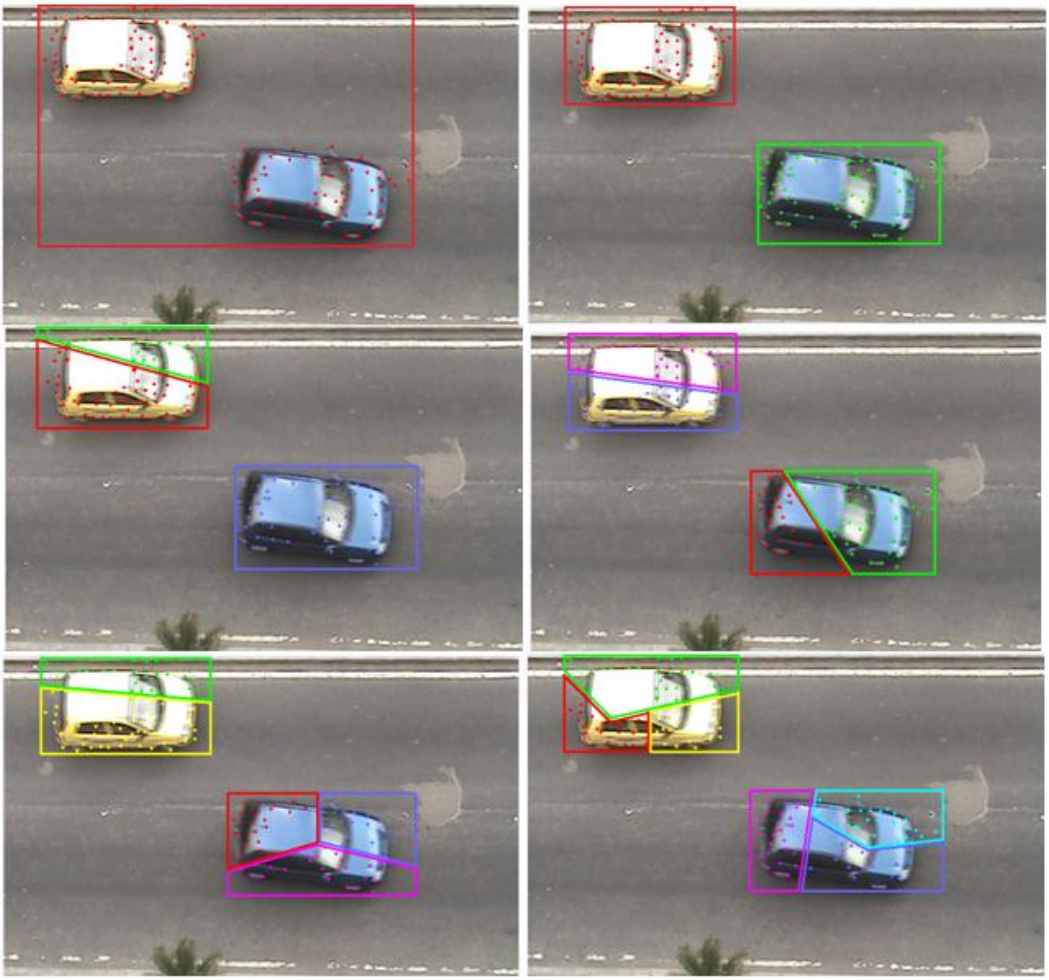


Figura 19. Puntos agrupados por *k-means* para k variando desde 1 hasta 6.

Cuando dos o más grupos pertenecen a un mismo objeto de interés, presentan distancias entre sus centros menores al tamaño esperado de dicho objeto. Basándonos en la condición presentada para los videos que no permite que se presenten traslapes entre los objetos, se realizó la unión de de estos grupos, como se aprecia en la Figura 20. En la Figura 20.a, se muestra la solución obtenida para $k = 3$ en una secuencia de *frames* que presenta dos vehículos en movimiento. A cada grupo encontrado se le asocia un rectángulo con las dimensiones del tamaño esperado del vehículo. Como se puede apreciar, los dos rectángulos asociados al mismo vehículo se intersecan. Si esta solución pertenecieran a dos objetos diferentes, se estaría incumpliendo con la condición de traslape entre objetos de interés, por lo que se asume que pertenecen a un mismo objeto. Finalmente se unen estos grupos como se muestra en la Figura 20.b.

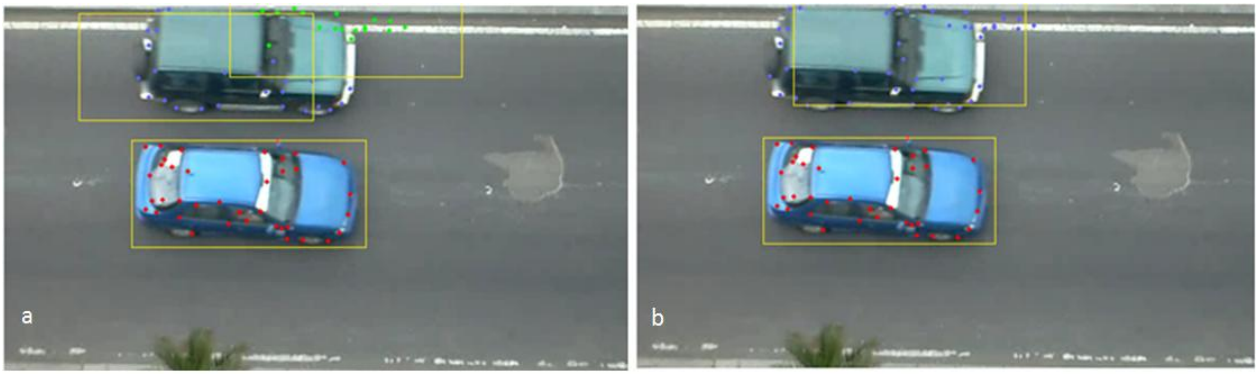


Figura 20. (a) Solución de k-means para k=3, (b) unión de grupos.

Cabe recordar que una de las condiciones para estimar el flujo óptico es que el movimiento de los píxeles que pertenecen a un objeto, está directamente relacionado con el movimiento del objeto; ésto implica que los puntos vecinos en la escena que pertenezcan a la misma superficie poseen un movimiento similar y se proyectan a puntos cercanos en el plano de la imagen [14].

Partiendo de lo anterior, utilizamos como uno de los criterios para determinar el número de objetos en la escena, la varianza de velocidad de cada grupo. Mediante este procedimiento buscamos grupos de puntos coherentes según la velocidad, a partir de una primera agrupación espacial realizada con el algoritmo de *k-means*. Para esto determinamos la varianza de velocidad de cada uno de los grupos obtenidos para los diferentes k y se determina su media. Si para un k dado se obtiene una varianza promedio menor que para el resto, esto implica que las velocidades asociadas a los puntos pertenecientes a cada grupo tienen valores similares de velocidad y espacialmente se encuentran cercanos según la distancia de Mahalanobis computada en la inicialización. La solución que minimice la varianza promedio de velocidad se almacena como posible solución de agrupación de puntos.

Por ejemplo, en la Figura 21 se muestran las mejores soluciones para k desde 1 hasta 6. En esta figura, cada grupo posee un color específico, además, se realizó una agrupación manual de los puntos de cada grupo mediante rectángulos con el fin de obtener una mejor visualización. Para este caso no hubo necesidad de realizar la unión de centros debido al número de objetos de interés en movimiento presentes en la escena. Para cada solución, se calculó la varianza de velocidad de cada grupo encontrado y se promedió. La solución que presentó la menor varianza promedio de velocidad fue para $k = 6$. En este caso esta solución se guardaría como posible agrupación y pasaría al siguiente bloque.

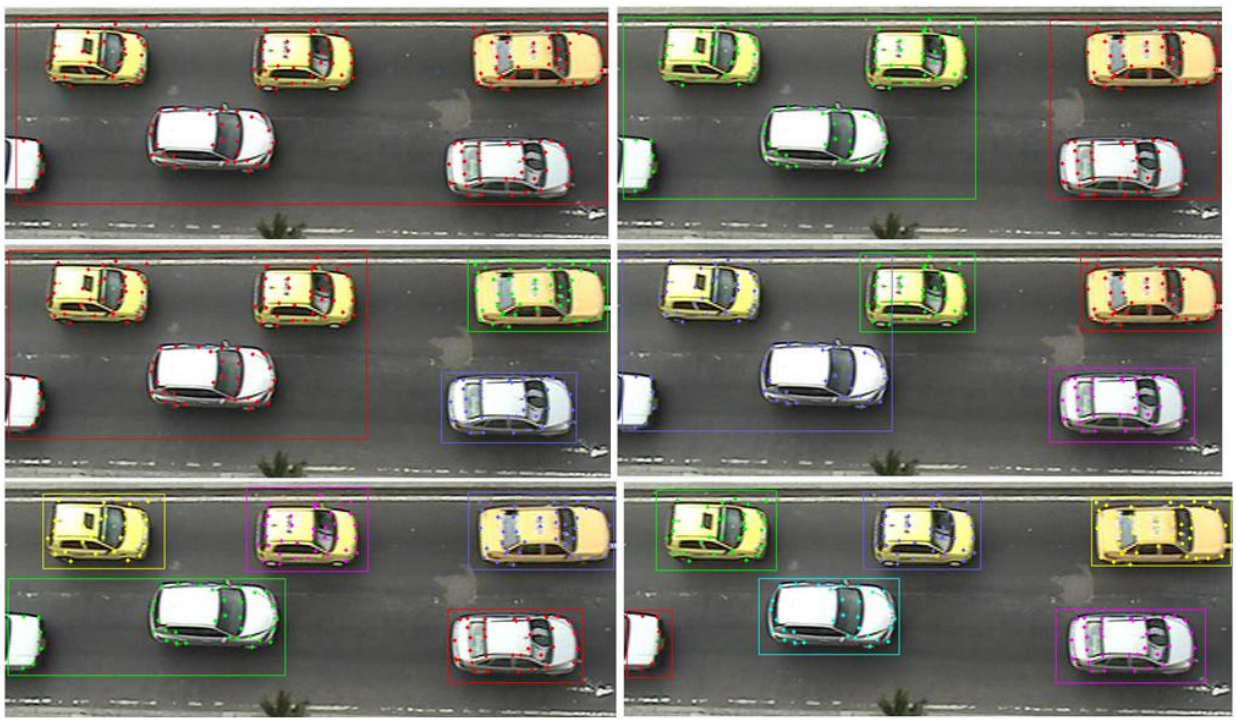


Figura 21. Mejores soluciones del algoritmo *k-means* para *k* desde 1 hasta 6.

En la Figura 22 se muestran algunos histogramas de la velocidad de puntos asociados a objetos de interés. En estos histogramas se puede apreciar que aunque los objetos presentan diversas velocidades, éstas se encuentran cercanas y suelen estar concentradas en uno o dos valores específicos. Las velocidades que se encuentran fuera de estas columnas, se podrían considerar eventualmente como ruido. Las velocidades presentadas se encuentran en valores de píxeles/*frame*.

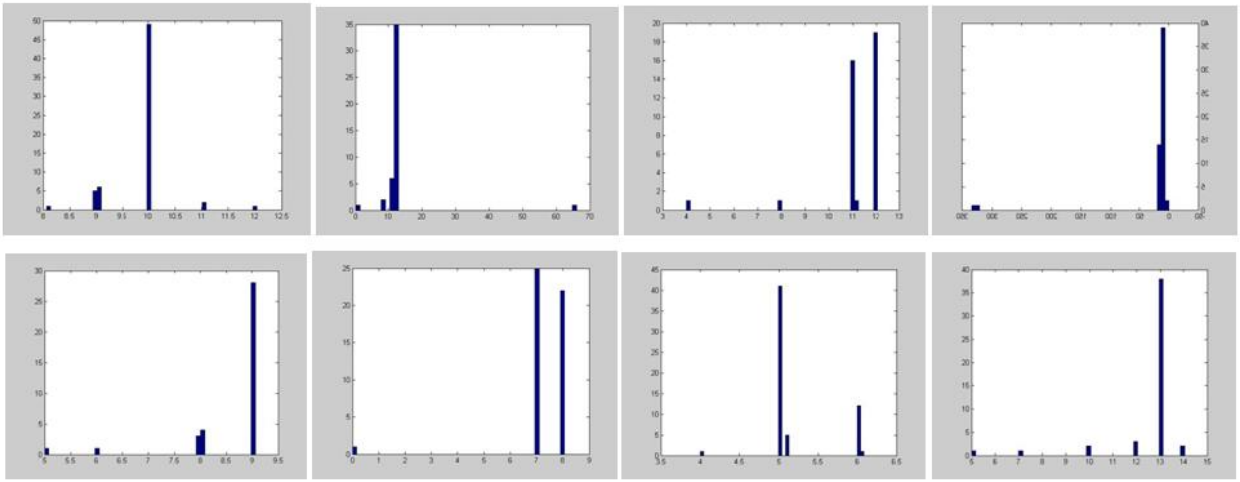


Figura 22. Histogramas de velocidad para objetos de interés.

Una de las características del algoritmo de *k-means* es que no arroja el mismo resultado para cada iteración. Debido a esto, para cada número *k* de objetos, se realizan cierto número de iteraciones con el fin de encontrar la mejor solución. Nuestro algoritmo utiliza 3 iteraciones.

3.3 AJUSTES Y REGLAS HEURÍSTICAS

3.3.1 Ajuste de centros con factor de similitud y probabilidad

La finalidad de este ajuste fue encontrar el centro real del objeto de interés mediante un peso asignado a cada punto según su distancia y velocidad correspondiente. Para esto, se parte de los puntos y el centro del objeto de interés encontrados en los pasos anteriores, y se busca ajustar los centros mediante una estrategia basada en EM (*Expectation-Maximization*) [17]. Este algoritmo busca encontrar el modelo que mejor represente una serie datos conocidos, mediante el ajuste de diferentes parámetros característicos. Este es un algoritmo de iteración que finaliza una vez se obtenga un modelo que reproduzca de forma similar los datos ingresados.

En la Figura 23, se muestra el diagrama de flujo de los ajustes de centros con factor de similitud y probabilidad.

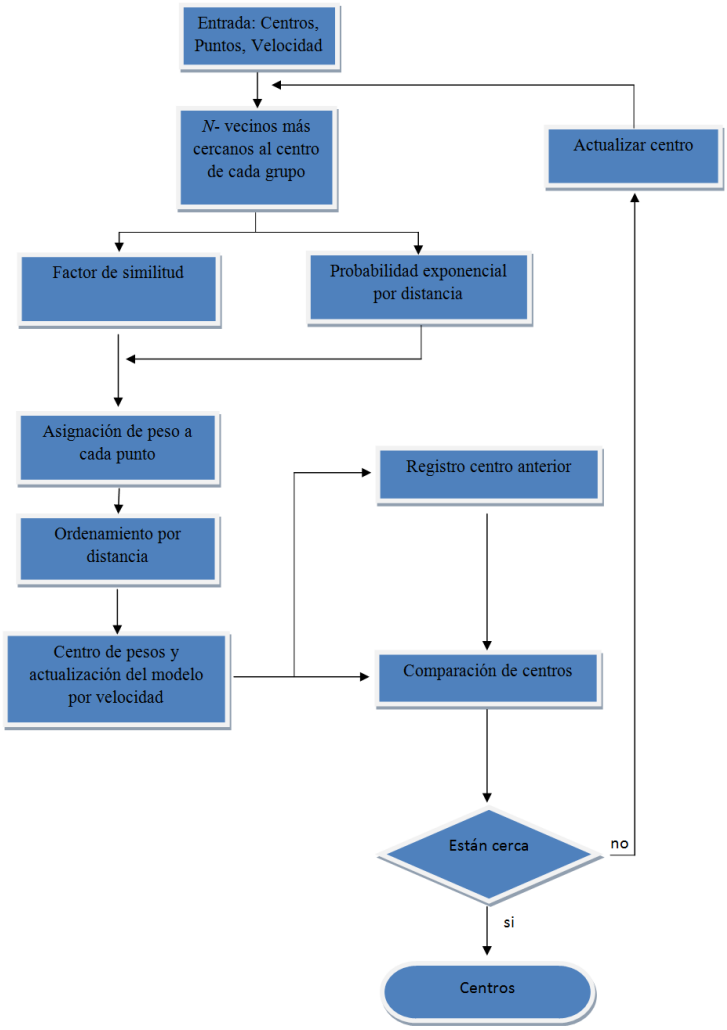


Figura 23. Diagrama de flujo: Ajuste de centros con factor de similitud y probabilidad.

Basados en lo anterior desarrollamos un algoritmo para tratar de encontrar el centro que mejor se ajuste al centro del objeto, utilizando la información de la velocidad y la distancia de los puntos más cercanos al centro encontrado mediante el algoritmo de *k-means*.

Para esto se parte de la suposición de que el centro que se encontró anteriormente está ubicado dentro del objeto de interés. A partir de lo anterior, el algoritmo ejecuta los siguientes pasos:

I. Se encuentra la velocidad y desviación promedio de los cinco puntos más cercanos al centro obtenido por medio de la agrupación de *k-means*, utilizando la distancia Euclidiana.

II. Se define un de factor de similitud en función de la velocidad de cada uno de los puntos sobre los que se estimó el flujo óptico. De igual forma se define una función de densidad de probabilidad exponencial por partes que tiene como parámetro la distancia de los puntos con respecto al centro.

III. Se asigna un peso a cada punto por medio del factor de similitud y la función de probabilidad, dándole un peso a los puntos que se encuentran dentro del tamaño esperado ( ), y otro peso a los que se encuentran por fuera mediante la siguiente función:

donde es el factor de similitud de la ecuación (8).

IV. Se ordenan los puntos según el valor absoluto de la distancia.

V. Se encuentra un nuevo centro haciendo una ponderación entre todos los puntos de interés, teniendo en cuenta el peso de cada uno de ellos. Esto se hace por medio de la siguientes ecuaciones:

donde y son las coordenadas del nuevo centro, es la distancia de punto *i* con respecto a su origen y es el peso del punto *i* definido en III.

VI. Se encuentra la diferencia entre el centro actual y el anterior. Si la diferencia es mayor a un mínimo se vuelve a repetir el algoritmo.

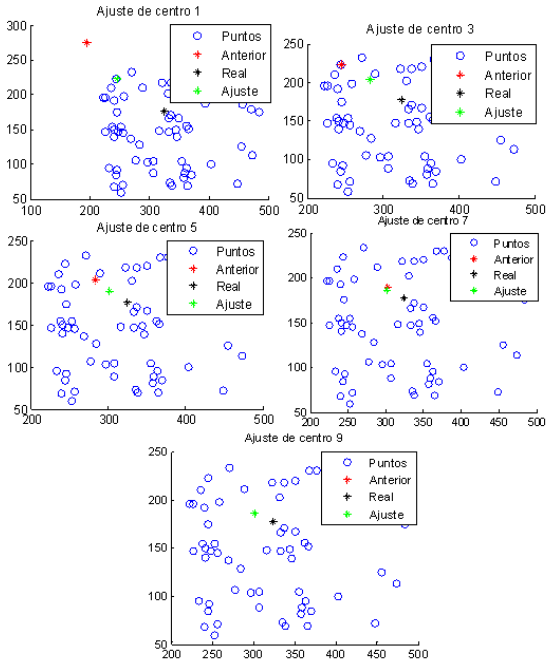


Figura 24. 5 iteraciones del algoritmo basado en EM. Los círculos azules representan los puntos asociados al vehículo después de la unión de centros. Los asteriscos rojos representan los centros que se guardan en el registro. El asterisco verde representa el centro después de realizar el ajuste y el asterisco negro son los centros del rectángulo que contiene el vehículo.

En la Figura 24 se muestran 5 iteraciones del algoritmo basado en EM y el factor de similitud implementado en Matlab® con puntos y centros de un vehículo real. Para la primera iteración, este centro es encontrado de manera aleatoria dentro de los puntos del vehículo.

Para la primera iteración (Ajuste de centro 1) se selecciona al azar un centro representado por el asterisco rojo, el cual se encuentra un poco alejado de la nube de puntos. Después de realizar el ajuste se encuentra el centro representado por el asterisco verde. En la iteración 5, el centro ajustado termina convergiendo, es decir, que la distancia con el centro estimado en la iteración anterior, es menor a un valor establecido, en este caso de 2 píxeles. Como suponemos que el centro que entra a este algoritmo esta en el objeto de interes, lo primero que hacemos es encontrar los 5 puntos mas cercanos como se muestra en la Figura 25. Luego se encuentra el promedio de la velocidad de estos puntos los cuales van a servir como referencia para encontrar más puntos pertenecientes al objeto de interes.

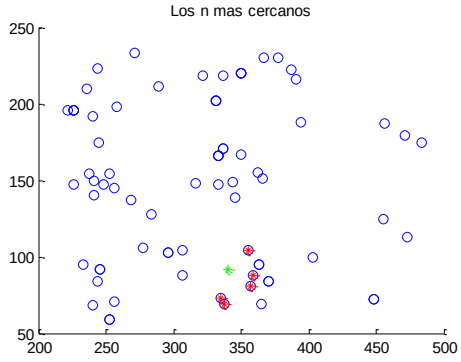


Figura 25. 5 puntos más cercanos al centro.

En la Figura 25, se muestra un ejemplo con puntos reales de un vehículo, en el cual se encuentran los 5 puntos más cercanos al centro por medio de un algoritmo implementado en Matlab ®. Los puntos de flujo óptico son círculos azules, el asterisco verde es un centro dado al azar y los asteriscos rojos son los 5 puntos más cercanos.

También se implementó el algoritmo utilizando una función de densidad Gaussiana en vez del factor de similitud y se encontró que si los puntos se seleccionan por la velocidad como distribución Gaussiana, no se obtienen buenos resultados, debido que se pretendía comparar la velocidad promedio con la velocidad de cada punto, y al analizar la probabilidad Gaussiana se encontró que cambian drásticamente los valores respecto a la desviación estándar. Esto ocasionó problemas para encontrar el umbral de pertenencia de cada punto ya que debería ser función de la desviación, por lo cual se tuvo que cambiar esta estrategia teniendo en cuenta la idea inicial.

Los resultados obtenidos al implementar los algoritmos no tuvieron buenos resultados, porque al intentar promediar los puntos con un peso determinado suponemos que la distribución de los puntos es simétrica respecto al centro con un peso constante, o el peso de los puntos es tal que los hace simétricos respecto al centro. Esto no se cumple para los objetos de interés, porque esperamos que todos los puntos se muevan a la misma velocidad del objeto independientemente de su centro. Además, se encontró que efectivamente los puntos obtenidos de los objetos no cumplen con esta simetría ya que lo que se busca son buenos puntos para seguir independientemente de la posición.

Además, cuando se encuentran dos o más objetos que se mueven a la misma velocidad y están muy cercanos, el objeto que tenga más puntos, o si su peso es muy grande, atrae los otros centros uniéndolos como un solo objeto.

3.3.2 Agrupación de puntos utilizando distancia de Chebyshev y factor de similitud

Como alternativa a la agrupación de puntos característicos realizada mediante el algoritmo de *k-means*, se realizó la agrupación de puntos utilizando un filtro de puntos lejanos con la distancia de Chebyshev y un factor de similitud. En la Figura 26, se muestra el diagrama de flujo para esta agrupación de puntos.

Basándonos en el tamaño esperado del objeto, este nuevo algoritmo de agrupación se aplica a los puntos sobre los cuales se computó el flujo óptico. A partir de estos puntos se escogen los que se encuentran a menos de 1.5 veces el tamaño esperado del centro obtenido de *k-means* y agrupación de centros. Luego estos puntos se ordenan según su distancia respecto al centro por medio de la distancia de Chebyshev, y se eliminan los puntos que se encuentra después de un espacio entre puntos de cuatro veces el tamaño esperado ya sea de los valores sobre el eje *x* como en el eje *y*.

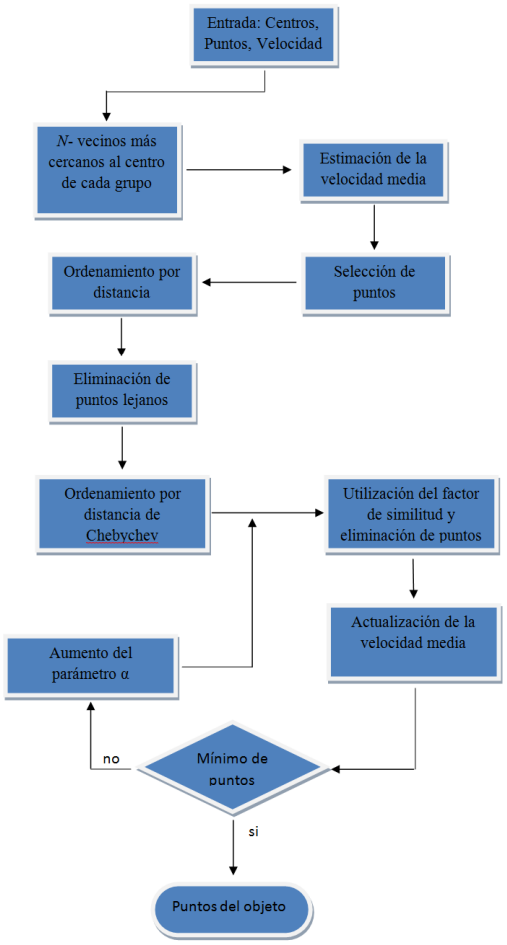


Figura 26. Diagrama de flujo: Agrupación de puntos utilizando distancia de Chebychev y factor de similitud.

Después de seleccionar los puntos que cumplen con las condiciones anteriores se realizan los siguientes pasos:

- Se encuentra la velocidad promedio de los cinco puntos más cercanos al centro, obtenidos por medio de la agrupación de *k-means*.

- Se agrupan nuevamente los puntos de interés mediante el factor de similitud. Para determinar por primera vez se utiliza el siguiente criterio:

donde es la deviación estándar de los valores de velocidad asociados a los objetos de interés y estimada en la inicialización. Cuando el resultado de aplicar el factor de similitud sobre un punto es mayor que 1, se selecciona dicho punto como característico del objeto asociado al centro sobre el cual se está trabajando

- Si los puntos asociados a un centro no son de al menos 1/2 del número de puntos esperados, el algoritmo aumenta en 1 el valor de α con el fin de permitir asociar un mayor número de puntos. Esto se realiza hasta que los puntos obtenidos sean mayores a la mitad de los puntos esperados, o hasta que α sea mayor a 10.

En la inicialización obtenemos un tamaño esperado, lo que nos indica que el objeto de interés está en cierto rango de tamaño dado por el tamaño esperado. Por tal motivo y suponiendo que el centro se encuentra dentro del objeto de interés, se eliminaron los puntos que se encuentran a una distancia mayor que 1.5 veces el tamaño esperado, ya que en el peor de los casos el centro puede quedar en un extremo.

Como un objeto tiene varios puntos de interés, ubicados en los mejores lugares de seguimiento que generalmente son esquinas, se pueden presentar errores como los que se observan en la Figura 27, en los que se detectan pocos puntos lejanos que no pertenecen al objeto de interés. Como solución a este error se propuso eliminar grupos de puntos en los cuales la distancia del punto más cercano al centro de este grupo se encentre a un cuarto de la distancia esperada del punto más lejano del grupo que contenga al centro.

Para implementar el algoritmo se ordenan los puntos según sus coordenadas *x* y *y* respecto al centro, separando tanto positivas como negativas. Después teniendo en cuenta el orden, se encuentra la diferencia entre dos puntos consecutivos. Si se encuentra una diferencia mayor a la mínima establecida, se eliminan los puntos que se encuentren después de dicho punto.

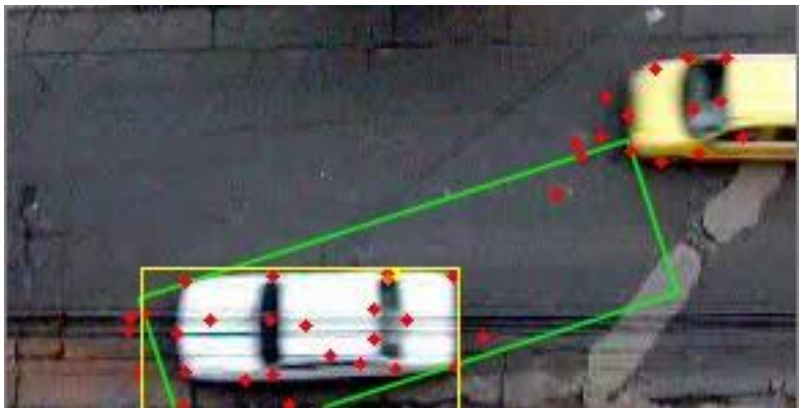


Figura 27. Agrupación de puntos lejanos. El rectángulo verde no tiene la restricción de distancia, mientras el amarillo sí.

En la Figura 27, se muestran un cuadro del video corregido por el algoritmo explicado anteriormente. En la escena hay 2 vehículos agrupados como uno solo. Este error debe a la agrupación de puntos lejanos. El rectángulo verde es la solución para el algoritmo de agrupación de puntos de interés por medio de *k-means* y el rectángulo amarillo es la solución para el algoritmo de agrupación de puntos utilizando distancia de Chebyshev y factor de similitud.

En la Figura 28, se observan 2 vehículos encerrados por medio de rectángulos. En verde se dibuja el mínimo rectángulo contenedor para la solución obtenida con *k-means* y en color blanco se dibuja el mínimo rectángulo contenedor para la solución obtenida después de hacer la eliminación por medio del factor de similitud. Como se observa en la Figura 28, el algoritmo basado en *k-means* agrupa más puntos que la otra solución tendiendo a segmentar completamente los vehículos.

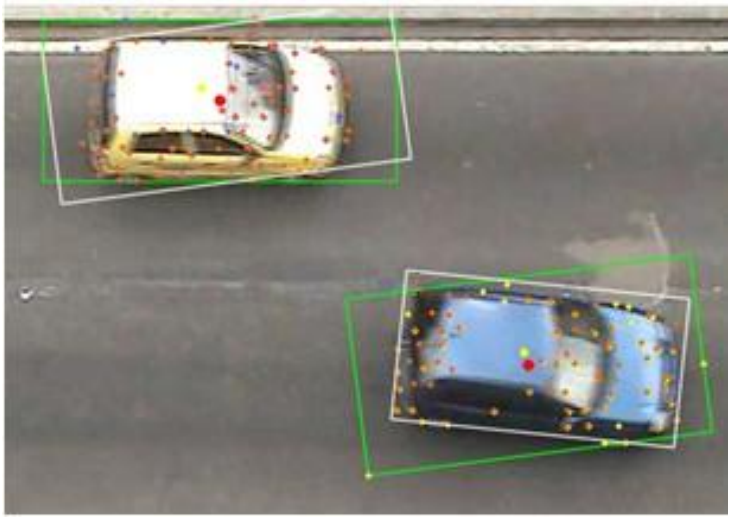


Figura 28. Corrección mediante eliminación de puntos lejanos.

3.3.3 Ajustes de rectángulos, ubicación de objetos y reglas heurísticas

Una vez determinados los puntos pertenecientes a los objetos de interés presentes en la escena, se ajusta el mínimo rectángulo contenedor para dichos puntos mediante la función `cvMinAreaRect2`. Este rectángulo está directamente relacionado con el objeto de interés.

Buscando mejores resultados en la segmentación de objetos de interés, se establecieron un conjunto de reglas para ajustar de mejor forma los rectángulos contenedores en casos en los cuales se encierran dos o más objetos dentro de un solo grupo debido a su cercanía espacial o descartar grupos que no pertenecen a objetos de interés.

Basándonos en el tamaño esperado del objeto y las observaciones de pruebas realizadas con los videos de entrenamiento, se determinaron los parámetros con los que se obtuvieron mejores resultados en segmentación. A partir de estos parámetros se definieron las siguientes reglas:

- Si el rectángulo contenedor supera 1.4 veces el tamaño esperado solo en dos de sus lados, este se divide en 2 rectángulos iguales por medio de una línea que divide los lados que supera dicho tamaño.
- Si el rectángulo contenedor supera 1.4 veces el tamaño esperado en todos sus lados, el rectángulo contenedor se divide en 4 rectángulos iguales.

Después de realizar los ajustes necesarios, a cada rectángulo obtenido se le asignan los puntos y su centro correspondiente, y por último se despliegan los rectángulos contenedores asociados a cada objeto detectado como resultado final del algoritmo.

Con el fin de eliminar soluciones que no pertenecieran a objetos de interés, se implementó un filtro en el cual se eliminan las soluciones con menos de 1/3 de los puntos esperados obtenidos en la inicialización.

En la Figura 29, se muestran imágenes de la utilización de dichas reglas.

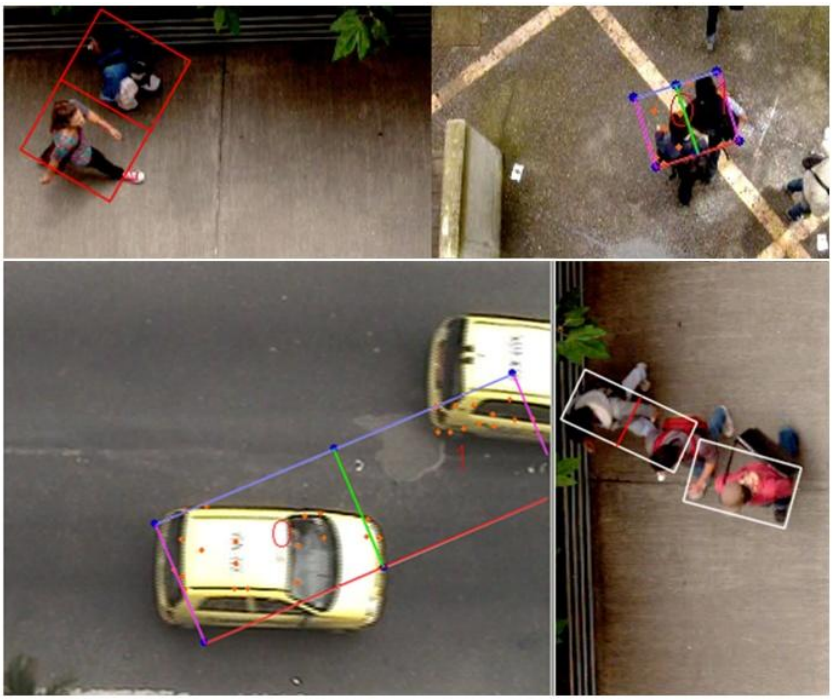


Figura 29. Implementación de reglas heurísticas. Objetos de interés segmentados por la heurística.

3.4 ESTIMACIÓN DE LA VELOCIDAD INSTANTÁNEA

Adicionalmente, el algoritmo ofrece la posibilidad de desplegar la velocidad instantánea de cada objeto de interés detectado en la escena. Para esto, partimos del flujo óptico estimado para los puntos asociados al objeto.

Para una mejor estimación de la velocidad instantánea, se realiza una corrección de perspectiva con el fin de corregir las distorsiones en el movimiento de los objetos, las cuales afectan la relación entre tamaños de los objetos presentes a diferentes distancias de la cámara.

Para la corrección de perspectiva, utilizamos una trasformación proyectiva, en donde se transforma un plano en otro plano equivalente en el que se conservan todas las propiedades invariantes a las proyectividades. Esto es aplicable debido a que en las imágenes obtenidas por las cámaras de video se conservan algunas propiedades, entre ellas la colinealidad (una línea recta se proyecta sobre una recta).

Para esto se computa la matriz *H* a partir de 4 puntos característicos de la escena y de la imagen, pertenecientes a un mismo plano. Durante la inicialización se seleccionan 4 puntos conocidos de la escena en la imagen desplegada, (ver figura 30), obteniendo los puntos del plano de la imagen. Luego se ingresan sus distancias reales medidas en la escena, dadas en cm, obteniendo así los puntos del plano mundo. Estos puntos deben formar un rectángulo dentro de la escena.

A partir de estos puntos se calcula el vector *h* por medio de la ecuación (7) el cual determina la matriz *H*.

Figura 30. Selección de puntos sobre la escena para corrección de perspectiva.

Una vez computada la matriz *H* se realiza la corrección de perspectiva mediante la ecuación (6) para los puntos iniciales y finales del flujo óptico asociados a un objeto de interés, entregando nuevos puntos dados en cm, sobre los cuales se vuelve a estimar su desplazamiento. Mediante la función `cvGetCaptureProperty` se obtiene la velocidad de captura del video en *frames* por segundo. A partir de este dato multiplicamos la velocidad de cada punto por un factor de conversión para obtener dicha velocidad en Km/h. La velocidad seleccionada para el objeto se estima mediante la moda de las velocidades asociadas a los puntos pertenecientes a dicho objeto. Finalmente se despliegan en el centro de cada objeto detectado.

Para obtener mejores resultados es importante trabajar sobre videos que mantengan una tasa estable *frames*/segundo en su captura como se puede apreciar en la Figura 31.

Figura 31. Velocidad estimada para los objetos de interés.

4. ANÁLISIS DE RESULTADOS

4.1 PROTOCOLO DE PRUEBAS

Para la evaluación del algoritmo, se utilizan videos con flujo vehicular y videos con flujo peatonal. Los videos fueron tomados ubicando la cámara de forma vertical, buscando estar lo más paralelo a la vía, con el fin de minimizar los traslapes y problemas de perspectiva de los objetos de interés. Así mismo, se utilizaron videos que cumplieran con las condiciones de brillo y movimiento mínimo para la mayoría de los objetos de interés presentes. Para el caso de los videos con flujo vehicular, se utilizaron vías de un solo sentido. Las pruebas se realizaron sobre 6 videos de entre 3 y 7 minutos de duración.

En estas pruebas, se evalúan 2 soluciones alternativas para el agrupamiento de puntos característicos de los objetos de interés: la agrupación de puntos mediante el algoritmo de *k-means* (Algoritmo 1), y la agrupación realizada por medio de la distancia de Chebyshev y el factor de similitud (Algoritmo 2).

Tanto el algoritmo 1 como el algoritmo 2, parten de una base en común, y es el desarrollo para encontrar los posibles centros de los objetos de interés. El algoritmo 2 pretende agrupar los puntos a partir de estos centros, de forma tal, que se realice un mejoramiento en la segmentación del objeto, dando como solución, una segmentación diferente con su respectivo centro.

Cada algoritmo utiliza criterios diferentes para la selección o agrupación de puntos pertenecientes a los objetos de interés. Una vez realizada la agrupación, para cada algoritmo se obtiene el mínimo rectángulo contenedor. De ser necesario, se aplican las reglas heurísticas y se realiza la segmentación a partir los rectángulos obtenidos. Para cada solución, se tiene un centro, una posición y un área asociados al objeto de interés detectado.

El algoritmo de prueba trabaja con el Algoritmo 1 y el Algoritmo 2 simultáneamente, y arroja los resultados de cada solución de manera independiente. La comparación se realiza bajo los mismos parámetros, sobre los mismos *frames*. Para el algoritmo de prueba, se requiere como inicialización la selección de la región de interés y la selección manual de 5 objetos que cumplan con las condiciones presentadas en la Sección 3.2.1.

Para cada video, se evalúo el 1% de los *frames* restantes después de la inicialización. Después de la etapa de entrenamiento del algoritmo, se escoge aleatoriamente entre los siguientes 50 *frames* la primera pareja a ser evaluada. A partir de la primera evaluación, el algoritmo se ejecuta sobre *frames* que se encuentran separando distancias iguales de forma que al recorrer todo el video se haya evaluado el 1% de los cuadros del video.

Para realizar la evaluación de cada algoritmo sobre un par de *frames*, se segmentan manualmente todos los objetos en movimiento de la escena que cumplan las siguientes condiciones:

- El objeto debe estar completamente dentro de la escena.
- El objeto debe haber presentado movimiento entre el *frame* anterior y el *frame* a ser evaluado.
- El objeto de interés no se debe traslapar con otro objeto.
- El objeto debe presentar buen contraste frente a la vía.

Sobre cada objeto válido, se realiza una segmentación manual, seleccionando 4 puntos sobre el objeto, de forma tal, que el mínimo rectángulo que los contenga, también contenga completamente al objeto, evitando enmarcar regiones que no pertenezcan al objeto de interés (ver Figura 32).

La verificación de los objetos a comparar es realizada por el evaluador, quien determina cuáles y cuántos de los objetos presentes en la escena cumplen con dichas condiciones.

A partir de los 4 puntos seleccionados de cada objeto válido, se determina el mínimo rectángulo contenedor, y se realizan los siguientes cálculos:

- Área segmentada por el algoritmo que pertenece al objeto.
- Área segmentada por el algoritmo que no pertenece al objeto.
- Área del objeto, que no fue detectada por el algoritmo.
- Distancia entre el centro estimada por el algoritmo y el centro del objeto.

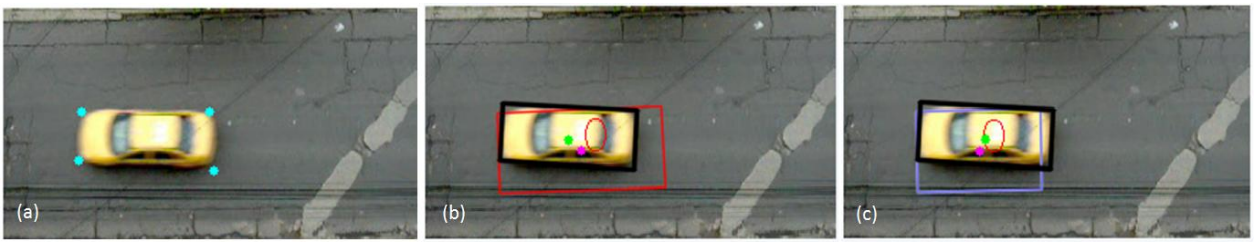


Figura 32. Evaluación de los algoritmos. (a) Selección manual de 4 puntos sobre el objeto, (b) solución Algoritmo 1, (c) solución Algoritmo 2.

La lista de los videos utilizados para realizar las pruebas, su locación, duración, si poseen flujo vehicular o de peatones y el número de *frames* evaluados, se encuentra en la Tabla 1. Se tiene en cuenta que los videos fueron grabados en la ciudad de Bogotá, D.C.

N°	Nombre del video	Locación	Duración	Vehículos	Peatones	N° de *frames* evaluados
1	53con13_1.avi	Cll 53 con Cra 13	00:04:46	X		52
2	45con8_1.avi	Cll 45 con Cra 8	00:05:13	X		57
3	45con8_2.avi	Cll 45 con Cra 8	00:04:05	X		78
4	53con13_2.avi	Cll 53 con Cra 13	00:05:12	X		63
5	Peatones_1.avi	Puente Facultad de Ingeniería PUJ	00:07:44		X	110
6	Peatones_2.wmv	Parqueadero PUJ	00:03:32		X	51

Tabla 1. La lista de los videos utilizados para realizar las pruebas, su locación, duración y si poseen flujo vehicular o de peatones

En la evaluación de los algoritmos, la segmentación de objetos que no son de interés, no se tendrán en cuenta para ninguno de los resultados. Por ejemplo, en los videos etiquetados como videos de flujo vehicular, las detecciones de peatones, ciclas y motos, se ignoraran como soluciones del algoritmo.

De igual forma, cuando el algoritmo detecte más objetos que los presentes en la escena, solo se tendrán en cuenta la solución que comparta mayor área con los objetos que cumple con las condiciones mencionadas anteriormente. Por ejemplo, si un bus es detectado como 2 objetos, solo se compara una de las soluciones, la que presente mayor área intersecada con dicho objeto.

Cuando el algoritmo, detecte menos objetos que los presentes en la escena, la comparación se realizará con el objeto más cercano. En el caso que se agrupen dos o más objetos como uno solo, se comparará con el que presente mayor área intersecada.

Buscando una relación entre la ubicación dentro del vídeo y los objetos no detectados, se tomaron datos de los lugares donde se encontraban los objetos no detectados, según se indica la Figura 33. En caso de que el objeto se encuentre entre dos o tres lugares, se le asignará el lugar donde se encuentre más área del objeto, y en caso de tener la misma área, se escogerá la que mas área tenga en los *frames* anteriores.

1	2	3
4	5	6

Figura 33. División de la escena.

Con el fin de realizar una evaluación más eficiente, el video 6 fue editado omitiendo las partes donde no hay peatones. Esto ayuda a que la mayoría de los *frames* evaluados tengan objetos de interés.

En la Figura 34, se muestran imágenes de cada uno de los videos de pruebas.

Figura 34. Imágenes de los videos de pruebas.

4.2 EVALUACIÓN DE DETECCIÓN DE OBJETOS

En la Tabla 2, se presentan los resultados de la detección de objetos de interés para cada uno de los videos de pruebas.

En la Figura 35, se muestra la comparación del número de objetos detectados por los dos algoritmos, y los encontrados por la detección manual.

Video N°	Número de Objetos de Interés que Presentan Movimiento Dentro de la Escena		
	Detección manual	**Detección con el Algoritmo 1**	**Detección con el Algoritmo 2**
1	40	39	38
2	24	25	24
3	20	21	22
4	80	76	74
5	69	41	41
6	35	27	27

Tabla 2. Resultados de la detección para los dos algoritmos.

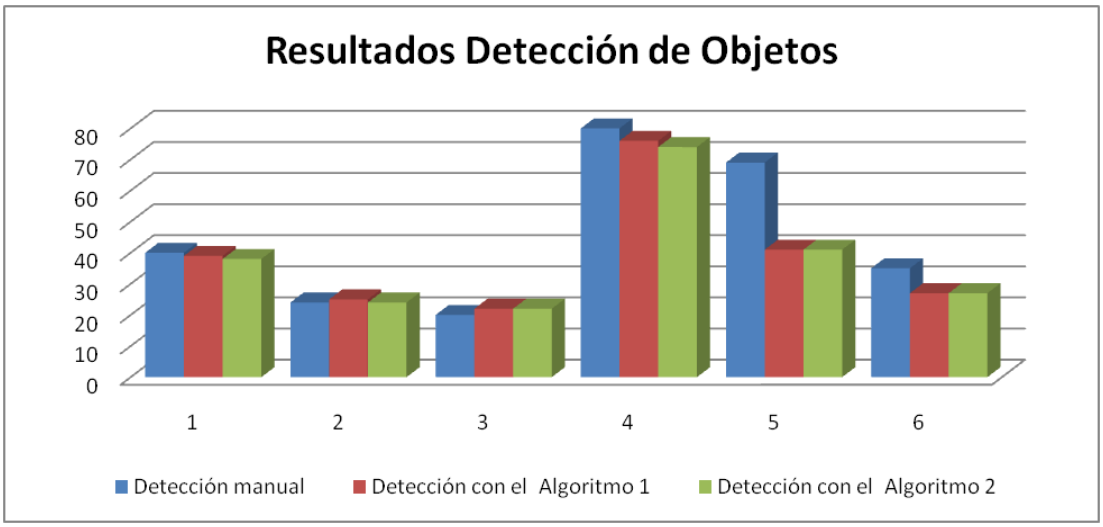


Figura 35. Resultado detección de objetos.

En la Figura 36, se presenta un grafico que muestra el porcentaje de error de cada algoritmo para los videos de prueba.

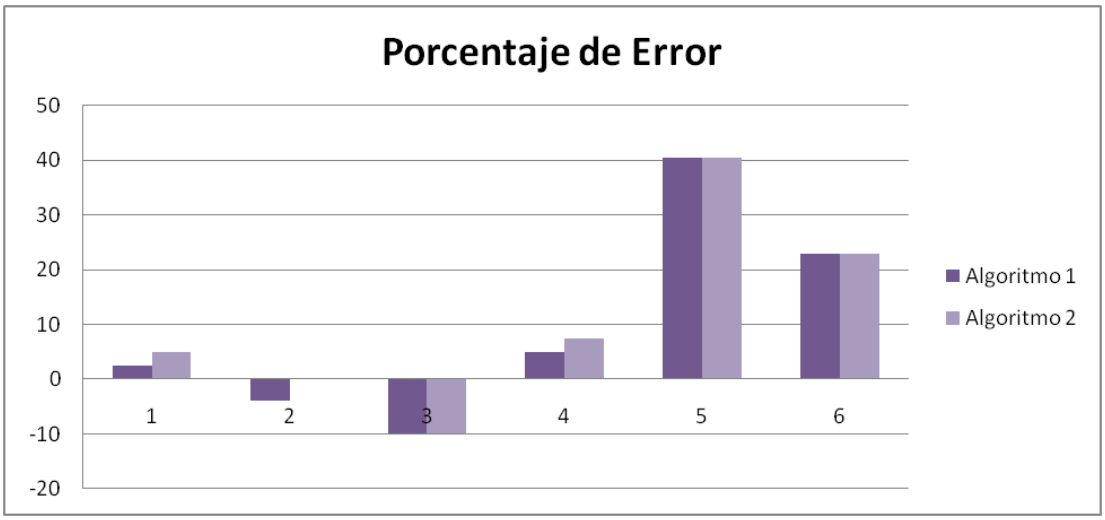


Figura 36. Porcentaje de error para la detección de objetos en cada video.

Como se puede observar en la Figura 36, los dos algoritmos arrojaron resultados similares en cuanto a detección de objetos de interés. Ambos presentaron un buen desempeño en los videos con flujo vehicular, sin embargo, se obtuvo un porcentaje de error alto para videos de peatones.
De igual forma, se puede observar que mediante el Algoritmo 1, se obtuvieron mejores resultados para los videos 1 y 4, y con el Algoritmo 2 se obtuvieron mejores resultados en el video 2.

La similitud en los resultados de detección para los dos algoritmos, se debe en parte, a que ambos utilizan los mismos centros propuestos como solución por la segmentación realizada con *k-means*. A partir de estos centros, los dos algoritmos buscan identificar cuáles de los puntos pertenecen a dichos centros mediante criterios diferentes, para finalmente, segmentarlos y entregarlos como solución utilizando las reglas heurísticas.

Para el video 2, el Algoritmo 1 presenta un error de – 4 %, lo cual implica que detectó más objetos de los que se encontraban en la escena. Esta clase de errores se presentan cuando se detectan cambios de intensidad en el video debidos a movimientos de cámara, o por la sombra de los mismos objetos de interés. De igual forma esto ocurre cuando dentro de la escena se presentan objetos más grandes que el tamaño esperado, como por ejemplo, buses, busetas, camiones, etc., ya que el algoritmo tiende a separar estos grupos de puntos muy grandes en subgrupos con tamaños cercanos al tamaño esperado, (ver Figura 37).

El Algoritmo 1, presenta una mayor tendencia a agrupar puntos generados por la sombra de los objetos de interés, por cambios de iluminación y movimientos de la cámara, debido a que no posee una restricción espacial muy estricta. Por el contrario, el Algoritmo 2, penaliza los puntos que se encuentren lejanos al centro del grupo. En la Figura 38, se muestran las soluciones para los dos algoritmos, y los puntos sobre los cuales se realiza la agrupación para un cuadro del video 2. Como se pude apreciar en la Figura 38.b, el Algoritmo 2 encerró solo cierta parte de los puntos. Por el contrario, como se muestra en la Figura 38.c, el Algoritmo 1 agrupa una buena parte de los puntos, incluyendo un punto lejano que se podría considerar como ruido. Debido a esto, para el Algoritmo 1, el rectángulo contenedor supera el tamaño esperado, por lo que se divide en 2 grupos mediante las reglas heurísticas.

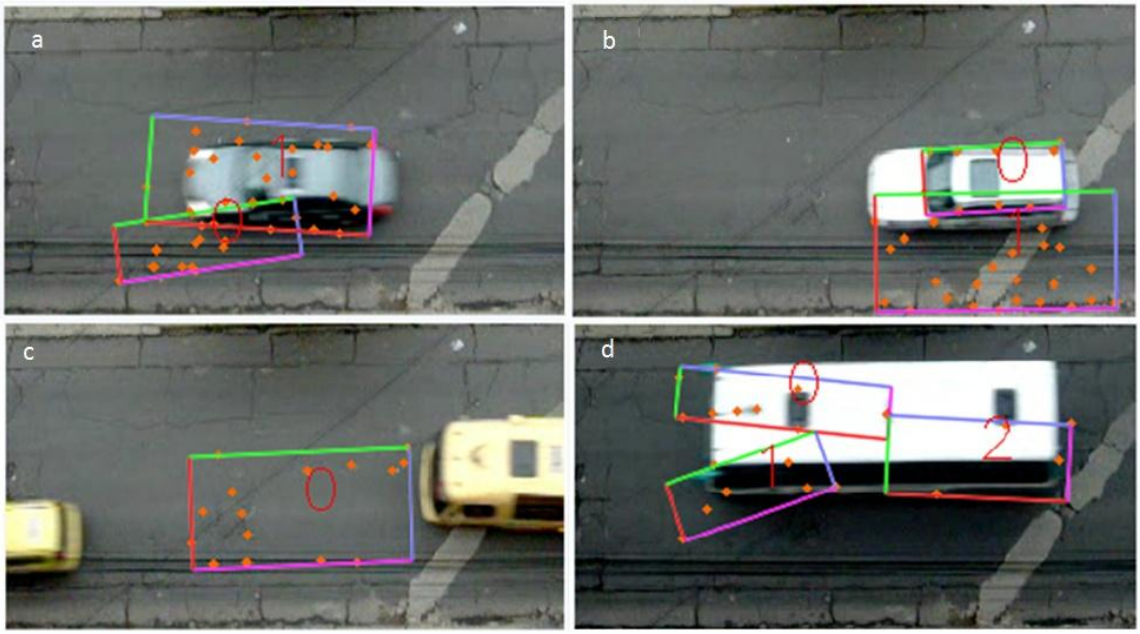


Figura 37. (a) y (b) problemas por detección de sobra de vehículos, (c) problemas por movimiento de cámara, (d) problemas por partición de objetos grandes.

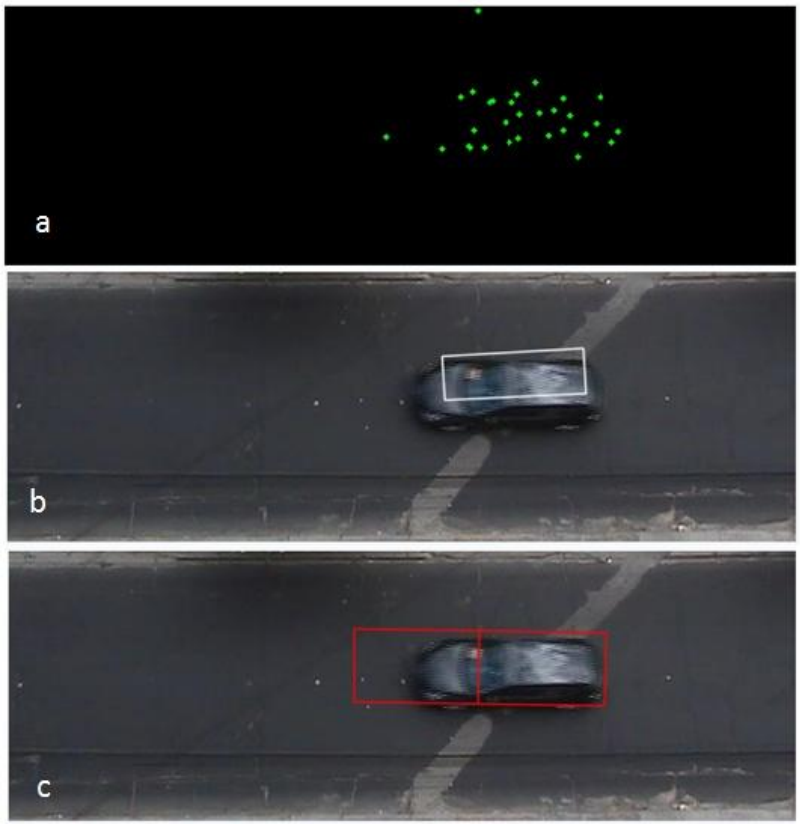


Figura 38. (a) puntos característicos, (b) solución Algoritmo 2, (c) solución Algoritmo 1.

Como se menciono anteriormente, los mayores errores en la detección de objetos se obtuvieron para los videos que presentan flujo de peatones. Por ejemplo, para el caso del video 5, se presentó un error del 40,5 %, el cual representa una cifra considerable.

Luego de revisar el video 5, se encontró que debido a la baja resolución del video (320x240), los objetos de interés presentaban tamaños pequeños por lo que fueron muy pocos los puntos asignados por al algoritmo de Shi y Tommasi [13] sobre cada objeto de interés.

A partir de la inicialización del video 5, se estableció un promedio de puntos asignados a cada objeto de aproximadamente 18 puntos, mientras que en los videos de configuración con flujo peatonal (604x480), los objetos de interés presentaron 41 puntos aproximadamente. Además, debido a que muchos de los puntos característicos se pierden entre un par de *frames* antes de volver a calcular *Good Features to Track*, y que existen partes del cuerpo de los peatones que no presentan movimiento como por ejemplo el pie de apoyo, algunos objetos de interés (no detectados) presentaron de 3 a 5 puntos finales asociados, por lo que se elimina dicha solución debido al filtro que descarta el objeto si el número de puntos es menor a 1/3 de los puntos del tamaño esperado.

Debido a esto se realizaron pruebas quitando el filtro de puntos mínimos para los videos 5 y 6, ya que los dos videos fueron tomados con la misma cámara, presentan la misma resolución y presentan flujo de peatones. Las nuevas pruebas se realizaron sobre los *frames* evaluados anteriormente, con el fin de determinar si se obtiene un mejor desempeño en detección de objetos. A partir de estos cambios, y con ayuda de las reglas heurísticas, se detectaron 4 objetos más utilizando el Algoritmo 1, y 8 objetos más para el Algoritmo 2, disminuyendo error de detección en un 5.7% y 11.5 % respectivamente. En la Tabla 4, se muestran los resultados de la prueba sin el filtro de puntos mínimos. En la Tabla 3 se presentan los valores de la resolución para cada video de prueba.

Video N°	Tamaño
1	72 x480
2	720x480
3	320x240
4	720 x 480
5	320x240
6	320x240

Tabla 3. Valores resolución de los videos.

Video N°	**Objetos encontrados por el evaluador**	**Algoritmo 1**		**Algoritmo 2**	
		Con filtro	Sin Filtro	Con filtro	Sin Filtro
5	69	41	45	41	49
6	35	27	31	27	32

Tabla 4. Comparación pruebas con filtro y sin filtro.

En la Figura 39, se presentan algunas imágenes de nuevos peatones detectados durante la prueba sin el filtro de puntos mínimos. En la Figura 39.a se detecta un peatón más que en la Figura 39.b. En este caso, el algoritmo 1 detecto un objeto el cual fue dividido en 2 por medio de las reglas heurísticas, mientras que en Algoritmo 2 fue más estricto en la segmentación luego solo se segmentó un peatón.

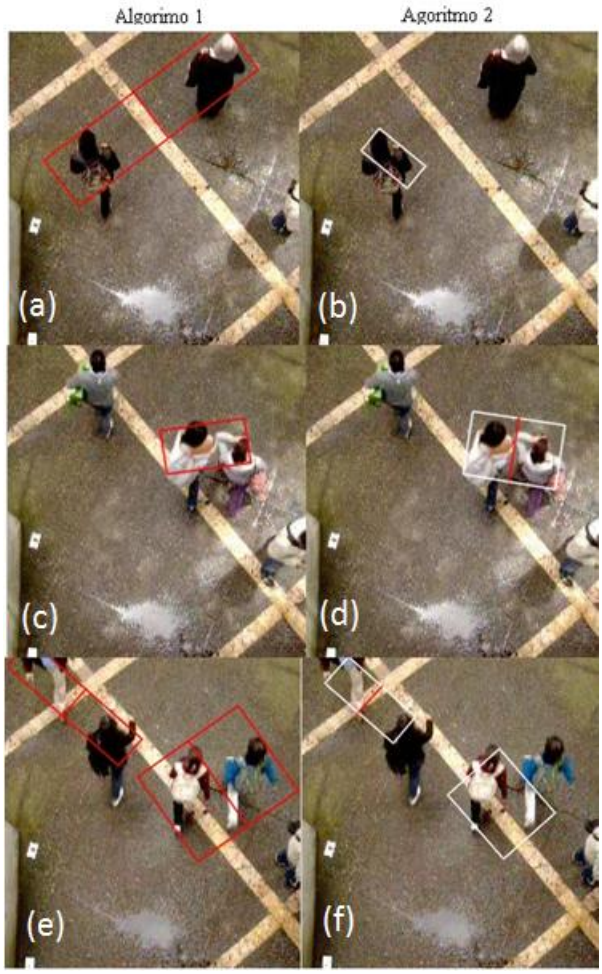


Figura 39. Prueba realizada para el videos 5 y 6 sin el filtro.

De igual forma, para el video 6, se obtuvo un porcentaje de error considerable (22,8 %). En este video, también se encontraron casos en los que el número de puntos asociados al peaton, era menor a 1/3 de los puntos esperados.

Después de realizar las pruebas sin el filtro de puntos mínimos, se detectaron 4 objetos más para el Algoritmo 1, y 5 peatones más, para el Algoritmo 2. Para las nuevas pruebas, el error disminuyó en un 11,4 % y 14,2 % respectivamente. En la Figura 40, se muestran algunos casos en los que el algoritmo sin el filtro de puntos mínimo detecto los peatones que no se habían detectado antes.

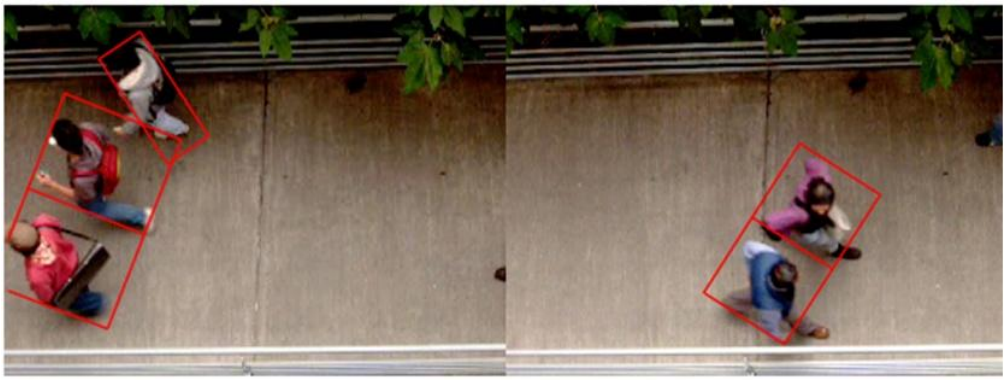


Figura 40. Prueba realizada para el video 6 sin el filtro.

Los videos que presentan flujo peatonal, poseen características diferentes a los videos que presentan flujo vehicular. Por ejemplo, los videos con peatones, suelen presentar mayores casos de traslape entre los objetos de interés. Para el video 5, el cual obtuvo el error más alto en detección, se presentaron casos en los cuales habían 6 peatones dentro de la escena (ver Anexo 1). En estos casos, los peatones se movilizaban en grupos de 3 personas. Esto dificulta la separación de los objetos, debido a que no se cumple un estándar para poder aplicar las reglas heurísticas. Para el caso de videos de prueba con flujo vehicular, se presentan máximo 4 objetos de interés dentro de la escena. Cuando esto ocurre, los vehículos mantienen una distancia prudente entre sí, facilitando su segmentación.

Aunque los dos videos de peatones presentaron errores altos en la detección de objetos, el video 6 presentó un porcentaje de error mucho más alto. Los factores que más influyeron para esto fueron la mayor cantidad de flujo de peatones y el tamaño de los objetos. El video 5, que presentó un error menor en un 43 % que en el video 6, posee un mayor número de puntos esperado para cada objeto, como se puede apreciar en la Tabla 5.

Video N°	Tamaño esperado en x (píxeles)	Tamaño esperado en y (píxeles)
1	229,4	109,8
2	126	49,2
3	94,8	48,8
4	124,6	60,8
5	60	55,8
6	67,4	61,2

Tabla 5. Tamaños esperados.

4.3 EVALUACIÓN DISTANCIA ENTRE CENTROS

Otro criterio de evaluación, fue la distancia promedio entre los centros detectados por el algoritmo y los centros de los objetos de interés. En la Tabla 6, se presentan los resultados para los dos algoritmos.

Video N°	Distancia promedio entre centros (píxeles)	
	Algoritmo 1	Algoritmo 2
1	15,92	14,5
2	19,49	21
3	13,5	12
4	14,15	12,1
5	11,58	10,21
6	13,6	14,4

Tabla 6. Resultados generales de distancia promedio entre centros

En la Figura 41, podemos apreciar la distancia promedio entre los centros para los dos algoritmos en los 6 videos de evaluación. Estos valores se encuentran en píxeles.

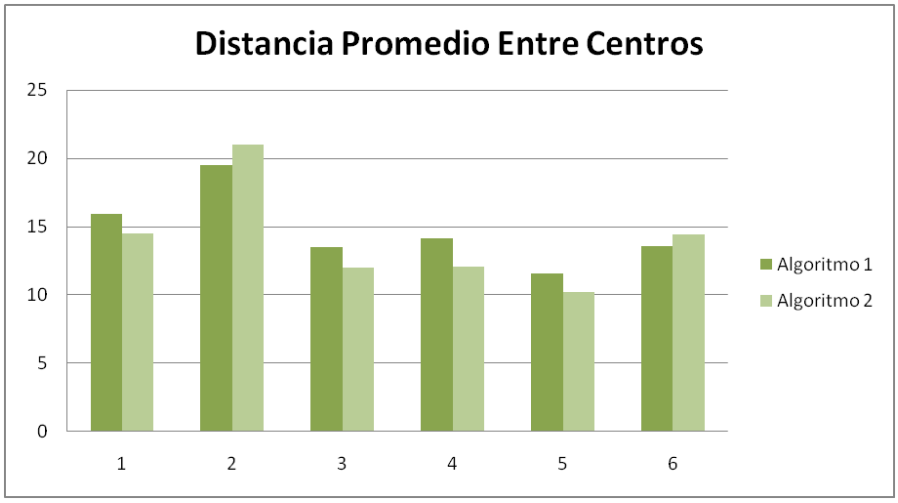


Figura 41. Distancia promedio entre centros.

El algoritmo que presentó mejores resultados respecto a la distancia entre los centros fue el Algoritmo 1, con un mejor desempeño especialmente en los videos con flujo vehicular.

Para poder realizar un mejor análisis, se determino el porcentaje de error de la distancia promedio de los centros respecto al tamaño esperado mostrado en la Figura 42. Según esta gráfica la distancia entre los centros fue mayor para los videos con flujo peatonal, obteniendo un error entre 18.29% y 23.5% respecto al tamaño del vehículo.

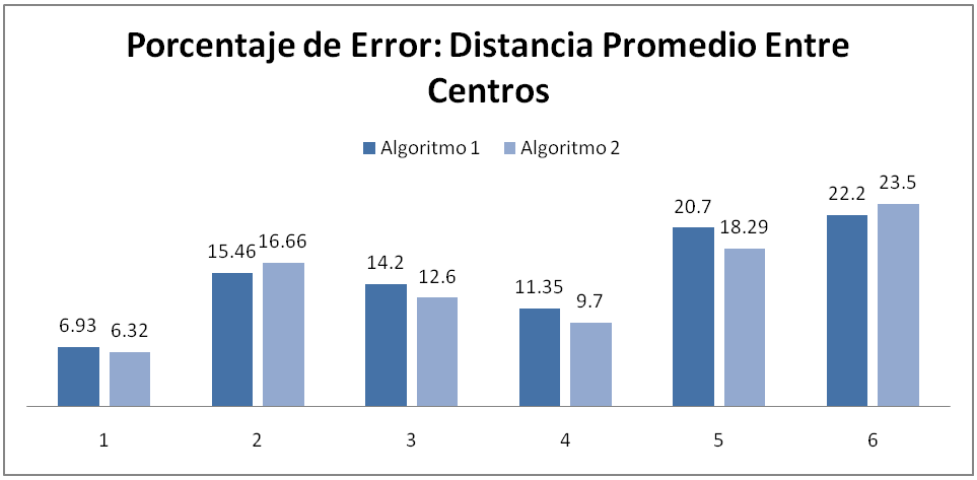


Figura 42. Porcentaje de error de distancia entre centros referenciados al tamaño esperado de los objetos.

Los mejores resultados se encuentran para el video 1 con un valor del 6,9 % para el Algoritmo 1 y 6,32 % para el Algoritmo 2. Para este mismo video, se obtuvieron los mejores resultados en la detección de objetos.

Los videos con flujo peatonal, presentaron un alto porcentaje de error en el desplazamiento promedio de sus centros respecto al tamaño esperado de los objetos. Esto es debido a que, para los casos en los que se detectaron dos o más peatones dentro de un solo grupo, resulta difícil determinar a cuál de los objetos se debe asocial la solución presentada. A demás, el centro determinado por el algoritmo suele caer en medio del grupo, alejándolo del centro de algún objeto presente (ver Figura 43).

Figura 43. Solución para objetos muy cercanos y que presentan el mismo movimiento.

Por esta misma razón, el Algoritmo 1, que tiende a asociar mayor número de puntos a los objetos de interés (ver Figura 38), suele generar rectángulos que no se encuentran alineados con respecto al objeto.

4.4 EVALUACIÓN DE ÁREAS

En la Tabla 7, se presentan los resultados de la comparación entre las áreas segmentadas por el Algoritmo 1, y las áreas de los objetos. En la Tabla 8, se presentan los resultados de las mismas comparaciones para el Algoritmo 2.

Video N°	Área del objeto detectada (%)	Área detectada por el algoritmo que no pertenece al objeto (%)	Área del objeto no detectada (%)
1	92,9	24,78	7
2	71,33	19,36	27,07
3	85,39	57,4	14,61
4	84,98	25,8	15,02
5	52,56	50,34	47,43
6	77,66	79,1	23,33

Tabla 7. Resultados generales en porcentajes de las áreas detectadas para el Algoritmo 1.

Video N°	Área del objeto detectada (%)	Área detectada por el algoritmo que no pertenece al objeto (%)	Área del objeto no detectada (%)
1	89,08	14,78	10,92
2	62,86	13,59	35,5
3	62,73	26,22	37,27
4	79,6	18,08	20,4
5	46,83	43	43,17
6	53, 55	42,44	46,45

Tabla 8. Resultados generales en porcentajes de las áreas detectadas para el Algoritmo 2.

En la Figura 44, se muestra el porcentaje del área del objeto detectada para cada uno de los algoritmos, y el porcentaje del área detectada por los algoritmos que no pertenece a los objetos de interés.

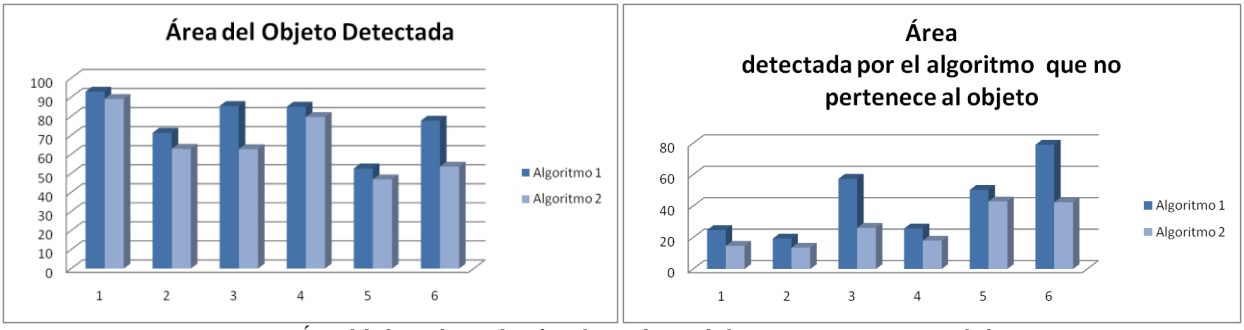


Figura 44. Área del objeto detectada y área detectada por el algoritmo que no pertenece al objeto.

Como se pude apreciar en la Figura 44, el Algoritmo 1 presentó un mejor desempeño en porcentaje de detección del área del objeto para todos los casos, encontrado desde el 52,56 % del área detectada para el video 5 hasta el 92,9 % para el video 1. De igual forma, el Algoritmo 1 presenta un mayor porcentaje del área detectada que no pertenece al objeto. Esto concuerda con los resultados anteriores, debido a la agrupación de puntos lejanos a los objetos de interés por medio del Algoritmo 1. Este algoritmo presenta un rectángulo contenedor más grande respecto al obtenido para el Algoritmo 2. Según lo anterior es de esperarse que el Algoritmo 1encierre un porcentaje mayor del área del mismo objeto, pero que a su vez, arroje como solución a la segmentación de los objetos de interés, grandes áreas que no pertenecen a dichos objetos. Recordemos que los dos algoritmos tienen como base los mismos centros determinados por el algoritmo de *k-means* y ciertas reglas de unión de grupos.

Para el video 5, se obtuvo un porcentaje bajo del área detectada, debido a los problemas que se presentaron por la unión de uno o más objetos de interés en un solo grupo. Cuando esto ocurre, muchas veces los rectángulos contenedores no coinciden con objetos asociados, por lo que existe una gran parte del área dada como solución del algoritmo que pertenece a otros objetos.

A partir del análisis anterior, se puede decir que para el Algoritmo 2, como se esperaba, se obtuvo un alto porcentaje de área del objeto que no fue detectada. En algunos casos como por ejemplo para el video 3, este porcentaje fue el doble respecto al Algoritmo 1, como se aprecia en la Figura 45.

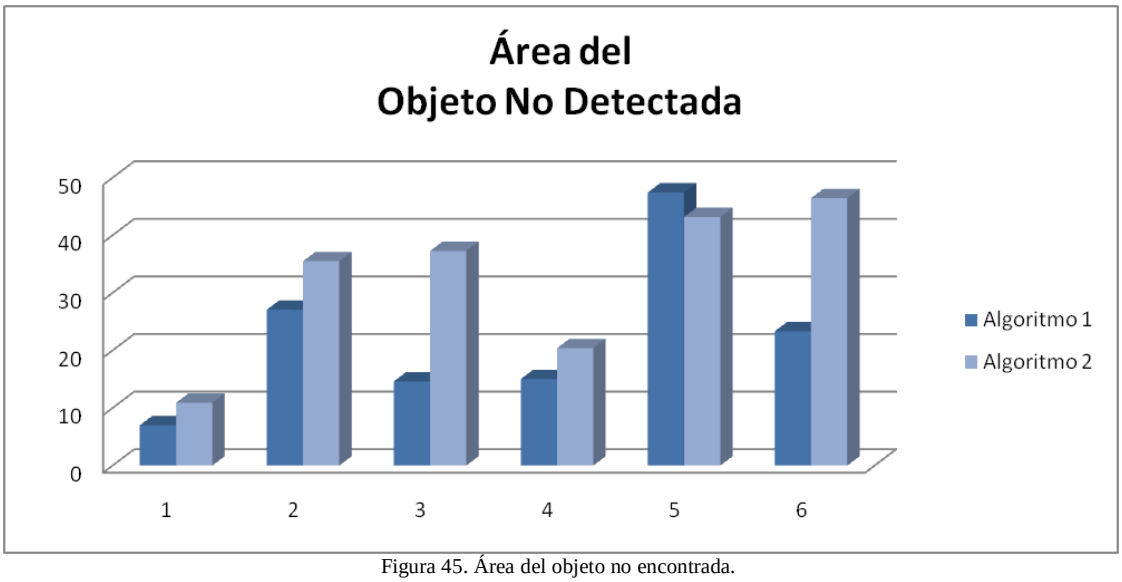


Figura 45. Área del objeto no encontrada.

4.5 ERROR Y NÚMERO DE OBJETOS EN LA ESCENA

En la Figura 46 se muestra el porcentaje de error del promedio de los objetos detectados en todos los videos por los dos algoritmos. Estos datos se encuentran agrupados según el número de objetos detectados por el evaluador en la escena.

En el eje *x* de esta gráfica se presentan el número de objetos en la escena. El color más oscuro representa el error según el número de objetos detectados obtenido con el Algoritmo 1, y el color más claro representa lo mismo para el Algoritmo 2.

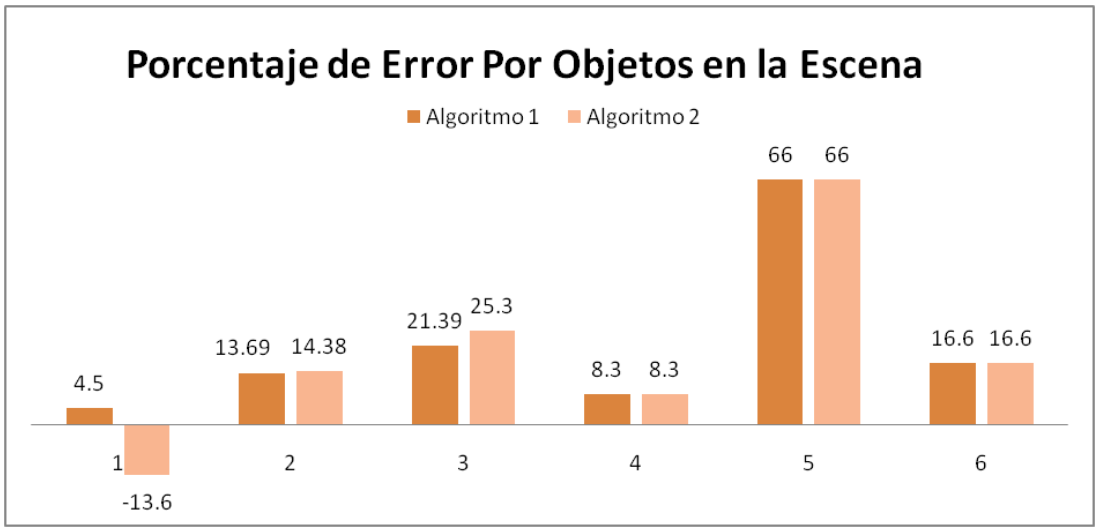


Figura 46. Porcentaje de error según número de objetos en la escena.

En esta Figura se puede apreciar que el error es menor para los dos algoritmos cuando en la escena se presentaba solo un objeto de interés. Se esperaba que estos valores crecieran a medida que aumenta el número de objetos. Esto se puede apreciar en la grafica para 1, 2 y tres objetos. Sin embargo, para el caso

de 4 y 6 objetos se presenta un porcentaje bajo. Esto se debe a que el evento 4 ocurrió 3 veces frente a 65 veces del primero. De igual forma, el evento 6 ocurrió una sola vez.

Por otro lado, el evento 5 se presentó en dos ocasiones para los videos que presentan flujo peatonal. Como se observa en la Figura 46, se tienen problemas de detección para estos videos, y este incrementa si aumenta el número de objetos de interés en la escena.

4.6 ERROR Y POSICIÓN

En la Tabla 9 se presenta el número de objetos según la región del video en donde se presentó el error. En la Figura 47, se muestran los resultados totales. En color azul se muestra el número de errores, y en rojo, el número de objetos presentes en dicha región.

Para esta prueba se esperaba que para los videos de vehículos, los errores más bajos se encontraran en la región 2 y 5. Esto teniendo en cuenta que *Good Features to Track* actualiza los puntos cada 10 *frames*, dejando muchas veces a los vehículos entrantes sin puntos para seguir. Sin embargo los resultados no siguen un patrón respecto a la ubicación y el número de errores de detección, para determinar si existe alguna región en la cual se presenten más errores que en otra.

Video N°	Regiones donde se presentaron los errores					
	Región 1	Región 2	Región 3	Región 4	Región 5	Región 6
1	2	1	2	3	2	0
3	2	0	0	2	0	0
4	0	2	1	0	0	0
5	10	14	6	12	6	6
6	2	2	2	6	2	2

Tabla 9. Región de objetos que presentaron error.

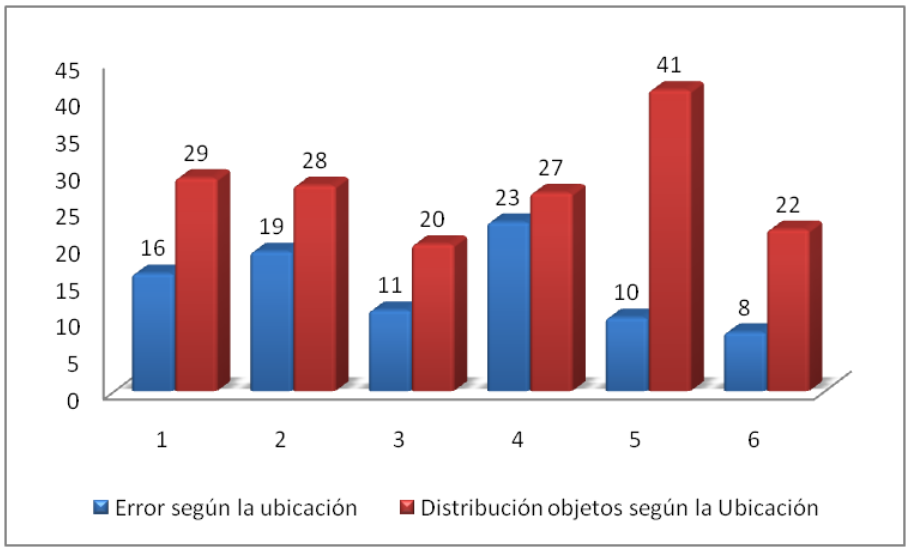


Figura 47. Región objetos que presentaron error.

4.6 TIEMPOS DE PROCESO

En la Tabla 10, se presentan los tiempos de proceso promedio para cada video. Estas pruebas se realizaron utilizando un computador Sony Vaio Modelo PCG-5L1P el cual tiene un procesador Pentium Dual Core de 1.73 GHz, 2 GB de memoria RAM y un disco duro de 200GB el cual trabaja con el sistema operativo de Microsoft Windows XP profesional 2002 Service Pack 3 .Como se puede observar, el algoritmo tiene una tasa de procesamiento promedio más alta que la tasa de adquisición para los videos 1, 2, 3 y 5. Estas velocidades de procesamiento del algoritmo dependen del número de objetos de interés dentro de la escena, luego los videos que presenten un flujo más alto, se tardarán en procesar más tiempo.

Nº del video	Velocidad de procesamiento (frames/Segundo)	Tasa de frames (frames/Segundo)
1	41.8	25
2	44.2	29
3	44.19	29
4	20.36	25
5	34.0	29
6	26.45	30

Tabla 10. Tiempos de proceso.

5. CONCLUSIONES

El presente trabajo de grado da continuidad a una serie de investigaciones realizadas por el grupo de investigación *SIRP* de la Pontificia Universidad Javeriana en el capítulo de "Adquisición Automática de Variables de Tráfico", al desarrollar e implementar un algoritmo de detección de objetos móviles presentes en una escena, basándose en la estimación del flujo óptico mediante la implementación de Lucas y Kanade piramidal.

Los mejores resultados en detección se presentaron para videos con flujo vehicular, en donde para uno de los casos se alcanzó el 2.5 % de error. Por el contrario, para los videos con flujo peatonal se alcanzaron errores considerables. El algoritmo presentó los mejores resultados en detección para los casos en los cuales se encontraba sólo un objeto de interés dentro de la escena.

Los factores más determinantes para la detección de vehículos y peatones en movimiento a partir del flujo óptico, son el tamaño en píxeles de los objetos, su cercanía espacial con otros objetos de interés, y el número de objetos de interés dentro de la escena. Los mejores resultados se presentaron para videos con tamaño 720x480. Los problemas presentados para los videos de menor resolución, se deben a que el método utilizado para la estimación del flujo óptico se basa en el seguimiento de ciertos puntos de interés determinados por el algoritmo de Shi y Tommasi, los cuales disminuyen en cantidad entre más pequeño sea el objeto de interés.

De igual forma se propusieron dos algoritmos alternativos para la agrupación de puntos pertenecientes a los objetos de interés. Los dos algoritmos presentaron resultados similares en cuanto a la detección de objetos, debido a que ambos parten del mismo centro. Sin embargo, como es de esperarse, los algoritmos presentaron diferencias respecto a la segmentación de los objetos. El algoritmo basado *k-means* y minimización de variables dependientes de la velocidad, presenta agrupaciones más grandes que para el algoritmo basado en la distancia de Chebyshev y el factor de similitud propuesto, obteniendo una segmentación que encierra el mayor porcentaje del área del objeto, pero así mismo, encierra mayor cantidad de área que no pertenece.

Las segmentaciones de objetos de interés basadas en la estimación del flujo óptico, no ofrecen la suficiente información para poder determinar el centro real de un objeto en movimiento. Sin embargo, el algoritmo basado en Chebyshev y el factor de similitud, presenta un error menor en cuanto a ubicación de los centros presentados como solución al algoritmo y los centros de los objetos de interés. Por esta razón, se seleccionó como algoritmo definitivo, el basado en la distancia de Chebyshev y el factor de similitud, ya que a partir de una mejor ubicación en los centros de los objetos de interés, es posible, establecer una mejor segmentación.

El algoritmo desarrollado no presenta iguales resultados cuando se trabaja con videos que presenten flujo vehicular y flujo peatonal, debido a las diferencias en forma y movimiento de dichos objetos. Lo que para los videos con flujo peatonal puede ser interpretado como un error, no lo es para los videos con flujo vehicular. También se pudo determinar que mediante el ajuste de parámetros que dependan de la resolución del video, tal como el filtro mínimo de puntos, se pueden obtener mejores resultados. De igual manera podemos concluir que no se presenta relación entre la ubicación del objeto en la escena y la probabilidad de ser detectado, siempre y cuando este objeto cumpla con las restricciones iniciales.

Este trabajo sirve como punto de partida para el desarrollo de algoritmos de seguimiento y conteo de objetos, basados en la estimación del flujo óptico. De igual forma, mediante la adición de algoritmos de seguimiento de objetos, se podrían obtener mejores resultados independientemente de la cantidad de movimiento que presenten los objetos dentro de la escena, ya que una vez detectados, se podría saber cuál es el comportamiento de dicho objeto durante su permanencia en el video. Adicionalmente, la información

de movimiento obtenida por el algoritmo propuesto puede ser utilizada para complementar las estrategias de detección y seguimiento de objetos que se basan en un modelo del fondo.

Algunos de los resultados obtenidos fueron presentados en el Simposio de Tratamiento de Señales, Imágenes y Visión Artificial (STSIVA), realizado entre los días 9 y 11 de septiembre en la Universidad Tecnológica de Pereira [18]. Como continuación a este trabajo de grado, se propone implementar diferentes reglas heurísticas, basadas en la segmentación de grupos de personas según su tamaño, o con el conteo de cabezas, para obtener mejores resultados al trabajar con videos que presenten flujo peatonal. Así mismo, se podrían desarrollar diferentes reglas para poder trabajar con videos que presenten traslape entre los objetos de interés, y videos que contengan diferentes clases de objetos en una misma escena tales como peatones y vehículos, basados en la detección previa de objetos en movimiento realizada por este algoritmo. También como trabajos futuros se propone hacer algoritmos de seguimiento, en los cuales se espera que sea más robusta la segmentación porque que se tiene más información de los objetos en las escenas anteriores. Además de este algoritmo, los trabajos futuros sirven como base para algoritmos de conteo.

6. BIBLIOGRAFÍA

[1] Kim-Sung Jie, Ming Liu. (2005) "Computer Vision Based Real-time Information acquicition for transport Trafic", Information Acquisition, 2005 IEEE International Conference on Volume, 27 paginas: 6 pp.

[2] Calderón Francisco, Urrego Germán. "Conteo Automático de Vehículos". Trabajo de Grado. Departamento Ing. Electrónica, Pontificia Universidad Javeriana. Bogotá 2008.

[3] Nestor Romero y Carolina Garcia, "Detección y Seguimiento de personas en un cruce peatonal". Trabajo de Grado. Departamento Ing. Electrónica, Pontificia Universidad Javeriana. Bogotá 2009.

[4] P. L. Rosin, "Thresholding for Change Detection", Sixth International Conference on Computer Vision, 1998.

[5] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground–background segmentation," *Real-Time Imaging 11*, pp. 167-256, 2005.

[6] G Chris Stauffer and W.E.L Grimson, "Adaptive background mixture models for real-time tracking" IEEE Computer Society Conference on. Volume 2, 23-25 June 1999.

[7] Michail Krinidis, Nikos Nikolaidis, Ioannis Pitas, " 2-D Feature-Point Selection and Tracking Using 3-D Physics-Based Deformable Surfaces" Circuits and Systems for Video Technology, IEEE Transactions on
Volume 17, Issue 7, July 2007 Page(s):876 – 888.

[8] Adrian G. Bors¸ and Ioannis Pitas. "Prediction and Tracking of Moving Objects in Image Sequences". IEEE Transactions on image processing, vol. 9, no. 8, August 2000.

[9] David G. Lowe, *"Distinctive Image Features from Scale-Invariant Keypoints"*, Computer Science Department, University of British Columbia, Vancouver, B.C., Canadá,lowe@cs.ubc.ca, January 5, 2004

[10] Jean-Yves Bouguet "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm" Intel Corporation Microprocessor Research Labs.

[11] B.K.P. Horn and B.G. Schunck, "Determining optical flow." *Artificial Intelligence*, vol 17, pp 185-203, 1981.

[12] Bruce D. Lucas Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". From *Proceedings of Imaging Understanding Workshop*, pp. 121-130 (1981). H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[13] Jianbo Shi and Tomasi C. "Good features to track". IEEE Computer Society Conference on 21-23 June 1994 Page(s):593 – 600.

[14] Gary Bradski E Adrian Kaebler, *Learning OpenCV: Computer Vision whit the OpenCV Library*, O´Reilly, Software That Sees.

[15]. Dinero (2008, 11 de Diciembre), "Multas Soluciones a los problemas de movilidad" [en línea], disponible en http://www.dinero.com/noticias-on-line/soluciones-problemas-movilidad/54260.aspx.

[16] W.S.P. Fernando, Udawatta Lanka, Pathirana Pubudo. *"Identification of Moving Obstacles with*

Pyramidal Lucas Kanade Optical Flow and k means".

[17] Yair Weiss, "Motion Segmentation using EM – a short tutorial", MIT E10-120, Cambridge. USA.

[18] Quiroga Julián, Mora David y Páez Andrés, "Detección de Objetos Móviles en una Escena Utilizando Flujo Óptico", XIV simposio de tratamiento de señales, imágenes y visión artificial – STSIVA 2009.

[19] Apuntes de clase: Sistemas Lineales, Patiño Diego, 2009, Pontificia Universidad Javeriana, Bogotá.

[20] Apuntes y diapositivas de clase: Señales y Sistemas, Quiroga Julián, 2009, Pontificia Universidad Javeriana, Bogotá.