ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN MEDIDOR DE VELOCIDAD BASADO EN VISIÓN ARTIFICIAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

DANNY JAVIER VELASCO SALAZAR

DIRECTOR: PhD. ROBIN ÁLVAREZ RUEDA

Quito, febrero 2020

.

AVAL

Certifico que el presente trabajo fue desarrollado por Danny Javier Velasco Salazar, bajo mi supervisión.

Ph.D. Robin Álvarez Rueda
DIRECTOR DEL TRABAJO DE TITULACIÓN

.

DECLARACIÓN DE AUTORÍA

Yo, Danny Javier Velasco Salazar, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamento y Normas vigentes.

Danny Javier Velasco Salazar

.

DEDICATORIA

Dedicado a quienes se adelantaron y aun así siguen siendo el corazón de mi familia.

.

AGRADECIMIENTO

A mi padre y madre Graciela por su apoyo y motivación para alcanzar cada una de mis metas.

A mis amigos por el tiempo y momentos compartidos.

Un especial agradecimiento al Dr. Robin Álvarez por el tiempo invertido en guiar este Trabajo de Titulación.

Danny Velasco

.

ÍNDICE DE CONTENIDO

.

.

RESUMEN

En este trabajo se presenta un método automático de medición de velocidad vehicular. Se tiene el dato real de su velocidad (observada en el tacómetro), y esto permite obtener el error cometido por el método. Una vez realizadas las pruebas, esta técnica podrá ser aplicada para mediar la velocidad de cualquier otro objeto.

Para realizar dicha medición de velocidad, se requiere tener la distancia recorrida por el objeto y el tiempo invertido, ambas variables considerando el campo de visión de la cámara empleada.

Para obtener la distancia abarcada por el campo de visión de la cámara, inicialmente fue hecho de manera manual dependiendo del escenario particular. Posteriormente su obtención fue automatizada mediante el uso de un medidor laser que mide la distancia perpendicular a la trayectoria del vehículo y, por otro lado, al ángulo del campo de visión de la cámara empleada que también es calculado experimentalmente. Con ambos parámetros, se obtiene trigonométricamente dicha distancia del campo de visión.

Para la medición del tiempo se emplea un algoritmo en Matlab considerando el tiempo desde que entra hasta que sale el objeto del mencionado campo de visión de la cámara, observándolo en los frames adquiridos. En base a estos dos parámetros, distancia y tiempo, se determina la velocidad del objeto.

Para determinar del tiempo se utilizaron dos métodos: algoritmo de sustracción de fondo y detección de bordes; este primero fue el que obtuvo menores porcentajes de error de 8 %.

Palabras clave: Medición de velocidad de un objeto, medición de velocidad de un vehículo, visión artificial.

.

ABSTRACT

This project presents an automatic method of vehicle speed measurement. It has the real data of its speed (observed in the tachometer), and this allows to obtain the error made by the method. Once the tests have been carried out, this technique can be applied to mediate the speed of any other object.

To perform this speed measurement, it is required to have the distance traveled by the object and the time invested, both variables considering the field of view of the camera used.

To obtain the distance covered by the camera's field of view, it was initially done manually depending on the particular scenario. Subsequently, its obtaining was automated through the use of a laser meter that measures the distance perpendicular to the trajectory of the vehicle and, on the other hand, to the angle of the field of view of the camera used which is also calculated experimentally. With both parameters, this distance from the field of vision is trigonometrically obtained.

For the measurement of time, an algorithm is used in Matlab considering the time from when it enters until the object of the mentioned field of vision of the camera comes out, observing it in the acquired frames. Based on these two parameters, distance and time, the speed of the object is determined.

Two methods were used to determine the time: background subtraction algorithm and edge detection; This first was the one that obtained the lowest error rates with 8%.

Keywords: Speed measurement of an object, speed measurement of a vehicle, artificial vision.

1. INTRODUCCIÓN

La visión artificial o también llamada visión por ordenador es un conjunto de procesos que se utilizan principalmente para procesar y analizar información relevante obtenida de imágenes digitales.

En el presente caso, estos procesos son: captación de imágenes, memorización de información, pre-procesamiento (operaciones morfológicas para poder mejorar la imagen), algoritmo de detección de objetos, algoritmo de cálculo de velocidad.

Los radares tradicionales de medición de velocidad de vehículos funcionan bajo el principio de emisión de una onda electromagnética. Esta se refleja en dicho vehículo y es recibida por el mismo equipo de manera que se mide el tiempo empleado. Por otro lado, ya que se conoce la velocidad de propagación de la onda electromagnéticas (la velocidad de la luz), basados en el efecto Doppler, se puede realizar la medición de velocidad del vehículo [1].

En la actualidad existen diferentes tipos de radares de tráfico vehicular pero su costo es elevado dependiendo de sus especificaciones, tipo y marca.

Los radares tipo laser barrera basan su funcionamiento en dos haces laser ubicados perpendicularmente a la carretera los cuales están separados una cierta distancia. Se mide el tiempo que tarda el vehículo en cruzar de un haz hacia el otro. Con ambos parámetros medidos (distancia y tiempo), se calcula su velocidad. Estos radares pueden operar de forma manual o automática. Realizan dos fotos por cada vehículo que supere la velocidad limite permitida, y lo hacen tanto en uno como en los dos carriles con un margen de error de 1% [2].

La principal desventaja de estos radares es su costo demasiado alto y el mantenimiento de requerido, sumado al trámite respectivo de importación desde otro país.

Otro tipo de radares más económicos y llamados móviles son los de pistola, por ejemplo, el modelo BUSHNELL II (101911), funcionan mediante láser y su principal característica es que permite medir la velocidad desde 16 Km/h hasta 322 Km/h [3]. Estos radares resultan ser más económicos que los mencionados anteriormente, poseen algunos problemas como es el ángulo en el que el vehículo se encuentra con respecto a la pistola, lo cual puede generar un error en la estimación de la velocidad. Otro inconveniente es el ambiente en el que se puede utilizar la pistola, ya que la niebla o la lluvia afecta de manera considerable su precisión.

Los radares mencionados anteriormente utilizan para su funcionamiento radiofrecuencia y láser respectivamente. Actualmente, se tiene otro tipo de radar vehicular denominado de tramo que para su funcionamiento utiliza dos cámaras que reconocen la matricula al inicio y al final de un tramo. Mediante este mecanismo se toma el tiempo que se demoró el vehículo en atravesar las dos cámaras separadas una distancia conocida, con lo que se calcula su velocidad [2].

En este trabajo se pretende presentar un sistema de medición de velocidad vehicular basado en visión artificial con el fin de poder obtener una opción más económica en comparación con los radares descritos anteriormente. El algoritmo será capaz de trabajar bajo diferentes condiciones de luz y utilizar dispositivos de bajo costo.

Como ya se mencionó, en este proyecto se desarrolla un método de medición de velocidad basado en visión artificial. El algoritmo empleado permite tener cualquier ángulo de ubicación de la cámara. Su funcionamiento depende del tiempo en el que el objeto se encuentre en el campo de visión de la cámara.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es:

- Implementar un medidor de velocidad de un objeto basado en visión artificial.

Los objetivos específicos del Proyecto Técnico son:

- Analizar los parámetros con los que se determinará la velocidad del objeto.
- Implementar el algoritmo para calcular la velocidad de un objeto.
- Determinar la máxima tasa de adquisición de imágenes empleando el computador y con ello saber hasta qué velocidad de un objeto en movimiento se puede obtener.
- Determinar la influencia sobre esta tasa de adquisición del uso algoritmos de detección de bordes (gradiente de Scharr con filtro gaussiano, algoritmo de Canny, etc) y observar cual posee mejor exactitud en el cálculo de la velocidad.
- Realizar las pruebas respectivas, utilizando el algoritmo seleccionado para el cálculo de la velocidad de un objeto en tiempo real.

1.2. ALCANCE

Existen varias técnicas que se utilizan para el procesamiento de imágenes, los algoritmos que se utilizan en el presente proyecto son operaciones morfológicas (dilatación, erosión, erosión+dilatación y dilatación+erosión), etiquetado de imágenes binarias, entre otras. El etiquetado es importante ya que permite representar al objetivo en movimiento y poder calcular el centro de masa de dichos objetos que en este caso sería el objeto del cual se va a medir su velocidad [4].

Se pretende determinar la máxima tasa de adquisición de imágenes real empleando el computador y con ello saber hasta qué velocidad de un objeto en movimiento se puede determinar.

Se desea determinar la influencia en esta tasa de adquisición cuando se emplean los siguientes algoritmos de detección bordes (gradiente Laplaciano con filtro Gaussiano, algoritmo de Canny, etc) y escoger el que obtenga mejor exactitud [5].

Se realizará la medición de velocidad solamente de un objeto en movimiento

El algoritmo de detección de bordes seleccionado será aquel que funcione de mejor manera en las respectivas pruebas y que entregue una lectura más fiable de la velocidad del objeto.

Para un adecuado procesamiento de imágenes se probarán diferentes cámaras como webcam, webcam infrarroja (IR) y se escogerá la más adecuada para tener el menor error en el cálculo de velocidad.

Las pruebas de funcionamiento se realizarán sobre un vehículo ya que este tiene su medidor de velocidad(tacómetro) incorporado lo cual nos permitirá establecer el error cometido por nuestro algoritmo.

Este proyecto contará con un producto final demostrable el cual consta de un computador, un medidor de distancia laser y la cámara adecuada. De esta manera se podrá calcular la velocidad de un vehículo en tiempo real.

1.3. MARCO TEÓRICO

1.3.1. ESTADO DEL ARTE DE ALGORITMOS Y TECNOLOGÍAS DE MEDICIÓN DE VELOCIDAD UTILIZANDO ORDENADOR

Por varios años se han realizado esfuerzos para calcular la velocidad de un objeto de una manera más económica y eficaz. A continuación, revisaremos algunos de ellos:

1.3.1.1. Medición de velocidad utilizando el Método de flujo óptico

En este método se tiene una imagen solo con dos niveles: uno que es el estático y otro que es el objeto en movimiento. Se calcula el centroide del objeto en movimiento y se mide el tiempo que toma en desplazarse una determinada distancia, con estos dos datos se estima la velocidad y se realiza un seguimiento de dicho centroide [6].

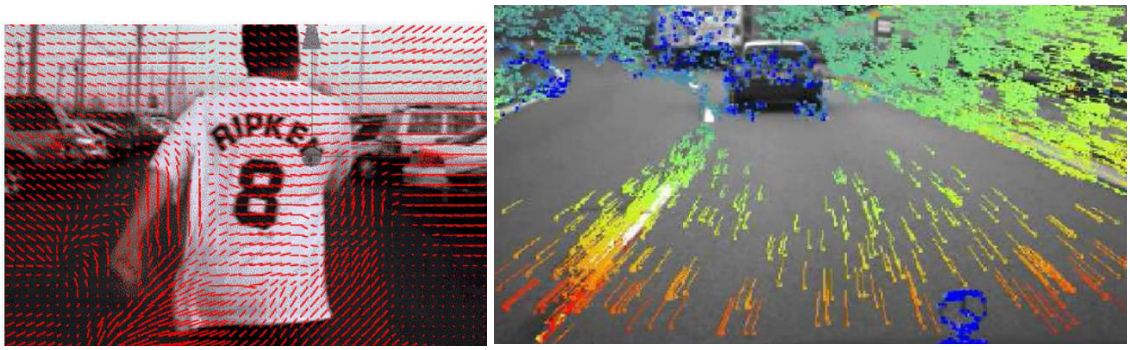


Figura 1.1 Ejemplo de flujo óptico [7]

Para la implementación de este método presenta dos fases principales: identificación y modelado.

La fase de identificación permite encontrar el o los objetos requeridos a lo largo de toda la secuencia de video. Para realizar esta fase se emplea técnicas como de los histogramas de color y el análisis de momentos. La fase de modelado realiza la estimación de trayectoria y la ubicación de objetos [19].

1.3.1.2. Medición de velocidad utilizando el Método visión estereoscópica (Vestro)

Este sistema se llama Vestro y usa geometría analítica estereoscópica. Su funcionamiento se basa en dos cámaras que procesan las dos imágenes adquiridas y se aplica un algoritmo

para calcular el desplazamiento realizado en un tiempo determinado, con lo cual se calcula el vector velocidad [8].

Para la realización del proceso de visión estereoscópica se presenta seis pasos principales que son:

- Adquisición de imágenes
- Modelado del cámara basado en la geometría del sistema
- Extracción de las características
- Correspondencia de las imágenes en base a sus características
- Determinación de la distancia mediante la profundidad
- Interpolación, en caso de ser necesaria

La adquisición de las imágenes se la realiza de varias formas que pueden ser de forma simultánea en el tiempo, o a través de intervalos de tiempo con una duración establecida. A partir de locaciones y direcciones que pueden ser diferentes. Como factores que pueden influir adicionalmente pueden ser las condiciones atmosféricas y un elemento cualquiera que cambie la escena que se considera en efecto [20].

1.3.1.3. Clasificación de vehículos y medición de velocidad mediante matching

En otro trabajo relacionado con visión artificial, se implementa un sistema de estimación de velocidad, clasificación de vehículos motorizados y se lo aplica a los videos adquiridos de cámaras de tráfico instaladas en autopistas.

En donde, se realizan los siguientes pasos para el diseño del algoritmo de cálculo de velocidad.

a) Se detecta al vehículo en movimiento a través de algoritmos de segmentación de fondo.

b) Se realiza la clasificación de vehículos mediante redes neuronales artificiales. Entre los diferentes vehículos que se puede clasificar se encuentran automóviles, motocicletas, camionetas, autobuses y camiones.

c) Para poder obtener la distancia se aplica un método de georreferencia utilizando imágenes obtenidas del satélite.

d) Se calcula la velocidad de los automóviles en base a los algoritmos antes mencionados [9].

Debido a que también se tiene que realizar la clasificación de los distintos vehículos su procesamiento es excesivo, así que no funciona en tiempo real, y además la clasificación se realiza con una media de las dimensiones de los vehículos.

Al realizar una comparación entre las diferentes maneras que se han implementado para poder calcular la medición de velocidad, se observa que ninguna de estas opciones puede trabajar en tiempo real, debido a que el procesamiento es alto al utilizar los diferentes algoritmos. Significa que para detectar la velocidad se necesite de un video externo al momento de procesar las imágenes. La ventaja del tercer método al utilizar redes neuronales artificiales es que: a parte de medir la velocidad de un vehículo, permite hacer un reconocimiento de este pudiendo identificar entre una moto, auto o camión basándose en las dimensiones, mientras la desventaja radica es en el alto procesamiento que conlleva.

1.3.2. TECNOLOGÍAS DE DIFERENTES RADARES VEHICULARES

1.3.2.1. Radar tipo barrera

Este radar funciona con láser, es un dispositivo que se coloca en el filo de la carretera como se observa en la Figura 1.2, contiene dos haces laser, separados por cierta distancia y mide la velocidad dependiendo del tiempo en que el vehículo entra y sale de los dos haces.

La desventaja que poseen estos radares es su alto costo y mantenimiento, además se requiere de trámites engorrosos de importación [10].

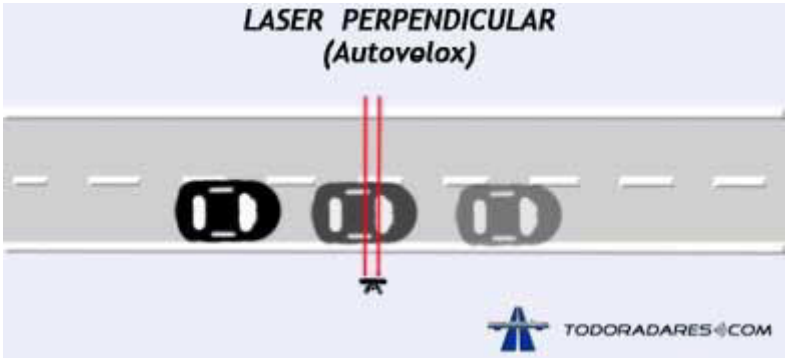


Figura 1.2 Radar laser tipo barrera [10]

1.3.2.2. Radar de tramo

Este tipo de radar consiste en un conjunto de cámaras como se muestra en la Figura 1.3 donde estas cámaras reconocen la matrícula de un vehículo al inicio y al final de un tramo, y con el tiempo que demora el vehículo en atravesarlo se calcula la velocidad [11].

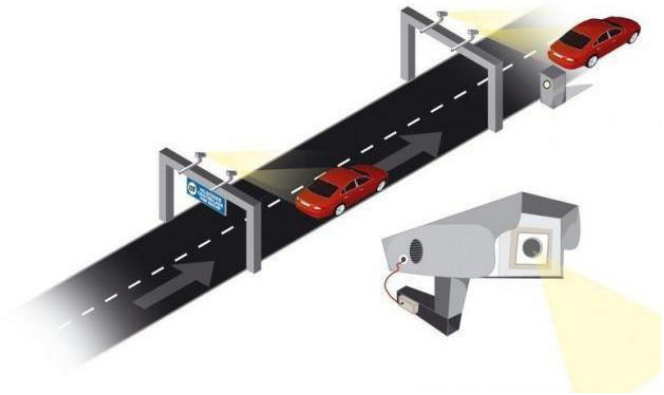


Figura 1.3 Radar de tramo [11]

1.3.2.3. Radar tipo pistola

Este radar debido a que es móvil se puede trasladar fácilmente e instalar en el interior de un auto, moto o incluso se puede utilizar un trípode como soporte para este.

El radar es un elemento compacto en el que viene incluido una cámara fotográfica, un subsistema láser, una pantalla táctil y también posee una batería como se observa en la figura 1.4. Para que funcione óptimamente, su ubicación debe estar como máximo a 200 metros. La desventaja que posee es el ángulo que tiene con respecto al vehículo ya que puede dar errores apreciables en la lectura de velocidad [12].

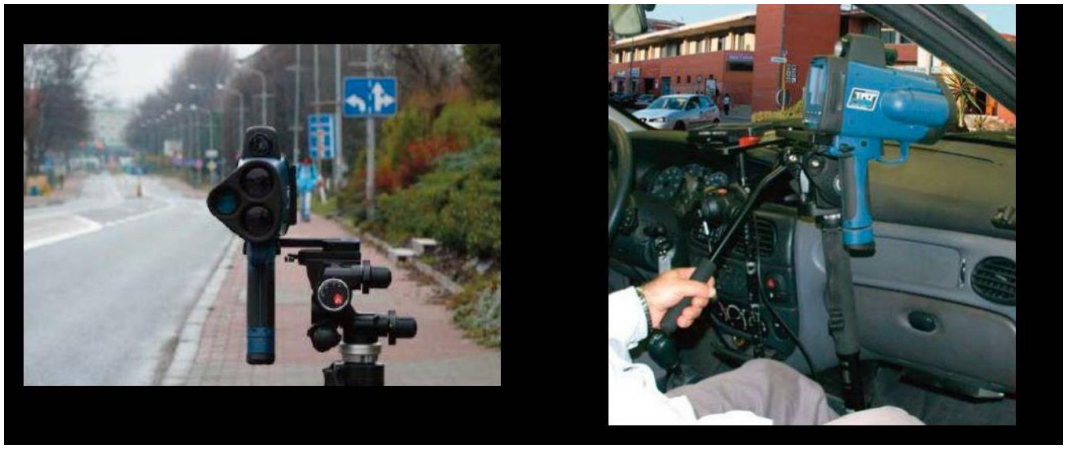


Figura 1.4 Radar tipo pistola [12]

2. METODOLOGÍA

En este capítulo se presenta el algoritmo desarrollado de medición de velocidad.

Este algoritmo pretende presentar una opción más económica y fácil de implementar en comparación con los radares tradicionales. Se implementarán dos algoritmos: en el primero se aplicará sustracción de fondo y en el segundo es un algoritmo de detección de bordes.

El algoritmo realizado es un sistema para poder medir la velocidad de un objeto en tiempo real. Para calcular la distancia se podrá contar con dos alternativas para realizar el cálculo, una será delimitando marcas físicas y la otra será utilizando un láser.

2.1. DISEÑO DEL ALGORITMO DE MEDICIÓN DE VELOCIDAD

El algoritmo consta de varias fases (Figura 2.1):

1. Adquisición de imágenes
2. Pre-procesamiento de imagen:
    a. Operaciones morfológicas
    b. Binarización de imágenes
3. Cálculo de distancia
4. Cálculo de tiempo
5. Cálculo de velocidad

En esta sección se detallan las fases para poder obtener el algoritmo final.

En la fase de cálculo de tiempo se desarrollarán los dos algoritmos ya que el proceso anterior es el mismo.

En la Figura 2.1 se muestra el procedimiento que se realizará para obtener los dos métodos para calcular la velocidad de un objeto, detallando cada una de las etapas por las cuales atraviesa el procesamiento de la señal y el cálculo de la velocidad del vehículo.

En la etapa de adquisición de imágenes se plantea revisar algunos comandos para realizar dicha acción revisando la tasa de adquisición de datos para escoger el más conveniente.

En la etapa de pre-procesamiento se rdetallan algunas operaciones morfológicas que existen para mejorar la imagen.

En la etapa de calculo de distancia cálculo de distancia se observa dos métodos para poder obtener el campo de visión de la cámara y así utilizar este dato para medir la velocidad.

En la etapa de cálculo de tiempo se realiza una introducción a los dos algoritmos utilizados: sustracción de fondo y detección de bordes, incluyendo un temporizador que inicia cuando un objeto entra en el campo de visión de la cámara.

En la etapa de cálculo de velocidad se utilizan los dos datos adquiridos anteriormente que es la distancia y el tiempo, con estos datos se puede calcular la velocidad que se actualizara cada vez que pase un nuevo objeto por el campo de visión de la cámara.

El algoritmo propuesto está constituido por 5 fases: Adquisición de imágenes, pre-procesamiento de imagen, cálculo de distancia, cálculo de tiempo y cálculo de velocidad.

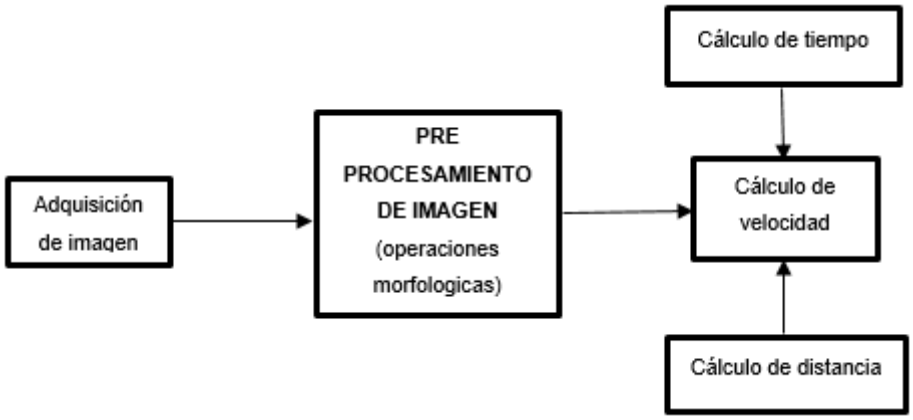


Figura 2.1 Diagrama de bloques para medición de velocidad.

Para diseñar este algoritmo se dispone de los siguientes recursos:

- Matlab 2018b con Image Acquisition Toolbox, Image Processing Toolbox, Computer Vision System Toolbox y Matlab support Package for USB Webcams.
- Ordenador Dell. Procesador core i7, 8 GB de RAM, Disco Duro de 500 GB.

- Cámara web delta de 60 fps.
- Cámara infrarroja

En la Figura 2.2 se observa el esquema de conexión de los recursos para la cámara infrarroja.

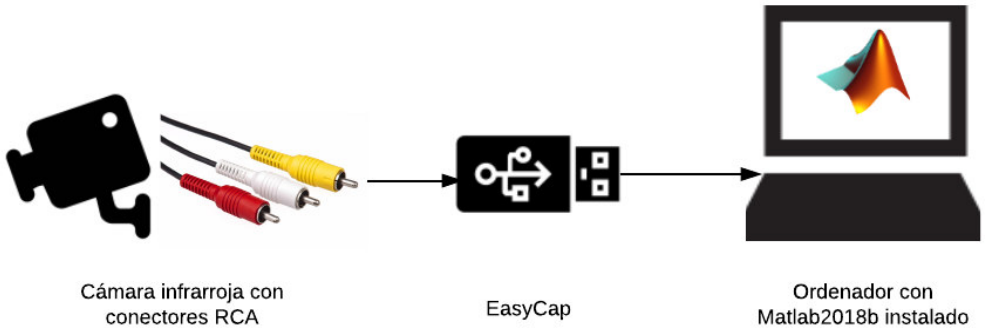


Figura 2.2 Esquema de conexión de los recursos. Se necesita un capturador de video que se conecta al ordenador mediante el puerto USB.

2.2. CONDICIONES DE FUNCIONAMIENTO

Para realizar la detección y cálculo de la velocidad del objeto se deben cumplir los siguientes requisitos:

- En un inicio no debe existir presencia del objeto a medir la velocidad.
- El objeto debe mantener una velocidad constante en el intervalo de tiempo que se encuentre en el campo de visión de la cámara
- Al inicio se debe esperar unos segundos hasta que la cámara se logre adecuar a la luz que exista en el ambiente.

2.3. FASE 1: ADQUISICIÓN DE IMÁGENES

Para empezar, el algoritmo funciona con algunas librerías adicionales para poder soportar la adquisición de imágenes obtenida desde una webcam.

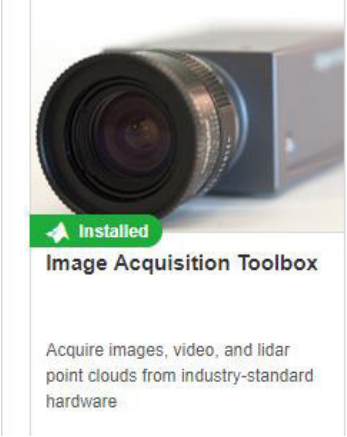


Figura 2.3 Librería Image Acquisition Toolbox [20].

Figura 2.4 Librería MATLAB Support Package for USB Webcams [20].

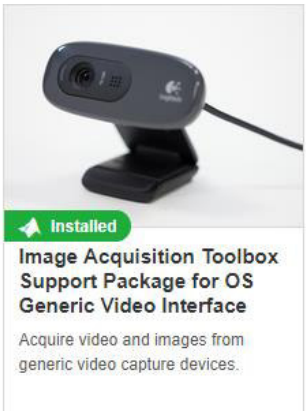


Figura 2.5 Librería Image Acquisition Toolbox Support Package for OS Generic Video Interface [20].

Estas tres librerías mostradas en las Figuras 2.3, 2.4, 2.5 se utilizarán para poder tener un adecuado procesamiento de las imágenes.

Al momento de realizar la adquisición de imágenes, se tienen dos opciones: la una es utilizar alta resolución para capturar imágenes bien definidas y la otra es utilizar baja resolución dando prioridad a la velocidad de adquisición. Ya que la aplicación está relacionada con detección del objeto en los diferentes frames, se utilizará la menor resolución posible, que es de 160x120.

Con el comando “imaqtool” se pueden obtener los diferentes formatos que poseen los dispositivos que se encuentran conectados al ordenador.

Como se puede ver en la Figura 2.6, de los diferentes formatos soportados por la cámara empleada, la que tiene menor resolución es la de formato YUY2 160x120, y es la que se utilizará para la aplicación.

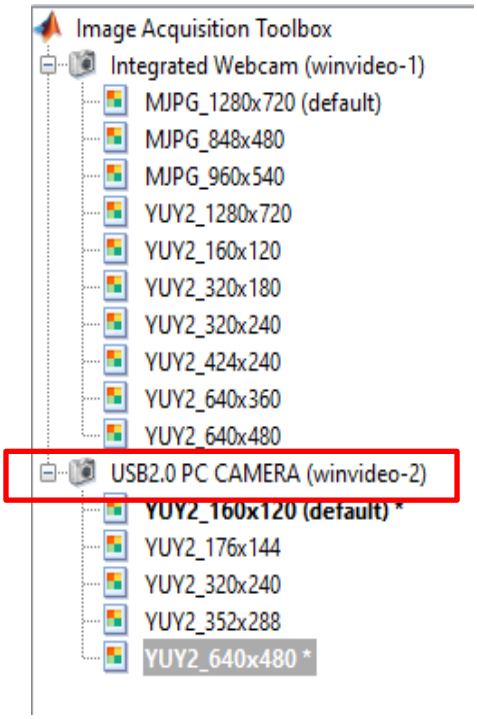


Figura 2.6 Diferentes formatos de los dispositivos conectados

Cuando se utiliza el comando imaqtool, se puede realizar una vista previa con la resolución escogida (Figura 2.7). Se puede observar además que la tasa de adquisición de imágenes es de 8.43 frames por segundo.

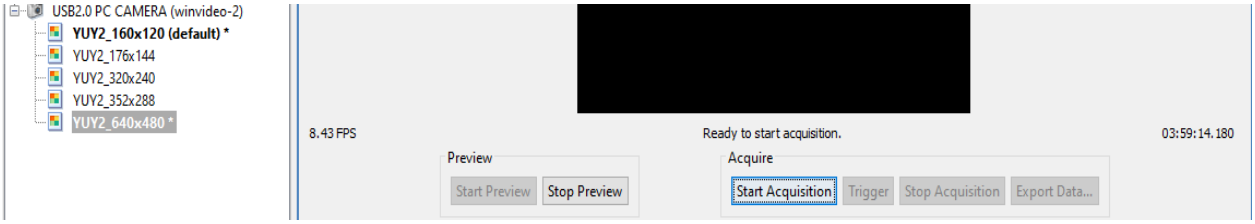


Figura 2.7 Vista previa del formato YUY_160x120

A continuación, se muestran algunos comandos para adquisición de imagen y la determinación de la cantidad de fps que se pueden obtener. Esto es posible hacerlo mediante la función "*tic toc*"

2.3.1. *Videoinput* [20]

Crea un objeto de video de entrada. Permite la adquisición de eventos en un buffer.

Sintaxis: obj=videoinput(*adaptorname*)

Donde "*adaptorname*" es el nombre del adaptador a ser usado en nuestro caso será winvideo.

Ejemplo de Script

```
cam=videoinput('winvideo', 2, 'YUY2_160x120');
frames=10;
h = figure;
subplot(1,1,1,'Parent',h); handles.img(1)=imshow(zeros(240,320));
tic
for i = 1:frames
%% Obtención de imagen
I = getsnapshot(cam);
set(handles.img(1),'CData',I);
end
toc
delete(cam)
imaqreset
```

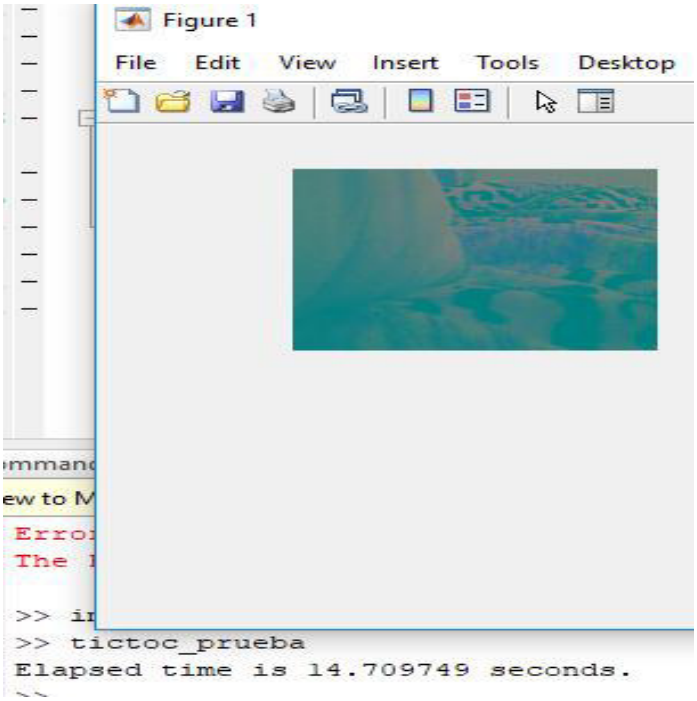


Figura 2.8 Comando Videoinput

Los resultados se muestran en la Figura 2.8 y son: 10 imágenes en 14.71 segundos; que equivale a 0,68 imágenes por segundo.

2.3.2. VideoDevice [20]

A través del commando imaq.VideoDevice(adaptorname, deviceid, format), se puede realizar la adquisición de imágenes que presentan un solo cuadro.

Script

```
cam=imaq.VideoDevice('winvideo', 2, 'YUY2_160x120');
frames=10;
h = figure;
subplot(1,1,1,'Parent',h); handles.img(1)=imshow(zeros(240,320));
tic
for i = 1:frames
%% Obtención de imagen
I = step(cam);
set(handles.img(1),'CData',I);
end
toc
release(cam)
imaqreset
```

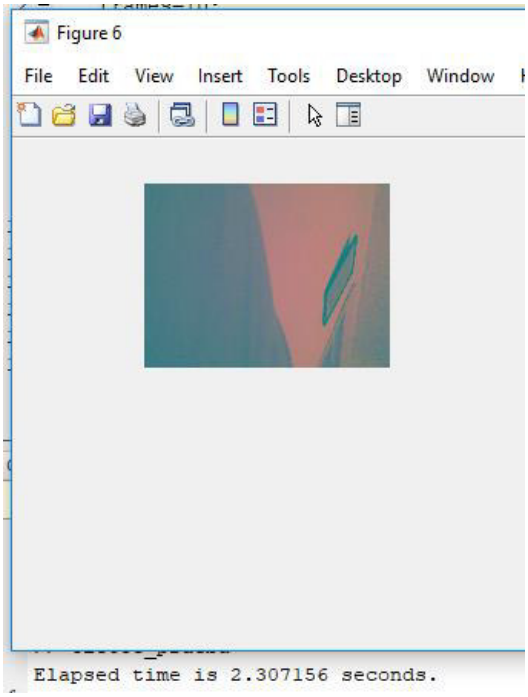


Figura 2.9 Comando VideoDevice

Los resultados se muestran en la Figura 2.9 y son: 10 imágenes en 2.3071 segundos; son 4.33 imágenes por segundo.

2.3.3. **Webcam** [20]

Script

```
cam=webcam;
cam.Resolution='160x120';
frames=100;
h = figure;
subplot(1,1,1,'Parent',h); handles.img(1)=imshow(zeros(120,160));
tic
for i = 1:frames
%% Obtención de imagen
I = snapshot(cam);
set(handles.img(1),'CData',I);
end
toc
```

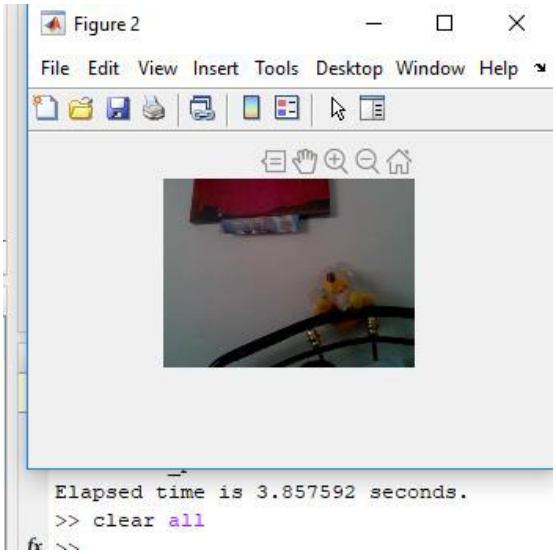


Figura 2.10 Ejecución de Comando Webcam.

Como se muestra en la Figura 2.10 se tiene 100 imágenes en 3,86 segundos; con lo que tendrían 25,91 imágenes por segundo.

El máximo número de fps que se puede adquirir con la cámara que se va a trabajar es de 100 imágenes en 3,86 segundos; es decir, 25,91 imágenes por segundo(fps).

2.4. FASE 2: PRE-PROCESAMIENTO DE IMAGEN

Antes de utilizar los algoritmos respectivos se pre-procesara la imagen para poder mejorarla y así no tener demasiado ruido en ella.

Los procesos que se aplicarán en este bloque son: erosión, open, close, que se detallan brevemente a continuación.

La dilatación morfológica define como incorporar pixeles cercanos al objeto.

La erosión se define como lo contrario de la dilatación morfológica, es decir, es una reducción de la imagen original, disminuyendo así el ruido que puede aparecer.

Con estas operaciones lo que cambia es el volumen del objeto por lo que se puede combinar estas operaciones.

Para el algoritmo se utilizará apertura y cierre para poder mejorar la imagen y poder procesarla adecuadamente.

En la apertura (open) primeramente se realiza erosión y luego una dilatación, mientras que en el cierre (close) se realiza primero una dilatación y luego erosión. Con el cierre se puede borrar las irregularidades del borde de una imagen.

2.5. FASE 3: CÁLCULO DE DISTANCIA

2.5.1. PROCEDIMIENTO PARA TRABAJAR CON MARCAS FÍSICAS

Debido a que el programa propuesto cuenta con un contador que se inicia cuando detecta un cambio en los frames para poder calcular la velocidad, también se debe tener como dato la distancia que recorre el objeto en ese intervalo de tiempo.

Desde el punto de vista de la cámara se vería como se muestra en la Figura 2.11. Lo que se realiza, es medir la distancia que se marca en el campo de visión de la cámara, todo lo que abarca el largo de la imagen adquirida por la cámara. Con lo que dependería de la distancia medida manualmente y no del ángulo de la cámara.

Figura 2.11 Posición de la cámara

También se puede tener otros ángulos como se muestra en la Figura 2.12 para poder medir la velocidad. Donde se mide la distancia que se tiene en el recuadro como se muestra a continuación.

Figura 2.12 Diferente ángulo de cámara [13]

La desventaja de este método es que tomaría más tiempo el procedimiento, porque la distancia se tendría que medir manualmente con flexómetro, y con ayuda de una persona para poder tener la distancia del campo de visión de la cámara.

Con este método se tiene que fijar la cámara y realizar las mediciones de distancia respectivas.

Ya que el programa implementado es un contador que detecta el movimiento cuando entra y sale de la cámara un automóvil, queda demostrado con estos ejemplos que para el cálculo de la velocidad de un vehículo no depende del ángulo de la cámara. Lo único que se necesita es medir la distancia que va a recorrer el vehículo dentro del recuadro en un determinado tiempo y realizar los cálculos con esos parámetros.

2.5.2. PROCEDIMIENTO ALTERNATIVO PARA TRABAJAR SIN DELIMITAR MARCAS FÍSICAS

Todas las cámaras trabajan con un ángulo de visión como se muestra en la Figura 2.13.

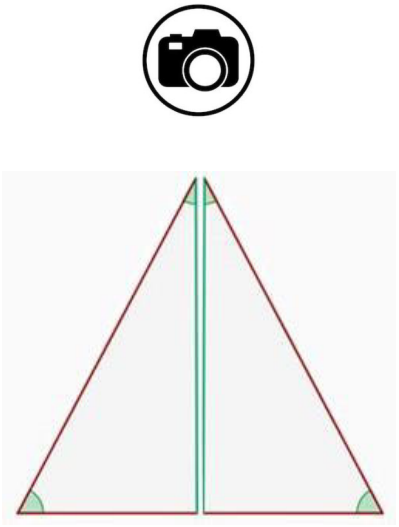


Figura 2.13 Angulo de visión de la cámara

Al considerar este ambiente, realizando las pruebas respectivas, se puede calcular el ángulo de visión de la cámara y con este ya obtenido como dato, se logra obtener la distancia del campo de visión de la cámara, evitando medir manualmente con un flexómetro.

Para poder realizar las pruebas respectivas de esta técnica de trabajar sin marcas físicas se utiliza un medidor de distancia láser debido a que agilita el proceso y se lo realiza más rápidamente que estar que al realizar la medición con un flexómetro.

Este medidor que se muestra en la Figura 2.14, de distancia laser Bosh puede abarcar hasta 20 metros de distancia y su costo es de 40 dólares.

Figura 2.14 Medidor de distancia Bosh

Este medidor consta de las características mostradas a continuación en la Tabla 2.1 detalladas en el manual del producto.

Tabla 2.1 Datos técnicos del medidor de distancia laser

Datos Técnicos	**Valores**
Diodo láser	635 nm, < 1 mW
Rango de medición	0,15 – 20,00
Peso, aprox.	0,13 kg
Tiempo de medición, habitual	< 0,5 s
Margen de medición, de	0,15 m
Margen de medición, hasta	20 m
Clase de láser	2
Precisión de medición, habitual	±3,0 mm
Fuente de alimentación	2 x 1,5 V LR03 (AAA)
Desconexión automática	5 mín.
Unidades de medición	m/cm, ft/in
Vida útil de las baterías, número de mediciones aprox.	5.000
Vida útil de las baterías, tiempo de servicio aprox.	5 h

A continuación, se muestra el procedimiento experimental para poder obtener el ángulo de la cámara:

Primeramente, se coloca marcas en una pared y estas coinciden con el recuadro de la cámara a utilizar como se muestra en la Figura 2.15, 2.16.

Figura 2.15 Marcas que coinciden con campo de visión de la cámara

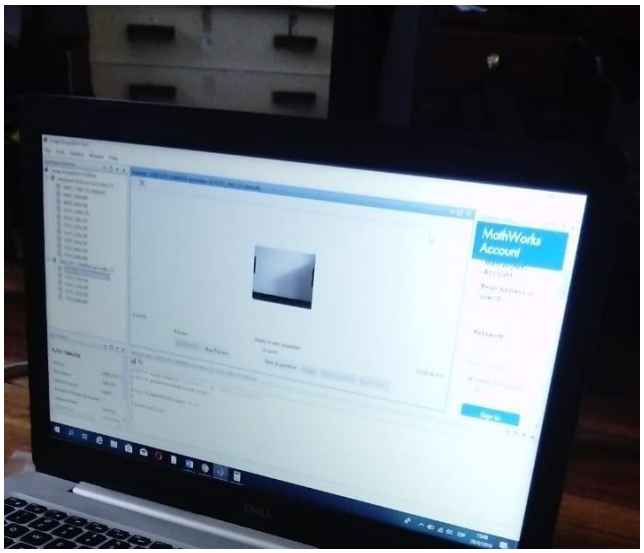


Figura 2.16 Marcas vistas desde el ordenador

Se midió la distancia de la marca izquierda a la marca derecha dando como resultado un valor de 0.491 metros.

Esto se realizó poniendo un cartón en la marca izquierda como tope para poder medir con el láser como se muestra en la Figura 2.17.

d_marcas=0.491 m

Figura 2.17 Distancia medida de una marca hacia la otra

Luego se calculó de la distancia de la cámara al centro de las marcas como se muestra en la Figura 2.18 dando como resultado 0.869 m.

d_laser=0.867 m

Figura 2.18 Distancia medida al centro de las marcas

A continuación, en la Figura 2.19, se muestran los resultados obtenidos.

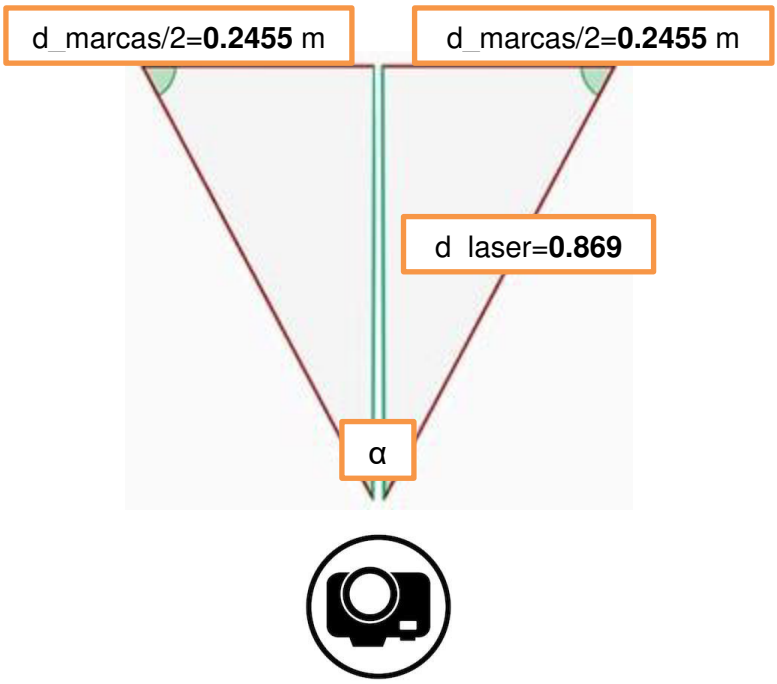


Figura 2.19 Resultados obtenidos de las mediciones

Como se puede observar se tiene un triángulo rectángulo con lo que aplicando funciones trigonométricas se puede calcular el ángulo de visión de la cámara.

$$tan\left(\frac{\alpha}{2}\right) = \frac{0.2455}{0.867}$$

$$tan\left(\frac{\alpha}{2}\right) = 0.2831$$

$$\left(\frac{\alpha}{2}\right) = \tan^{-1} 0.2831$$

$$\left(\frac{\alpha}{2}\right) = 15.81$$

$$\alpha = 2 * 15.81$$

$$\alpha = 31.62$$

Se muestra una prueba utilizando el medidor de distancia laser, para poder calcular la distancia del recuadro de la cámara.

2.5.3. Método manual (medición de distancia del campo visual de la cámara)

La distancia medida manualmente del recuadro es de 6.22 m como se observa en la Figura 2.20.

Figura 2.20 Distancia medida de forma manual.

Método automático basado en láser

La distancia medida con el medidor láser desde la cámara hasta el centro del recuadro es de 11.84 m (Figura 2.21).

Figura 2.21 Distancia medida al centro del recuadro de la cámara

d_laser= 11.84 m

En la Figura 2.22 se muestran los datos obtenidos.

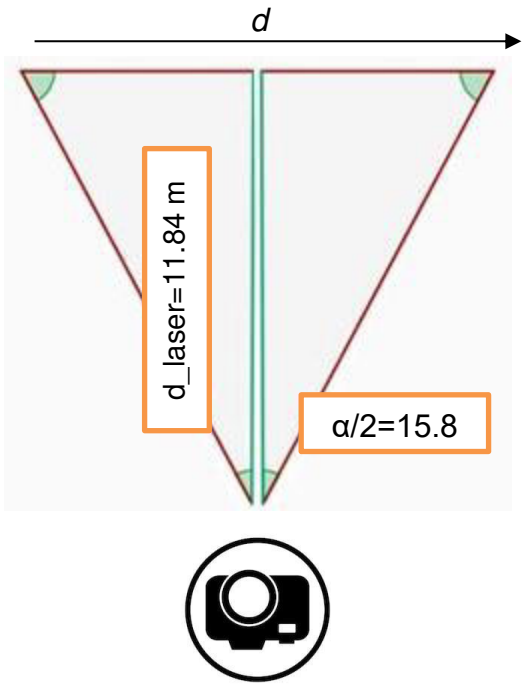


Figura 2.22 Resultados obtenidos (método basado en laser)

Dados estos datos se puede calcular la distancia del recuadro de la cámara de la siguiente manera.

$$\frac{d}{2} = \tan(15.81) * 11.84\, m$$

$$\frac{d}{2} = 3.35\, m$$

$$d = 6.70\, m$$

Dado que la distancia media manualmente fue de 6.22 el error es de 7.71%. Este error se puede dar debido a que el láser no es estable y se mueve bastante mientras la distancia a medir es más grande.

2.6. FASE 4: CÁLCULO DE TIEMPO

2.6.1. MÉTODO 1: SUSTRACCIÓN DE FONDO

El método de sustracción de fondo o también llamado ("background subtraction") se lo utiliza bastante para realizar detección de objetos mediante la diferencia entre un conjunto de pixeles y otro de referencia, este conjunto de pixeles de referencia es también llamado imagen de background o imagen de fondo. [14]

Al aplicar esta técnica existen áreas donde la diferencia de pixeles es significativa, como al cambiar de color e indicando que el objeto ha sufrido movimiento como se observa en la Ecuación 2.1:

$$|\text{Background}_{\,t} - \text{Frame}_t| > \tau \qquad (2.1)$$

donde $\boldsymbol{\tau}$ es un umbral predefinido.

Existen algunos métodos para separar la imagen de fondo de los objetos que pueden aparecer. Entre los métodos más representativos principales tenemos:

- Métodos básicos

- Running Average

- Running Gaussian Average

- Mezcla de Gaussianas (Mixture of Gaussians)

- Estimación de densidad del Kernel ("Kernel Density Estimation")

- Estimación basada en la técnica mean-shift

- Aproximación secuencial de densidad del Kernel ("Sequential Kernel Density Approximation)

- Coocurrencias de variaciones en la imagen - Autobackgrounds ("Eigenbackgrounds")

Se detalla a continuación los métodos básicos para la aplicación de un sistema medición de velocidad de un objeto con sus respectivas características.

2.6.1.1. Métodos básicos

2.6.1.1.1. Diferencia entre píxeles

El background se actualiza continuamente del instante anterior al actual como se muestra en la Ecuación 2.2:

$$B_t = F_{t-1} \tag{2.2}$$

donde F_t son los pixeles de la imagen en un instante t y B_t son los pixeles de la imagen de background (fondo).

Cada determinado tiempo t, si se cumple la siguiente desigualdad, F_t se convierte en un pixel denominado de foreground, eso significa que existe cambio en los pixeles.

$$|B_t - F_t| > \tau \tag{2.3}$$

Si la desigualdad de la Ecuación 2.3 no es cumplida, F_t es denominada como pixel de background, lo que significa que no hubo cambio en los pixeles.

El proceso es descrito en la Figura 2.23.

-Primeramente, se toma como background la primera imagen y después, compararla con las siguientes imágenes que se van obteniendo, ahí se comprueba si la desigualdad se cumple los pixeles son foreground (objeto) mientras que si no se cumple los pixeles son denominados background.

- Si un pixel F_t es foreground no aplica el modelo de background, es decir:

$$B_t = B_t - 1 \tag{2.4}$$

Este pixel F_t se guarda como pixel de objeto Como se muestra en la Ecuación 2.5 .

$$(Ot = F_t) \tag{2.5}$$

- Si un pixel Ft es denominado como background, entonces la imagen de fondo background se actualiza. Como se observa, el método es muy sensible al umbral τ, debido a que si se varia este valor la sustracción de fondo puede ser más eficiente [14].

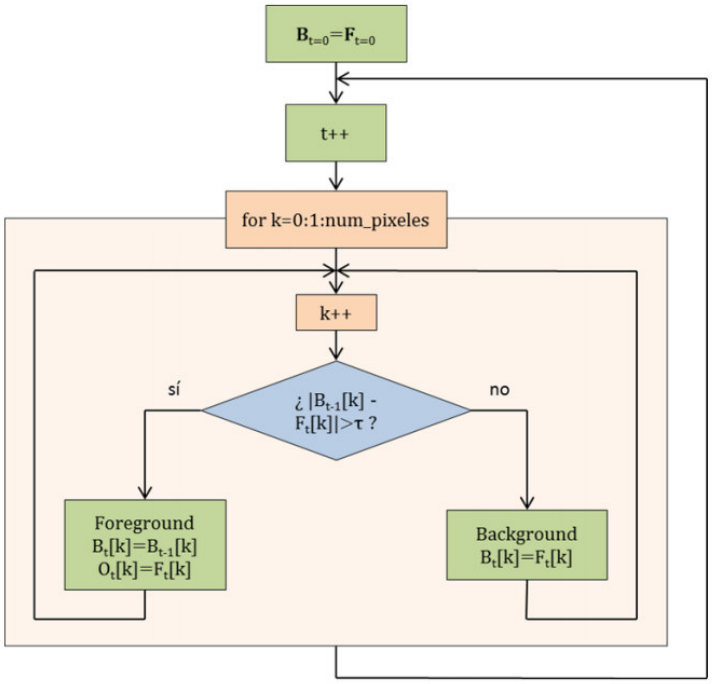


Figura 2.23 Diagrama de flujo del método diferencia entre pixeles [14]

2.6.1.1.2. Media de n píxeles

El background se actualiza en cada interacción como la media de los pixeles en n instantes consecutivos como se observa en la Ecuación 2.6:

$$Bt = media\ F_t, F_t{+}1\ , \ldots , F_t{+}n \tag{2.6}$$

donde F_t son los pixeles de la imagen en un instante t y B_t son los pixeles de la imagen de background (fondo).

Cada determinado tiempo t, si se cumple la siguiente desigualdad mostrada en la Ecuación 2.7, F_t se convierte en un pixel denominado de foreground, eso significa que existe cambio en los pixeles.

$$|\, Bt - F_t| > \tau \tag{2.7}$$

Si la desigualdad no es cumplida, F_t es denominada como pixel de background, lo que significa que no hubo cambio en los pixeles.

El procedimiento de este método se muestra en la siguiente Figura 2.24:

- Primeramente, se toma como imagen de fondo background la media de las n primeras series de puntos. Luego, con la siguiente serie de puntos pixeles se comprueba la desigualdad y se determina si los pixeles son foreground (objeto en movimiento) o background.

- Si un pixel F_t se clasifica como foreground entonces se ignora en el modelo de background (B= B). Este pixel F_t se almacena como pixel objeto como se describe en la Ecuación 2.8.

$$(Ot = F_t) \tag{2.8}$$

- Si un pixel F_t es denominado como background, significa que la imagen de fondo background se actualiza.

La desventaja de este método es que necesita almacenar n conjunto de puntos, lo que significa, que utiliza más memoria, lo que no conviene para nuestra aplicación [14].

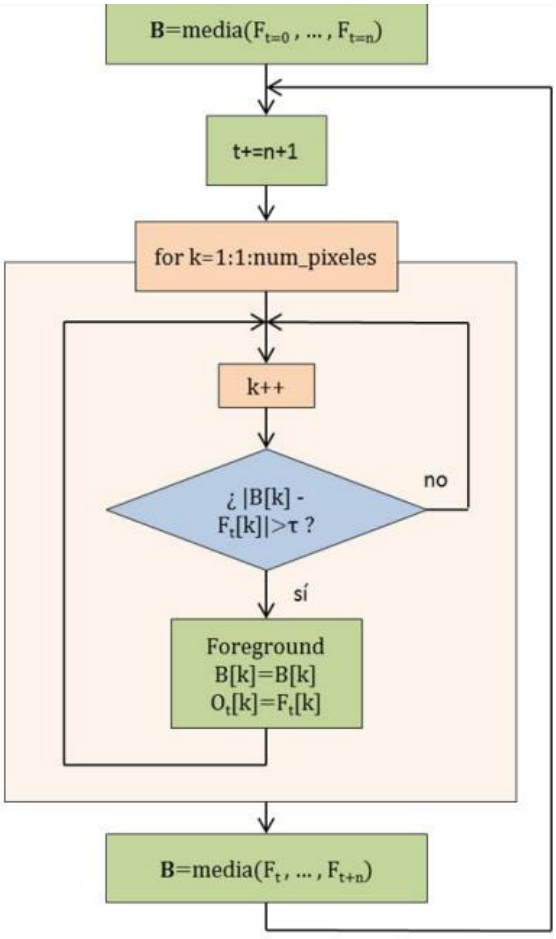


Figura 2.24 Diagrama de flujo del método media de n-píxeles [14]

2.6.1.1.3. Mediana de n píxeles

Este método es similar al anterior, pero en vez de calcular la media de los pixeles en n instantes de tiempo, se calcula la mediana. Esto es, el background se actualiza mediante la Ecuación 2.9 [14].

$$Bt = mediana\,(F_t, F_t{+}1\ , \ldots , F_t{+}n) \tag{2.9}$$

Se detalla a continuación el código utilizado línea por línea para realizar el medidor de velocidad de un objeto mediante sustracción de fondo.

Código utilizado

% Cierra todo, limpia el workspace

clc, clear all, close all

Desconecta y borra todos los objetos de la adquisición de imágenes

***Problema**:* Cada vez que se adquiere imágenes no son eliminados con los comandos anteriores, entonces aparece un mensaje de error**.**

***Solución**:* Eliminar las imágenes previamente adquiridas lo cual se puede hacer a través del comando "imaqreset".

El comando "*imaqreset"* elimina cualquier adquisición de imágenes que se encuentre en la memoria y actualiza los adaptadores cargados por el tollbox.

Imaqreset

Determinación de los dispositivos de adquisición de imágenes actualmente conectados al computador

Problema: Puede haber varias cámaras conectadas al computador y necesitamos determinar cuál de ellas vamos a usar en nuestro proyecto.
Solución: Emplear una función que nos dé esta información.

Ya que las últimas versiones de matlab no tienen la librería para utilizar winvideo, se debe bajar la librería del internet teniendo una cuenta para el caso.

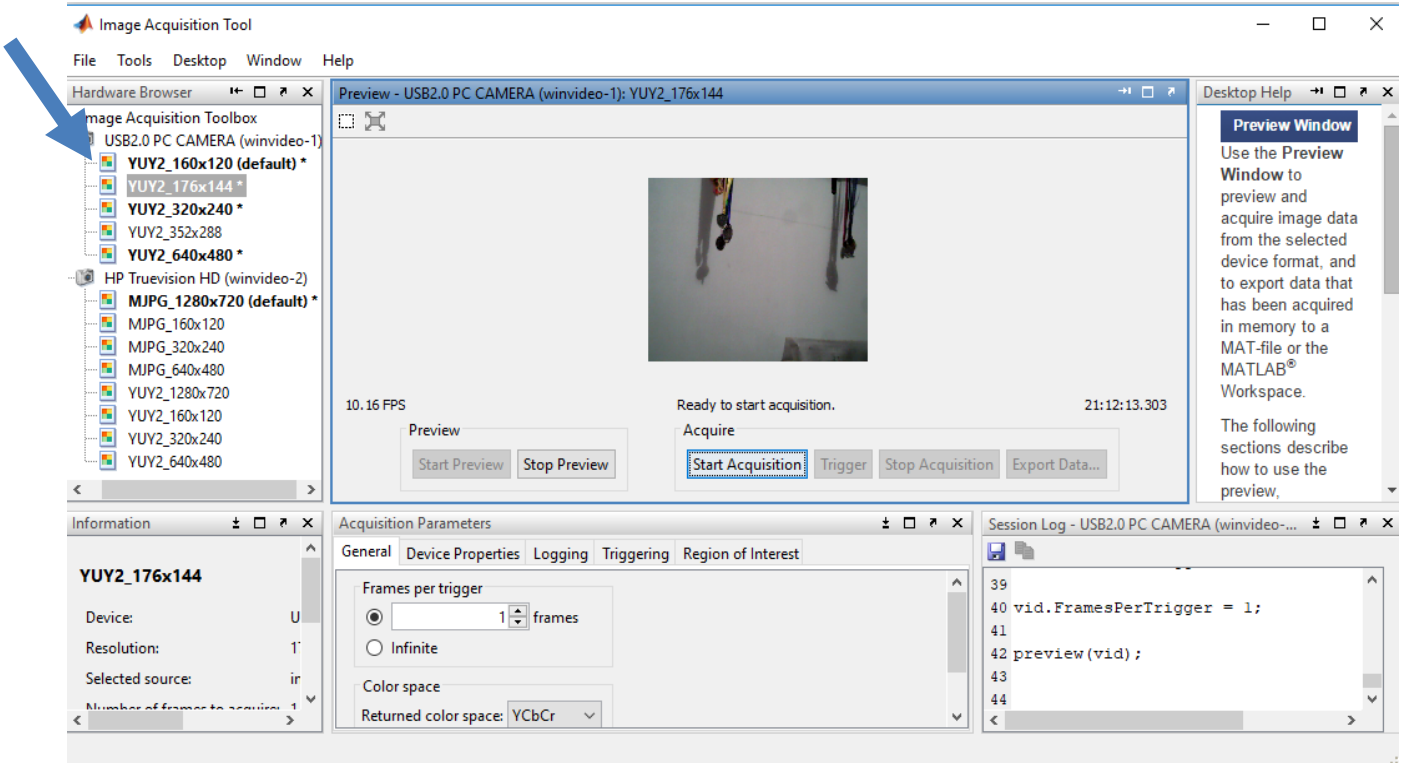


Figura 2.25 Comando imaqtool

Existe un comando llamado "*imaqtool*" con el cual se puede observar todos los formatos y resolución que tiene cada dispositivo de video como se muestra a continuación.
Hay dos opciones para realizar el proyecto propuesto: la una es utilizar alta resolución para capturar imágenes bien definidas y la otra utilizar de baja resolución dando prioridad a la velocidad de adquisición. En el presente caso se utiliza menor resolución porque se necesita velocidad de procesamiento antes que calidad.

Al escoger un formato se genera un código donde consta la resolución utilizada y se puede hacer una prueba para comprobar las diferentes resoluciones del dispositivo como se observa en la Figura 2.25. Permite crear un objeto empleando la cámara y el formato anteriormente seleccionados.

2.6.1.1.4. Creación de un objeto

Videoinput

Permite crear un objeto empleando la cámara y el formato anteriormente seleccionados.

La estructura del comando es la siguiente:

obj = videoinput (adaptorname, deviceID, format)

Construye un objeto de entrada de video, donde formato es un vector de caracteres que especifica un formato de video particular admitido por el dispositivo o la ruta completa de un archivo de configuración de dispositivo (también conocido como archivo de cámara).

El formato es el que sirve para transmitir la imagen en directo desde el dispositivo a una computadora, MJPEG o YUY2. El códec MJPEG utiliza más de compresión, lo que resulta en una imagen de menor calidad y tasa de cuadros más alta. El YUY2 utiliza menos de compresión, lo que resulta en una imagen de calidad más alta y una velocidad de trama menor.

En el caso práctico no se necesita tener buena resolución y más bien es un problema de velocidad de adquisición de imágenes por esta razón se escoge la mínima que es 160x120.

Se crea un "*videoinput*" con una resolución a color de 160x120, ya que si se aumenta la resolución el video empieza a entre cortarse más, y se hace más lento debido a que se realiza mayor procesamiento.

Comando de Matlab:

```
vid=videoinput('winvideo',1, 'YUY2_160x120');
```

Confirmación de que se utilizó la resolución escogida.

Obtiene las dimensiones del video. En este caso es 160x120 y para comprobar se los guarda en la variable "reso".

Comando de Matlab:

```
reso=vid.VideoResolution;
```

```
>> reso

reso =

   160   120
```

Figura 2.26 Confirmación de la resolución utilizada en Command Window de Matlab.

2.6.1.1.5. Configuración básica del objeto

La propiedad "*FrameGrabInterval*" [20] especifica con qué frecuencia el objeto de entrada de video adquiere un marco de la secuencia de video. De forma predeterminada, los objetos adquieren cada cuadro en la secuencia de video, pero puede usar esta propiedad para especificar otros intervalos de adquisición.

Por ejemplo, cuando especifica un valor de "*FrameGrabInterval"* de 3, el objeto adquiere cada tercer fotograma de la secuencia de video, como se ilustra en esta Figura 2.27 de la ayuda que ofrece Matlab. El objeto adquiere el primer fotograma en la transmisión de video antes de aplicar "*FrameGrabInterval"***.**

Comando de Matlab:
vid.FrameGrabInterval = 1;

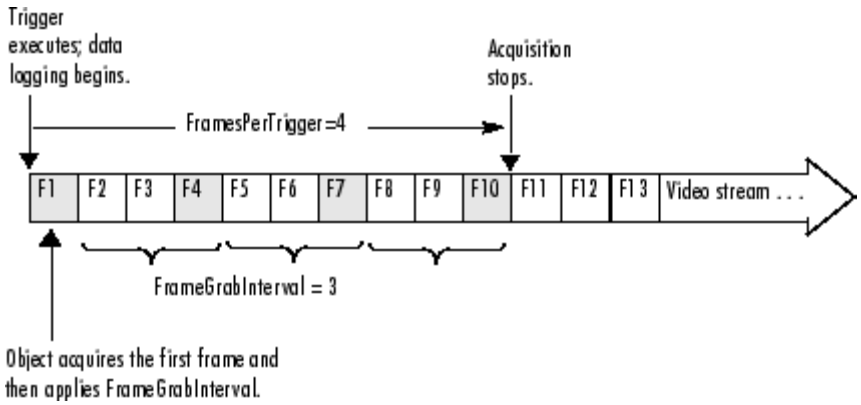


Figura 2.27 "FrameGrabInterval"

Para este caso se aplicó el valor de 1 para obtener todos los fotogramas posibles.

Cuando el valor de la propiedad "FramesPerTrigger" se establece en Inf, el objeto continúa adquiriendo marcos hasta que se produce un error o se emite un comando de detención.

Comando de Matlab:
set(vid, 'FramesPerTrigger', Inf);

La propiedad "ReturnedColorSpace" especifica el espacio de color que desea que use toolbox cuando devuelve datos de imagen al MATLAB workspace.

Comando de Matlab:
set(vid,'ReturnedColorspace', 'rgb')

Inicio del video

Para indicar el inicio de la captura de imagen principal, se coloca un "disp" e inicia el video con el comando "*start()"* con la adquisición de las imágenes.Comando de Matlab:
disp('Obtener imagen original')
start(vid);

Adquisición imágenes del video.

Con "*getdata"* se puede obtener imágenes en el workspace de Matlab [20].

data = getdata(obj,n), devuelve n marcos de datos asociados con el objeto de entrada de video en este caso vid. Se utiliza el valor de 1 para que solo tome una imagen que será de 120*160.

Comando de Matlab:
data = getdata(vid,1); %Adquiere imagenes del video.

Data contiene la imagen original de 120*160 en formato unit8 como se muestra en la Figura 2.28 a continuación.

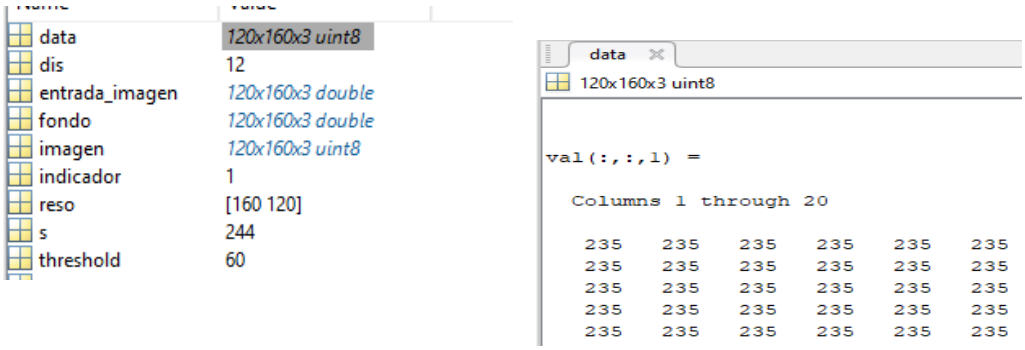


Figura 2.28 La variable data que contiene todo el primer fotograma

Conversión a formato double de la variable data e indica la terminación de obtención de la imagen original.

Comando de Matlab:
fondo=double(data(:,:,:)); %convierte a formato double
disp('Fin obtencion de imagen original')

Definición de un contador llamado s para poder simular el tiempo donde se inicia con el valor de 0.

El "threshold" es un valor que se toma como referencia al momento de realizar la resta de frames, ya que prácticamente se resta la matriz de pixeles y este valor se obtuvo mediante experimentación para que detecte solo los cambios de pixeles grandes en la imagen.

Comando de Matlab:
threshold=60;
figure(1);
s=0;

2.6.1.1.6. Inicio del lazo

Indicación del número total de fotogramas

"FramesAcquired" indica el número total de fotogramas que el objeto ha adquirido, mientras sea menor a 350 fotogramas se mantendrá el lazo, después saldrá del lazo y se detendrá la adquisición de imágenes, ya que si no se restringe seguirá tomando datos indefinidamente y se guardaran en el buffer.

Comando de Matlab:
while (vid.FramesAcquired<350)

Obtención de una imagen de 120*160.

Obtiene una imagen de 120*160 y la muestra posteriormente. Para ello se utiliza el comando “getdata()”.

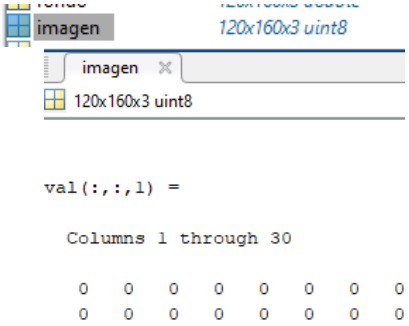


Figura 2.29 Variable imagen antes de ser convertida a formato double en Workspace.

Comando de Matlab:
imagen = getdata(vid,1);

En donde, “vid” es el objeto de entrada de video y n el número de “frames”.

Observación de la imagen

Mediante la función “imagesc(C)” se tiene que la matriz C se la puede mostrar como una imagen que presenta todo el rango de colores. En donde cada elemento de C representa a 1 pixel de la imagen [20].

Comando de Matlab:
imagesc(imagen);
title('Radar de velocidad')

Conversión de la imagen a tipo double [20]

Se emplea la función double(x), en donde x es el argumento de entrada a convertir.

Comando de Matlab:
entrada_imagen = double(imagen);

Utilización de la función comparet para comparar las imágenes.

Utiliza una función (“compare(data1,data2,…)”) para realizar la diferencia entre la imagen que se obtuvo al principio y las posteriores, se envían parámetros como la imagen de entrada, las imágenes que se siguen obteniendo en tiempo real, el límite y el contador s para poder calcular el tiempo. Esta función tiene como parámetros de salida el indicador y

el contador. Ya que el contador debe estar dentro de la función para que solo funcione cuando detecta movimiento.

Comando de Matlab

```
[indicador,s] = comparet(entrada_imagen,fondo,threshold,s);

pause(0.01);
end
```

Detener y borrar la variable vid

Para detener el video se utiliza el comando “stop()”. Para borrar el contenido de la variable se utiliza el comando “clc”.

Fragmento de código:

```
stop(vid); %Detiene el video
   delete vid %Borra la variable vid
   clear vid %
% catch
%     stop(vid);
%     disp('Sucedió un ERROR')
% end
Clc

disp('FIN') %Muestra el fin del programa.
% Desconecta y borra toda la adquisicion de objetos
imaqreset
```

2.6.1.1.7. Función Compare

Definición parámetros de entrada y salida de la función.

La función “compare()” [20] tiene como parámetros de entrada la imagen original, las imágenes que va tomando después, el contador t para poder obtener el tiempo en segundos y threshold que es un valor límite para la diferencia. Como parámetros de salida tiene un indicador que indica el inicio y fin del bucle y el temporizador s.

Fragmento de Código:

```
    function [indicador,t]=compare(entrada_imagen,fondo,threshold,t)%Se asignan las
    entradas de la funcion
indicador = 0;
```

Resta de imágenes para obtener las diferencias

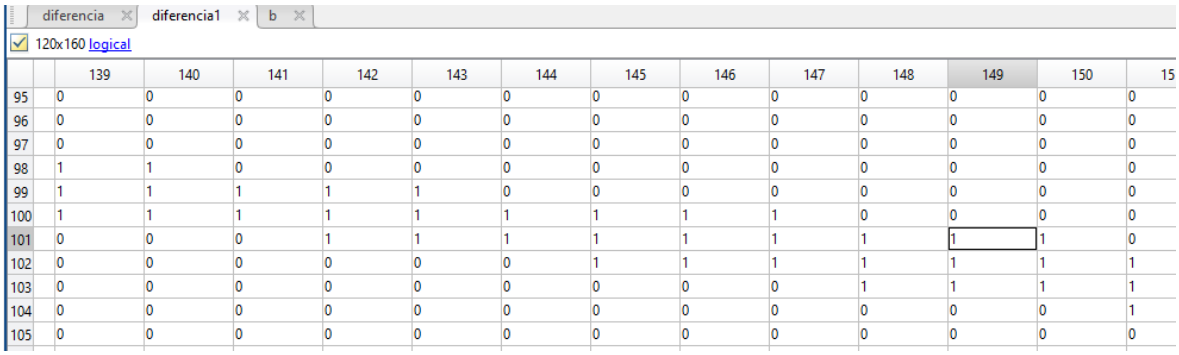
“Threshold” es un valor límite donde la resta de toda la matriz de la original menos las imágenes que se siguen obteniendo debe ser mayores a 60.

Si es que la resta es menor a 60 se obtiene un 0 lógico.

Mientras que si la resta es mayor a 60 se obtiene un 1 lógico como se muestra en la figura 2.30.

Fragmento de código:

```
diferencia = (abs(entrada_imagen(:,:,1)-fondo(:,:,1)) > threshold) |
(abs(entrada_imagen(:,:,2) - fondo(:,:,2)) > threshold) ...
        | (abs(entrada_imagen(:,:,3) - fondo(:,:,3)) > threshold);
% Elimina el ruido (eliminando pequeños agujeros y rellena las aberturas)
% a = bwlabel(diference,8);
```



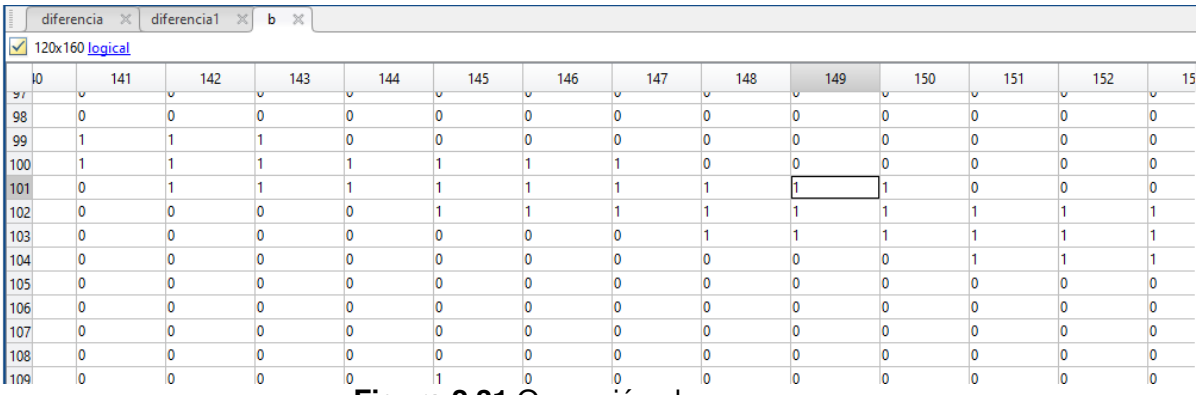
	139	140	141	142	143	144	145	146	147	148	149	150	15
95	0	0	0	0	0	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0	0	0	0	0	0
97	0	0	0	0	0	0	0	0	0	0	0	0	0
98	1	1	0	0	0	0	0	0	0	0	0	0	0
99	1	1	1	1	1	0	0	0	0	0	0	0	0
100	1	1	1	1	1	1	1	1	1	0	0	0	0
101	0	0	0	1	1	1	1	1	1	1	1	1	0
102	0	0	0	0	0	0	1	1	1	1	1	1	1
103	0	0	0	0	0	0	0	0	0	1	1	1	1
104	0	0	0	0	0	0	0	0	0	0	0	0	1
105	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 2.30 Sustracción de fondo. Datos en Workspace.

2.6.1.1.8. Utilización de operaciones morfológicas

bwmorph [20]
Aplica una operación morfológica específica a la imagen

Fragmento de código:
BW2 = bwmorph(BW,operation), binaria BW.



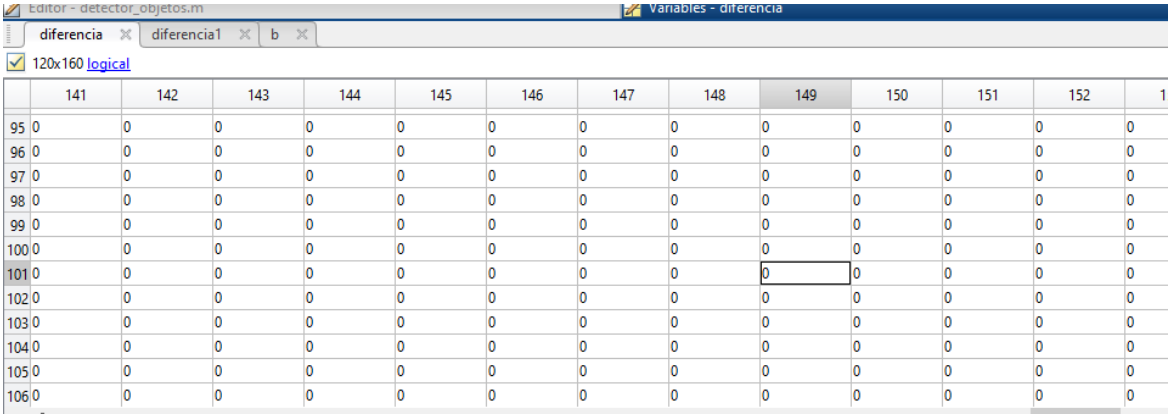
	141	142	143	144	145	146	147	148	149	150	151	152	15
98	0	0	0	0	0	0	0	0	0	0	0	0	0
99	1	1	1	0	0	0	0	0	0	0	0	0	0
100	1	1	1	1	1	1	1	0	0	0	0	0	0
101	0	1	1	1	1	1	1	1	1	1	0	0	0
102	0	0	0	0	1	1	1	1	1	1	1	1	1
103	0	0	0	0	0	0	0	1	1	1	1	1	1
104	0	0	0	0	0	0	0	0	0	0	1	1	1
105	0	0	0	0	0	0	0	0	0	0	0	0	0
106	0	0	0	0	0	0	0	0	0	0	0	0	0
107	0	0	0	0	0	0	0	0	0	0	0	0	0
108	0	0	0	0	0	0	0	0	0	0	0	0	0
109	0	0	0	0	1	0	0	0	0	0	0	0	0

Figura 2.31 Operación close

Comando de Matlab:

```
b = bwmorph(diferencia,'close');
```

Se realiza una apertura morfológica con open (erosión seguida de dilatación) y con erode realiza una erosión.



	141	142	143	144	145	146	147	148	149	150	151	152	1
95	0	0	0	0	0	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0	0	0	0	0	0
97	0	0	0	0	0	0	0	0	0	0	0	0	0
98	0	0	0	0	0	0	0	0	0	0	0	0	0
99	0	0	0	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0	0	0
101	0	0	0	0	0	0	0	0	0	0	0	0	0
102	0	0	0	0	0	0	0	0	0	0	0	0	0
103	0	0	0	0	0	0	0	0	0	0	0	0	0
104	0	0	0	0	0	0	0	0	0	0	0	0	0
105	0	0	0	0	0	0	0	0	0	0	0	0	0
106	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 2.32 Operación open

Comando de Matlab:

```
diferencia = bwmorph(b,'open');
diferencia = bwmorph(diferencia,'erode',2);
% Seleccione el objeto más grande
```

Etiquetación de una imagen [20]

Para el proceso de etiquetado de componentes se tiene el comando “”Bwlabel”.
Bwlabel.- permite etiquetar los componentes conectados en una imagen binaria bidimensional

```
BW = lógico ([1 1 1 0 0 0 0 0
              1 1 1 0 1 1 0 0
              1 1 1 0 1 1 0 0
              1 1 1 0 0 0 1 0
              1 1 1 0 0 0 1 0
              1 1 1 0 0 0 1 0
              1 1 1 0 0 1 1 0
              1 1 1 0 0 0 0 0]);
```

Crea la matriz de etiquetas usando objetos conectados a 4.

```
L = bwlabel (BW, 4)
```

```
L = 8 × 8

     1 1 1 0 0 0 0 0
     1 1 1 0 2 2 0 0
     1 1 1 0 2 2 0 0
     1 1 1 0 0 0 3 0
     1 1 1 0 0 0 3 0
     1 1 1 0 0 0 3 0
     1 1 1 0 0 3 3 0
     1 1 1 0 0 0 0 0
```

Figura 2.33 Ejemplo de etiquetamiento

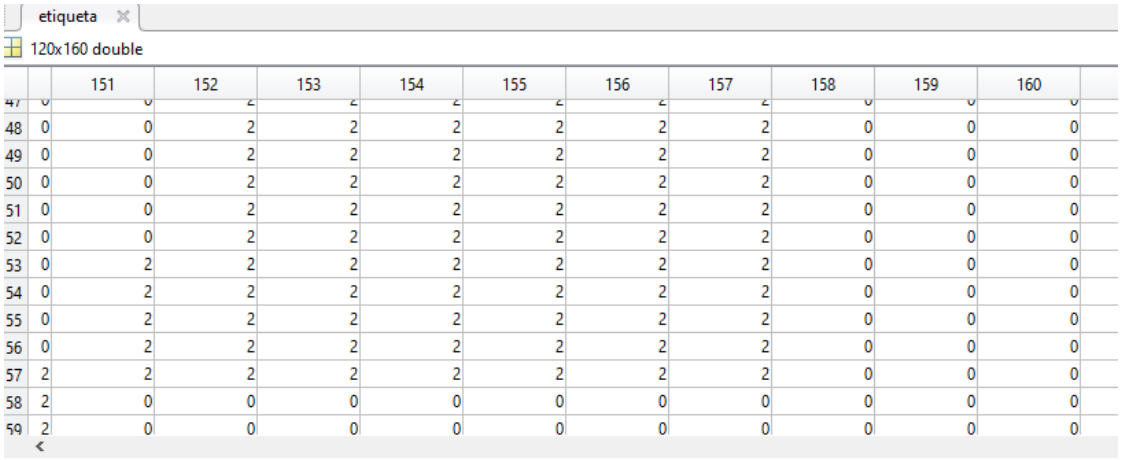


Figura 2.34 Etiquetamiento de la imagen en nuestro código

Comando de Matlab

```
etiqueta = bwlabel(diferencia,8);
```

Medición de las propiedades de las regiones de la imagen, como: 'Área', 'Centro' y el 'Rectangulo'

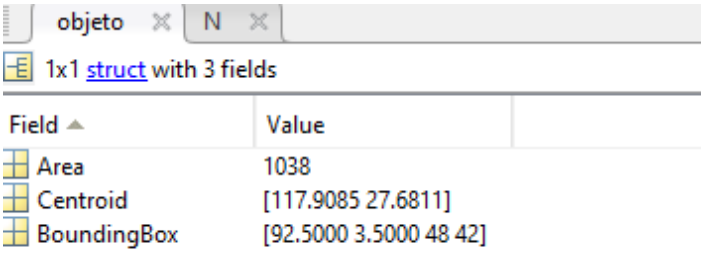


Figura 2.35 Variables resultantes al ejecutar Comando “regionprops” en Workspace.

Comando de Matlab:

```
objeto = regionprops(etiqueta);
```

Numero de objetos en la imagen [20]

Para determinar el número de objetos de la imagen se emplea el comando size(A,dim), en donde “A” es la matriz y “dim” es la longitud.

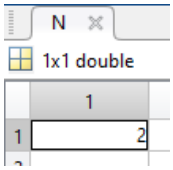


Figura 2.36 Comando para observar el número de objetos en movimiento en la imagen

Fragmento de código:

```
N = size(objeto,1);
```

El comando "*isempty*" permite determinar si la matriz sin elementos. Si se encuentra vacía devuelve un 1 lógico caso contrario un 0 lógico.

Fragmento de código:

```
if N < 1||isempty(objeto) %
```

Inicia un contador de tiempo, pero si no encuentra diferencias este se pondrá cero de nuevo y se reiniciará siempre que no haya movimiento.

```
t=0;

   return
end
```

Eliminar agujeros de menos de 200.

```
s=find([objeto.Area]<200);
```

Si no está vacía, le rellena con ceros

```
if ~isempty(s)
   objeto(s)=[ ];
end
```

Numero de objetos en la imagen.

```
N=size(objeto,1);
if N < 1 || isempty(objeto)
```

Inicia un contador de nuevo pero después de eliminar los pixeles y si de igual manera sigue sin encontrar movimiento, este se pondrá en cero.

```
        t=0;

   return
end
```

Dibuja un rectángulo y un punto central para cada objeto en la imagen

```
for n=1:N
   hold on
   centroid = objeto(n).Centroid;
   C_X = centroid(1);
   C_Y = centroid(2);
   rectangle('Position',objeto(n).BoundingBox,'EdgeColor','g','LineWidth',2)
   plot(C_X,C_Y,'Color','g','Marker','+','LineWidth',2)
   title(['X= ',num2str(round(C_X)),' Y= ',num2str(round(C_Y))])
```

Muestra en el command window el temporizador

```
   fprintf('%d\n',t);
```

```
    t=t+1;
    pause(0.01);

    hold off
end
indicador = 1;
```

Cálculo del tiempo

```
tiem=s/10;

dis=12;
vel=dis/tiem;
vel1=(vel*3600)/1000;
```

Muestra los resultados

```
fprintf('La distancia en metros es : %6.1f\n',dis);
fprintf('El tiempo en segundos es: %6.1f\n',tiem);
fprintf('La velocidad en metros/segundos es: %6.1f\n',vel);
fprintf('La velocidad en kilometros/hora es: %6.1f\n',vel1);

disp('FIN') %Muestra el fin del programa.

Return
```

2.6.2. MÉTODO 2: DETECCIÓN DE BORDES

Los bordes de una imagen proporcionan información relevante sobre el contorno de un objeto y se utiliza para realizar segmentación, reconocimiento de objetos, etc.

Los bordes se definen como el cambio en los niveles de intensidades, como se observa en la figura 2.37.

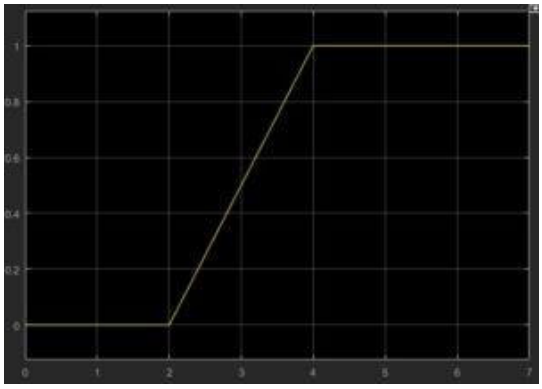


Figura 2.37 Niveles de grises [15]

En la siguiente sección se revisaron algunos algoritmos para elegir el mejor que se acople a la aplicación de medidor de velocidad de un objeto.

Entre los algoritmos más comúnmente conocidos para detección de bordes por gradientes se tiene: Laplaciano, Scharr, adicionalmente se presentará el método de detección de bordes mediante el algoritmo de Canny [18].

2.6.2.1. Gradiente Laplaciano

Esta aplicación permite la extracción de bordes de una imagen, Figura 2.38, el gradiente laplaciano es la segunda derivada bidimensional de los pixeles como se muestra en la Ecuación 2.10, las imágenes en escala de grises se pueden expresar como una función bivariable f(x, y), donde x e y son las coordenadas de ubicación de cada pixel (fila, columna).

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x^2} + \frac{\partial^2 f}{\partial^2 y^2} \tag{2.10}$$

Algunos inconvenientes con este método son: tiene sensibilidad al ruido, por esta razón necesita suavizado para poder funcionar con eficiencia, tiene tendencia para redondear las esquinas, no tiene orientación de los bordes por lo cual, los detecta en todas las direcciones, y principalmente tiene una alta carga computacional [16].

2.6.2.2. Gradiente de Scharr

Se presenta como la gradiente de primer orden, Figura 2.39, entre los filtros que utilizan gradientes de primer orden se tienen: Prewitt, Sobel, Scharr entre otros, para esta revisión se escogió el de Scharr debido a que es uno de los algoritmos que mejor rendimiento posee.

La principal ventaja de este filtro es que se puede emplear en diferentes direcciones, con lo cual se puede tener algunos Kernels para diversas situaciones como, por ejemplo: si se requiere detección de bordes en x, y o en algún ángulo en específico [16].

El filtro que utiliza Sharr se basa en la Ecuación 2.11.

$$\nabla f = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y} \tag{2.11}$$

2.6.2.3. Algoritmo de Canny

Matlab posee en su toolbox de procesamiento de imágenes una herramienta para poder detectar bordes utilizando el algoritmo de Canny con la función “edge”, Figura 2.40. Es similar a los mencionados anteriormente, la diferencia es que utiliza un filtro Gaussiano y

lo aplica en todas las direcciones según las Ecuaciones 2,12, 2.13, 2.14.

$$GaussKernel_{5x5} = \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{2.12}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.13}$$

$$\theta = \arctan\left(\frac{Gx}{Gy}\right) \tag{2.14}$$

El código se implementa de manera sencilla, pero presenta algunos inconvenientes como: hay demasiados falsos negativos que son superiores a los filtros antes mencionados, su carga computacional también es demasiada alta lo que dificulta poder utilizar este filtro para la aplicación de un medidor de velocidad en tiempo real [16].

A continuación, se presenta imágenes referenciales de ejemplos de los métodos de detección de bordes antes mencionados.

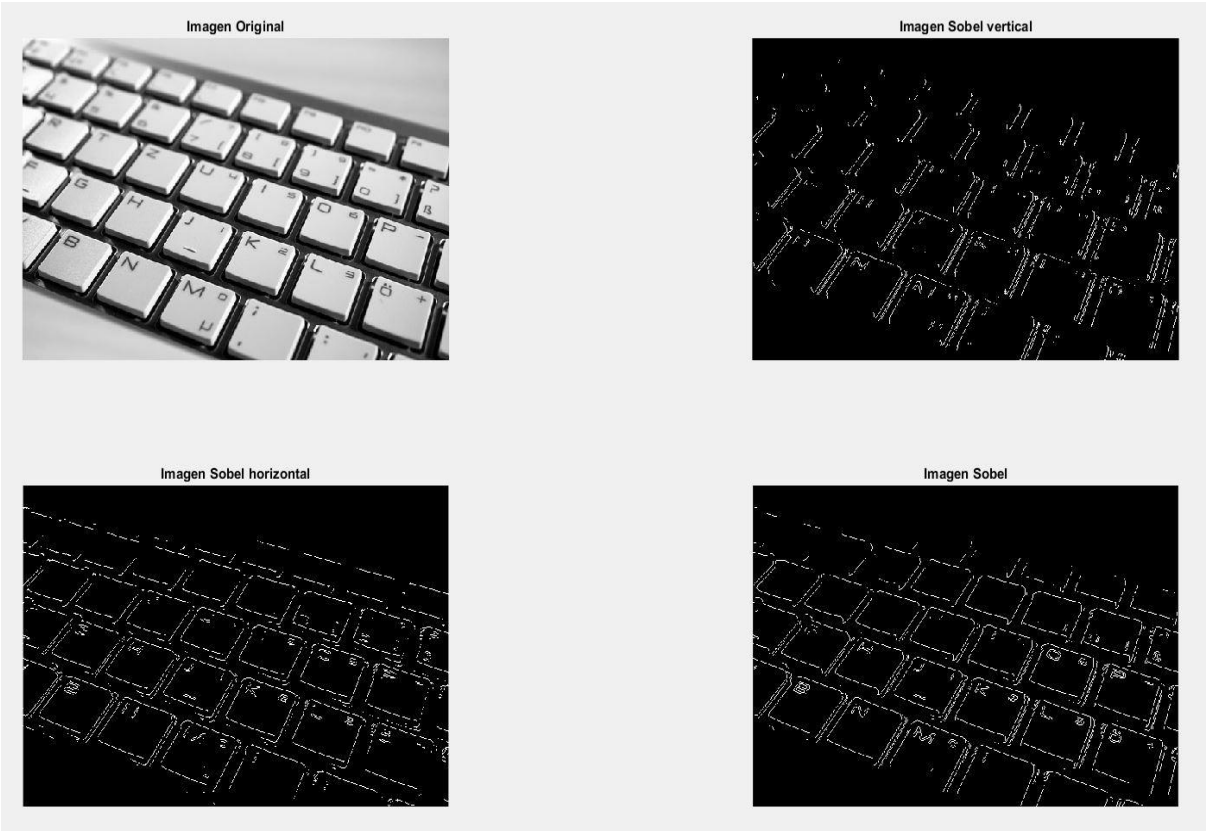


Figura 2.38 Gradiente Laplaciano o gradiente de primer orden [17].

Figura 2.39 Gradiente de Scharr o gradiente de segundo orden [17]

Figura 2.40 Algoritmo de Canny [16]

La diferencia principal entre estos algoritmos es que, en el Laplaciano se realiza la detección de bordes en todos los sentidos mientras que en el de Scharr se puede escoger en qué sentido se realizará la detección de bordes, es decir Scharr es un filtro direccionado mientras Laplaciano no.

Con los métodos antes mencionados para realizar la detección de bordes se decidió escoger el método que ocupa menor carga computacional, este es el algoritmo de Scharr, debido a que se puede realizar detección de bordes en una dirección y para la aplicación solo se necesitaría detección de bordes en el eje X, pudiendo hacer posible la aplicación de este algoritmo en tiempo real.

Utilizando el mismo procedimiento para calcular la velocidad sin marcas físicas se utilizó un algoritmo de detección de bordes para poder ver el cambio en el algoritmo de cálculo de velocidad.

Los pasos que se utilizaron para poder desarrollar el algoritmo fueron los siguientes: primeramente, se desarrolló un filtro Gaussiano con el objetivo de suavizar la imagen y eliminar el ruido.

Luego se aplicó una segmentación pudiendo ser realizada por Scharr.

Para la realización de este algoritmo se aplicó un filtro Gaussiano para suavizar la imagen y luego, dado que para la aplicación solo será necesario detectar los bordes en X. A continuación, se muestran las máscaras que se utilizan para los diferentes casos:

$$Scharr_x = \frac{1}{16}\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \tag{2.15}$$

$$Scharr_y = \frac{1}{16}\begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{2.16}$$

$$Scharr_{45\ grados} = \frac{1}{16}\begin{bmatrix} -10 & -3 & 0 \\ -3 & 0 & 3 \\ 0 & 3 & 10 \end{bmatrix} \tag{2.17}$$

Ya que la detección del automóvil se realizará solo en el plano X, se aplicó el algoritmo Scharr con filtro Gaussiano en el plano X con lo que omite en gran medida los bordes en el plano Y, se ahorra carga computacional, para que el programa no sea tan pesado y pueda calcular la velocidad en tiempo real.

Luego de utilizar detección de bordes se aplicó el algoritmo utilizado en el método uno, es decir el de sustracción de bordes para poder detectar el movimiento con la variación que en este caso se encuentra modificado, ya que aquí no existe gama de colores, solo unos y ceros.

Código utilizado

```
vid=videoinput('winvideo', 2, 'YUY2_160x120'); % Se utiliza el puerto de
video winvideo 2 para generar una entra de resolución de 160x120
set(vid,'ReturnedColorspace','rgb'); % Se transforma a imagen RGB
set(vid,'TriggerRepeat',Inf); %
set(vid,'FramesPerTrigger',1); %
% vid.FrameGrabInterval=1; %
```

```
triggerconfig(vid,'manual'); % Se configura como manual

start(vid); % se inicia el objeto vid que son las imágenes que se van
obteniendo
tic %
trigger(vid);while islogging(vid)==1, end % se obtura siempre mientras no
salga del lazo
%
k=1;% Se inicializa el contador
%     k=0;
% Ip1 = zeros(216,384,'double');
threshold=0;
s1=0;
while k<120 % Se seguirá obteniendo imágenes mientras los fotogramas que
se obtienen sean menores a 120

    k=k+1;
    data=flip(getdata(vid,1),2); % Se obtiene cada frame de la secuencia
de video
    trigger(vid); % Se realiza la obturación sobre las imágenes que se
van obteniendo
    im=imresize(data,0.5);

    %Detector bordes

    b=rgb2gray(im);% imagen a escala de grises
    a1=double(b);% Se transforma a formato double es decir cambia el
valor de cada pixel a 8 bytes
    s=size(b);%dimensiones de la matriz
    arc2=a1*0;%  matriz de ceros igual a las dimensiones de la imagen

            kerv=[1;2;1];%componente kernel vertical
            kerh=[1 2 1];%componente kernel horizontal
            kernel1=kerv*kerh/16;%filtro gausiano y se obtiene un valor
promedio al dividir;

            for i=2:s(1)-1%alto de la imagen
                for j=2:s(2)-1%ancho de la imagen
                    vent=a1(i-1:i+1, j-1:j+1);%se aisla la matriz que de
la imagen resultante y que será amultiplicada por el kernel
                    produc=vent.*kernel1;% multiplica el kernel por la
ventana
                    pix=sum(sum(produc));%se suman los valores de la
matriz
                    arc2(i,j)=pix;%se sustituye todos los pixeles de la
imagen por este nuevo valor
                end
            end
            arcz=zeros(s(1) , s(2));% matriz de ceros de las mismas
dimensiones de la imagen
            %Scharr
            kernelscharrX=[-3 0 3;-10 0 10;-3 0 3]/16;

            for i=2:s(1)-1%el alto de la imagen
                for j=2:s(2)-1%ancho de la imagen
                    vent=arc2(i-1:i+1, j-1:j+1);%s
                    produc=vent.*kernelscharrX;%se multiplica el
laplaciano por la ventana
```

```
                    pix=sum(sum(produc));% uman los valores de la matriz
                    arcz(i,j)=(pix+255)/2;%se sustituye todos los pixeles
por este nueevo valor
                end
            end
    Ient=uint8(arcz);%transforma la imagen  en valores enteros
    Ibin=im2bw(Ient,0.5059);%se binariza la imagen
    Egr=strel('disk',2);%crea un disco de grosor 2 para los bordes
    Ia=imerode(Ibin,Egr);%expande el grosor de los bordes si se resta la
imagen original menos la erosionada
    Ips=abs(Ibin-Ia-1);%gradiente morfológico
    imagesc(Ips);


if k==10 %para poder contrastar la luminosidad al momento de adecuarse la
camara, se escoge un fotograma cuando el contraste de la cámara se haya
estabilizado
    Ipd=Ips;
end
if k>10
[indicador,s1,diferencia] = compare2(Ips,Ipd,threshold,s1);%Utiliza una
función para la comparación , la misma que se utilizó para sustracción de
fondo pero con dos niveles (1 y 0)
       pause(0.01);
end
end
tiem=s1/100;

    dis=12;
    vel=dis/tiem;
    vel1=(vel*3600)/1000;

    stop(vid); %Detiene el video
    delete vid %Borra la variable vid
    clear vid %
% catch
%     stop(vid);
%     disp('Sucedió un ERROR')
% end
clc
fprintf('La distancia en metros es : %6.1f\n',dis);
fprintf('El tiempo en segundos es: %6.1f\n',tiem);
fprintf('La velocidad en metros/segundos es: %6.1f\n',vel);
fprintf('La velocidad en kilometros/hora es: %6.1f\n',vel1);

disp('FIN') %Muestra el fin del programa.
% Desconecta y borra toda la adquisicion de objetos

figure(2)
imagesc(Ipd);
imaqreset
```

Función Compare

Se utiliza la misma función que en el código de sustracción de fondo para realizar la detección de movimiento, pero la única variación que existe es la variable threshold, que en este caso debido a que la imagen es en blanco y negro tiene un valor de 0.

```
function
[indicador,t,diferencia]=compare2(entrada_imagen,fondo,threshold,t)%Se
asignan las entradas de la funcion
indicador = 0;
%
diferencia = (abs(entrada_imagen(:,:,1)-fondo(:,:,1)) > threshold);
% Elimina el ruido (eliminando pequeños agujeros y rellena las aberturas)
% a = bwlabel(diference,8);
% Realiza cierre morfológico (dilatación seguida de erosión).
b = bwmorph(diferencia,'close');
% Realiza una apertura morfológica (erosión seguida de dilatación).
diferencia = bwmorph(b,'open');
diferencia = bwmorph(diferencia,'erode',2);
% Seleccione el objeto más grande
etiqueta = bwlabel(diferencia,8);
% Mide las propiedades de las regiones de la imagen, como: 'Área', 'Centro'
y 'Rectangulo'
objeto = regionprops(etiqueta);
N = size(objeto,1); %Número de objetos en la imagen.
if N < 1||isempty(objeto) % Devuelve si no hay ningún objeto en la imagen
    return
end
% Eliminar agujeros de menos de 200 píxeles
s=find([objeto.Area]<200);
if ~isempty(s)
    objeto(s)=[ ];
end
N=size(objeto,1);% Contador de objetos
if N < 1 || isempty(objeto)
    return
end
% Dibuja un rectángulo y un punto central para cada objeto en la imagen
for n=1:N
    hold on
    centroid = objeto(n).Centroid;
    C_X = centroid(1);
    C_Y = centroid(2);

rectangle('Position',objeto(n).BoundingBox,'EdgeColor','g','LineWidth',2)
    plot(C_X,C_Y,'Color','g','Marker','+','LineWidth',2)
    title(['X= ',num2str(round(C_X)),' Y= ',num2str(round(C_X))])

    fprintf('%d\n',t);
    t=t+1;
    pause(0.01);
    hold off
end
indicador = 1;
return
```

3. RESULTADOS Y DISCUSIÓN

3.1. CÁLCULO DE DISTANCIA

En esta sección se presentan pruebas con diferentes distancias, al utilizar el medidor láser sin delimitar marcas físicas comparado con la distancia real medida manualmente para calcular el porcentaje de error.

Los resultados se muestran en la tabla 3.1.

Tabla 3.1. Resultados de medición de distancia

Distancia medida manualmente	**Distancia medida usando trigonometría.**	**Porcentaje de error %**
6 m	5.64 m	6
10 m	10.22 m	2.2
12 m	10.68 m	11
5 m	4.62 m	7.6
13 m	12.39 m	4.69

Como se observa en la tabla, mientras la distancia aumenta presenta un mayor margen de error, esto se debe a que mientras exista mayor distancia el laser es más sensible al movimiento, lo que produce una lectura con mayor porcentaje de error Según las pruebas realizadas pasado 10 metros medidos desde la cámara al centro del campo de visión de la cámara este empieza a presentar errores mayores.

3.2. CÁLCULO DE TIEMPO

En esta etapa se presentan las imágenes del software obtenidos con el método 1 (sustracción de fondo) y método 2 (detección de bordes).

Método 1

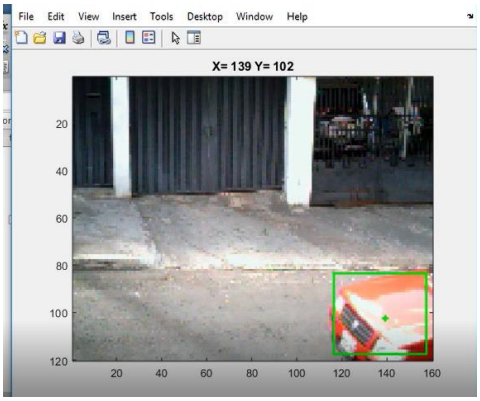


Figura 3.1 Vehículo ingresando al campo de visión de la cámara

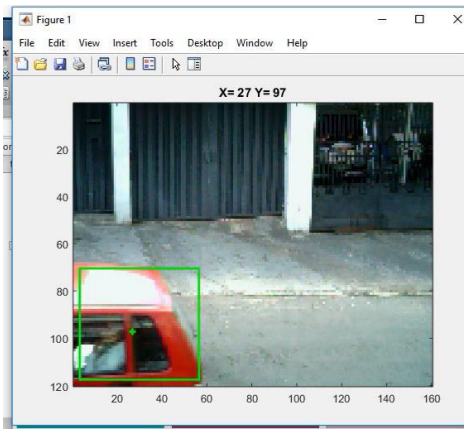


Figura 3.2 Vehículo saliendo del campo de visión de la cámara

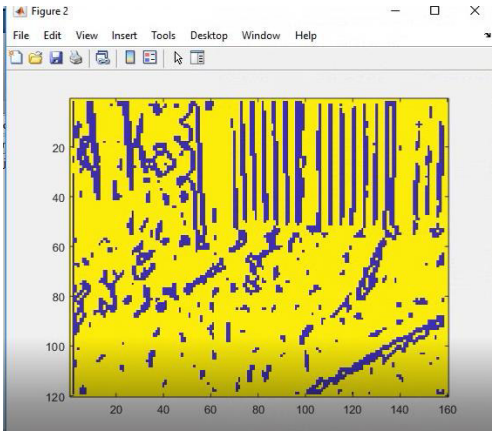


Figura 3.3 Detección de bordes solo en dos niveles

Se realizaron pruebas adicionales con la cámara infrarroja tal como se muestra en la Figura 3.4, para observar el desempeño de los algoritmos en un ambiente nocturno.

Figura 3.4 Cámara infrarroja

En la Figura 3.5, 3.6 se observa el rendimiento del algoritmo cuando el objeto entra y sale del campo de visión de la cámara utilizando el método 1.

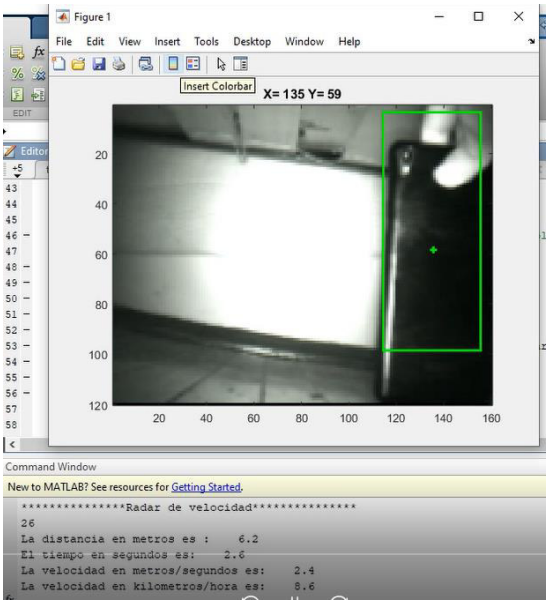


Figura 3.5 Objeto entrando al campo de visión de la cámara (método 1)

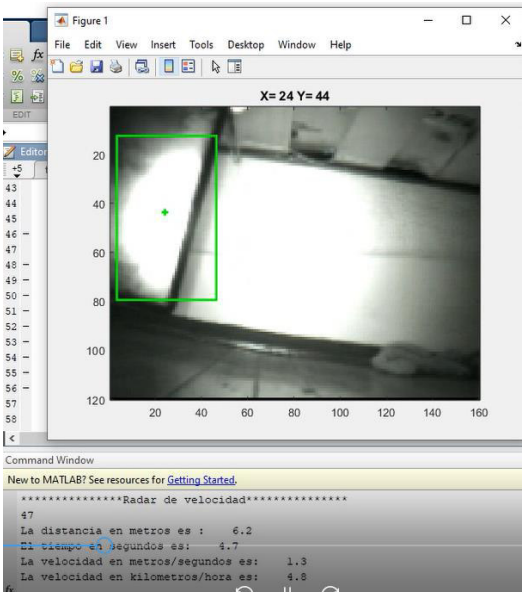


Figura 3.6 Objeto saliendo del campo de visión de la cámara (método 1)

Se puede observar que la luminosidad afecta en gran medida al algoritmo, debido a que el más mínimo cambio de luminosidad o el reflejo en el objeto provoca que el temporizador se reinicie y empiece de nuevo desde cero.

En la figura 3.7,3.8 Se muestra el rendimiento del segundo algoritmo con el mismo ángulo de visión de la cámara.

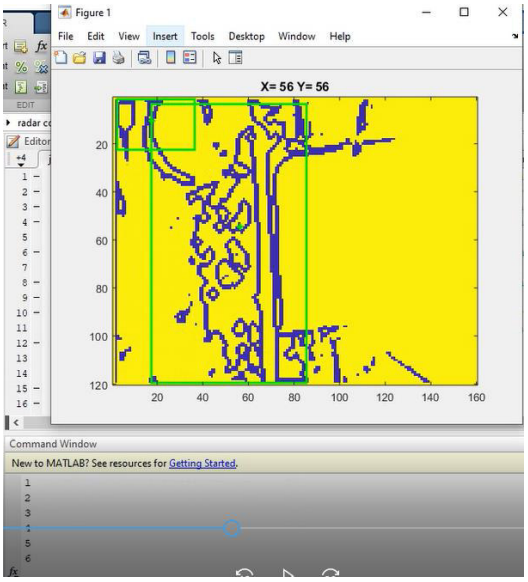


Figura 3.7 Objeto entrando al campo de visión de la cámara (método 2)

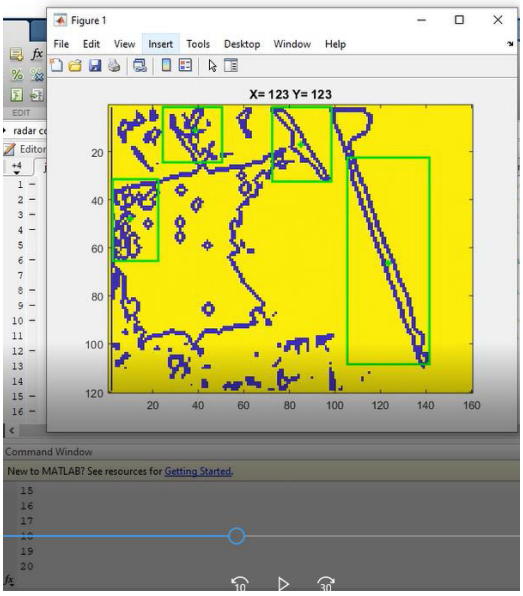


Figura 3.8 Objeto saliendo del campo de visión de la cámara (método 2)

Se observa que cuando entra un objeto en el campo de visión de la cámara, existen demasiados falsos positivos, donde no existe movimiento ni cambio en los pixeles en los dos niveles (1 y 0), lo que demuestra que este algoritmo aplicado a un ambiente nocturno es muy sensible a la luminosidad y cualquier cambio leve hace que existan errores e inicie el temporizador.

Al observar el desempeño de los dos algoritmos en distintos ambientes se concluye que el que mejor rendimiento posee es el método 1 (sustracción de fondo), y es el que se escogió para realizar la medición de velocidad.

3.3. CÁLCULO DE VELOCIDAD

Para realizar las pruebas finales se utilizó el tacómetro de un vehículo para comparar si la velocidad medida es la correcta, a continuación, se muestra una tabla con las pruebas realizadas, donde en el vehículo marca la velocidad real y el software marca el valor estimado.

Los resultados alcanzados con el método 1 se observan en la tabla 3.2.

Tabla 3.2. Resultados de valor real(tacómetro) y con el software

Numero de prueba	Valor real (km/h)	Valor estimado (km/h)
1	40	43.5
2	50	48.5
3	60	52.3
4	55	51.4
5	30	28.4

El porcentaje de error de cada prueba se muestra a continuación.

Tabla 3.3. Porcentaje de error de las pruebas

Numero de prueba	Porcentaje de error
1	8.75
2	3
3	12.83
4	6.54
5	5.33

Método 2: Detección de bordes

Tabla 3.4. Pruebas realizadas con detección de bordes

Número de prueba	Velocidad (Software)	Velocidad (Vehículo)	Porcentaje error
1	10.9 km/h	5 km/h	118
2	19.1 km/h	10 km/h	91
3	27.0 km/h	15 km/h	80
4	29.4 km/h	20 km/h	47
5	37.1 km/h	25 km/h	48.4

Para la realización de este método se optó por utilizar el algoritmo que requiera menos carga computacional, debido a que el objetivo es realizar las pruebas en tiempo real.
El algoritmo que mejor se adoptó para el caso es de Scharr, y puesto que el ambiente de prueba es solo en el eje X, este se implementó para detectar bordes mayormente en dicho eje.
Dado que la tasa de adquisición de datos es de 9.75, esta es muy baja, lo que provoca que el procesamiento, aun siendo con el algoritmo con menos carga computacional de detección de bordes sea realmente lento para procesar en tiempo real, haciendo que frames se pierdan en cada interacción y el contador se ralentice.

Para el método de sustracción se fondo se obtuvo una tasa de adquisición de datos de 13.75 fps, y aunque es bajo, permite ejecutar el programa sin problema.
Actualmente existen cámaras que llegan a 60 fps, pero para este proyecto se utilizo una cámara que soporta 30 fps.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Como se ha observado han existido esfuerzos para poder calcular la velocidad de un vehículo utilizando las propiedades y procesamiento de un computador, se puede utilizar las características de los diferentes vehículos como dimensiones para poder clasificarlos como motos, carros, camiones, pero la desventaja es que el procesamiento es complejo y es difícil poder realizarlo en tiempo real por lo cual se opta por obtener los videos y de ahí poder procesarlos.

El presente proyecto se centra en un ambiente en específico donde se debería tener al otro extremo una pared o algo para que con el medidor de distancia láser poder medir la distancia que existe de la cámara a la pared y así, ya conocido el ángulo de visión, poder calcular la distancia del recuadro de la cámara. Se utiliza este dato en el software para obtener la velocidad en base a la distancia que recorre el vehículo en un determinado instante de tiempo, ese instante se observa desde que el vehículo entra al recuadro hasta que sale.

Los errores que se obtienen son debido al procesamiento que se realiza en tiempo real, ordenadores con más nivel de procesamiento podrían reducir el error ya que poseen mayor memoria RAM y procesaría las frames rápidamente.

Dados los resultados que se observan en el capítulo 3, se concluye que el primer método de sustracción de fondo, aunque utiliza toda la gama de pixeles para realizar la sustracción de fondo es más eficiente, se obtiene menores márgenes de error (8%) que el segundo método de detección de bordes con un error superior al 40%, las razones se deben a que al realizar detección de bordes y convertir la imagen en dos niveles de 1 y 0, se requiere mayor procesamiento, lo que hace que haya ciertos retardos al momento de realizar el cálculo de la velocidad que son de 0.5 segundos.

Otro factor importante son las condiciones de luz, si el ambiente está muy iluminado afecta a la cámara que debe adecuarse continuamente a la nueva iluminación con lo que hace que se produzcan errores ya que la gama de colores cambia.

El de sustracción de fondo posee menor carga computacional lo que da una gran ventaja en velocidad de procesamiento haciendo posible que este método sea apto para la medición de velocidad en tiempo real.

Para un ambiente donde predomine la oscuridad los algoritmos propuestos presentan fallas con una cámara infrarroja, porque la luminosidad varia demasiado y existen reflejos en los objetos que se desea calcular la velocidad, lo cual dificulta que el objeto sea localizado con eficiencia produciéndose ruido y falsos positivos en el intento.

4.2. RECOMENDACIONES

Para futuros proyectos se recomienda utilizar el método de sustracción de fondo para realizar un radar de velocidad. Para el caso de una vía de doble carril se recomienda utilizar redes neuronales artificiales con el fin de poder identificar a un vehículo y no confundirlo con un transeúnte que este cruzando por el campo de visión de la cámara.

Se recomienda utilizar el método 2 de detección de bordes Scharr en diferentes ángulos o aplicar un gradiente Laplaciano, con un ordenador que posea mayores recursos de procesamiento (8GB de RAM, CORE I7) y observar el desempeño que estos algoritmos tendrían para aplicaciones de visión artificial.

Si se requiere realizar la medición de velocidad de un objeto en un ambiente nocturno se debe utilizar algoritmos diferentes a los aplicados en el día, o modificar los algoritmos de sustracción de fondo y detección de bordes propuestos con el fin de eliminar el ruido existente.

5. REFERENCIAS BIBLIOGRAFICAS

[1] O. Morales, " Eproceso de homologación y calibración de dispositivos y equipos tecnológicos foto radar y la notificación de las infracciones de tránsito", Repositorio Universidad Técnica de Ambato, 2016. [Online]. Disponible: http://repositorio.uta.edu.ec/bitstream/123456789/25059/1/FJCS-DE-1000.pdf. [Accessed: 14- Mar- 2019].

[2] Dirección General de Tráfico, "La DGT compra 16 nuevos radares para controlar la velocidad por un millón de euros", Dirección General de Tráfico, España, 2018. [Online]. Disponible: https://www.20minutos.es/noticia/1953066/0/dgt-compra/16-nuevos-radares/velocidad-multas/. [Accessed: 16- Mar- 2019].

[3] "Pistola / Medidor radar de velocidad Bushnell II (101911)", ServoVendi, 2018. [Online]. Disponible: https://www.servovendi.com/es/pistola-medidor-radar-de-velocidad-bushnell-ii-101911.html. [Accessed: 16- Mar- 2019].

[4] F. Sisalema, “Sistema para detección y conteo vehicular aplicando técnicas de visión artificial”, Repositorio Universidad Nacional de Loja, Loja, 2018. [Online]. Disponible: http://dspace.unl.edu.ec/jspui/bitstream/123456789/20892/1/Sisalima%20Ortega%2C%20Fabricio%20Roberto.pdf. [Accessed: 25- May- 2019].

[5] E. Herrera, G Loor, “Desarrollo e implementación de un sistema de visión artificial y seguimiento de objetivos humanos por un cuadricóptero de exteriores utilizando Matlab”, Repositorio Escuela Politécnica Nacional, Quito, 2018. [Online]. Disponible: https://bibdigital.epn.edu.ec/handle/15000/19432. [Accessed: 27- May- 2019].

[6] S.Indu, Mañjari Gupta y el Prof. Asok Bhattacharyya, " Seguimiento de vehículos y estimación de la velocidad usando el método de flujo óptico ", Revista Internacional de Ciencia y Tecnología de Ingeniería (IJEST), India, 2011. [Online]. Disponible: https://www.researchgate.net/publication/50392061_Vehicle_Tracking_and_Speed_Estimation_using_Optical_Flow_Method. [Accessed: 05- Jul- 2019].

[7] A.Otero, J. Otero , "Flujo optico", Departamento de Informatica, Oviedo, 2007. [Online]. Disponible: http://di002.edv.uniovi.es/~jotero/foclases2007.pdf. [Accessed: 18- Jul- 2019].

[8] Sheikh Ullah, Syed Ab Rahman Abu Bakar, Rudi Heriansyah," Vestro: estimación de la velocidad mediante visión estereoscópica", IEEE, Malasia, 2005. [Online].

Disponible: https://ieeexplore.ieee.org/document/4977172. [Accessed: 25- Jul- 2019].

[9] G. Yabo, F. Safar, "Clasificación del vehículo y la estimación de la velocidad utilizando técnicas de visión artificial", IEEE, Argentina, 2016. [Online]. Disponible: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-49932009000200002. [Accessed: 13- Ago- 2019].

[10] J.Camos, " Los nuevos radares de tráfico son tan pequeños que se colocan como una lapa en las barreras de la carretera", Motorpasion, 2018. [Online]. Disponible: https://www.motorpasion.com/seguridad/los-nuevos-radares-de-trafico-son-tan-pequenos-que-se-colocan-como-una-lapa-en-las-barreras-de-la-carretera. [Accessed: 14- Oct- 2019].

[11] J. Ibañes, " Radares de tramo: qué son y cómo funcionan", xataka, 2019. [Online]. Disponible: https://www.xataka.com/automovil/radares-de-tramo-asi-funcionan. [Accessed: 18- Oct- 2019].

[12] D. Villareal, " Los radares "pistola" láser están en España: ¿Cómo funcionan este tipo de radares?", diariomotor, España, 2015. [Online]. Disponible: https://www.diariomotor.com/2015/06/26/radares-laser-pistola-espana/. [Accessed: 24- Oct- 2019].

[13] D. L, " Un paso más: la carretera inteligente para los coches autónomos", el telescopio, España, 2017. [Online]. Disponible: https://www.elplural.com/el-telescopio/innovacion/un-paso-mas-la-carretera-inteligente-para-los-coches-autonomos_100058102. [Accessed: 24- Oct- 2019].

[14] J. D'Amato, " Un método eficiente para la sustracción de fondo en videos usando gpu", Universidad Nacional del Centro de Buenos Aires, Argentina, 2014. [Online]. Disponible: https://www.researchgate.net/publication/268519388_UN_METODO_EFICIENTE_PARA_LA_SUSTRACCION_DE_FONDO_EN_VIDEOS_USANDO_GPU. [Accessed: 24- Oct- 2019].

[15] J. Contreras, " Deteccion de bordes con Matlab", aprendiendo ingenieria, España, 2017. [Online]. Disponible: http://aprendiendoingenieria.es/deteccion-bordes-matlab/. [Accessed: 26- Oct- 2019].

[16] E. Herrera, G Loor, “Desarrollo e implementación de un sistema de visión artificial y seguimiento de objetivos humanos por un cuadricóptero de exteriores utilizando

Matlab", Repositorio Escuela Politécnica Nacional, Quito, 2018. [Online]. Disponible: https://bibdigital.epn.edu.ec/handle/15000/19432. [Accessed: 27- Oct- 2019].

[17] V. Sanchez, F. Caballero, M. Perez, M. Vidal, E. Rodrigues, "DETECCIÓN DE BORDES DE UNA IMAGEN USANDO MATLAB", Repositorio Tecnologico Nacional de Mexico, Celaya, 2016. [Online]. Disponible: file:///C:/Users/Danny/Downloads/655-2016-3-PB.pdf. [Accessed: 27- Oct- 2019].

[18] J. Valverde, " Deteccion de bordes utilizando el algoitmo de canny", Universidad Nacional de Trujillo, Perú , 2007. [Online]. Disponible: . [Accessed: 28- Oct- 2019].

[19] A. Parra, "Análisis de Movimiento en 3 dimensiones en video de alta velocidad para evaluación de sistemas dinámicos", Centro de Investigaciones Ópticas, México, 2012. [Accessed: 28- Oct- 2019].

M. Montalvo, "Técnicas de visión estereoscópica para determinar la estructura tridimensional de la escena", Universidad Complutense de Madrid, España, 2010.

[20] "MATLAB Documentation", The Mathworks, 2020. [Accessed: 28- Oct- 2019].

ANEXOS

ANEXO A. Adquisición de imágenes de las cámaras con sus diferentes resoluciones de video.

ANEXO B. Código fuente del proyecto (Sustracción de fondo)

ANEXO C. Código fuente del proyecto (Detección de bordes)

ANEXO A

Las dos cámaras que se utilizaron se muestran a continuación con sus diferentes características y resoluciones:

Cámara Web delta

Figura 0.1 Cámara web

Formatos de resolución disponibles:

- 160x120
- 176x144
- 320X240
- 352x288
- 640x480

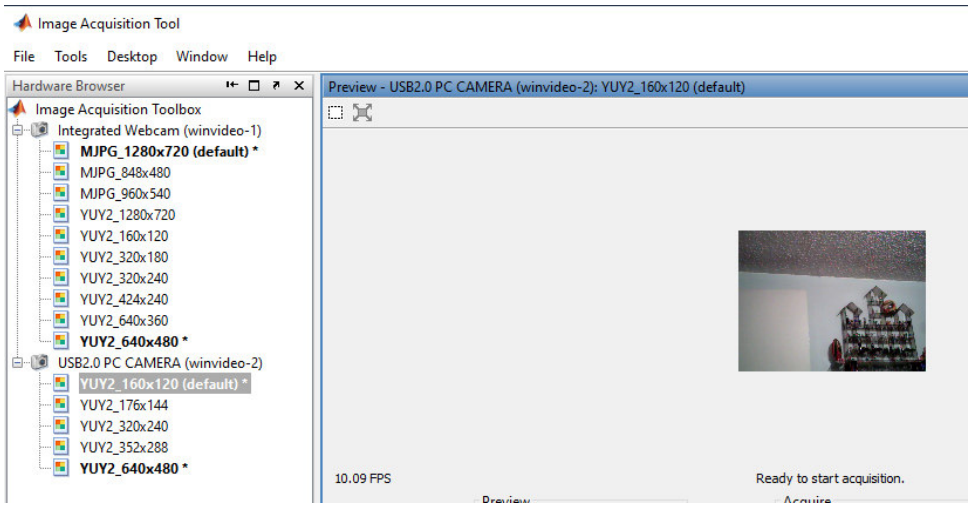
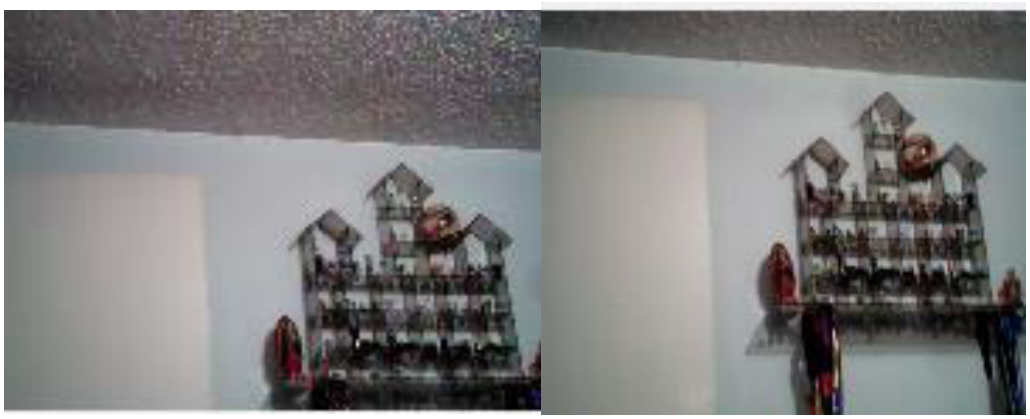
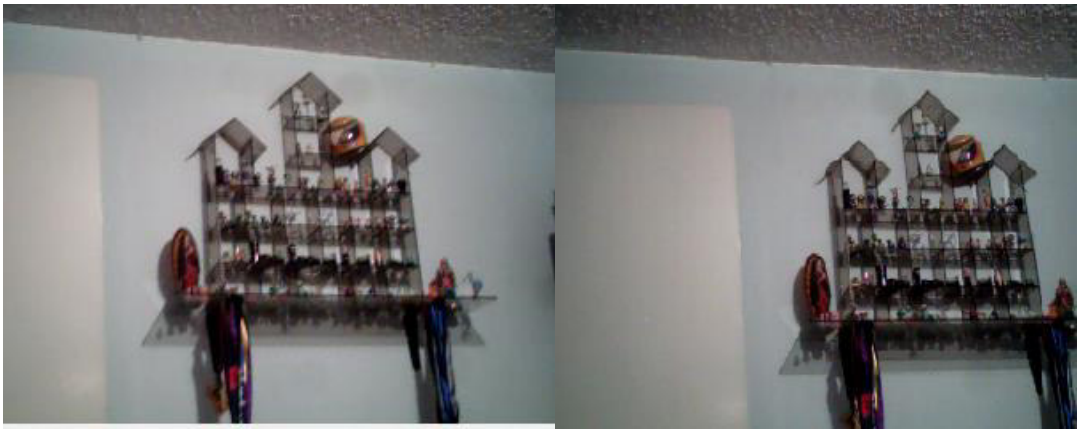


Figura 0.2 Vista previa de la cámara con los formatos de resolución

En la Figura 0.2 se observan los diferentes formatos que posee la cámara, que se utilizó para el proyecto.



a) Resolución 120x160 b) Resolución 176x144



c) Resolución 320x240 d) Resolución 352x288

Figura 0.3 Formatos de resolución soportados por cámara

Cámara Infrarroja

Figura 0.4 Cámara Infrarroja

Formatos de resolución disponibles:

- 160x120
- 320x240
- 640x480
- 720x480
- 720x576

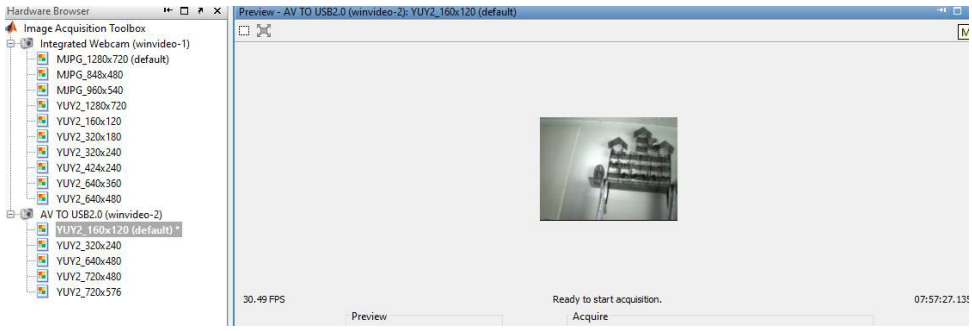


Figura 0.5 Vista previa de la cámara con los formatos de resolución

Las dos cámaras presentan formatos similares, pero dado que la aplicación se realiza en tiempo real es necesario escoger la que posee menor resolución para que no exista demasiada carga computacional.

La cámara infrarroja es inalámbrica lo que produce un retraso al momento de adquirir imágenes a comparación de la cámara web USB.

ANEXO B

Código utilizado

```
imaqreset
% Detector de objetos con webcam
% _____________________________________

clc, clear all, close all % Cierra todo, limpia workspace

imaqreset%  Desconecta y borra toda la adquisicion de objetos

vid=videoinput('winvideo',2, 'YUY2_160x120'); % Ya que las ultimas
versiones de matlab no tienen la libreria para
% utilizar winvideo, se debio bajar la libreria del internet.
% Crea un videoinput con una resolucion a color de 320x240, ya que si se
% aumenta la resolucion el video empieza a entre cortarse mas, y se hace
mas lento.


vid.FrameGrabInterval = 1;% Especifica la frecuencia con la que debe
adquirir el marco de la secuencia de video
%set(vid,'TriggerRepeat',Inf);
reso=vid.VideoResolution; %Obtiene las dimensiones del video. En este
caso es 320x240
set(vid, 'FramesPerTrigger', Inf);%Configuracion de propiedades del video
set(vid,'ReturnedColorspace', 'rgb') %Imagenes de color
% _______Toma de antecedente_________ Obtiene una imagen previa para
% despues compararla enn la funcion.
disp('Obtener imagen original')
%vid.FramesPerTrigger=20; %Especifique el número de fotogramas que se
deben adquirir utilizando la fuente de video seleccionada
start(vid); %Empieza el video

disp('Fin obtencion de imagen original')
% _______________________________
% inicia
    threshold=60;
%     figure(1);
    s=0;
    v=0;
    %k=vid.FramesAcquired;
    k=1;
    tic
    while (k<350) % Se utiliza un buble para restringir el video, ya que
si no se restringe seguira tomando datos indefinidamente.
        k=k+1;
        % obtiene imagen
        ky=vid.FramesAcquired;
        imagen = getdata(vid,1); %Obtiene imagenes y las muestra
posteriormente
%         imshow(imagen) %Muestra la imagen
        imagesc(imagen);

        title('Medidor de velocidad')
%         text(reso(1),reso(2)+35, ....
%         'Detector de imagenes', ...
```

```
%         'FontSize',12,'HorizontalAlignment','right');
        entrada_imagen = double(imagen);%Convierte imagen a tipo double
        %
        if k==20
        data = getdata(vid,1); %Adquiere imagen original del video.
        fondo=double(data(:,:,:)); %convierte a formato double
        end
        if k>20
        [indicador,s,diferencia,N] =
comparet(entrada_imagen,fondo,threshold,s);%Utiliza una funcion para la
comparacion
        pause(0.01);
        end
    end

    stop(vid); %Detiene el video
    delete vid %Borra la variable vid
    clear vid %
% catch
%     stop(vid);
%     disp('Sucedió un ERROR')
% end
clc


disp('FIN') %Muestra el fin del programa.
% Desconecta y borra toda la adquisicion de objetos
imaqreset
toc
```

Función utilizada

```
function
[indicador,t,diferencia,N]=comparet(entrada_imagen,fondo,threshold,t)%Se
asignan las entradas de la funcion
indicador = 0;

diferencia1 = (abs(entrada_imagen(:,:,1)-fondo(:,:,1)) > threshold) |
(abs(entrada_imagen(:,:,2) - fondo(:,:,2)) > threshold) ...
                    | (abs(entrada_imagen(:,:,3) - fondo(:,:,3)) >
threshold);
% Elimina el ruido (eliminando pequeños agujeros y rellena las aberturas)
% a = bwlabel(diference,8);
% Realiza cierre morfológico (dilatación seguida de erosión).
diferencia=diferencia1;
b = bwmorph(diferencia,'close');
% Realiza una apertura morfológica (erosión seguida de dilatación).
diferencia = bwmorph(b,'open');
diferencia = bwmorph(diferencia,'erode',2);
% Seleccione el objeto más grande
etiqueta = bwlabel(diferencia,8);
% Mide las propiedades de las regiones de la imagen, como: 'Área',
'Centro' y 'Rectangulo'
objeto = regionprops(etiqueta);
N = size(objeto,1); %Número de objetos en la imagen.
if N < 1||isempty(objeto) % Devuelve si no hay ningún objeto en la imagen
```

```
    t=0;
    return
end
% Eliminar agujeros de menos de 200 píxeles
s=find([objeto.Area]<200);
if ~isempty(s)
    objeto(s)=[ ];
end
N=size(objeto,1);% Contador de objetos
if N < 1 || isempty(objeto)
    t=0;
    return
end
% Dibuja un rectángulo y un punto central para cada objeto en la imagen
for n=1:N
    hold on
    centroid = objeto(n).Centroid;
    C_X = centroid(1);
    C_Y = centroid(2);

rectangle('Position',objeto(n).BoundingBox,'EdgeColor','g','LineWidth',2)
    plot(C_X,C_Y,'Color','g','Marker','+','LineWidth',2)
    title(['X= ',num2str(round(C_X)),' Y= ',num2str(round(C_Y))])

%     fprintf('%d\n',t);
    t=t+1;
    pause(0.01);

    hold off
end
indicador = 1;
tiem=t/10;

    dis=5.80;
    vel=dis/tiem;
    vel1=(vel*3600)/1000;
disp('***************Medidor de velocidad***************');

fprintf('%d\n',t);
fprintf('La distancia en metros es : %6.1f\n',dis);
fprintf('El tiempo en segundos es: %6.1f\n',tiem);
fprintf('La velocidad en metros/segundos es: %6.1f\n',vel);
fprintf('La velocidad en kilometros/hora es: %6.1f\n',vel1);
return
```

ANEXO C

Código utilizado

```
vid=videoinput('winvideo', 2, 'YUY2_160x120'); % Se utiliza el puerto de
video winvideo 2 para generar una entra de resolución de 160x120
set(vid,'ReturnedColorspace','rgb'); % Se transforma a imagen RGB
set(vid,'TriggerRepeat',Inf); %
set(vid,'FramesPerTrigger',1); %
% vid.FrameGrabInterval=1; %
triggerconfig(vid,'manual'); % Se configura como manual

start(vid); % se inicia el objeto vid que son las imágenes que se van
obteniendo
tic %
trigger(vid);while islogging(vid)==1, end % se obtura siempre mientras no
salga del lazo
%
k=1;% Se inicializa el contador
%     k=0;
% Ip1 = zeros(216,384,'double');
threshold=0;
s1=0;
while k<120 % Se seguirá obteniendo imágenes mientras los fotogramas que
se obtienen sean menores a 120

    k=k+1;
    data=flip(getdata(vid,1),2); % Se obtiene cada frame de la secuencia
de video
    trigger(vid); % Se realiza la obturación sobre las imágenes que se
van obteniendo
    im=imresize(data,0.5);

    %Detector bordes

    b=rgb2gray(im);% imagen a escala de grises
    a1=double(b);% Se transforma a formato double es decir cambia el
valor de cada pixel a 8 bytes
    s=size(b);%dimensiones de la matriz
    arc2=a1*0;%  matriz de ceros igual a las dimensiones de la imagen

            kerv=[1;2;1];%componente kernel vertical
            kerh=[1 2 1];%componente kernel horizontal
            kernel1=kerv*kerh/16;%filtro gausiano y se obtiene un valor
promedio al dividir;

            for i=2:s(1)-1%alto de la imagen
                for j=2:s(2)-1%ancho de la imagen
                    vent=a1(i-1:i+1, j-1:j+1);%se aisla la matriz que de
la imagen resultante y que será amultiplicada por el kernel
                    produc=vent.*kernel1;% multiplica el kernel por la
ventana
                    pix=sum(sum(produc));%se suman los valores de la
matriz
                    arc2(i,j)=pix;%se sustituye todos los pixeles de la
imagen por este nuevo valor
                end
            end
```

```
            arcz=zeros(s(1) , s(2));% matriz de ceros de las mismas
dimensiones de la imagen
            %Scharr
            kernelscharrX=[-3 0 3;-10 0 10;-3 0 3]/16;

            for i=2:s(1)-1%el alto de la imagen
                for j=2:s(2)-1%ancho de la imagen
                    vent=arc2(i-1:i+1, j-1:j+1);%s
                    produc=vent.*kernelscharrX;%se multiplica el
laplaciano por la ventana
                    pix=sum(sum(produc));% uman los valores de la matriz
                    arcz(i,j)=(pix+255)/2;%se sustituye todos los pixeles
por este nueevo valor
                end
            end
    Ient=uint8(arcz);%transforma la imagen  en valores enteros
    Ibin=im2bw(Ient,0.5059);%se binariza la imagen
    Egr=strel('disk',2);%crea un disco de grosor 2 para los bordes
    Ia=imerode(Ibin,Egr);%expande el grosor de los bordes si se resta la
imagen original menos la erosionada
    Ips=abs(Ibin-Ia-1);%gradiente morfológico
    imagesc(Ips);

if k==10 %para poder contrastar la luminosidad al momento de adecuarse la
camara, se escoge un fotograma cuando el contraste de la cámara se haya
estabilizado
    Ipd=Ips;
end
if k>10
[indicador,s1,diferencia] = compare2(Ips,Ipd,threshold,s1);%Utiliza una
función para la comparación , la misma que se utilizó para sustracción de
fondo pero con dos niveles (1 y 0)
        pause(0.01);
end
end
tiem=s1/100;

    dis=12;
    vel=dis/tiem;
    vel1=(vel*3600)/1000;

    stop(vid); %Detiene el video
    delete vid %Borra la variable vid
    clear vid %
% catch
%     stop(vid);
%     disp('Sucedió un ERROR')
% end
clc
fprintf('La distancia en metros es : %6.1f\n',dis);
fprintf('El tiempo en segundos es: %6.1f\n',tiem);
fprintf('La velocidad en metros/segundos es: %6.1f\n',vel);
fprintf('La velocidad en kilometros/hora es: %6.1f\n',vel1);

disp('FIN') %Muestra el fin del programa.
% Desconecta y borra toda la adquisicion de objetos

figure(2)
```

```
imagesc(Ipd);
imaqreset
```

Función Compare

Se utiliza la misma función que en el código de sustracción de fondo para realizar la detección de movimiento, pero la única variación que existe es la variable threshold, que en este caso debido a que la imagen es en blanco y negro tiene un valor de 0.

```
function
[indicador,t,diferencia]=compare2(entrada_imagen,fondo,threshold,t)%Se
asignan las entradas de la funcion
indicador = 0;
%
diferencia = (abs(entrada_imagen(:,:,1)-fondo(:,:,1)) > threshold);
% Elimina el ruido (eliminando pequeños agujeros y rellena las aberturas)
% a = bwlabel(diference,8);
% Realiza cierre morfológico (dilatación seguida de erosión).
b = bwmorph(diferencia,'close');
% Realiza una apertura morfológica (erosión seguida de dilatación).
diferencia = bwmorph(b,'open');
diferencia = bwmorph(diferencia,'erode',2);
% Seleccione el objeto más grande
etiqueta = bwlabel(diferencia,8);
% Mide las propiedades de las regiones de la imagen, como: 'Área', 'Centro'
y 'Rectangulo'
objeto = regionprops(etiqueta);
N = size(objeto,1); %Número de objetos en la imagen.
if N < 1||isempty(objeto) % Devuelve si no hay ningún objeto en la imagen
    return
end
% Eliminar agujeros de menos de 200 píxeles
s=find([objeto.Area]<200);
if ~isempty(s)
    objeto(s)=[ ];
end
N=size(objeto,1);% Contador de objetos
if N < 1 || isempty(objeto)
    return
end
% Dibuja un rectángulo y un punto central para cada objeto en la imagen
for n=1:N
    hold on
    centroid = objeto(n).Centroid;
    C_X = centroid(1);
    C_Y = centroid(2);

rectangle('Position',objeto(n).BoundingBox,'EdgeColor','g','LineWidth',2)
    plot(C_X,C_Y,'Color','g','Marker','+','LineWidth',2)
    title(['X= ',num2str(round(C_X)),' Y= ',num2str(round(C_X))])

    fprintf('%d\n',t);
    t=t+1;
    pause(0.01);
    hold off
end
indicador = 1;
return
```

ORDEN DE EMPASTADO