UNIVERSIDAD POLITÉCNICA SALESIANA

Carrera de Ingeniería Mecánica Automotriz

DESARROLLO DE UN ALGORITMO PARA LA ADAPTACIÓN INTELIGENTE DE LA VELOCIDAD DE UN VEHÍCULO AUTOMOTOR MEDIANTE EL USO DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN DEL TRAZADO DE LA VÍA Y LA PROXIMIDAD ENTRE VEHÍCULOS.

TESIS DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO MECÁNICO AUTOMOTRIZ

AUTORES: **JHONNY FELIX BAQUE LEÓN**

NELSON MARTIN LEÓN COBOS

LUIS LIZANDRO SARMIENTO BARRERA

DIRECTOR: **ING. NESTOR RIVERA C.**

Cuenca –Ecuador

2014

DECLARACIÓN

Nosotros: Jhonny Félix Baque León, Nelson Martin León Cobos y Luis Lizandro Sarmiento Barrera, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Jhonny Félix Baque León

Nelson Martin León Cobos

Luis Lizandro Sarmiento Barrera

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Jhonny Félix Baque León, Nelson Martin León Cobos y Luis Lizandro Sarmiento Barrera, bajo mi supervisión.

Firma: _____

Ing. Néstor Rivera C.

DIRECTOR DE PROYECTO

AGRADECIMIENTO:

En primera instancia queremos agradecer a nuestras familias por estar siempre a nuestro lado apoyándonos incondicionalmente en el transcurso de nuestras vidas y, para cumplir nuestras metas en la trayectoria estudiantil.

Se le agradece a nuestro Director de Tesis Ing. Néstor Rivera, ya que siempre nos ha estado guiando en el desarrollo de nuestra Tesis impartiendo sus conocimientos.

Agradecemos al Director de Carrera Ing. Cristian García, por el apoyo brindado durante la realización de nuestra Tesis con la prestación del laboratorio y recursos.

Un gran agradecimiento a todos los que forman parte de las labores dentro del Taller Automotriz y, a todos nuestros amigos tanto dentro como fuera de la Universidad por el gran apoyo brindado en el transcurso del desarrollo de nuestro proyecto de Tesis.

Los Autores

DEDICATORIA:

Este proyecto se lo dedico en primer lugar a mis Padres María León y Félix Baque que me han apoyado en toda mi vida estudiantil, gracias a su ayuda he alcanzado otra meta en mi vida, siempre con su apoyo incondicional, también dedico a mi hermana Fernanda por su apoyo moral e incondicional.

Dedico y agradezco a todos los amigos dentro y fuera de la universidad que me han ayudado y apoyado a culminar el proyecto, con apoyo moral y recursos, gracias a su ayuda, fue las buenas vibras y su información importante para resolver y concluir el proyecto.

Dedico también a los grupos de la Universidad, Club Automotriz y Jausp. Donde tuve la oportunidad de pertenecer y conocer a personas muy importantes que me han enseñado hacer cada día mejor persona y seguir con mis metas planteadas en la vida.

Gracias a mis compañeros de tesis, que tuve la suerte de conocerlos casi desde el inicio de la universidad, hasta realizar el proyecto de Tesis, muchas gracias por su apoyo.

Dedico y agradezco a todo el grupo de amigos, profesores y personal dentro de la universidad, que he tenido la oportunidad de conocer, ha sido muy grato ser compañero, estudiante y amigo, gracias a todos.

Jhonny Félix

DEDICATORIA:

Este proyecto se lo dedico a mi padre Ezequiel quien por el apego a sus hijos a sabido sacrificarse para que salgamos adelante, a mi madre Rosa por saber apoyarme en esos momentos difíciles y guiarme para que sea una persona de bien.

A mis hermanos y hermanas porque a pesar de no compartir el tiempo suficiente con ustedes siempre han estado a mi lado brindándome su apoyo. Dedico también este proyecto a mi abuelita quien con sus concejos y cariño han sabido darme ánimos para seguir adelante.

Al grupo de amigos que durante el paso por la universidad he tenido la oportunidad de conocerles, les agradezco porque han sabido brindarme su apoyo cuando lo necesitaba.

Nelson Martin

DEDICATORIA:

El siguiente proyecto va dedicado en primera instancia a Dios por ser una guía en mi camino diario, bendecirme con salud y por darme las fuerzas para salir adelante a pesar de las enfermedades y poder cumplir mis metas.

A mis padres Digna Barrera y Alfonso Sarmiento ya que por medio de ellos eh conseguido cada logro de mi vida con su apoyo incondicional a lo largo de mi vida y, por su apoyo a lo largo de realización del proyecto de Tesis. A mis hermanos Orlando, Isidro y Diana y mi sobrino Ronaldo que en cada momento me han estado incentivando en la continuación de mis estudios. A mi abuelita Zoila Torres que la llevo siempre a mi lado y la recuerdo mucho a lo largo de mi camino. A mi abuelito José Barrera por estar ahí apoyándome en cada momento. A todos mis amigos que han estado apoyándome y compartiendo conocimientos a lo largo del proyecto y han estado en las buenas y en las malas y gracias por esa amistad tan sincera.

Luis Lizandro

RESUMEN

El siguiente proyecto de Tesis presenta el desarrollo de un algoritmo para la adaptación inteligente de velocidad, que proporciona información al conductor para regular la velocidad de circulación teniendo en cuenta varios factores como el comportamiento dinámico del automóvil, circulación en curvas, distancia de seguridad entre vehículos y los límites de velocidad. El algoritmo es desarrollado usando visión artificial para la detección de automóviles y bordes de la vía por medio del Algoritmo de Viola Jones y la Transformada de Hough respectivamente.

El capítulo uno trata sobre el estudio de la visión artificial, de sus sistema de adquisición de imagen, el procesamiento que consta de su filtrado, segmentación, sus operaciones morfológica.

En el capítulo dos se encuentra la distancia de frenado analizando a un vehículo automotor Zuki forza 1 que se utiliza para el funcionamiento del prototipo, con el mismo vehículo se analiza la velocidad de derrape y vuelco mientras circule en una curva.

El capítulo tres se realiza lo que es la adquisición, procesamiento de imagen, el filtrado, segmentación y la operación morfológica que se los puede encontrar en Matlab® y Simulink®. Lo que funciona para la detección del trazado de la vía y la aproximación a vehículos.

En el capítulo cuatro se especifica la realización de los algoritmos para la aproximación de vehículos como también de la detección de líneas del trazado de la vía que se lo realizo en el entorno de Simulink® y Matlab®.para completar el algoritmo se culmina relacionando las cuatro señales lo que es aproximación de curvas, detección de curvas, velocidad del vehículo y la señal de freno

Como capitulo cinco tenemos como se adaptó el sensor de velocidad los dos algoritmos, la señal de freno y las tarjetas electrónicas y los indicadores lumínicos y acústicos para el conductor.

En los capítulos sextos se menciona todos los resultados que se obtuvo y el análisis estadístico mediante el proceso ANOVA, para comprobar la efectividad del sistema y con ello la Hipótesis de 20%.

ÍNDICE DE CONTENIDOS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

ÍNDICE DE FIGURAS.

CAPITULO 1

CAPITULO 2

CAPITULO 3

CAPITULO 4

CAPITULO 5

CAPITULO 6

ÍNDICE DE TABLAS.

CAPITULO 2

CAPITULO 4

CAPITULO 5

CAPITULO 6

CAPITULO 1

FUNDAMENTOS TEÓRICOS DEL USO DE VISIÓN ARTIFICIAL.

1. FUNDAMENTOS TEÓRICOS DEL USO DE VISIÓN ARTIFICIAL.

El siguiente capítulo tiene el objetivo de explicar los fundamentos teóricos para el uso de visión artificial. Hay un sin número de herramientas, dependiendo la escena captada y los objetos a identificar, por ejemplo: Sistemas de parqueo, detección de huellas digitales, detección de temperatura, aplicaciones en robótica, etc.

Se explicará lo más relevante para el desarrollo del algoritmo de adaptación inteligente de velocidad como es: formación y representación de la imagen, tipos de imágenes, efectos de muestreo y la cuantificación, relación entre pixeles, entre otros. Todo esto es necesario debido a las ideas principales del proyecto, que es detección del trazado de la vía y la proximidad de vehículos en tiempo real.

1.1 FORMACIÓN Y REPRESENTACIÓN DE LA IMAGEN.

La formación de una imagen es la información en 3D (escena) proyectada en un plano 2D, para realizar este proceso intervienen los siguientes elementos: el objeto, la fuente radiante y el sistema de formación de la imagen que puede ser: sistema óptico, un sensor, video cámara.

Una imagen es la representación de un objeto por una fuente luminosa. La representación depende de la cantidad de luz que reflejen los objetos, por esto se considera, dos componentes importantes para representar una imagen que es: la fuente luminosa de la escena en la imagen y la luz reflejada por los objetos. Dichas componentes reciben el nombre de luminancia y reflectancia, con representación de $I(x, y)$ y $R(x, y)$ respectivamente, donde ambas funciones se combinan para dar un producto de $f(x, y)$[1].

La imagen es representada por medio de una matriz f de dimensiones NXM donde cada elemento es un pixel, que da la intensidad en cada punto de la imagen.

$$f = \begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & \dots & f(1,M) \\ f(2,1) & f(2,2) & f(2,3) & \dots & f(2,M) \\ f(3,1) & f(3,2) & f(3,3) & \dots & f(3,M) \\ \vdots & \vdots & \vdots & & \vdots \\ f(N,1) & f(N,2) & f(N,3) & \dots & f(N,M) \end{bmatrix}$$

Figura 1.1: Representación de Imagen en Matriz**.**
Fuente: [1].

1.1.1 PIXEL.

Viene de la abreviatura (Picture element). En la representación de la imagen el pixel es una cuadricula capturada de esta forma por cámaras digitales. Está formada por varios colores el rojo (R), verde (G) y el azul (B), se tiene por ejemplo 1028*615, los 1028 representan el número horizontal de pixeles y los 615 representa el número vertical de pixeles. Al momento que se realiza un zoom en la imagen se puede observar los pixeles como en la Figura 1.2.

Figura 1.2: Pixel.
Fuente: Los Autores.

En cuanto a la resolución de la imagen esta se expresa en pixeles por pulgada (ppp), mientras que en las cámaras digitales son expresadas en Mega Pixeles, por ejemplo una cámara de 6MP, toma una fotografía con 6 millones de pixeles.

La profundidad de color trata sobre el número de bits con que guarda la información de cada pixel de la imagen tratada. El bit puede tener el valor de **0** o **1**, 1 bit para imágenes en blanco y negro, 2 bits para 4 colores, 3 bits para 8 colores, 8 bits para 256 colores, 24 bits para 17.7 millones de colores[2].

1.2 EFECTOS DEL MUESTREO Y LA CUANTIFICACIÓN.

Para el procesamiento digital de imagen debemos de tener presente dos factores importantes, ya que estos pueden ocasionar perdida de información, estos son la naturaleza discreta de los pixeles de la imagen y le rango limitado de valores de

intensidad luminosa, que corresponden respectivamente al muestreo y la cuantificación.

1.3 EFECTO DE MUESTREO.

El muestreo de una imagen tiene el efecto de reducir la resolución espacial de la misma. Se podría decir que cambia el tamaño de la imagen representada en matriz, lo cual cambia el número de pixeles de la imagen, en la siguiente Figura 1.3. puede observarse la pérdida de información introducida con el aumento del paso de muestreo, así como el ruido que se va introduciendo en forma de patrones rectangulares sobre la imagen [3].

Figura 1.3: Izquierda a Derecha de Arriba Hacia Abajo: Muestreo 256x256, 128x128, 64x64, 32x32 Pixeles.
Fuente: Los Autores.

1.3.1 EFECTO DE LA CUANTIFICACIÓN.

El efecto de cuantificación se da por la imposibilidad de tener un rango infinito para la intensidad de cada pixel dentro de la imagen, hasta el momento con que el

dispositivos capturar la imagen solo se consigue los 10 bits de información, pero por lo general se ocupa 8 bits o 256 niveles de gris para codificar este valor lumínico [1].

Dependiendo del procesamiento digital de imagen y la información que se quiera procesar, se establece el rango de bits a utilizar para un proceso correcto, por lo cual no hay norma para el valor de intensidad de cada pixel a utilizar, en la siguiente Figura 1.4. Podemos observar la misma imagen en diferentes valores de bits.

Figura 1.4: Efecto de Cuantificación: Izquierda a Derecha: 8 a 1 bits.
Fuente: [1].

1.4 RELACIONES BÁSICAS ENTRE PIXELES.

En este tema se describe las relaciones entre pixeles, es elemental para la obtención de información, por lo cual a continuación vemos algunas relaciones.

1.4.1 VECINOS DE UN PIXEL.

La vecindad se define como la relación que tiene un pixel de manera posicional con los pixeles más cercanos a él como dice [4]. En una imagen existen dos tipos de vecindad que posee cada pixel, 4-vecinos y la de 8-vecinos.

El pixel de 4-vecinos tiene pixeles (V1, V2, V3 y V4), donde cada uno se encuentra arriba, abajo, izquierda y derecha del pixel principal P. el pixel de 8-vecinos tiene pixeles (V1, V2, V3, V4, V5, V6, V7 y V8), donde tiene como el pixel de 4-vecinos y 4 más en diagonal con el pixel principal P. La siguiente Figura 1.5. Muestras cada una de las vecindades.

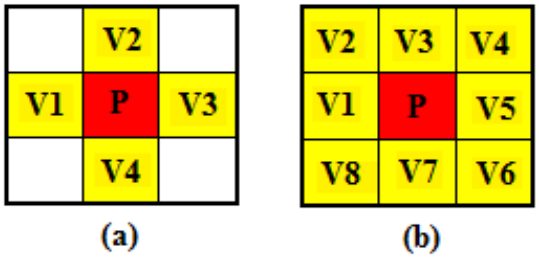


Figura 1.5: Vecindad de un Pixel P. (a) Vecindad 4 y (b) Vecindad-8.
Fuente: [4].

1.4.2 CONECTIVIDAD.

La conectividad entre pixeles es utilizada en la detección de regiones u objetos presentes en una determinada imagen como dice [4]. La conectividad se define como una situación de adyacencia y vecindad. Por lo cual existen dos tipos de conectividad, conectividad 4 y la conectividad 8. Dentro del sistema de visión artificial es importante, considerar la conectividad como una imagen binaria, por lo cual los pixeles serán solo uno y cero, estos pasaran a representar una característica en lugar de luminosidad.

En la interpretación de imágenes y el reconocimiento de objetos, se puede ver la importancia de este concepto en la siguiente Figura 1.6. Ya que se representan uno o dos objetos en imagen binaria según el tipo de conectividad utilizada. La conectividad-4 tendrá dos objetos donde en el punto más cercano no está en relación de vecindad 4-vecinos. En cambio aplicando el mismo concepto para una vecindad de 8-vecinos, los dos pixeles de contacto estarían conectados, por lo que se considera un solo objeto.

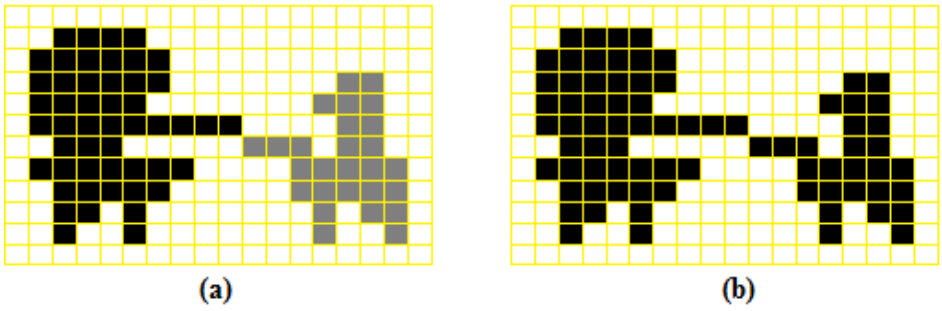


Figura 1.6: Conectividad de pixeles. (a) Conectividad- 4 y (b) Conectividad- 8.
Fuente: Los Autores.

1.5 TIPOS DE IMÁGENES.

Los tipos de imágenes a utilizar en sistemas de visión artificial, depende de la información a ser procesada.

1.5.1 IMÁGENES DE COLOR RGB.

La imagen de color RGB se entiende por tener tres canales de colores, comparando con una imagen en escalas de grises solo tiene un canal, donde los canales están coloreados de rojo, verde y azul cada uno. Las siglas RGB son sus siglas en ingles de cada color. En la siguiente Figura 1.7. (Imagen a color) y Figura 1.8. (Tres canales).

Figura 1.7: Imagen a Color.
Fuente: Los Autores

Figura 1.8: Imagen con Tres Canales.
Fuente: Los Autores.

1.5.2 IMAGEN A ESCALA DE GRISES.

La escala de grises es la que se emplea a una imagen de [0, 255] o también llamada imagen de intensidad, donde cada pixel representa un valor en la escala como podemos ver en la Figura 1.9., teniendo en cuenta que la escala va desde **0** que representa el color negro hasta **255** que representa el color blanco.

Figura 1.9: Escala de Grises o Intensidad.
Fuente: Los Autores.

1.5.3 IMÁGENES BINARIAS.

Imagen binaria es muy importante para procesos de filtrado, ya que solo es un arreglo que solo contiene unos y ceros, donde cada pixel toma valores de ceros y unos para la representación de ciertos objetos, como se puede ver en la Figura 1.10. Estos unos y ceros son especiales, porque no implican valores numéricos sino más bien banderas que indican el estado de falso (**0**) o verdadero (**1**) [4].

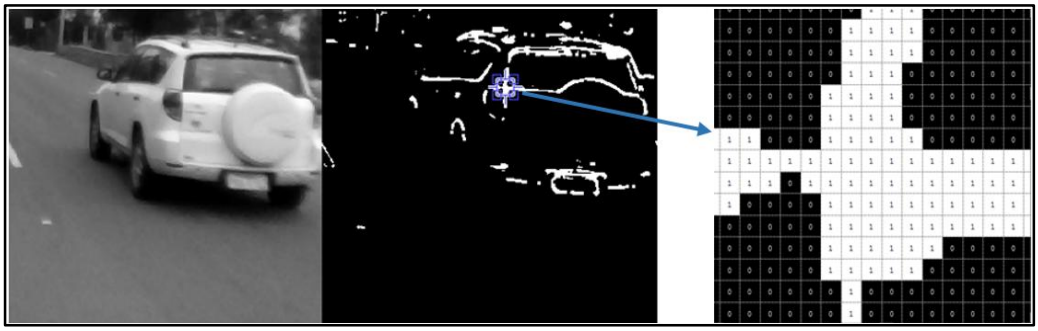


Figura 1.10: Imagen Binaria.
Fuente: Los Autores.

1.6 CARACTERÍSTICAS.

En visión artificial detectar objetos se da por medio de sus características como color, tamaño o la forma, pero esto no suele ser suficiente porque en la mayoría de los casos, varios objetos pueden compartir estas mismas características o incluso que estas características varíen para un mismo objeto en función de las condiciones en las que se perciban (distancia, perspectiva, iluminación).

Pero hay técnicas desarrolladas como la umbralización o la detección de bordes para encontrar ciertas características para cada uno de los objetos. Hay casos que dependiendo de la aplicación no son suficiente [5].

A continuación se mencionaran técnicas desarrolladas por programas de visión artificial que son necesarias para la detección de características como: características Haar, características LBP y características HOG.

1.6.1 CARACTERISTICAS HAAR.

Las características Haar según estudios realizados solo fueron para la detección caras y peatones [6], pero su uso fue significativo más tarde cuando Viola y Jones adaptaron la idea desarrollando una nueva versión a la que denominaron **características Haar (Haar-Like Features)** [7]**.**

Esta técnica es en detectar los objetos en base a las estructuras de los niveles de intensidad de los pixeles de una imagen. Para obtener la información se implementará una serie de funciones dado las denominadas características Haar. La ventaja es que permite detectar la estructura del objeto aunque esta no sea uniforme.

La característica Haar se define como una ventana de pixeles de tamaño y orientación variables, dividida en regiones rectangulares, estos se los denomina positivos y negativos como se puede ver en la siguiente Figura 1.11 donde están 4 pixeles [5].

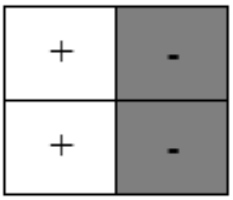


Figura 1.11: Imagen Binaria.
Fuente: [5].

Esta ventana de pixeles se va desplazando sobre la imagen de detección evaluando, por una parte, la suma de los pixeles que se caen sobre las regiones positivas y la suma de los pixeles que caen sobre las regiones negativas. A la diferencia entre la suma de las regiones positivas y la suma de las regiones negativas será lo que se denomine el valor de la característica. Así pues, el valor de una característica *Haar* en un punto $H(x, y)$, se puede representar mediante la siguiente fórmula:

$$H(x,y) = \sum_{p} I(x,y) - \sum_{n} I(x,y)$$

Donde $I(x, y)$ representa la imagen a evaluar, $\boldsymbol{p}$ y $\boldsymbol{n}$ representan respectivamente las regiones positivas y negativas de la característica *Haar* sobre la imagen. Mediante esta operación se obtendrá un mapa con los valores de la característica en cada posición de la imagen de detección. Estos valores son utilizados después y junto con el umbral para clasificar las diferentes regiones de la imagen.

Figura 1.12: Proceso de aplicación de una característica Haar a una imagen.
a) Desplazamiento de la característica sobre la imagen. b) Cálculo de los valores de la característica.
c) Umbralizado de los valores de la característica.
Fuente: Los Autores.

1.6.2 CARACTERISTICAS LBP.

Las características LBP (Local Binary Patterns) comienzan como una técnica para encontrar las texturas de las imágenes a través de las llamadas unidades de texturas. Esta técnica consiste en evaluar la disposición espacial y el contraste de los pixeles en pequeñas regiones de la imagen. Esta permite obtener un espectro de textura por cada región que permite encontrar la textura global de la imagen. Pero años más tarde utilizaron las unidades de textura, designándoles un número binario a cada unidad, de tal manera que es más fácil identificarlas [8]**.**

Para encontrar el valor de característica LBP, comienza con un entorno de la imagen de 3x3 pixeles, donde se compara la intensidad de cada uno de los ocho pixeles vecinos con la intensidad del pixel central. Si la intensidad del pixel vecino es mayor o igual que la del pixel central, se le asigna a la posición del pixel vecino un valor de **1** o, en caso contrario, un valor de **0**. Una vez comparados todos los pixeles, se tendrá un conjunto de ceros y unos que representarán un número binario. Multiplicando cada posición del número binario por su peso en decimal y sumando todo, se obtiene el valor de la característica *LBP* con el que se etiquete al pixel central. En la siguiente Figura 1.13 se muestra un ejemplo para la explicación.

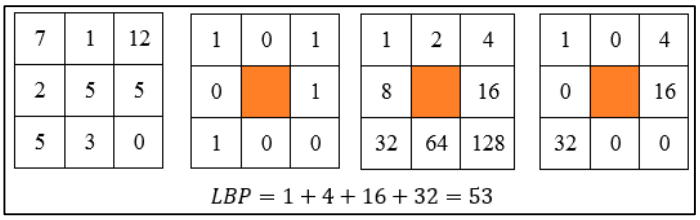


Figura 1.13: Característica LBP con el que se Etiquete al Pixel Central.
Fuente: Los Autores.

Una descripción más formal para encontrar la característica *LBP* para el pixel central, puede darse mediante la siguiente expresión:

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} s(I_p - I_c)\, 2^p$$

Con (x_c, y_c) la posición del pixel central, el número de pixeles de la vecindad, I_p la intensidad de los pixeles vecinos, I_c la intensidad del pixel central y la siguiente función:

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & si\ no \end{cases}$$

En definitiva, para cada pixel de la imagen se obtiene un valor asociado que indica el tipo de textura que presentan sus 8 pixeles vecinos. Con esta combinación de valores, a lo máximo se podrán representar 28 o 256 tipos de unidades de texturas. Así pues, calculando el histograma de la característica *LBP* y analizando sus valores, se puede extraer un patrón identificativo de texturas que permitan clasificar la imagen como observamos en la Figura 1.14.

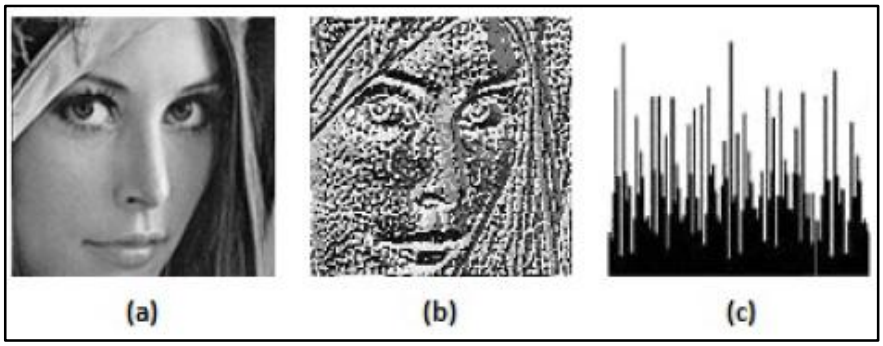


Figura 1.14: Proceso de cálculo de la característica LBP de una imagen.
(a) Imagen a evaluar. (b) Característica LBP. (c) Histograma de la característica LBP.
Fuente: [5].

1.6.3 CARACTERISTICAS HOG.

Las características HOG cuyas siglas significan Histogramas de los Gradientes Orientados, consiste en la división de la imagen en sub-bloques distribuidos a lo largo y ancho de la misma y con cierto cubrimiento entre ellos. Cada bloque se subdivide en sub-bloques o celdas y sobre estos se calcula la magnitud y orientación de los gradientes en cada píxel. Sobre cada uno de estos bloques se calcula el histograma de los gradientes promediado por un peso gaussiano, y luego se almacena en el vector de características de la imagen [9].

La diferencia añadida que representa los métodos HOG consiste en que los gradientes no se calculan uniformemente sobre un mallado denso, sino que se divide la imagen en bloques, y a su vez cada bloque en diversos sub-bloques, y se calcula en cada uno de ellos los gradientes y el histograma.

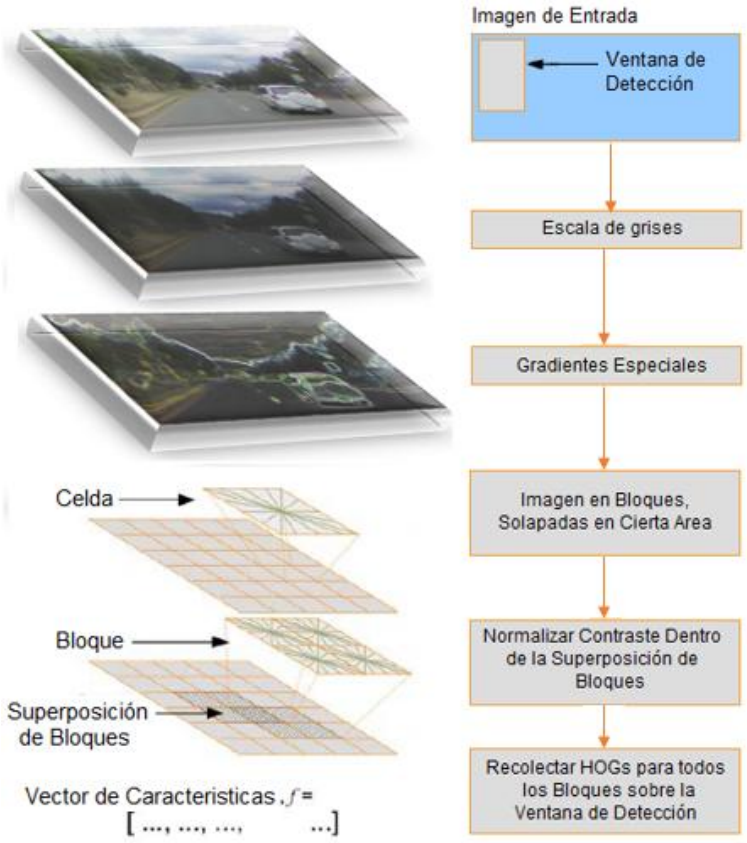


Figura 1.15: Proceso de Extracción de Características para una Ventana de Detección.
Fuente: [9].

Observamos en la Figura 1.15 que da una imagen en color, se transforma a escala de grises. A continuación se calculan los gradientes espaciales sobre toda la imagen.

Posteriormente, dividimos la imagen en bloques, solapadas cierta área. Cada bloque, a su vez, se divide en sub-bloques o celdas. Para cada celda calculamos los histogramas de los gradientes orientados. Finalmente se aplica una ventana gaussiana sobre cada bloque que se repite para todos los bloques de la imagen.

1.7 ETAPAS DE UN SISTEMA DE VISION ARTIFICIAL.

El sistema de visión artificial intenta representar el comportamiento del ser humano cuando observa una imagen, esta imagen se procesa por medio del cerebro para realizar una acción. Las etapas de un sistema de visión artificial se generalizan para todas sus aplicaciones, a continuación veremos cada etapa.

1.7.1 LA CAPTURA O ADQUISICIÓN DE IMÁGENES.

La captura de imágenes es únicamente sensorial, donde se dará por medio de una video cámara para captar cada escena al conducir, dentro de esta es muy importante la iluminación, de acuerdo al tema del proyecto la iluminación será por el ambiente, en la noche se dará la iluminación frontal direccional con faros del automóvil.

1.7.2 TRATAMIENTO DIGITAL DE LA IMAGEN O PREPROCESAMIENTO.

Esta ayuda a facilitar el proceso de etapas posteriores. Mediante filtros y transformaciones geométricas, se eliminan partes indeseables o se realzan partes interesantes de la imagen.

1.7.2.1 Filtros.

El filtrado tiene la función de reducir el ruido para un cambio de intensidad, sombras del mismo objeto y cualquier efecto que se presente en la imagen digitalizada como consecuencia del muestreo, cuantización, transmisión o de acuerdo a otras perturbaciones presentes en la imagen. Dentro de ellos tenemos dos clases de filtros los lineales y los no lineales que a continuación se describen [10]-[11].

1.7.2.2 Filtro Lineales.

Son designados filtros lineales ya que los valores de intensidad de los pixeles se los realiza una combinación linealmente para generar un pixel resultado[11].

Se obtiene dos clases de filtros el de suavizado y el de diferencia.

En los filtros de suavizado tenemos el filtro "*Box*" el cual consta de una caja como su nombre lo dice y todos sus coeficientes tienen el mismo valor. También el filtro "*Gaussiano*" el elemento central es el peso máximo y mientras que los otros valores tienen una menor influencia a medida que se alejen del central.

Como también se obtiene lo que es la convolución y la correlación.

1.7.2.2.1 Convolución.

El sistema de convolución se da haciendo rotar 180° la matriz, lo que cambiaría el orden de la secuencia más no el punto de referencia lo cual permanecerá en la misma posición.

En la figura siguiente se tiene el punto de referencia en 5 lo cual nos queda.

2	3	5	1	0

0	1	5	3	2

Figura 1.16: Convolución.
Fuente:[11].

1.7.2.2.2 Correlación.

La correlación a diferencia de la convolución es que no se da un giro de 180°. Se trata de desplazamientos de las matrices y se realiza una serie de intercambios y se añade ceros tanto al final como al inicio de la función y luego se realiza una suma de los diferentes números de la otra función. A continuación se realiza un ejemplo.

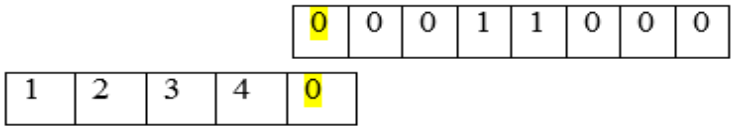
Funciones:

0	0	0	1	1	0	0	0

1	2	3	4	0

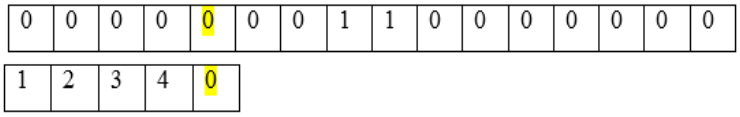
Primer Paso.

Se los ponen en el mismo sentido los puntos fijos que están descritos de color amarillo.



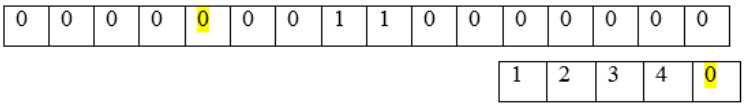
Segundo Paso:

Secuencia de ceros tanto al lado derecho e izquierdo de la función, con ceros tanto a la derecha como la izquierda de la función, la cantidad de ceros se realiza viendo cuantos casilleros queda desde el punto fijo hacia la izquierda de la segunda función.



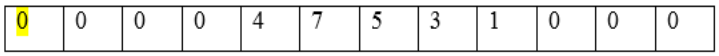
Tercer Paso:

Desplazamiento de la segunda función hacia la derecha.



Cuarto Paso:

Sumar en secuencia los valores de la segunda función y nos queda la correlación del a siguiente forma[11].



1.7.2.3 Filtro no Lineal.

Los filtros lineales tienen una gran desventaja para suavizar y quitar perturbaciones tales como puntos, bordes y líneas. En los filtros no lineales se puede dar en escalón o líneas.

Tenemos algunos filtros no lineales a continuación.

1.7.2.3.1 Filtro de la Mediana.

El filtro permite eliminar puntos no deseados o figuras dentro de la imagen ya que esto afecta a los bordes que queremos obtener de la imagen. Se trata de un filtro estadístico.

En estadística la mediana es el valor de la variable que deja el mismo número de datos antes y después que él.

Para el cálculo de la media se realiza, acomodar los valores de intensidad de la imagen luego acomodarlos en forma creciente, si existen valores repetidos también se los repite en el arreglo como en el siguiente ejemplo.

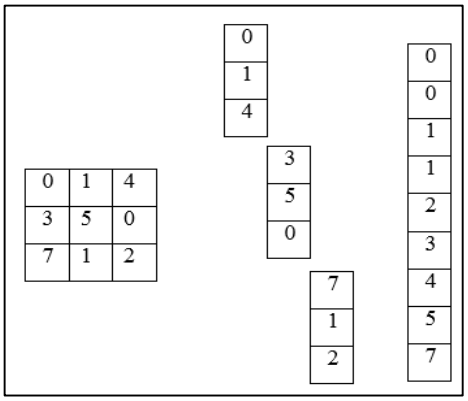


Figura 1.17: Desarrollo de un filtro de la Mediana.
Fuente:[11].

1.7.3 SEGMENTACIÓN.

Es un proceso muy importante ya que se trata de la detección de bordes de una imagen, se da esta detección ya que el ojo humano distingue los objetos mediante los contornos de la figura, y por ello es que realiza la detección de los bordes. Los filtros ocupan la primera derivado tanto en sus filas y en sus columnas de la imagen. Los bordes son considerados como puntos en una imagen donde la intensidad cambia drásticamente de dirección. De acuerdo al cambio de intensidad será el valor del borde en la imagen para ese punto, esto se realiza mediante el cálculo de la derivada[11].

1.7.3.1 Operadores Prewitt y Sobel.

Estos son los dos métodos más utilizados para la detección de bordes. Los dos operadores utilizan una matriz de 3*3 ya que nos permite que el filtro no sea tan vulnerable al ruido que se encuentra en la imagen.

El operador Prewitt nos presenta resultados más ruidosos, el filtro está definido por las siguientes matrices tanto en ***x*** con en ***y***.

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{Y} H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Figura 1.18: Filtro Prewitt.
Fuente:[11].

El filtro Sobel tiene un filtrado prácticamente idéntico al Prewitt con una diferencia que se le da una mayor peso al renglón o columna central del filtro como se define a continuación. Los pixeles que tiene un valor de **1** son los bordes y los que son **0** son los que no tienen bordes [11].

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{Y } H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Figura 1.19: Filtro Sobel.
Fuente:[11].

1.7.3.2 Operadores Roberts.

Este filtro es uno de los más antiguos ya que no se lo considera muy importante para el tratamiento de imágenes ya que solo consta con una matriz de 2*2.

1.7.4 RECONOCIMIENTO O CLASIFICACIÓN.

En ella se pretende distinguir los objetos segmentados [12], analizando características que se establecen previamente para el reconocimiento.

Las cuatro etapas mencionadas no siempre siguen la misma secuencia, en donde se deben retroalimentar, ya que dependiendo de las imágenes o el video en la etapa de captura, dependerá el proceso de las etapas, por ejemplo en la segmentación, puede cambiar de acuerdo como cambie el video, entonces la programación del algoritmo debe ser lo más efectiva posible, dependiendo lo que se quiera reconocer.

En la Figura 1.20 nos muestra las cuatro etapas de un sistema de visión artificial, en donde se puede ver una retroalimentación o como otros Autores lo llaman base del conocimiento [13]. Donde todas las etapas tienen en común ya que si una cambia las demás se adaptan.

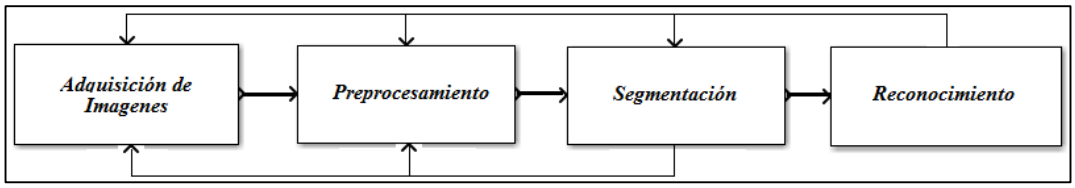


Figura 1.20: Diagrama de Bloques de las Etapas Típicas en un Sistema de Visión Artificial.
Fuente: Los Autores.

1.8 COMPONENTES DE UN SISTEMA DE VISION ARTIFICIAL.

Los componentes para un sistema de visión artificial en general son cuatro que son: sensor óptico, tarjeta de adquisición de imagen, computador, monitor de video.

1.8.1 SENSOR ÓPTICO.

Estas pueden ser cámaras de video, u otros componentes dependiendo de la aplicación como cámaras de color o monocromo, cámara scanner. En el proyecto a presentar se utilizaran dos webcams para aumentar el campo visual y mejorar la seguridad del conductor.

1.8.2 TARJETA DE ADQUISICIÓN DE IMAGEN.

Permite digitalizar la señal de video por el subsistema anterior. En nuestro caso no se utilizará tarjeta de adquisición ya que las webcams serán conectadas directamente al computador llegando la imagen digitalizada.

1.8.3 COMPUTADOR.

Para almacenar el video digitalizado para el procesamiento respectivo con el software.

1.8.4 MONITOR.

Permite observar imágenes o escenas captadas, como los resultados del procesamiento de imagen.

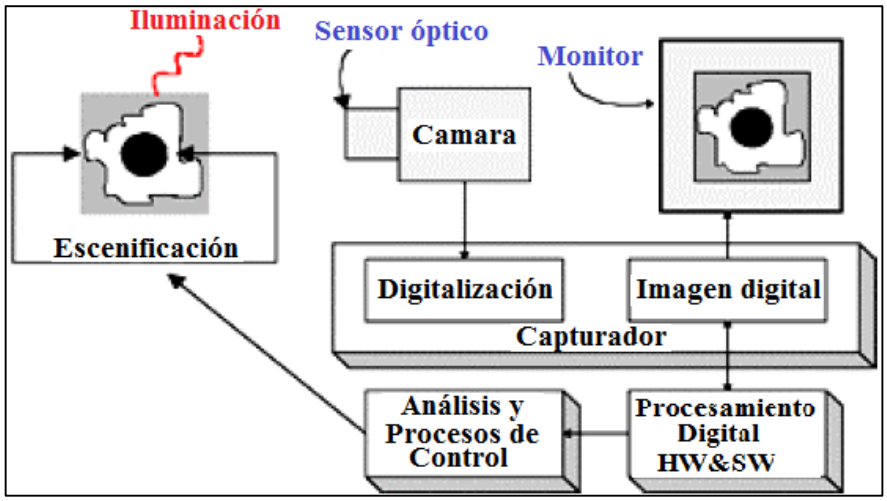


Figura 1.21: Diagrama de bloques de un SVA
Fuente: [1].

Al mencionar los componentes de un sistema de visión artificial, se escucha muy simple, pero debemos de tener en cuenta que para un procesamiento correcto de la escena captada, se debe realizar con mucha precisión el algoritmo para detección del trazado de la vía y la proximidad entre vehículos. Posterior a esto se realizará con una tarjeta electrónica de adquisición de datos para realizar los indicadores que ayudaran al conductor a la prevención de accidentes.

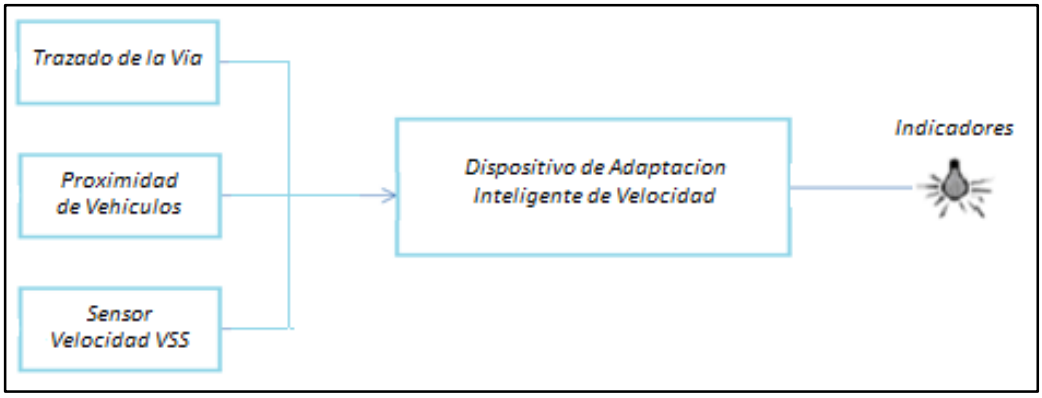


Figura 1.22: Diagrama de Bloques de Adaptación Inteligente de Velocidad.
Fuente: Los Autores.

CAPITULO 2

ANÁLISIS DE LA RESPUESTA DINÁMICA DEL VEHÍCULO EN CIRCULACION EN CURVAS Y FRENADO.

2. ANÁLISIS DE LA RESPUESTA DINÁMICA DEL VEHÍCULO EN CIRCULACION EN CURVAS Y FRENADO.

Para el análisis de circulación en curvas y frenado del vehículo, se considera primero los conceptos básicos para realizar un análisis de la respuesta dinámica, en conjunto con el sistema de visión artificial debe trabajar para dar el aviso informativo de control inteligente de velocidad.

Para circulación en curvas se analizarán todo lo referente a velocidad de vuelco, velocidad de derrape, donde nos dará los límites de circulación del vehículo cuando circule en curva los cuales son comparados con los límites establecidos en la ley orgánica de transporte terrestre. Para el análisis de distancia de frenado se analizarán todos los aspectos necesarios para encontrar la distancia de frenado como es tiempos de reacción del conductor y del sistema de frenado, centro de gravedad del automóvil, deslizamiento, entre otros.

2.1 CIRCULACIÓN EN CURVA.

La forma de las vías y curvas es una de las causas principales para accidentes de tránsito, esto se puede confirmar según datos del ECU911, en la Figura 2.1. Donde el color rojo refleja los altos índices de accidentes de tránsito.

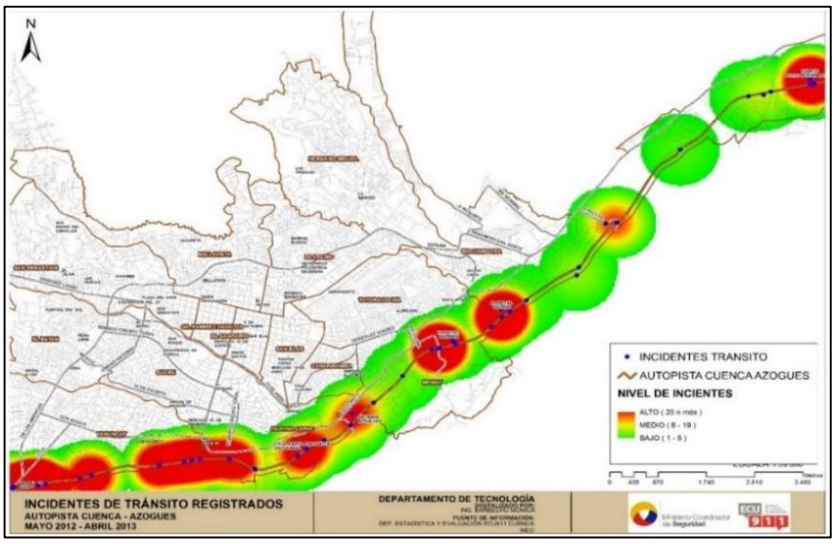


Figura 2.1: Incidentes de Tránsito en Autopista Cuenca – Azogues.
Fuente: ECU 911 Cuenca.

La velocidad de circulación en curvas para vehículos livianos es establecida en la Ley Orgánica de Transporte Terrestre, como se puede observar en la Tabla 2.1, este valor servirá para los análisis posteriores en este capítulo.

Tabla 2-1: Límites de Velocidad Para Vehículos Livianos
Fuente: Ley Orgánica de Transporte Terrestre.

Tipo de vía	***Límite máximo***	***Rango moderado (Art. 142.g de la Ley)***	***Fuera del rango moderado (Art. 145e de la Ley)***
Urbana	50 Km/h	>50 Km/h - <60 Km/h	>60 Km/h
Perimetral	90 Km/h	>90 Km/h - <120 Km/h	>120 Km/h
Rectas en carreteras	100 Km/h	>100 Km/h - <135 Km/h	>135 Km/h
Curvas en carreteras	60 Km/h	>60 Km/h - <75 Km/h	>75 Km/h

2.1.1 VELOCIDADES LIMITE DE DERRAPE Y DE VUELCO.

Cuando el vehículo circula en una trayectoria curva, la fuerza centrífuga se aplica sobre el centro de gravedad del automóvil, este se encuentra a una altura ***h*** de la calzada donde circula. Esto genera un esfuerzo lateral y un momento de vuelco. El esfuerzo lateral es compensado por las fuerzas de adherencia entre los neumáticos y la calzada [14]. Al tomar la curva a velocidades límites se incrementa ambos efectos, donde el vehículo puede perder su trayectoria dependiendo de las condiciones de la calzada.

Para la determinación de velocidades se puede considerar la suspensión rígida donde el desplazamiento del centro de gravedad no es notable, pero considerando en real es despreciable la flexibilidad de la suspensión. También se supone que la calzada en la curva tiene un ángulo de inclinación con respecto a la horizontal, como se observa en la Figura 2.2.

2.1.1.1 Cálculo Aproximado de la Velocidad Límite de Derrape.

El siguiente cálculo sirve para determinar la velocidad limite derrape en una curva, en este caso para encontrar las velocidades límites al cual el vehículo ingresara a la curva, el vehículo puede volcar, ya que al derrapar y encontrar un obstáculo o deformaciones de la vía, puede ocasionar un accidente.

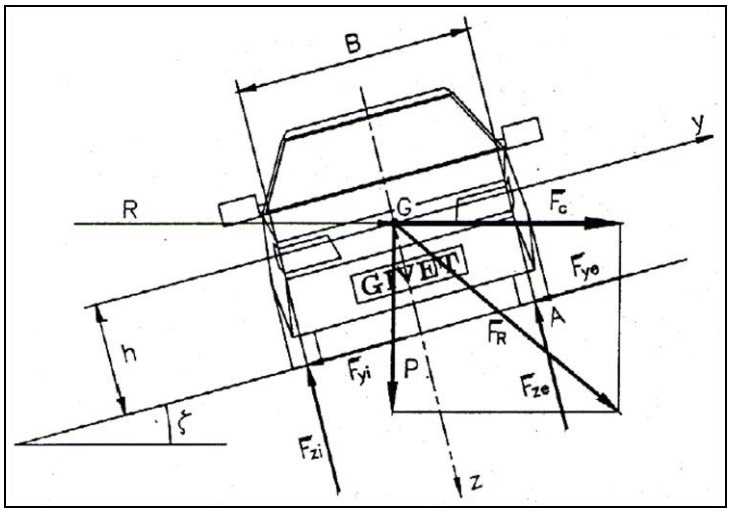


Figura 2.2: Modelado Bidimensional para el Cálculo Aproximado de las Velocidades Limite de Derrape y de Vuelco.
Fuente: [14].

Observamos la Figura 2.2 donde ***B*** es el ancho del vehículo, ***R*** es el radio de la curva y ***h*** es la altura del centro de gravedad. Las ecuaciones encontradas en [14] de la velocidad límites de derrape, donde el μ_{ymax} es el límite de adherencia para que los neumáticos comiencen a derrapar, teniendo en cuenta que la siguiente ecuación es con un peralte de curva (grados de inclinación de la vía).

$$V_{ld} = \sqrt{gR\frac{\mu_{ymax} + tg\zeta}{1 - \mu_{ymax}.tg\zeta}}$$

Si la curva no está peraltada $\zeta = 0$.

$$V'_{ld} = \sqrt{gR\mu_{ymax}}$$

2.1.1.2 Cálculo Aproximado de la Velocidad Límite de Vuelco.

Para el siguiente cálculo se considera las fuerzas que actúan sobre el centro de gravedad del vehículo que son Peso (P) y Fuerza centrífuga (Fc), analizando con la Figura 2.2. Desde el punto **A** donde corta a la superficie de rodadura en el punto exterior de la huella contacto del neumático exterior [14].

Las ecuaciones de velocidad límite de vuelco son.

Con un peralte de curva:

$$V_{lv} = \sqrt{gR\frac{\mathrm{B/2h} + tg\zeta}{1 - \mathrm{B/2h}\,.\,tg\zeta}}$$

En el caso del peralte nulo:

$$V_{lv} = \sqrt{gR\frac{\mathrm{B}}{2h}}$$

Ahora como podemos observar, las ecuaciones de velocidad límite de derrape y vuelco, son análogas, pudiéndose obtener una de la otra sin más que sustituir μ_{ymax} por B/2h. Esto nos da el siguiente análisis.

a) Si $\mu_{ymax} = \frac{B}{2h}$; $V_{ld} = V_{lv}$ ambos efectos ocurrirían para el mismo valor de velocidad, suponiendo valores determinados de R y ζ.

b) Si $\mu_{ymax} > \frac{B}{2h}$; $V_{ld} > V_{lv}$ esto significa que el vehículo volcaría al alcanzar la velocidad superior a velocidad de vuelco sin llegar a derrapar. Esto se puede dar en vehículos cuyo centro de gravedad está a una altura alta en relación con la vía y siempre que la adherencia sea suficientemente alta.

c) Si $\mu_{ymax} < \frac{B}{2h}$; $V_{ld} < V_{lv}$ esto da que el vehículo comenzará a derrapar antes de volcar, esto puede darse en vehículos de turismo o industriales circulando con adherencia no muy elevada.

A continuación veremos el cálculo de las velocidades límites de derrape y vuelco para el respectivo análisis, considerando un automóvil experimental Suzuki Forsa 1.

Figura 2.3: Suzuki Forsa 1.
Fuente: Autores

Especificaciones generales de SUZUKI FORSA 1:

Tabla 2-2: Especificaciones Dimensionales y de Peso de Suzuki Forsa 1.
Fuente: Autores.

DIMENSION	
Longitud total	3670mm (144.5in)
Ancho total	1545mm (60.8in)
Altura Total	1350mm (53.1in)
Distancia entre ejes	2245mm (88.4in)
Vía Delantera	1335mm (52.6in)
Vía Trasera	1300mm (51.2in)
Altura libre sobre el suelo	180mm (7.1in)
PESO	
Peso del vehículo sin carga	*680 kg (1499 lb.)
	*700 kg (1543 lb.)
Peso bruto del vehículo	*1170 kg (2580 lb.)
	*1200 kg (2645 lb.)

Analizamos el Suzuki Forsa 1, con un análisis de cuerpo libre, con los respectivos pesos del automóvil, considerando el coeficiente de adherencia máximo de las ruedas en asfalto seco, el radio de curva que se establecerán posteriormente. Todos estos datos nos sirven para encontrar las distancias del Centro de Gravedad, necesarios para el cálculo de las Velocidades de Derrape y Vuelco.

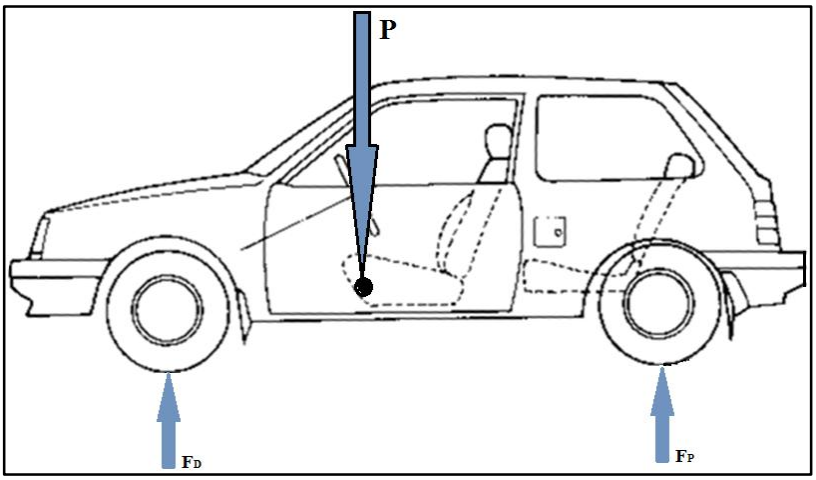


Figura 2.4: Esquema del Suzuki Forsa 1.
Fuente: Autores.

2.1.1.3 Calculo del Centro de Gravedad.

Para encontrar la distancia del centro de gravedad, hallamos los pesos del eje delantero y el posterior y el total del automóvil de prueba, en este caso se toma el peso real por medio de una plataforma para medir los pesos reales.

Figura 2.5: Peso de Suzuki Forsa 1.
Fuente: Autores.

Ahora encontramos la distancia del CG a la línea central del eje delantero (x), donde tenemos un peso total de WT = 680kg, donde el peso del eje delantero es WF = 410kg, y del eje posterior es WR = 270kg, la distancia que existe entre ejes (batalla) es de LB = 2245mm, remplazando en la fórmula.

$$x = \frac{WR * LB}{WF + WR}$$

$$x = \frac{270kg * 2245mm}{410kg + 270kg}$$

$$\boldsymbol{x = 892\ mm}$$

Para encontrar la distancia en el eje y, según la designación de los neumáticos, parámetros fundamentales [14], encontramos el radio del neumático.

Neumático del vehículo:

Figura 2.6: Descripción de Medidas del Neumático.
Fuente: Autores.

Tiene las Siguientes Características:

- Ancho Nominal de la Sección 175mm.
- Relación Nominal de Aspecto 70%.
- Estructura – Radial.
- Diámetro de la llanta Nominal – 12".
- Posee una Capacidad de Carga de 475 Kg. (Índice 82).
- Pertenece a la categoría de velocidad H (210Km/h).

2.1.1.3.1 Calculo del Radio Total de la Rueda:

Teniendo en cuenta la Relación Nominal:

$$Relacion\ Nominal = 100\ \frac{hn}{bn}$$

$$70\% = 100\ \frac{hn}{bn}$$

$$hn = 0.7\ bn$$

Ahora con Ancho nominal ***hn***.

$$hn = 0.7 * (175mm)$$

$$hn = 122.5mm$$

Entonces el Radio de la rueda es formado el ancho nominal más el diámetro del rin dividido para dos.

$$r = \left(\left(\frac{R}{2}\right) * 25.4mm\right) + hn$$

$$r = \left(\left(\frac{12"}{2}\right) * 25.4mm\right) + 122.5mm$$

$$r = 274.9\ mm$$

A continuación calculamos la altura del centro de gravedad (**ac**), con la siguiente formula.

$$ac = \frac{(Cambio\ de\ Peso) * LB * Cos\alpha}{WT * Sen\alpha}$$

- ***Cambio de peso:*** es la diferencia de pesos del eje con el eje posterior cuando se levanta la parte posterior del vehículo (puede ser un 80% del radio del neumático).
- ***LB:*** distancia entre ejes (Batalla).
- ***α:*** Angulo formado al levantar la parte posterior del vehículo.
- ***WT:*** Peso total del vehículo.

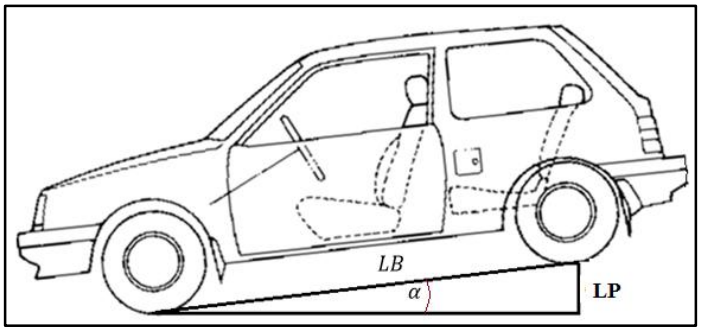


Figura 2.7: Descripción al Levantar la Parte Posterior del Vehículo.
Fuente: Autores.

Longitud posterior de levantamiento (LP), es el 80% del radio del neumático. Tenemos.

$$LP = 0.8 * r$$

$$LP = 0.8 * 274.5\ mm$$

$$LP = 219.92\ mm$$

Con los valores de LB = 2245 mm y LP = 219.92 mm, encontramos el ángulo de inclinación del vehículo al levantar la parte posterior.

$$sen\alpha = \frac{LP}{LB}$$

$$\alpha = Sen^{-1}\frac{LP}{LB}$$

$$\alpha = Sen^{-1}\frac{219.92}{2245}$$

$$\alpha = 5.62^{\circ}$$

Ahora para encontrar el cambio de peso, levantamos la parte posterior del vehículo, la distancia de LP = 219.92mm y procedemos a medir el nuevo peso en la parte delantera del vehículo.

Figura 2.8: Peso al Levantar la Parte Posterior del Vehículo.
Fuente: Autores.

El peso medido es de WF1 =420 kg de la parte delantera, por consiguiente el cambio de peso es el nuevo peso levantado la parte posterior menos el peso delantero cuando el vehículo está en reposo.

$$Cambio\ de\ Peso = WF1 - WF$$

$$Cambio\ de\ Peso = 420kg - 410kg$$

$$Cambio\ de\ Peso = 10kg$$

Ahora reemplazando en la ecuación de la altura calculada del CG.

$$ac = \frac{(Cambio\ de\ Peso) * LB * Cos\alpha}{WT * Sen\alpha}$$

$$ac = \frac{10kg * 2245mm * Cos(5.62^{\circ})}{680kg * Sen(5.62^{\circ})}$$

$$ac = 335.50mm$$

Para encontrar la altura del centro de gravedad (***h***), sumamos altura calculada del centro de gravedad sumada el radio del neumático.

$$h = Rn + ac$$

$$h = 274.9mm + 335.5mm$$

$$\boldsymbol{h = 610.4\ mm}$$

Ya encontradas las distancias del centro de gravedad, se establece el coeficiente de adherencia entre neumático y superficie de rodadura [14] (μ_{max}) para asfalto seco de 0.8, consideramos un peralte de 0˚ ya que solo será una demostración teórica.

Para el análisis de velocidades de derrape y vuelco se analizará los radios de curva en la Autopista Cuenca-Azogues, donde hay mayor incidentes de transito registrados según datos de ECU-911 Cuenca.

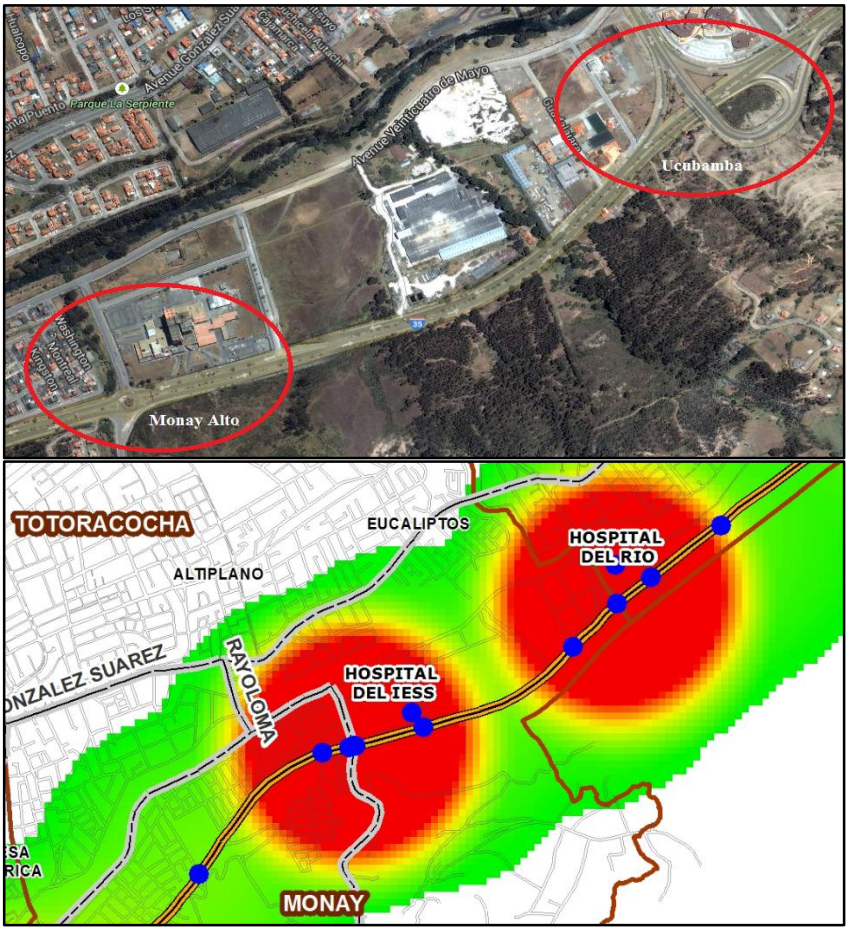


Figura 2.9: Incidentes de Transito Registrados
Fuente: ECU 911 Cuenca.

Los radios serán encontrados a través del casco urbano de la ciudad de Cuenca como se ve en la Figura 2.10 Proporcionado por la empresa municipal ETAPA, donde se da un levantamiento de vías a medidas reales.

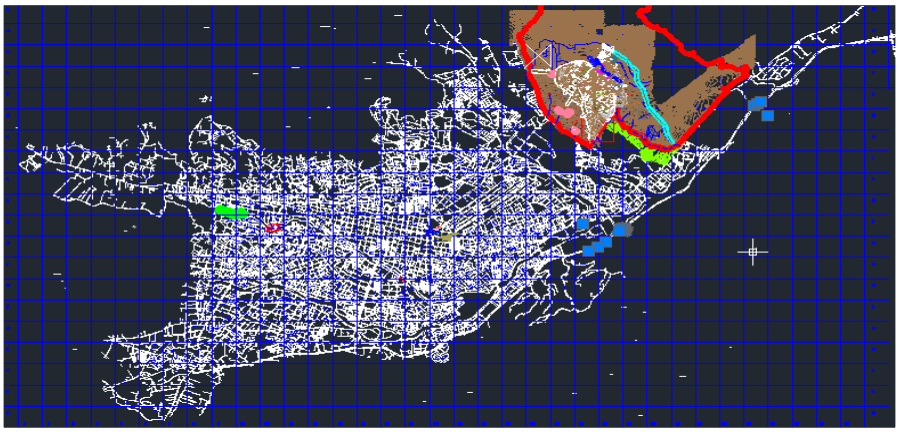


Figura 2.10: Casco Urbano de la Ciudad de Cuenca.
Fuente: Empresa Municipal Etapa.

En la siguiente Figura 2.11 muestra el radio real de dos curvas de los dos sectores de mayor incidentes de transito registrados.

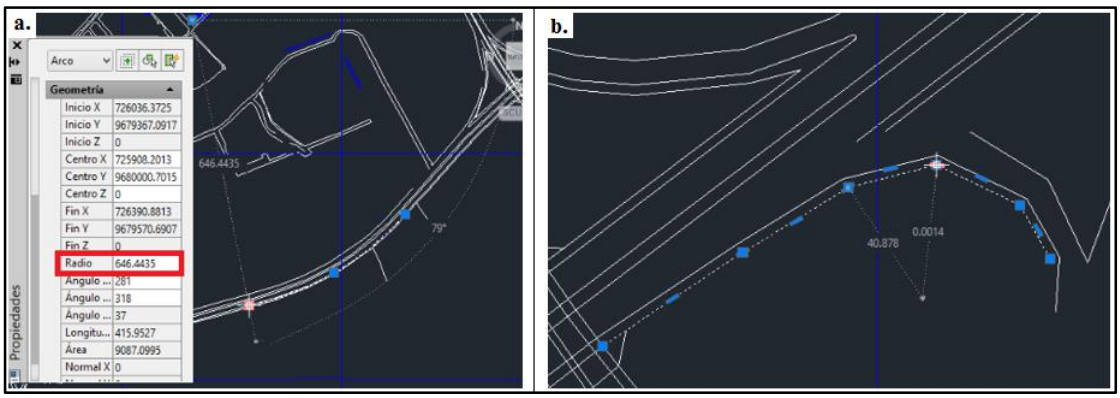


Figura 2.11: a. Monay Alto, b. Intercambiador de Ucubamba.
Fuente: Empresa Municipal Etapa.

Con los radios encontrados Ra = 646.44m, Rb = 40m, y coeficiente de adherencia con asfalto seco es 0.8, con gravedad de 9.81m/s^2, el ancho del vehículo B = 1.54m y con una altura del centro de gravedad de h = 0.61m, remplazamos en las siguientes formulas.

$$\boldsymbol{Velocidad\ de\ Derrape} = V'_{ld} = \sqrt{gR\mu_{ymax}}$$

$$\boldsymbol{Velocidad\ de\ Vuelco} = \boldsymbol{V'_{lv}} = \sqrt{\boldsymbol{gR\frac{B}{2h}}}$$

Tabla 2-3: Velocidades de Derrape y Vuelco.
Fuente: Autores.

	V_{ld} (Km/h)		V_{lv} (Km/h)	
	Ra = 646.44m	Rb = 40m	Ra = 646.44m	Rb = 40m
μ_y = 0.8	256.46	63.78	322.14	80.12

Al observar las velocidades de derrape y vuelco, con radios diferentes donde es una curva abierta y una cerrada que pertenecen a la carretera y la otra aun intercambiador respectivamente, podemos ver que las velocidades son mayores a un radio mayor de curva pero en situación real de manejo no se llega a dichas velocidades, en cambio las velocidades con el radio de curva del intercambiador, son más a pegadas a la realidad ya que son velocidades que por lo común se dan en situaciones de manejo real.

Comparando la velocidad de derrape y velocidad de vuelco con una radio de curva de 40m, y según el análisis de las velocidades, vemos que el vehículo tenderá a derrapar antes de volcar. Esto coincide con la teoría planteada anteriormente en el caso de los turismos y de vehículos industriales circulando sobre calzadas con adherencia no muy elevada.

2.2 DISTANCIA DE FRENADO.

La distancia de frenado es el espacio que recorre el vehículo desde cuando se acciona el pedal de freno hasta que se detenga completamente. Este depende de varios factores:

- Velocidad a la que el vehículo circula.
- Estado de los neumáticos.
- Carga del vehículo.
- Estado del pavimento.
- Condiciones meteorológicas.

La distancia de frenado considera para el cálculo asfalto seco, el vehículo en excelente estado. El factor de masas rotativas (γ_f = 1.05) y el conjunto de fuerzas retardadoras del movimiento, puede establecerse según la Figura 2.12.

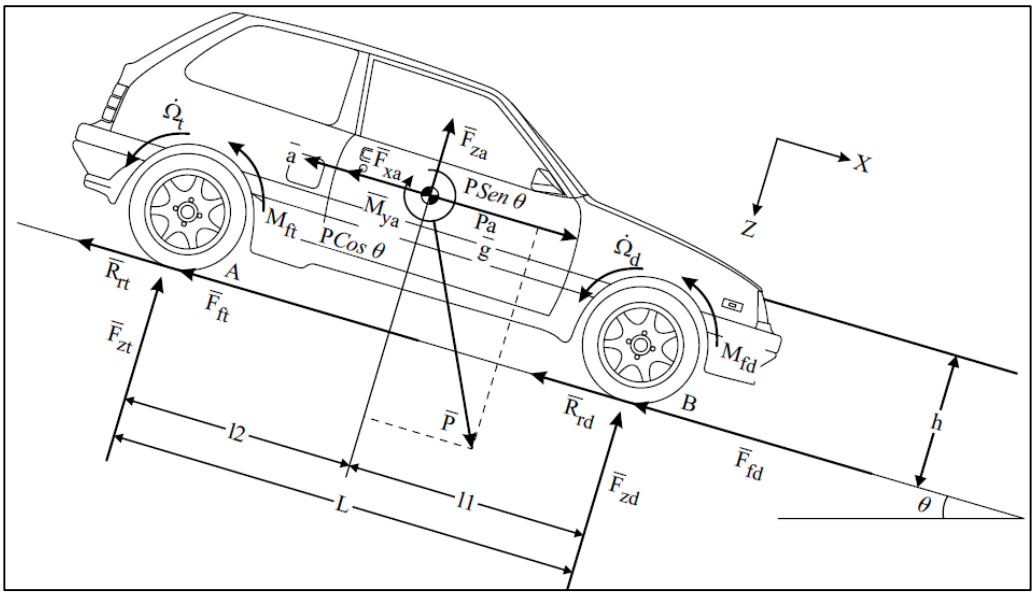


Figura 2.12: Modelo de Cuerpo Libre de un Vehículo de Dos Ejes Para el Estudio del Frenado.
Fuente: Autores.

De acuerdo con la fórmula de la desaceleración y el diagrama de cuerpo libre del vehículo tenemos.

$$a = -\frac{F_f + \sum R}{\gamma_f\, m}$$

$$a = -\frac{F_f + R_g + R_r + R_a}{\gamma_f\, m}$$

Donde cada componente de la ecuación es:

- F_f = Fuerza de frenado en los neumáticos.
- R_g = Resistencia Gravitatoria.
- R_r = Resistencia a la Rodadura en los neumáticos.
- R_a = Resistencia Aerodinámica del vehículo.
- γ_f = Factor de masas rotativas del vehículo.
- m = Masa del vehículo donde es Peso sobre la gravedad (P/g).

Se reemplazará cada uno de los factores según el diagrama de cuerpo libre.

$$a = -\frac{F_f + Psen\theta + Pf_r + CV^2}{\gamma_f\ P/g}$$

En donde se considera θ > 0 en ascensos en donde la resistencia aerodinámica es C, tenemos:

$$C = \frac{1}{2}.\rho.C_x.A_f$$

Donde:

- ρ = Densidad del Aire.
- C_x = Coeficiente Aerodinámico.
- A_f = Área Frontal del Vehículo.
- P = Peso del Vehículo.
- V = Velocidad de Avance del Vehículo.
- f_r = Coeficiente de Resistencia a la Rodadura.

Po otra parte, si ***S*** es la distancia recorrida del vehículo:

$$adS = VdV$$

A continuación sustituyendo el valor de desaceleración, donde se obtiene la distancia para frenar entre una velocidad inicial (V_1) y otra de valor final (V_2).

$$S_{V_1-V_2} = \int_{V_1}^{V_2} \frac{VdV}{a}$$

$$S_{V_1-V_2} = -\frac{P\gamma_f}{g}\int_{V_1}^{V_2} \frac{VdV}{F_f + Psen\theta + Pf_r + CV^2}$$

Suponiendo la fuerza de Frenado (F_f) y coeficiente de resistencia a la rodadura (f_r), independientes de la velocidad.

$$S_{V_1-V_2} = \frac{P\gamma_f}{2Cg} Ln \frac{F_f + Psen\theta + Pf_r + CV_1^2}{F_f + Psen\theta + Pf_r + CV_2^2}$$

La distancia hasta detener el vehículo ($V_2 = 0$) y sustituyendo F_f por $\eta_f \mu P$.

$$S_P = \frac{P\gamma_f}{2Cg} Ln\left[1 + \frac{CV_1^2}{\eta_f \mu P + Psen\theta + Pf_r}\right]$$

2.2.1 TIEMPO DE REACCIÓN DEL CONDUCTOR.

Ya encontrada la distancia de frenado, el proceso de frenado interviene reacciones del conductor y del sistema de frenos. Desde que ocurre la circunstancia imprevista, que obliga a frenar, hasta que el conductor acciona el pedal, se llama tiempo de reacción del conductor [14] (t_{rc}), cuyo valor varía entre 0.5 y 2s. Para cálculos posteriores se utilizara el tiempo de reacción del conductor alrededor de un 1s.

2.2.2 TIEMPO DE REACCIÓN DEL SISTEMA.

El tiempo de reacción del sistema (t_{rs}), este comienza cuando se acciona el pedal de freno hasta que la fuerza requerida llegue a las diferentes ruedas para el proceso de frenado, este valor es de 0.3s.

En consecuencia la distancia de parada va desde que reacciona el conductor y acciona el pedal, más el tiempo de reacción del sistema, la fórmula de distancia de frenado se modifica con estos dos tiempos.

$$S_{PT} = S_P + V_1(t_{rc} + t_{rs})$$

Remplazando S_P tenemos:

$$S_{PT} = \frac{P\gamma_f}{2Cg} Ln\left[1 + \frac{CV_1^2}{\eta_f \mu P + Psen\theta + Pf_r}\right] + V_1(t_{rc} + t_{rs})$$

La fórmula encontrada será calculada posteriormente para un vehículo Suzuki Forsa 1, en la cual serán necesarios todos los datos para la posterior programación y calibración del algoritmo y comprobar la efectividad del mismo. Esto nos servirá para el análisis y comportamiento del vehículo en situaciones de conducción real.

2.2.3 CALCULO DE LA DISTANCIA DE FRENADO.

Con la ecuación anterior se procede a encontrar cada uno de los datos para que la fórmula se transforme en función de la velocidad en donde a mayor velocidad aumentará la distancia de frenado, teniendo en cuenta que C tiene la siguiente formula.

$$C = \frac{1}{2}.\rho.C_x.A_f$$

La densidad del aire ($\rho = 1.225\, kg/m^3$) a condiciones normales de presión y temperatura ($1.074\ Pa\ y\ 25^{\circ}C$).

Utilizando el Software ANSYS® para el modelado de la carrocería del Suzuki Forsa 1, este nos proporciona el área frontal del vehículo (A_f) y el coeficiente de resistencia aerodinámico (C_x), debemos de tener en cuenta que los datos obtenidos son de un modelado a medidas reales como se puede observar en la Figura 2.13.

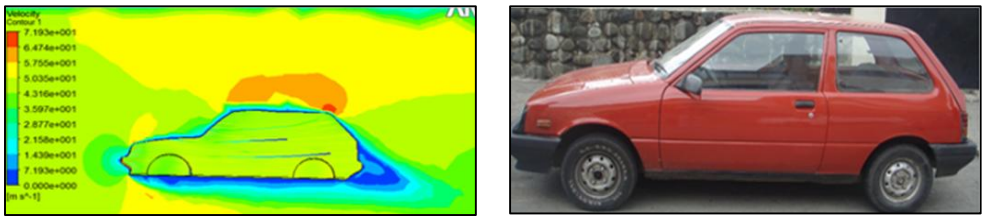


Figura 2.13: Modelado de un Suzuki Forsa 1.
Fuente: Autores.

Para el análisis de la carrocería los datos son obtenidos solo con la mitad de la carrocería, entonces estos datos se pasaran a medidas reales para los cálculos posteriores con carrocería completa.

	A	B	C	D
1	-9.826e+02 [N]	fuerza en x	Area en x (m^2)	0.7795376
2	1.139e+03 [N]	fuerza en z	Area en z (m^2)	3.432395
3	8.473e+02 [N]	fuerza en y	Area en y (m^2)	2.573051

Figura 2.14: Datos de Software ANSYS®.
Fuente: Autores.

Utilizando los datos anteriores, tenemos que $A_f = 0.77953$ y la fuerza aplicada en la parte frontal del automóvil $f_{xa} = 982.6\, N$ donde no tomamos en cuenta el signo negativo por la disposición de ejes del Software.

Se determinó los coeficientes de resistencia aerodinámica Cx aplicando las fórmula de resistencia aerodinámica al avance (1), teniendo en cuenta que en la simulación se tomó el eje x el eje de circulación del auto, entonces tenemos:

$$f_{xa} = \frac{1}{2} * Cx * \rho * Af * V^2 \quad (1)$$

$$C_x = \frac{2 * f_{xa}}{\rho * Af * V^2}$$

Teniendo en cuenta la f_{xa} es negativa por que está en contra a la circulación del automóvil para los siguientes cálculos con velocidad de 50 m/s que se efectuó la simulación, y con densidad del aire de 1.225 kg/m^3. Calculamos los coeficientes de resistencia aerodinámica al avance (C_x).

$$C_x = 0.8231$$

A continuación reemplazamos cada uno de los datos para encontrar $\boldsymbol{C}$, donde el área frontal será la real del vehículo por lo cual el valor anterior será el doble del proporcionado por el software.

$$C = \frac{1}{2}.\rho.C_x.A_f$$

$$C = \frac{1}{2}.\left(1.225\frac{kg}{m^3}\right) * (0.8231) * (1.6m^2)$$

$$C = 0.806\frac{kg}{m^2}$$

Para encontrar la eficacia de frenado (η_f), se obtuvo el dato mediante la revisión técnica vehicular del Suzuki Forsa 1. Como podemos ver en la siguiente Figura 2.15.

HIDROCARBUROS NO COMBUSTIONADOS (HC) 2500 RPM	ppm	300.00	0.00<=X<=750.00	OK
DESEQUILIBRIO DE FRENADO EN 1° EJE	%	9.00	0.00<=X<=16.00	OK
DESEQUILIBRIO DE FRENADO EN 2° EJE	%	13.00	0.00<=X<=15.00	OK
EFICACIA DE FRENADO	%	80.00	60.00<=X<=100.00	OK
EFICACIA FRENO DE ESTACIONAMIENTO	%	29.00	20.00<=X<=100.00	OK

Figura 2.15: Datos de Revisión Técnica Vehicular.
Fuente: Autores.

El valor medio en la revisión es de $\eta_f = 0.8$, que es la eficacia de frenado, que está dentro del rango de 60% y 100%.

El coeficiente de resistencia a la rodadura (f_r), se obtendrá de la Tabla 2.4 donde se ofrecen algunos valores de este coeficiente, en función del tipo de neumático, según el vehículo que se destina y la naturaleza del suelo o calzada.

Tabla 2-4: Valores Aproximados del Coeficiente de Resistencia a la Rodadura (f_r) de los Neumáticos.
Fuente: Autores.

Tipo de Vehículo	***Superficie***		
	Hormigón o Asfalto	**Dureza Media**	**Arena**
Turismo	0.015	0.08	0.30
Camiones	0.012	0.06	0.25
Tractores	0.02	0.04	0.20

Para el cálculo utilizamos $f_r = 0.015$, ya que los demás datos como coeficiente de adherencia se considera que le vehículo se traslada en una superficie de asfalto, ya que este se realizan pruebas en autopista.

Para encontrar la distancia de frenado se consideran todos los datos ya encontrados, en donde tenemos la formula y sus respectivos términos, considerando la velocidad variable, por lo cual la distancia de frenado es proporcional a la velocidad del vehículo.

$$S_{PT} = \frac{P\gamma_f}{2Cg} Ln\left[1 + \frac{CV_1^2}{\eta_f \mu P + Psen\theta + Pf_r}\right] + V_1(t_{rc} + t_{rs})$$

Tabla 2-5: Valores para Encontrar la Distancia de Frenado.
Fuente: Autores.

Términos	**Valor**
P	$680kg$
$\boldsymbol{\gamma_f}$	1.05
C	$0.806\ kg/m^2$
g	$1.98\ m/s^2$
$\boldsymbol{\eta_f}$	0.8
μ	0.8
θ	0
$\mathbf{f_r}$	0.015
$\mathbf{t_{rc}}$	1s
$\mathbf{t_{rs}}$	0.3s

A continuación se analizarán a diferentes valores de velocidad V_1 (m/s) para determinar las distancia de frenado con los datos ya remplazados en la fórmula como vemos a continuación sin olvidar las unidades de cada valor.

$$S_{PT} = \frac{680kg * 1.05}{2 * (0.806\,kg/m^2) * (9.81\ m/s^2)} Ln\left[1 + \frac{(0.806\,kg/m^2) * V_1^2}{(0.64 * 680kg) + (680kg * 0.015)}\right] + V_1(1s + 0.3s)$$

$$S_{PT} = (45.15ms^2) * Ln\left[1 + \frac{(0.806\,kg/m^2) * V_1^2}{445.4kg}\right] + V_1(1.3s)$$

Con la siguiente formula encontramos las diferentes distancias de frenado a diferentes velocidades.

Tabla 2-6: Distancias de Frenado.
Fuente: Autores.

Velocidad (V_1)		**Distancia (S_{PT})**
km/h	***m/s***	***m***
30	8.33	16.16
50	13.89	31.57
70	19.44	48.8
80	22.22	57.7
90	25	66.66
100	27.78	75.57
120	33.33	93.08

La tabla anterior muestra las diferentes distancias de frenado a diferentes velocidades del automóvil, esos son expresados en una función cuadrática como vemos en la Figura 2.16.

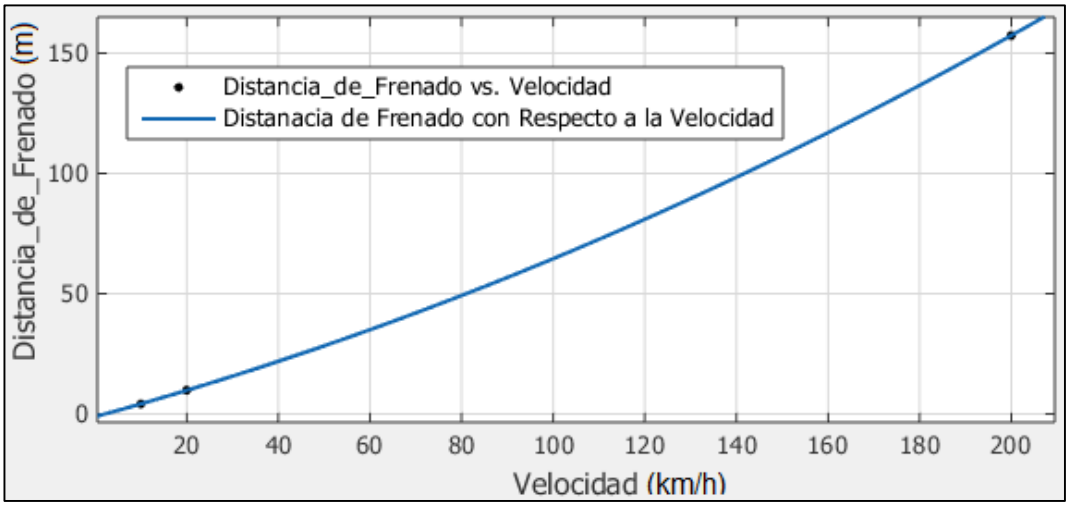


Figura 2.16: Distancia de Frenado con Respecto a la Velocidad.
Fuente: Autores.

CAPITULO 3

ADQUISICION Y PROCESAMIENTO DE LAS SEÑALES DE VIDEO.

3. ADQUISICIÓN Y PROCESAMIENTO DE LAS SEÑALES DE VIDEO.

La adquisición y procesamiento de señales de video es muy importante para el desarrollo del algoritmo de adaptación de velocidad ya que tenemos en cuenta factores como: rutas para la comprobación del dispositivo, donde hay mayor cantidad de vehículos, circulación peligrosa y tipo de vehículos automotores, entre otras.

También se mencionará herramientas necesarias para la preparación de las imágenes y de los objetos a seleccionar, y todos los aspectos necesarios para obtener señales de video que contengan la mayor información.

3.3 FUNDAMENTOS DE ADQUISICIÓN DE IMAGEN.

La adquisición trata de captar una imagen digital del medio o campo del que se pretende desarrollar el control o el trabajo de dicha imagen, es la primera etapa del sistema de visión artificial, y depende de los siguientes factores.

- Iluminación.
- Cámara.

3.1.1 ILUMINACIÓN.

El sistema de iluminación es muy importante para trabajar con el sistema de visión artificial, ya que nos presenta las características de la imagen. En esta investigación la iluminación ambiente en que se desarrolla el trabajo, pero se debe de utilizar algoritmos para corregir la imagen, o también se debe de utilizar una iluminación adecuada, con ello se va regulando el contraste en las áreas que son de mayor interés del sistema de visión artificial.[15]

3.3.2 CÁMARA.

Es la que se encarga de transformar las señales luminosas de la escena, en señales lógicas. Está compuesta por:

3.3.2.1 El Sensor.

Captura las propiedades de la escena en forma de señales luminosas y la trasforma en señales analógicas [15].

3.3.2.2 Óptica.

"Proyecta los elementos adecuados de la escena ajustando una distancia focal adecuada", basado de [15], esta puede ser de lente fijo ya que consta de una distancia focal fija que es la distancia del extremo del objetivo a la película y la de lente móvil que tiene una distancia focal manipulable con su máximo y mínimo de distancia focal.

Para establecer el diseño y la construcción del sistema de visión artificial, teniendo en cuenta la obtención de la imagen de las líneas, bordes de la vía y el reconocimiento de vehículos para la detección de la proximidad con otros vehículos. Utilizamos dos sensores, para el lado izquierdo y derecho. Estos dos sensores se escogieron de acuerdo a su bajo costo, su fácil montaje y manipulación. De acuerdo a ello tenemos una cámara, "Omega Webcam C11" colocadas en el automóvil como se muestra en la Figura 3.1 ya que presenta una fácil conexión con el entorno de Matlab® y su librería de Simulink®.

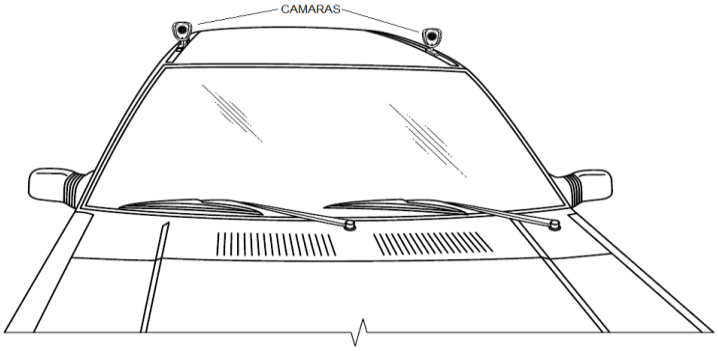


Figura 3.1: Cámaras Instaladas en el Automóvil.
Fuente: Autores.

La Webcam, Omega C11 en la Figura 3.2 tiene definición de 640 * 480 pixeles, veremos posteriormente la utilización dependiendo del funcionamiento.

Figura 3.2: Webcam Omega C11.
Fuente: Autores.

3.4 ADQUISICIÓN Y PROCESAMIENTO DE SEÑAL DE VIDEO PARA LA DETECCIÓN DE PROXIMIDAD ENTRE VEHÍCULOS.

3.4.1 CONEXIÓN DE CÁMARA A MATLAB®.

Para obtener la señal de video utilizaremos el software de Matlab® el cual por medio Image Processing Toolbox™ (Herramientas de Procesamiento de Imágenes) permite trabajar con imágenes y videos.

La conexión de la cámara se la realiza por medio de los puertos USB de la portátil y podemos usar la aplicación ¨Image Acquisition Toolbox¨ (Herramienta de Adquisición de Imágenes), en la Figura 3.3 se ve dónde encontrar la aplicación mencionada en la pestaña de aplicaciones del software, que permite ver las propiedades de las cámaras con las que podemos trabajar en Matlab®.

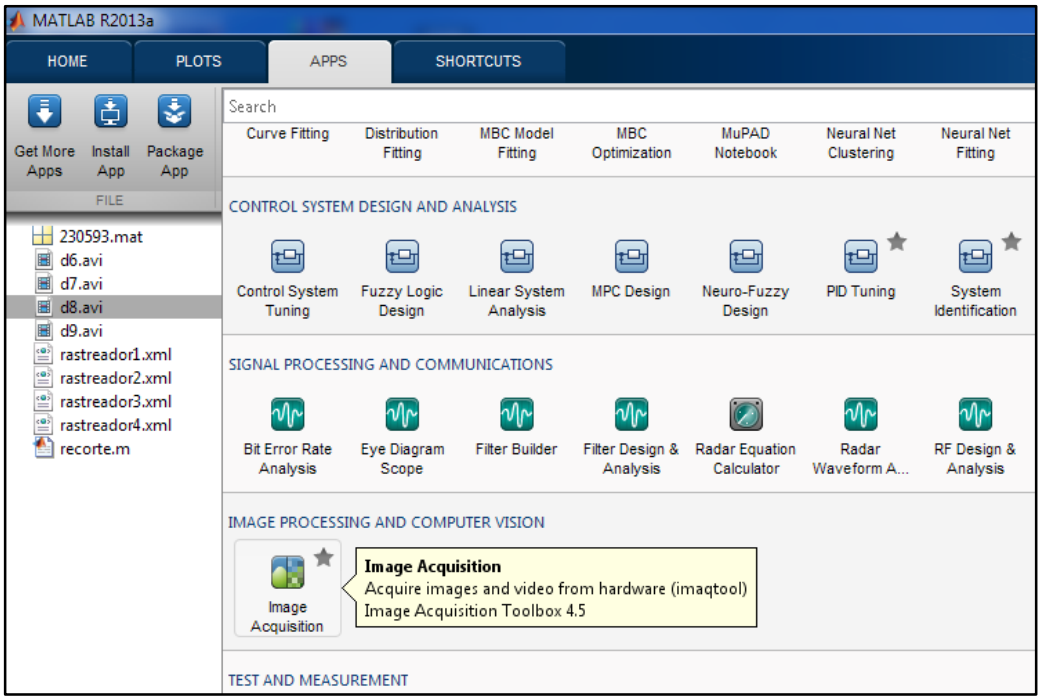


Figura 3.3: Localización de Image Acquisition.
Fuente: Autores.

La ventana abierta de Image Acquisition Figura 3.4 proporciona varias herramientas como se observa en el cuadro (1) que es la ventana *Hardware Browser* donde se observa dos cámaras conectadas, la una es ***Vimicro USB Camera (Altair) (winvideo-***

1), esta cámara es con la que se trabajara posteriormente, la otra es ***TOSHIBA Web Camera-HD (winvideo-2)*** es la cámara del computador.

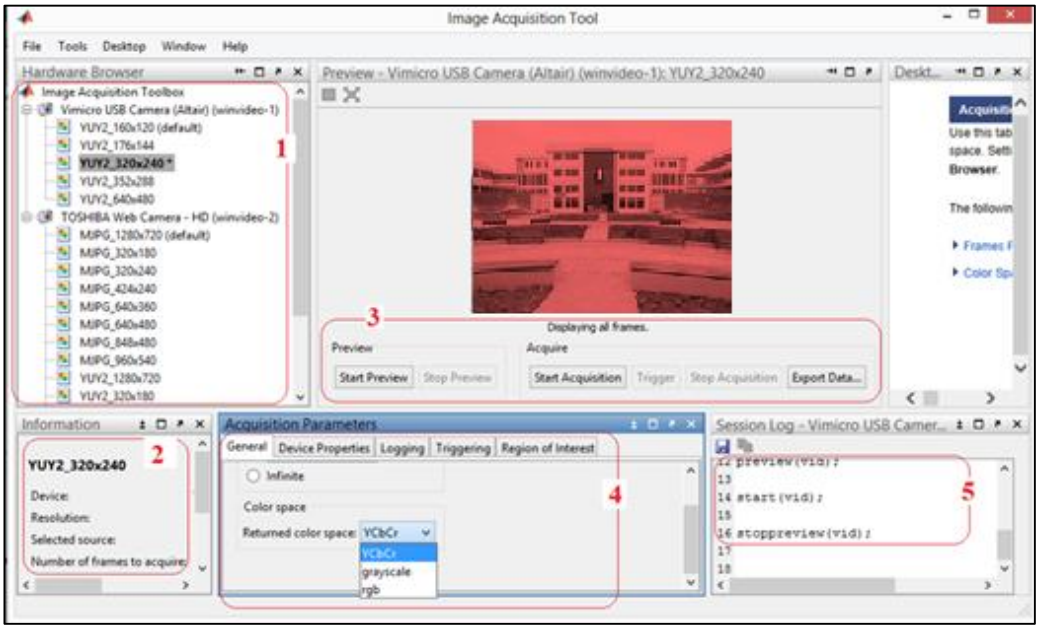


Figura 3.4: Ventana Image Acquisition.
Fuente: Autores.

La cámara ***Vimicro USB Camera*** utiliza el nombre del adaptador *winvideo-1* en el cuadro (1) en la parte de abajo del se despliega una lista que indica el tamaño de la imagen en pixeles (ancho x alto) (160x120, 176x144, 320x240, 352x288, 640x480) en conjunto con el tipo de dato YUY2 que posee cada cámara, en el cuadro (2) tenemos la ventana de *Information* que muestra los detalles y la información de la cámara seleccionada en la ventana *Hardware Browser*, mientras que en el cuadro (3) se encuentran herramientas que permite observar o capturar vistas previas de lo que está captando la cámara, el cuadro (4), ventana *Acquisition Parameters* permite utilizar *Returned color space* que permite determinar el tipo de dato con el que va a retornar el video, la cámara va a captar videos de tipo de dato YUY2 , este tipo de dato es el único que la cámara dispone como se observar en el cuadro (1) de la Figura 3.4 para poder cambiar esta opción existen dos opciones el tipo de video en RGB o grayscale (escala de grises o intensidad). Para la programación vamos a trabajar con un tipo de video en RGB. Por ultimo en el cuadro (5) se observa la ventana *Seccion Long* el cual permite observar los comandos en *Matlab*® *Editor* de las modificaciones que realizamos en la ventana de *Image Acquisition*.

Con los comandos dados por la ventana *Seccion Long* la programación en *Matlab®* *Editor* para la adquisición del video y detección de vehículos es la siguiente, como podemos observan en la Figura 3.5 y su explicación se la va a realizar paso a paso a continuación.

```
4 -     vid = videoinput('winvideo',1,'YUY2_320x240');
5
6 -     vid.FrameGrabInterval = 10;
7
8 -     set(vid,'TriggerRepeat',inf);
9
10 -    vid.ReturnedColorspace = 'rgb';
```

Figura 3.5: Matlab Editor programa para llamar a la cámara.
Fuente: Autores.

3.4.1.1 Selección de Cámara Retorno del Color del Video.

En la primera línea de programación utilizamos el comando ***videoinput*** el mismo permite crear un objeto en la variable *vid* para la entrada del video, esta permite la conexión de la cámara de video y Matlab®.

```
4 -     vid = videoinput('winvideo', 1, 'YUY2_320x240');
```

Dentro de los paréntesis colocamos el nombre del adaptador para comunicarse con la cámara, en este caso utilizamos 'winvideo' seguido por el número 1 que va a permitir utilizar la cámara ***Vimicro USB Camera (Altair) (winvideo-1)*** y utilizamos un tamaño de 320x240 que está en tipo de video YUY2.

```
6 -     vid.FrameGrabInterval = 10;
7
8 -     set(vid,'TriggerRepeat',inf);
```

La línea 6 de programación el ***FrameGrabInterval*** permite especificar con qué frecuencia adquiere un fotograma (imagen) de la escena captada por la cámara, podemos utilizar valores entre (3-10), ya que con valores mayores el video tiende a no mostrar algunos fotogramas generando un efecto de retardo en las acciones captadas por la cámara y las mostradas en la pantalla. Entonces cada 10 fotogramas captados se mostrara el fotograma 10.

El comando de la línea 8 se utiliza para especificar el tiempo que deseamos que la cámara siga captando el entorno, se realiza modificando la tercera opción que se

encuentra entre los paréntesis, si colocamos un valor de ***0*** la cámara captará una foto, otra opción es colocar un tiempo determinado si colocamos un valor de 10 la cámara captará la escena durante 10 segundos y se detendrá la última opción es colocar ***inf*** que permite captar un video durante un tiempo indefinido hasta que activemos un comando de stop o se dé un problema dentro de la programación.

```
10 -    vid.ReturnedColorspace = 'rgb';
```

En la línea 10 modificamos el tipo de color de video con el que se trabajara para ello ocupamos el comando **ReturnedColorspace**, cambiando así el formato de color del video de YUY2 a RGB, ya que la programación trabaja con tipo de color de video RGB y la cámara proporciona videos de tipo de color YUY2.

3.4.1.2 Pantalla de Salida del Video.

```
18 -    start(vid);
19
20 -    while(vid.FramesAcquired<=2000)
21
22 -        data = getdata(vid,1);
23
24 -        videocor=imcrop(data, [70 0 200 250]);
25
26 -        contraste=imadjust(videocor, [.2 .3 0; .7 .7 1]);
27
28 -        imshow(contraste);
29 -    end
```

Figura 3.6: Matlab Editor Programa Mostrar a Pantalla y Ajuste del Video.
Fuente: Autores.

Para observar el video podemos usar el comando de **start** el mismo permite que la cámara comience a funcionar.

```
18 -    start(vid);
```

Para que el video pueda ser visualizado en tiempo real, se utiliza un operador condicional **while,** este permite repetir varias veces una acción mientras la condición sea verdadera, la condición para que se repita, va a ser que si el número de cuadros de imágenes sea menor o igual a 2000 esta se siga dando. Nota al final debe de

terminar la condición **while** con **end**. En la línea 22 el **getdata** permite extraer datos de entrada de la componente vid.

```
20 -    while(vid.FramesAcquired<=2000)
21      
22 -        data = getdata(vid,1);
```

El video de salida va a tener un tamaño de 320x240 como observamos en la Figura 3.7 observamos que la imagen es muy grande y existen partes donde el análisis es innecesario aumentando el tiempo computacional, es necesario recortar la imagen original solo a la región de interés, para ello utilizamos el comando **imcrop** para delimitar la región de interés entre corchetes [x y ancho alto] donde x-y son las coordenadas de la parte superior izquierda de la imagen respectivamente como muestra la Figura 3.8 luego se delimita el ancho y alto de la región de interés.

```
24 -        videocor=imcrop(data, [70 0 200 250]);
```

Figura 3.7: A la Izquierda Tenemos la ROI de la Imagen.
Fuente: Autores.

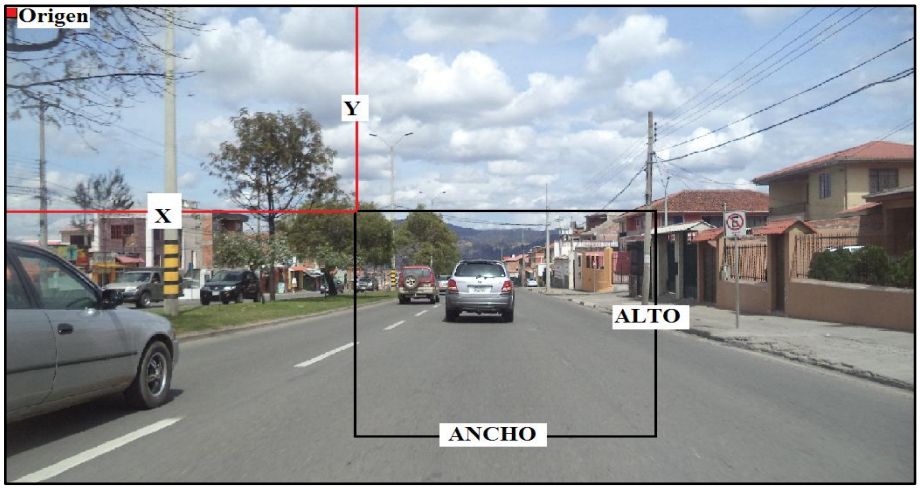


Figura 3.8: Recorte ROI de la Imagen.
Fuente: Autores.

3.4.2 CALIBRACIÓN DE LA ILUMINACIÓN.

La iluminación es ajustada con el comando **imadjust,** es controlada ya que la cámara al captar el transcurso del recorrido del vehículo vamos a tener diferente luminosidad.

```
26 -     contraste=imadjust(videocor, [.2 .3 0; .7 .7 1]);
```

Para el ajuste de iluminación de la imagen en RGB, debemos de especificar dentro de corchetes los valores en: R 0.2 a 0.7, G 0.3 a 0.7 y B 0 a 1.

Para poder observar el video obtenido utilizamos el comando **imshow** y terminamos la programación con **end** para terminar la condición dada en **while**.

```
28 -     imshow(contraste);
29 - end
```

3.4.3 ADQUISICIÓN DE MUESTRAS PARA ENTRENAR AL DETECTOR DE VEHÍCULOS.

Para poder detectar vehículos debemos de crearnos un archivo .xml[1] la cual contiene las características que permite reconocer a los vehículos dentro de las escenas captada por las cámaras, para ello debemos de contar con dos grupos de imágenes, el primer grupo es conocido como imágenes positivas, las mismas van a tener la región de interés que nosotros deseamos reconocer (en este caso la parte posterior del vehículo); el otro grupo de imágenes va a ser reconocido como imágenes negativas que va a contener imágenes donde su entorno puede hacer confundir a nuestro detector generando falsas detecciones ver Figura 3.9.

[1] **.xml (eXtensible Markup Language)._** o lenguaje de marca extensiva, utilizada para almacenar datos en forma legible. Es un estándar para dar el intercambio de información estructurada entre diferentes plataformas, permite almacenar grandes cantidades de información. [http://www.hipertexto.info/documentos/lenguajes_h.htm].

a) b)

Figura 3.9: a) Imagen Positiva; b) Imagen Negativa.
Fuente: Autores.

3.4.3.1 Lugar de Adquisición de Imágenes.

La ruta elegida para la recolección de imágenes es en la Autopista Cuenca-Azogues como se observa en la Figura 3.10, debido que tienen mayor flujo vehicular y es una vía de circulación rápida, por ende los índices de accidentes por exceso de velocidad son notorios. La ruta comienza desde la Universidad Politécnica Salesiana sector el vecino, sigue por la Avenida de las Américas hasta salir a la Autopista Cuenca-Azogues por el Sector del Hospital del Rio, se conduce en la autopista hasta el intercambiador del Descanso Sector de Chaullabamba y de regreso, teniendo así la mayor información posible.

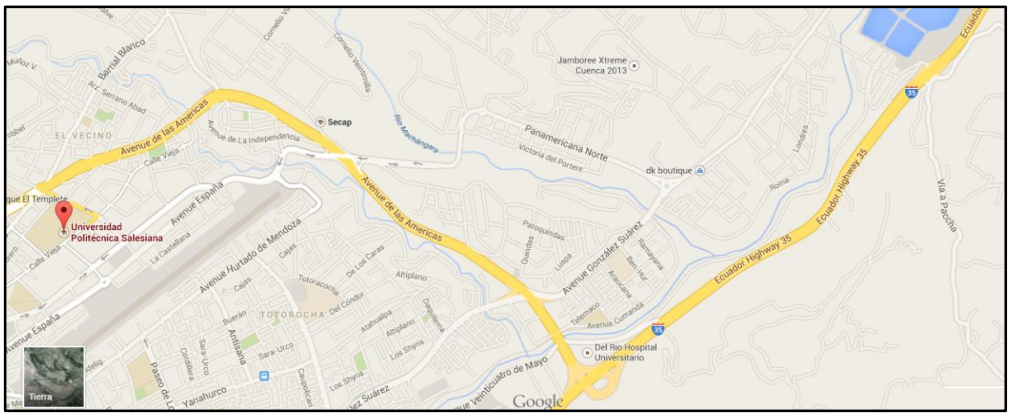


Figura 3.10: Recorrido para Recolección de Imágenes.
Fuente: Autores.

3.4.3.2 Componentes para Adquisición de las Imágenes.

Para obtener las imágenes utilizamos dos cámaras del mismo tipo, las imágenes lo obtuvimos grabando videos y luego por intermedio de la aplicación Free Video to JPG Converter como se ve en la Figura 3.11 del programa Free Studio,

transformamos a imágenes, esto permite obtener mayor cantidad de imágenes en menor tiempo.

Figura 3.11: Programas Free Video to JPG Converter.
Fuente: Autores.

La primera cámara que utilizamos es conectada a través de Simulink® como se observa en la Figura 3.12.

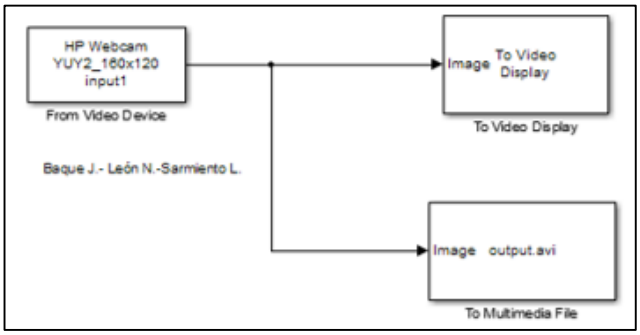


Figura 3.12: Esquema para Obtener el Video en Simulink.
Fuente: Autores.

Para la grabación de videos, la cámara fue colocada en un soporte sujetado sobre el capot del vehículo, como vemos en la Figura 3.13. Esta posición fue escogida ya que el detector debe de reconocer la parte posterior de los vehículos que circulen por delante al vehículo en el que va montado el sistema, la altura de la cámara y la posición es similar al sistema visual del conductor.

Figura 3.13: Posición de la Cámara Web en el Vehículo.
Fuente: Autores.

Realizamos una segunda grabación utilizando una cámara Sony para obtener unas imágenes más nítidas, grabando videos con la mis cámara digital, en este caso la misma no necesita un programa extra para guarda lo que graba como lo es con la Cámara Web, ya que la cámara cuenta con su propia memoria para almacenamiento de datos, la ubicación de la cámara digital no varía como se observa en la Figura 3.14.

Figura 3.14: Posición de la Cámara Digital en el Vehículo.
Fuente: Autores.

En los resultados de las imágenes con las dos cámaras, muestra una variación de nitidez de una imagen de una webcam (a) con una imagen de la cámara digital (b) como vemos en la Figura 3.15 cabe mencionar que ambas cámaras grabaron con un tamaño de 640 X 480 y tienen una resolución de 96 ppp. Podemos ver una diferencia de calidad entre fotografías esto puede darse debido a que las cámaras digitales

cuentan con sistema mecánico de captura que se denomina obturador[2] este permite obtener mejor calidad de imagen de acuerdo a la velocidad que el obturador se acciona, mientras más lenta es la obturación más clara es la fotografía y mientras más rápida sea la obturación más oscura es la fotografía. En cambio las webcam tienen un sistema digital que solo captura la fotografía.

a) b)

Figura 3.15: a) Imagen con Cámara Web; b) Imagen con Cámara Digital.
Fuente: Autores.

Una vez obtenida las Imágenes debemos de separar las imágenes positivas de las negativas y descartar las que no usaremos, para entrenar al detector posteriormente.

3.5 ADQUISICIÓN Y PROCESAMIENTO DE SEÑAL DE VIDEO PARA LA DETECCIÓN DEL TRAZADO DE LA VIA.

Para la adquisición de imagen se ocupan los siguientes bloques tanto para el lado izquierdo como para el lado derecho:

3.5.1 ADQUISICIÓN DE IMAGEN DEL TRAZADO DE LA VIA.

Para la adquisición de imagen utilizamos dos cámaras, el uso de dos cámaras nos ayuda a la percepción de profundidad y aumentar el ángulo de visión, logrando así captar el trazado de la vía. Todos los bloques descritos posteriormente trabajaran en una ventana de Simulink® como se muestra en la Figura 3.16.

[2] **Obturador._** es el dispositivo que controla el tiempo durante el que llega la luz al dispositivo fotosensible. [http://www.wordreference.com/definicion/obturador]

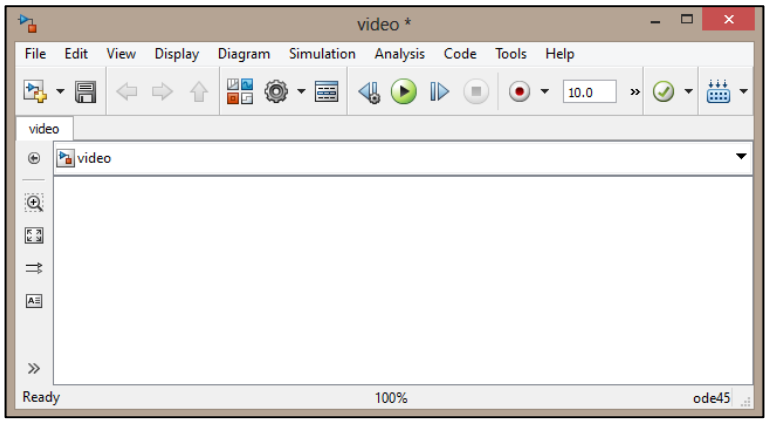


Figura 3.16: Ventana de trabajo de Simulink®.
Fuente: Simulink®.

3.5.1.1 From Video Device.

El bloque "*From video device*" que se encuentra en el Software de Matlab®, donde puede captar la imagen proveniente de la Webcam que vamos a utilizar, la librería que contiene el bloque es "*Image Acquisition Toolbox"* este se obtiene en la librería de Simulink®. Como se ve en la Figura 3.17.

Figura 3.17: From Video Device.
Fuente: Simulink®.

Para poder modificar los parámetros del bloque anterior damos doble click en el mismo bloque donde tenemos los siguientes parámetros. "*Device*" que es el dispositivo para escoger la Webcam con la que estamos trabajando Vimicro USB Camera (Altair). En "*Video Format*" nos permite escoger con el número de Pixeles que queramos trabajar en este caso tenemos uno de YUY2_160*120, esta es la resolución escogida ya que ocupa menos espacio computacional. En "*Output color space*" es donde podemos seleccionar el tipo de color en el que queramos trabajar, en nuestro caso es RGB ya que la cámara funciona solo en formato YUY2. El "*Block simple time*" es el tiempo de muestreo del bloque durante la simulación en este caso es de 1/30, todas estas opciones se ve en la Figura 3.18.

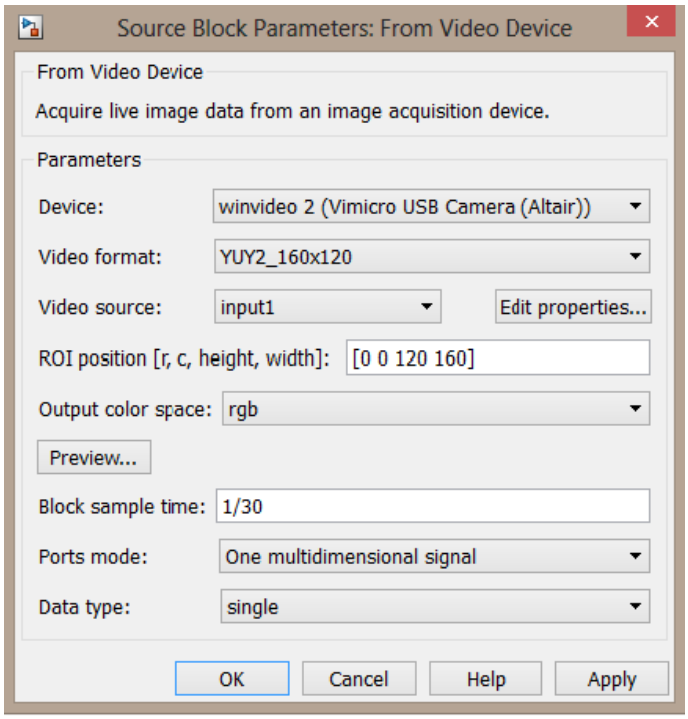


Figura 3.18: Introducción los parámetros en el From Video Device.
Fuente: Simulink®.

3.5.1.2 To Video Display.

El bloque de visualización de video de la Webcam, encontramos en la librerías de Simulink®, en ¨*Computer Vision System Toolbox*¨ y en la subcarpeta ¨*Sinks*¨. Con ella podemos visualizar la configuración de la cámara hasta obtener una adecuada visualización.

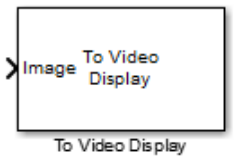


Figura 3.19: To Video Display.
Fuente: Simulink®.

3.5.1.3 To Multimedia File.

Se encuentra en la misma librería del bloque anterior. Este bloque es para guardar los videos ya que nos sirven para trabajar en las diferentes pruebas del algoritmo. Primero seleccionamos la carpeta donde la vamos a guardar, luego el tipo de formato que se quiere utilizar, en este caso utilizamos el de formato AVI, como también el formato de color RGB. En la Figura 3.20 se presenta el bloque y en la Figura 3.21 la ventana para escoger los parámetros de video.

Figura 3.20: To Multimedia File.
Fuente: Simulink®.

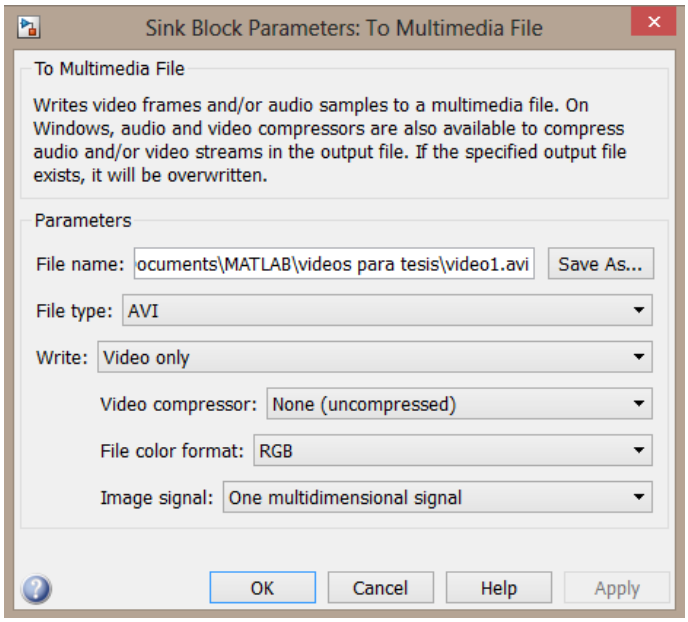


Figura 3.21: Parameters to Multimedia File.
Fuente: Simulink®.

Para la adquisición de imágenes del trazado de la vía, las cámaras están colocadas a una altura de 1.35m que es la altura del vehículo de pruebas, debemos de considerar que a mayor altura se captaran mejor el trazado de la vía mejorando así el funcionamiento del algoritmo. Para poder trabajar en el algoritmo de detección del trazado de la vía, se grabó videos de la vía rápida en diferentes tramos. Entre los videos grabados se tomó en cuenta la vía Cuenca – Azogues y la vía Cuenca – Tarqui debido a que esta última posee mejor señalización en el trazado de la vía.

Se construyó el conjunto de bloques para la adquisición de imágenes que nos muestra en la Figura 3.22. Teniendo en cuenta el tiempo de grabación, en este damos la opción *inf* para tener un tiempo ilimitado de grabación.

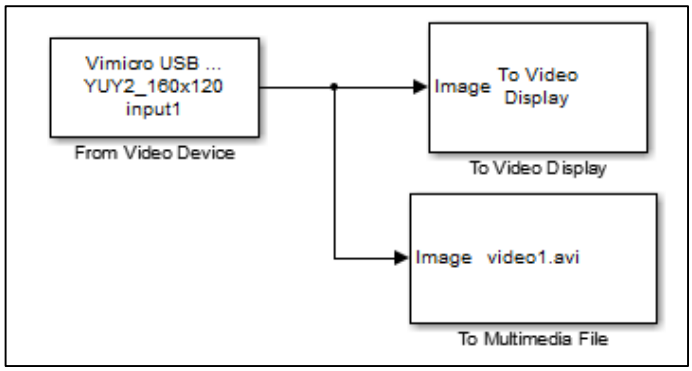


Figura 3.22: Adquisición de los Videos para la Programación.
Fuente: Simulink®.

3.5.2 PROCESAMIENTO PARA LA DETECCION DEL TRAZADO DE LA VÍA.

Dentro del procesamiento se logra obtener una transformación de la imagen que se obtuvo en la adquisición por medio de etapas como: eliminar el ruido mediante el filtrado y control de la iluminación. Obteniendo la imagen adecuada para una utilización dentro de los objetivos planteados.

3.5.2.1 Conversión a escala de grises.

Cuando adquirimos la imagen, esta se presenta en el formato RGB[3], la cual hay que convertirla en una imagen de intensidad o escala de grises. Este cambio se da para suprimir datos ya que cada pixel va tener un valor entre **0** a **255**, esto también nos ayuda para el fácil procesamiento de datos, ya que en procesos posteriores, requieren datos de pixel en intensidad. Este bloque es utilizado tanto para el video del lado izquierdo como el del lado derecho como se menciona en la adquisición de imágenes. Para obtener la conversión se lo realiza mediante el bloque "*Color Space Conversión*".

3.5.2.1.1 Color Space Conversion.

El bloque lo podemos encontrar en la librería de Simulink® en "*Computer Vision System Toolbox"* y en la subcarpeta ¨*Conversions*¨, como se ve en la Figura 3.23.

[3] RGB: Es la combinación de los colores primarios rojo (R), verde (G) y azul (B)[11].

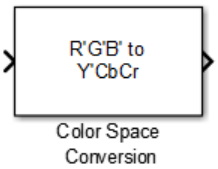


Figura 3.23: Color Space Conversion.
Fuente: Simulink®.

El bloque presenta diferentes conversiones de color, para el parámetro ¨*Conversion*¨ se escogió la de "*RGB to intensity*". En ¨*Image signal*¨ escogemos "*One multidimensional signal*". Como vemos en la Figura 3.24.

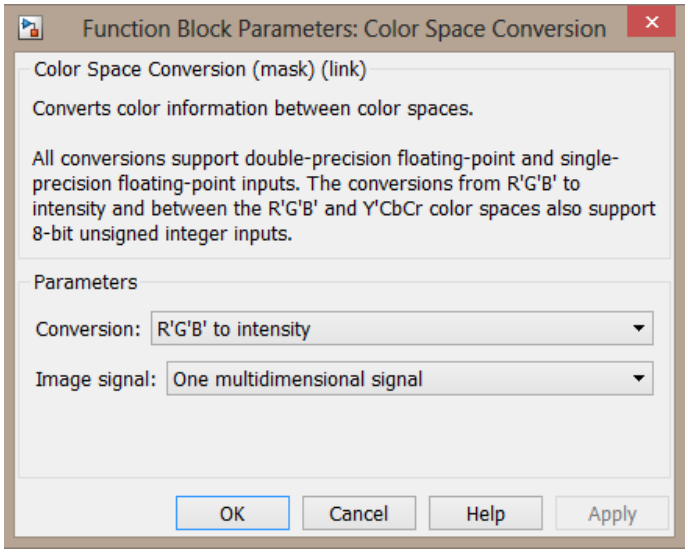


Figura 3.24: Parameters Color Space Conversion.
Fuente: Simulink®.

El modelo RGB, muestra imágenes con un valor grande en la componente de rojo o verde tengan una apariencia obscura. El efecto contrario sucede en aquellos píxeles donde el contenido del plano azul es grande, mostrado en su versión a escala de grises una apariencia más clara [11].

3.5.2.2 Recorte de la Región de Interés de la Imagen.

Recortamos la zona de interés de la imagen, en este caso recortamos la imagen obteniendo solo la imagen de la vía. El resto de la imagen es eliminada, evitando así mayores procesos para la detección del trazado de la vía. El bloque utilizado es el ¨*Submatrix*¨, que se encuentra en las librerías de Simulink®.

a) b)

Figura 3.25: a) Imagen de Intensidad, b) Recorte de Región de Interés.
Fuente: Autores.

3.5.2.2.1 Submatrix.

Este bloque nos permite recortar la imagen de entrada, tanto de sus filas como de sus columnas, para así solo trabajar con el cuadro de la imagen que es necesaria para detección del trazado de la vía. Este bloque se encuentra en la librería de Simulink® ¨DSP "S*ystem Toolbox*¨, la subcarpetas ¨*Signal Management*¨ y por último en ¨*Indexing*¨, como vemos en la Figura 3.26.

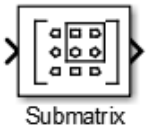
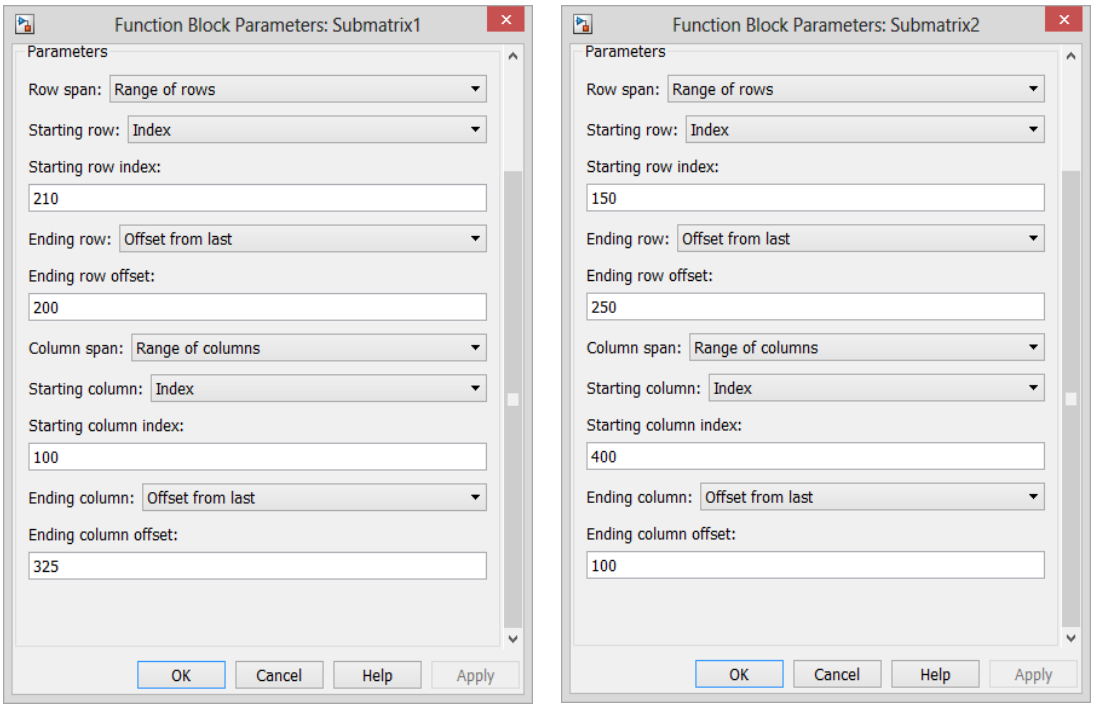


Figura 3.26: Submatrix.
Fuente: Simulink®.

Para realizar el recorte tanto de las filas y de las columnas de la imagen se procede a cambiar en la siguiente ventana los parámetros, cada uno de los valores en la ventana son conseguidos después de varias pruebas llegando a obtener los mejores para el recorte de la región de interés. Estos valores dependen de la región de interés que se obtendrá de la imagen en este caso la vía como se observó en la Figura 3.25.

"*Los recortes de la imagen son diferentes tanto de la Cámara Derecha como de la Izquierda, esto se debe a que los enfoques de las Cámaras son muy diferentes, como por ejemplo: la profundidad y el Angulo de visión. Sus valores varían dependiendo del tipo de vía en la que se conduzca ya que sus recortes pueden ser menos o más extensos.*"

Los valores tanto de la parte derecha como de la izquierda se observan en la Figura 3.27 que es la ventana de parámetros de ¨*Submatrix*¨.



a) b)

Figura 3.27: Parametros Submatrix (a) Cam Derecha, (b) Cam Izquierda.
Fuente: Simulink®.

3.5.2.3 2D FIR Filter.

Los (FIR) son filtros de repuesta finitos al impulso, posee diferentes características como son.

- Son fáciles de expresar en matrices de coeficientes.
- Son consideradas como extensiones naturales de los filtro FIR de una sola dimensión.
- No produce distorsión en la imagen[11].

El bloque se encuentra en la librería "*Computer Vision SystemToolbox*" en "*Filtering*" ver Figura 3.28.

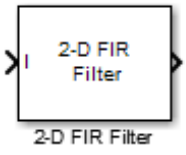


Figura 3.28: 2-D FIR Filter.
Fuente: Simulink®.

En la Figura 3.29 se puede observar los parámetro que se cambiaron en el "*Coeficient source"* se lo dejo en "*Specify via dialog"* que no especifica la vía de

diálogo para introducir los coeficientes del cuadro de dialogo de parámetros del bloque. "*Coeficients"*, mediante los diferentes cambios que se realizó optamos por una matriz de [1 0 1]. El "*Filtering base on"* este puede ser convolución o correlación en nuestro caso se escogimos convolución.

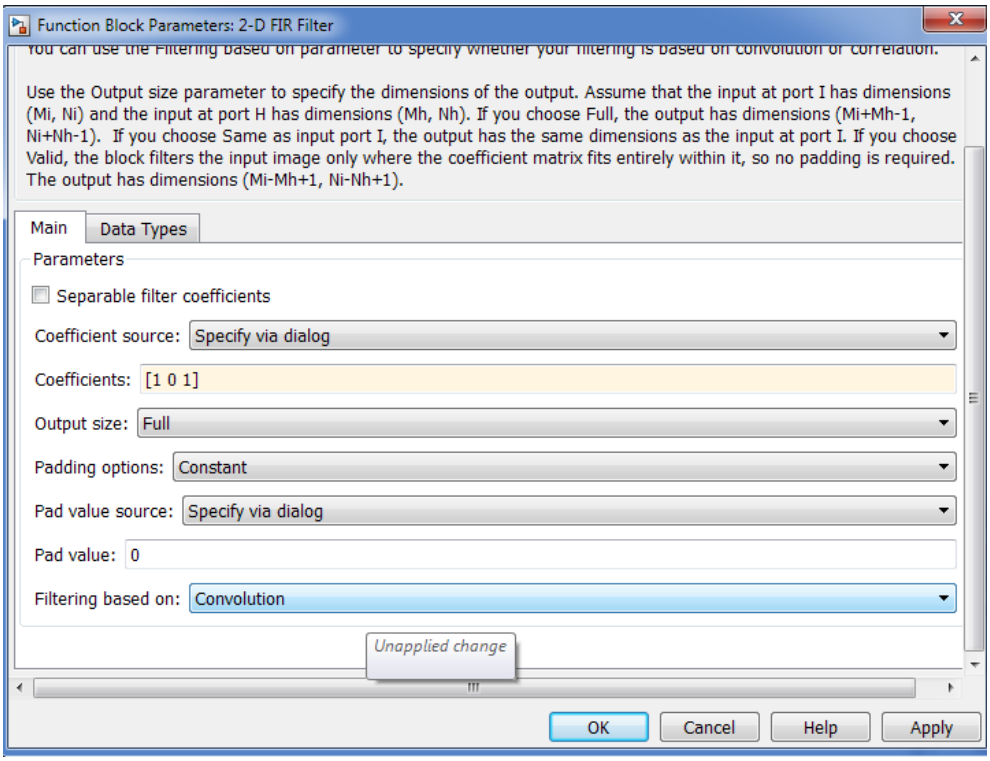


Figura 3.29: Parámetros 2D FIR Filter.
Fuente: Simulink®.

Coeficiente = [-1 0 1]

Se observa que nos presenta un color negro ya que un valor de la matrtiz es -1. Estos valores no nos presentan un buen filtrado de la imagen ya que no se observa las líneas de la vía ni los bordes como se muestra en la figura 3.30.

a) b)

Figura 3.30: Parámetros a cambiar en el bloque 2-D FIR Filter en [-1 0 1], (a) FIR y (b) Filtrado de la imagen.
Fuente: Simulink®.

Coeficiente = [-1 1 1]

Si se cambia solo el valor de cero por uno, notamos un cambio ya que cero nos puede hacer más negro y uno un color más blanco como se observa en la Figura 3.31 Pero tampoco hay un buen filtrado de la imagen ya que tampoco logramos observar las líneas de la vía.

a) b)

Figura 3.31: Parámetros a cambiar en el bloque 2-D FIR Filter en [-1 1 1], (a) FIR y (b) Filtrado de la imagen.
Fuente: Simulink®.

Coeficiente = [1 0 1]

Solo realizando el cambio de **1** y **0** en la matriz anterior notamos un cambio tanto en su claridad como también en el filtrado de la imagen ya que podemos observar con más claridad la línea de la vía y sus bordes. En la utilización para nuestro filtrado nos quedamos con estos valores de la matriz ya que es el más conveniente como se mencionó anteriormente ver Figura 3.32.

a) b)

Figura 3.32: Parámetros a cambiar en el bloque 2-D FIR Filter en [1 0 1], (a) FIR y (b) Filtrado de la imagen.
Fuente: Simulink®.

3.5.2.4 Edge Detection.

El siguiente bloque es utilizado para la detección de bordes, en la librería "*Computer Vision System Toolbox*" y en la subcarpeta "*Analysis & Enhancement*", dentro de los parámetros la opción "*Method*" se encuentran las opciones de operador Sobel, Prewitt, Roberts y Candy, como se observa en las siguientes Figuras 3.33 y 3.34.

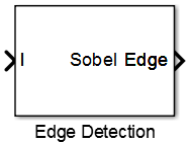


Figura 3.33: Edge Detection.
Fuente: Simulink®.

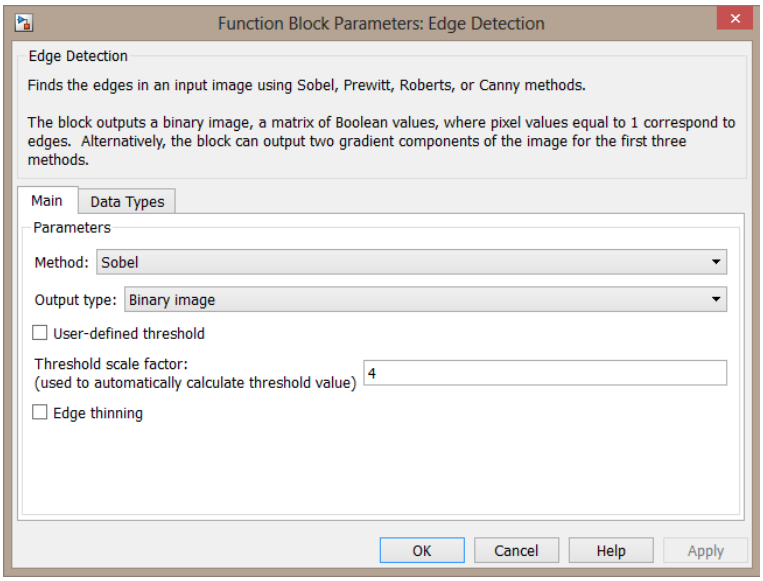


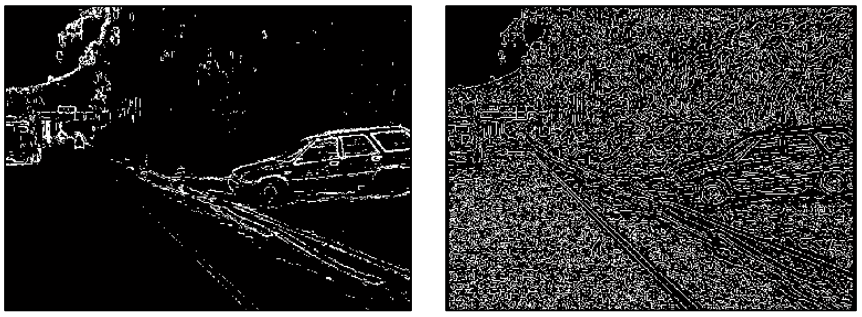
Figura 3.34: Parámetros Edge Detection.
Fuente: Simulink®.

Al seleccionar cada una de los métodos para la detección de bordes, observamos en la siguiente Figura 3.35.

a)

b)

c)



d) e)

Figura 3.35: a) Imagen de Intensidad, b) Imagen con Método Sobel, c) Imagen con Método Prewitt, d) Imagen con Metodo Roberts e) Imagen con Método Canny.
Fuente: Autores.

Al observar las imágenes optamos por escoger el detector de bordes Sobel por lo que es un filtro que realiza un suavizado sobre los datos, aunque el detector Prewitt también nos muestra los bordes pero no con un suavizado como el detector Sobel.

3.5.2.5 Operaciones Morfológicas.

Otro de las partes para el procesamiento de imagen son las operaciones morfológicas, estas son encargadas de convertir las características de los cuerpos en datos, por lo cual se utilizará para la detección del trazado de la vía, a continuación en la detección de bordes por erosión se utilizan los bloques explicados a continuación.

3.5.2.5.1 Compare to Constant.

Este bloque lo único que hace es comparar la señal de entrada con la que se encuentra dentro del bloque. Determina si la señal de entrada es mayor o igual que la constante especificada dentro del bloque en este caso 0.9. Este bloque se lo encuentra en la librería *Simulink*®, luego en la subcarpeta "*Logic and Bit Operations*" ver Figura 3.36.

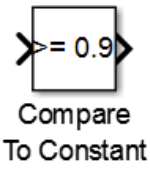


Figura 3.36: Compare To Constant.
Fuente: Simulink®.

3.5.2.5.2 Erosion.

Este bloque se encuentra en la librería de "*Computer Vision System Toolbox*", y en la siguiente subcarpeta "*Morphological Operations*" este bloque también nos permite la

detección de bordes pero en una imagen binaria. En este caso los parámetros no son modificados observar Figura 3.37.

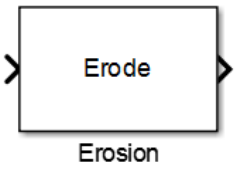


Figura 3.37: Erosion.
Fuente: Simulink®.

3.5.2.5.3 Imagen Complement.

Es muy importante dentro del sistema de visión artificial, como una etapa de procesamiento ya sea para detección de objetos o determinación de características en la imagen. Este bloque permite encontrar el complemento de la imagen o video [11]. Este se lo encuentra en la librería "*Computer Vision System Toolbox*" y en la subcarpeta "*Conversions*".

Figura 3.38: Imagen Complement.
Fuente: Simulink®.

3.5.2.5.4 Logical Operator.

Este bloque realiza una operación AND entre la imagen original y la inversa de la versión erosionada de la imagen, se obtendrá los bordes del objeto, ya que los pixeles de cada imagen a comparar son comunes y con ello obtenemos los bordes. Este bloque lo encontramos en la librería "*Simulink*", y en la subcarpeta "*Commonly Used Blocks*".

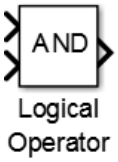


Figura 3.39: Logical Operator.
Fuente: Simulink®.

El Sistema de detección de bordes con la utilización del bloque erosión, queda estructurado en un diagrama de bloques como se observa en la Figura 3.40 igual el cambio de la imagen en la Figura 3.41 donde observamos el cambio.

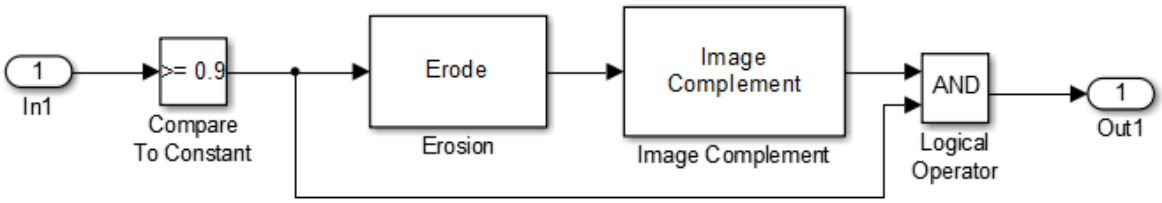
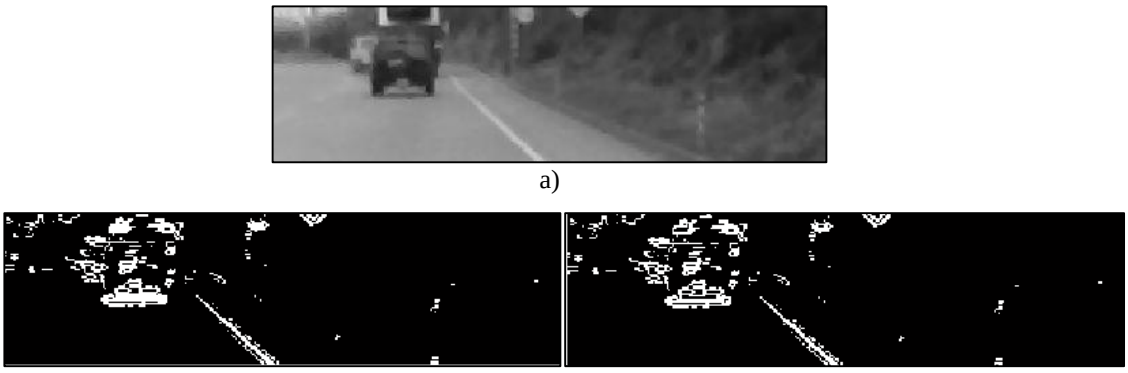


Figura 3.40: Detección de Bordes por Erosión.
Fuente: [11].



a)

b) c)

Figura 3.41: a) Imagen recortada en Escala de Grises, b) Imagen con Edge Detection, c) Imagen con Operación Morfológica.
Fuente: Autores.

3.5.2.6 Filtros no Lineales.

3.5.2.6.1 Median Filter.

Dentro de los filtros no lineales tenemos el "*Median filter*", este bloque se encuentra en la librería "*Computer Vision System Toolbox*", en la subcarpeta "*Analysis & Enhancement*" tiene la función de eliminar el ruido dentro de la imagen, ya que podemos introducir el tamaño de la región de influencia sobre la cual considera el cálculo de la mediana y el tamaño del resultado [11].

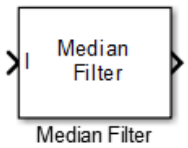


Figura 3.42: Median Filter.
Fuente: Simulink®.

Modificamos el parámetro "Neighborhood" podemos introducir el tamaño de la matriz y de acuerdo a ella se elimina el ruido de la imagen para nuestro caso se introdujo una matriz de [1 1]. Y mientras que en el "Output size" es el tamaño de la salida, solo expresa con las fronteras de la imagen por lo que este campo siempre se configura a "same as imput port I" [11].

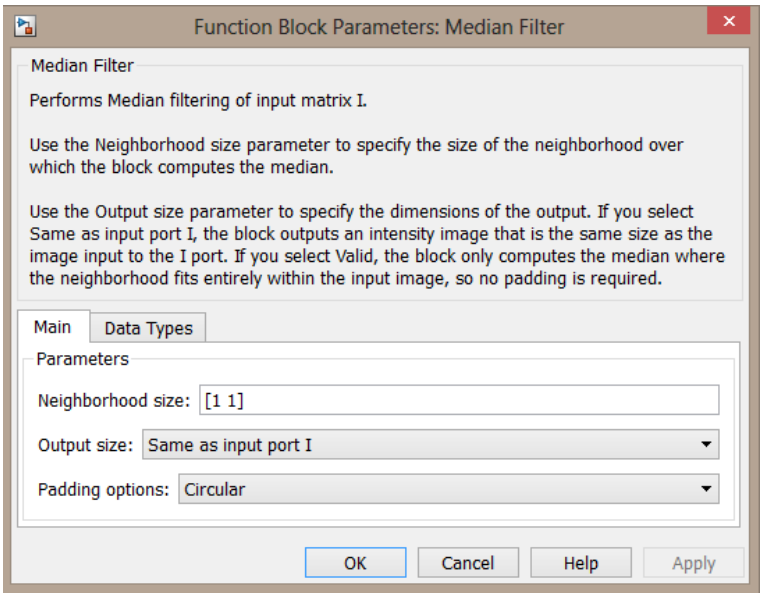


Figura 3.43: Parameters Median Filter.
Fuente: Simulink®.

Al utilizar el bloque "*Median Filter*", y con el parámetro "*Neighborhooh*" observamos en la siguiente Figura 3.44.

a)

b)

c)

Figura 3.44: a) Imagen recortada en Escala de Grises, b) Imagen con Operación Morfológica, c) Imagen con Median Filter.
Fuente: Autores.

El diagrama de bloques para la detección del trazado de la vía, se observa en la Figura 3.45.

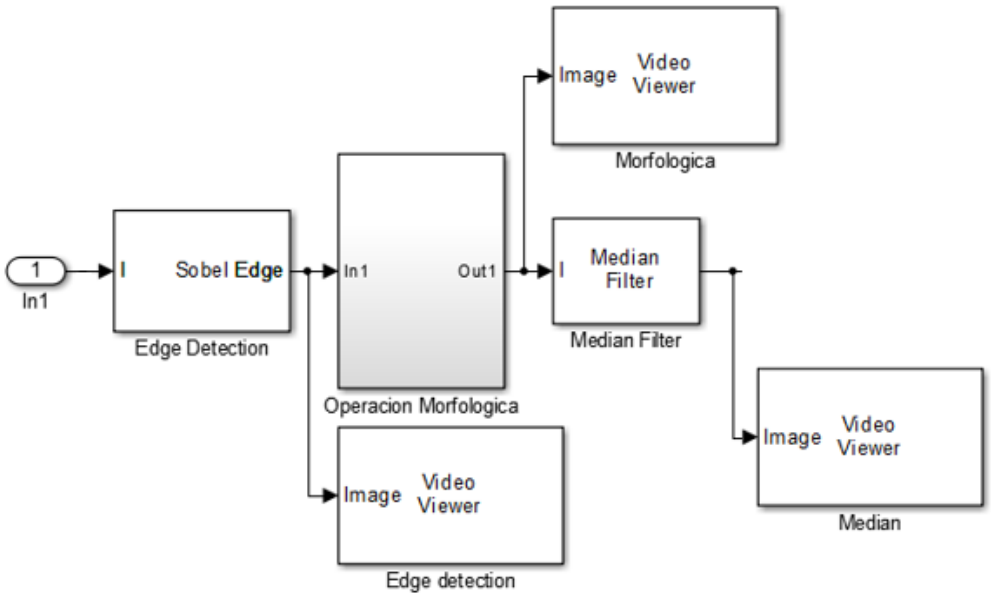


Figura 3.45: Diagrama de Bloques del Procesamiento de la Imagen para el Trazado de la Vía.
Fuente: Autores.

CAPITULO 4

DESARROLLO E IMPLEMENTACIÓN DEL ALGORITMO.

4. DESARROLLO E IMPLEMENTACION DEL ALGORITMO.

El capítulo cuatro menciona el desarrollo completo del algoritmo, que va desde las imágenes procesadas hasta el funcionamiento del algoritmo al funcionar de tal forma que, pueda reconocer vehículos, reconocer el trazado de la vía para la detección de curvas y la lectura de la velocidad. En base a la velocidad y las detecciones respectivas, enviara una señal de aviso al conductor para la adaptación de velocidad respectiva.

Se presentara todo el procedimiento para la detección de vehículos, en base a esto determinar la distancia de seguridad de circulación entre vehículos, con respecto al reconocimiento del trazado de la vía, se hará el reconocimiento de la vía para determinar las curvas cuando se esté en circulación. Estos dos reconocimientos se los realiza para la comparación con la velocidad de circulación, comparando así las velocidades y las situaciones de manejo. Esto proporcionara un aviso al conductor para que él tenga la oportunidad de adaptar la velocidad de circulación.

4.1 DETECCIÓN DE VEHÍCULOS.

Para la detección de vehículos por intermedio de visión artificial comenzaremos explicando el método de coincidencia de características, que consiste en buscar una característica en particular o grupo de características del objeto que deseemos identificar. Este método es eficiente debido a que las características de un mismo objeto varía desde su tamaño, forma o color, además de estos tres se puede presentar otras alteraciones en las características dentro de la misma escena como por ejemplo: si deseamos reconocer los vehículos de la Figura 4.1 observamos una escena donde podemos reconocer dos vehículos, ambos tienen similares características en su forma y color pero existe una diferencia en su tamaño, ya que el vehículo de la izquierda está más cerca de la cámara que el vehículo de la derecha; esta diferencia de distancia que existe de los vehículos con respecto a la cámara hace que las características sean distintas una con relación a la otra, a su vez la iluminación entre los dos vehículos son diferentes ya que existe una sombra sobre el vehículo de la

derecha; la variación de características mencionadas que existe entre vehículos hace que la identificación de ambos vehículos sean complicadas o nulas de realizarse por métodos comunes como: Filtros Gaussianos, Segmentación, Correlación, entre otros. Ya que los mencionados no pueden identificar las diferentes características que los objetos poseen en una misma escena o cambian en diferentes escenas (en el caso de un video), para ello se utiliza el método de coincidencia de características [5][16].

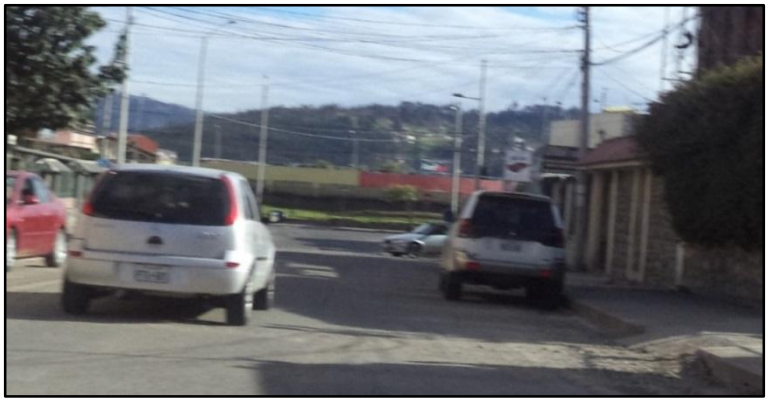


Figura 4.1: Variación de Distancia e Iluminación.
Fuente: Autores.

Para identificar una característica en una escena, la característica del objeto deberá recorrer por toda la escena hasta encontrar alguna coincidencia, ahora si tenemos un banco de características que cubra todas las posibilidades que mencionamos anteriormente como es tamaño, forma, iluminación, color, entre otras. Esto permite mejorar el rango de búsqueda y que la identificación sea veras, pero en si el método puede resultar ineficiente ya que el tiempo en recorrer todas las características para reconocer un solo objeto resultaría computacionalmente grande. Siendo inútil para una aplicación en tiempo real, de ahí que se implementa los clasificadores simples.

Estos permiten agrupar las características de forma que si se desea identificar un objeto se va a realizar en base a un umbral, si las características coinciden con la escena están fuera de este umbral, no se considera que este sea el objeto a identificar; este tipo de clasificador simple puede ser muy ineficiente ya que el umbral de este grupo de características puede ser muy flexible e identificar objetos similares como el objeto que deseamos. Es por ello que en el Algoritmo de Viola –Jones se utiliza varios clasificadores simples para mejorar el rango de detecciones verdaderas, de ahí que se desarrolla los Clasificadores en Cascada.

4.1.1 CLASIFICADOR EN CASCADA.

El clasificador en cascada como ya mencionamos fue implementado por Viola-Jones en el año de 2001 con el fin de mejorar la detección de rostros por intermedio de uso de características. En el artículo que realiza Viola-Jones especifican tres aportes para mejorar la detección de rostros. Primero la implementación de una imagen integral, esto permite que la búsqueda de características en una imagen sea más rápida. La segunda contribución es utilizar clasificadores simples que permite clasificar las características, dependiendo de las condiciones en las que se encuentra o también clasifica de acuerdo a la cantidad de características existentes. La tercera aportación que realizó Viola-Jones es utilizar los clasificadores en forma de cascada que permite desechar de manera inmediata las escenas que no contienen el objeto que deseamos identificar. Con estas tres aportaciones hechas por Viola-Jones, obtenemos un alto índice de detecciones verdaderas. Esto también disminuye el tiempo computacional requerido para las identificaciones, siendo útil en tiempo real [17][18].

El clasificador en Cascada consiste en etapas donde cada etapa tiene clasificadores simples que en su conjunto forman un clasificador fuerte; las primeras etapas consiste en un clasificador que van a identificar todos los objetos que tengan características parecidas al objeto a identificar; al transcurrir las etapas los clasificadores van a ir descartando los objetos que no sea el objeto deseado. Y si la escena no tiene ninguna característica, esta es descartada de inmediato y pasa así a la siguiente escena a analizar, según como vaya aumentando las etapas los clasificadores van a ser más selectivos y descartando los objetos que no sean el objeto a identificar ver Figura 4.2 [19].

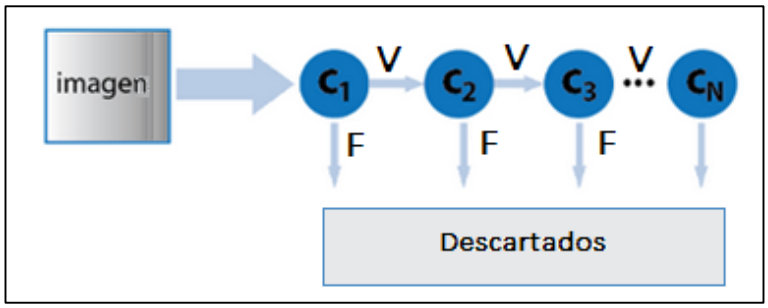


Figura 4.2: Funcionamiento Clasificador en Cascada.
Fuente: MathWorks®.

4.1.2 PROCESO DE ENTRENAMIENTO DE UN CLASIFICADOR EN CASCADA PARA VEHÍCULOS.

Para el proceso de programación para la detección de vehículos utilizamos el entorno de Matlab® Editor, en la cual por medio de comandos permite realizar detecciones mediante el algoritmo de Viola-Jones para ello debemos utilizar un entrenador, en vista que el comando ***vision.CascadeObjectDetector(' ')*** está configurada solo para detectar rostros y características faciales, como: ojos, boca o nariz.

Para poder crear un clasificador de vehículos debemos entrenar al mismo, al entrenar un clasificador obtenemos un documento con extensión **.xml**. Este documento va a ser nuestro banco de características ordenadas en varios clasificadores de tal manera que vaya descartando las escenas que no contengan el objeto a identificar como lo explicaremos posteriormente.

4.1.2.1 Entrenamiento del Clasificador.

Para entrenar un clasificador utilizamos una aplicación de Matlab® llamada "CascadeTrainingGUI", esta aplicación podemos descargarnos de la página MathWorks® en [18].

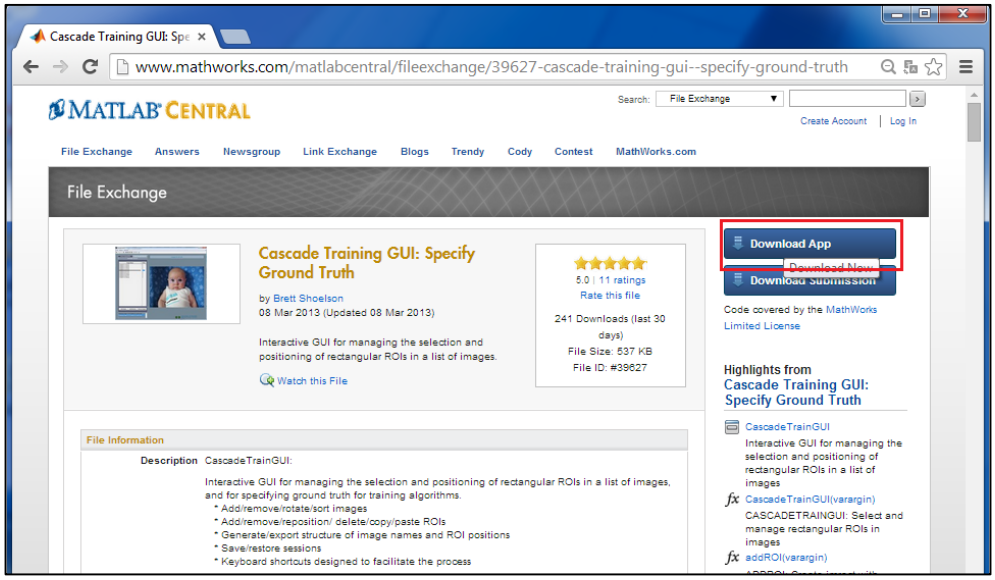


Figura 4.3: Pagina para Descargar el Entrenador.
Fuente: Autores.

Para descargar se crea una carpeta con nombre "MY APP" dentro de la carpeta de Matlab® que se encuentra en el "Disco Duro C" en el computador. En esta carpeta podemos guardar las aplicaciones que se desea.

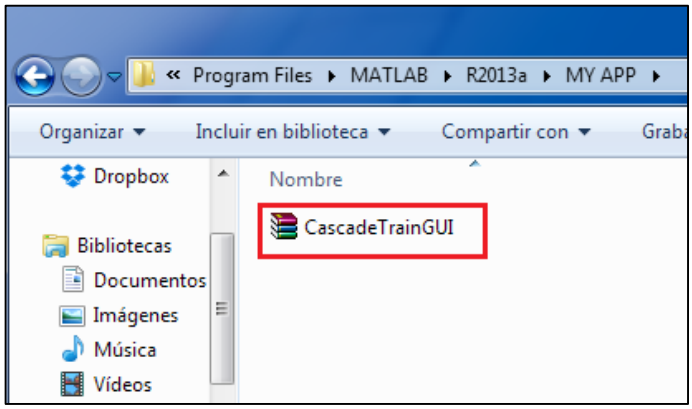


Figura 4.4: Carpeta en Matlab®.
Fuente: Autores.

A continuación abrimos el Matlab®, en la barra superior encontramos **"APPS"**, luego en la parte izquierda superior, damos clic en ***"Install App"***, y buscamos el archivo que descargamos y guardamos en la carpeta de nombre Matlab®.

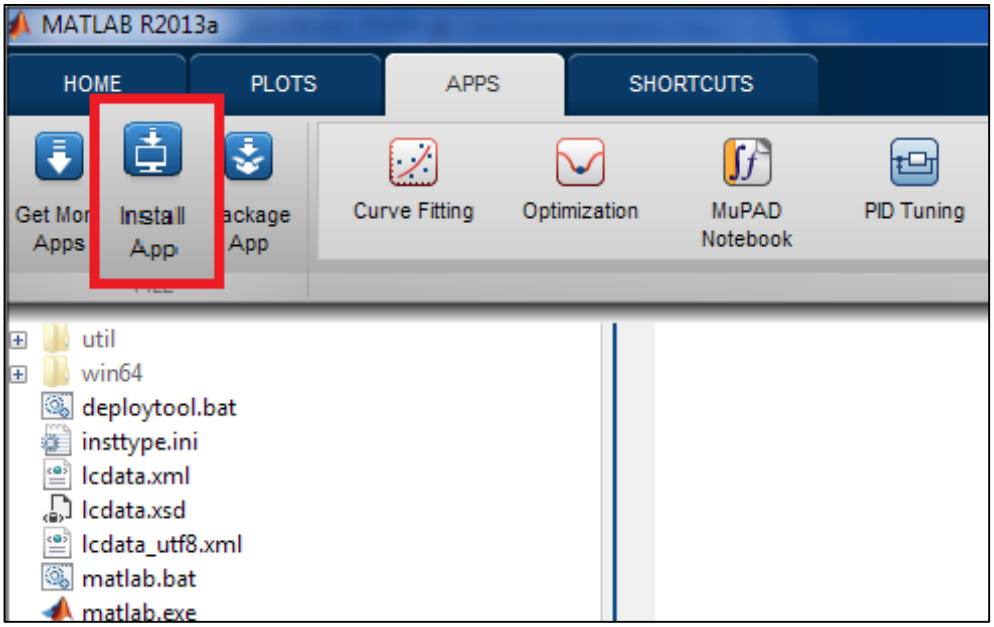


Figura 4.5: Instalar una Nueva APP.
Fuente: Autores.

Una vez terminado de instalar al "*CascadeTrainingGUI*", podemos encontrarle en la lista de APPS que tiene Matlab®, a continuación abrimos la aplicación.

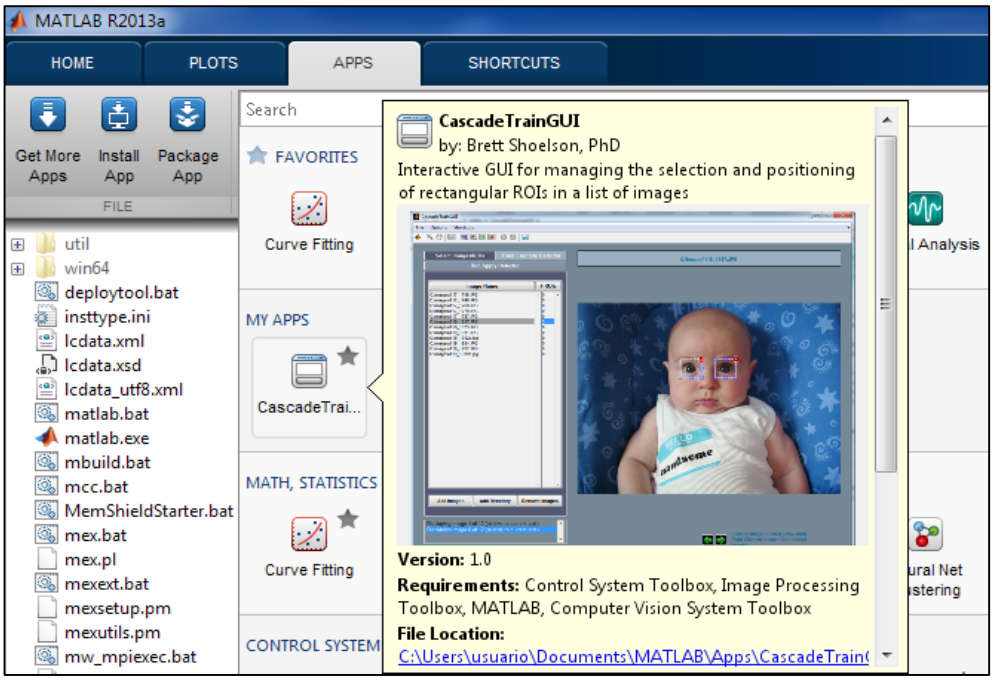


Figura 4.6: Abrir el Entrenador.
Fuente: Autores.

El primer paso, una vez abierto el "*CascadeTrainingGUI*", seleccionamos la opción de "*Select Images/ROIs*", una vez seleccionado esta opción nos dirigimos a la opción "*Add Directory*" que se encuentra en la parte izquierda inferior, aquí debemos buscar la dirección donde guardamos las imágenes positivas, que fueron previamente separadas, en el capítulo anterior y procedemos a cargarlas al programa.

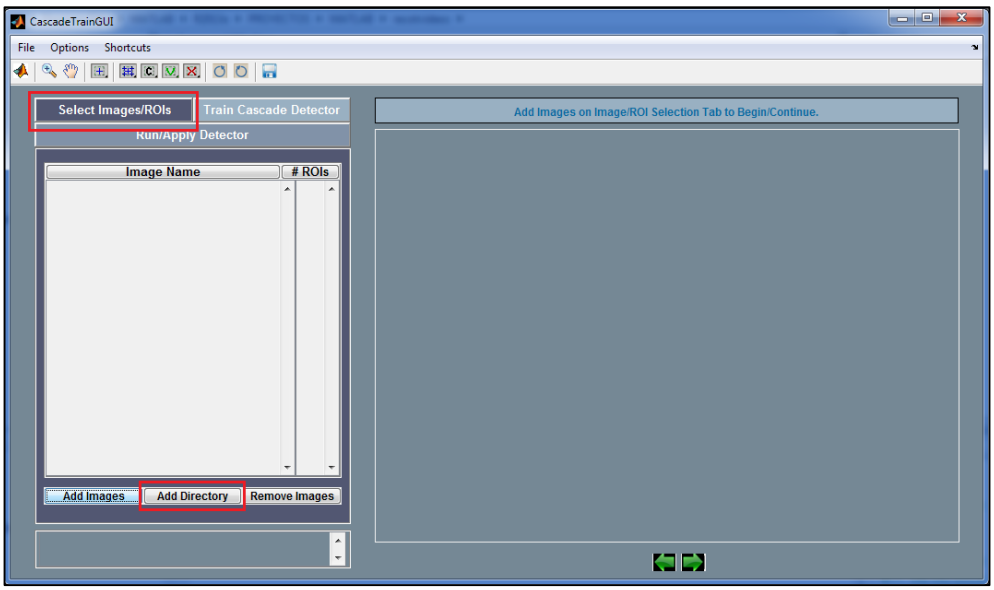


Figura 4.7: Cargar la Imágenes Positivas al Entrenador.
Fuente: Autores.

Cargadas las imágenes positivas debemos de seleccionar la región de interés (ROI) en cada una de las imágenes, para ello damos un clic sobre la imagen y seleccionamos la región de interés; para pasar de imagen damos clic en las flechas que se encuentra en la parte inferior de la imagen cuando las mismas se encuentran de color verde.

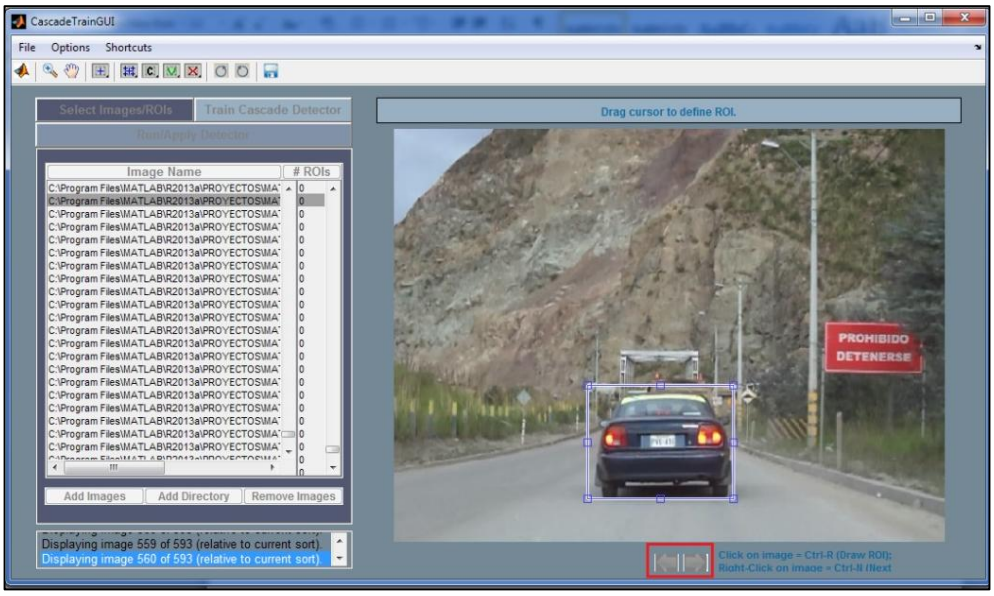


Figura 4.8: Selección la ROI de Cada Imagen.
Fuente: Autores.

Una vez seleccionado las ROIs de todas las imágenes positivas cargadas en el entrenador debemos seleccionar la opción "*Train Cascade Detector*", donde debemos de cargar las imágenes negativas, para ello seleccionamos la opción ¨ADD DIR¨.

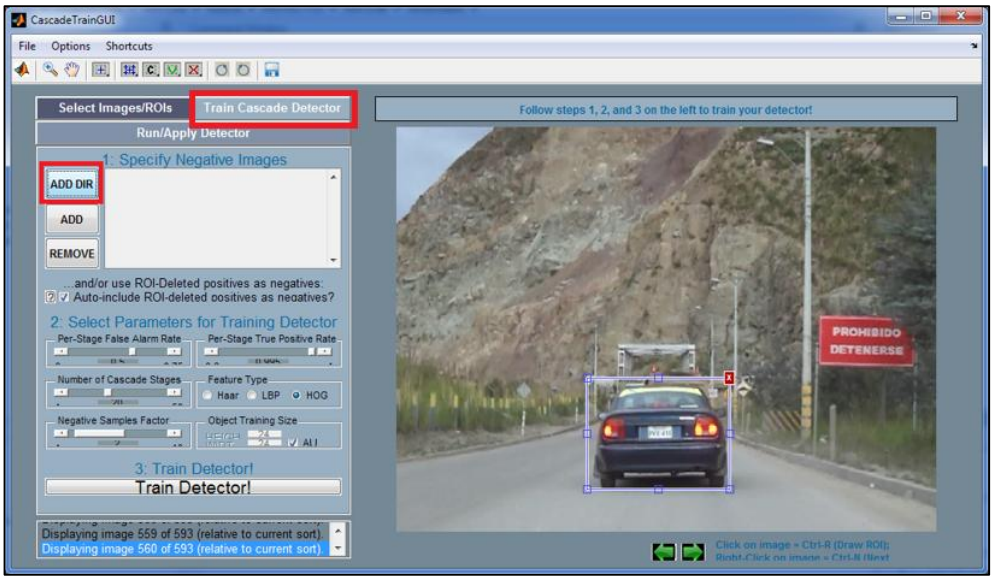


Figura 4.9: Cargar las Imágenes Negativas.
Fuente: Autores.

Una vez cargadas las imágenes negativas debemos de modificar los parámetros que tenemos en "*Select Parameters for Training Detector*" como podemos ver en la Figura 4.10. En la primera Opción (1) tenemos la tasa de falsas detecciones; en la (2) el número de tasas de detecciones verdaderas; en la opción (3) seleccionamos las etapas para la cascada; en la opción (4), podemos elegir el tipo de característica que vamos emplear; en la opción (5), podemos poner el número de imágenes negativas que vamos emplear y por último en la opción (6) colocamos el tamaño mínimo del objeto a detectar [20].

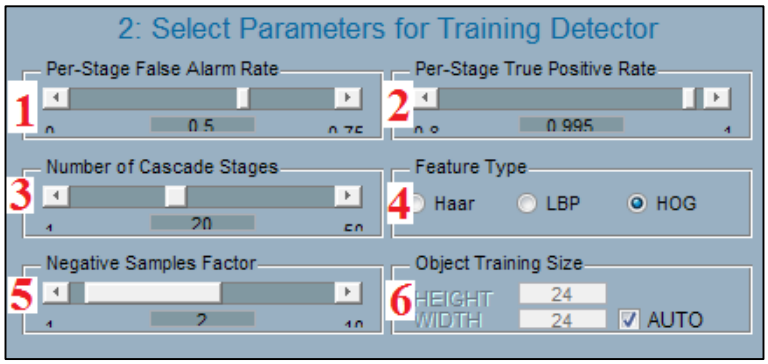


Figura 4.10: Modificación de los Parámetros para Entrenar el Clasificador.
Fuente: Autores.

Para garantizar que un clasificador realice reconocimientos eficaces, va a depender de las siguientes consideraciones.

4.1.2.1.1 Falsas Detecciones.

Es la tasa de detecciones erróneas que genera el clasificador, mientras mayor sea esta, mayor probabilidad existe que el clasificador identifique el objeto a encontrar aumentando también así el número de falsas detecciones.

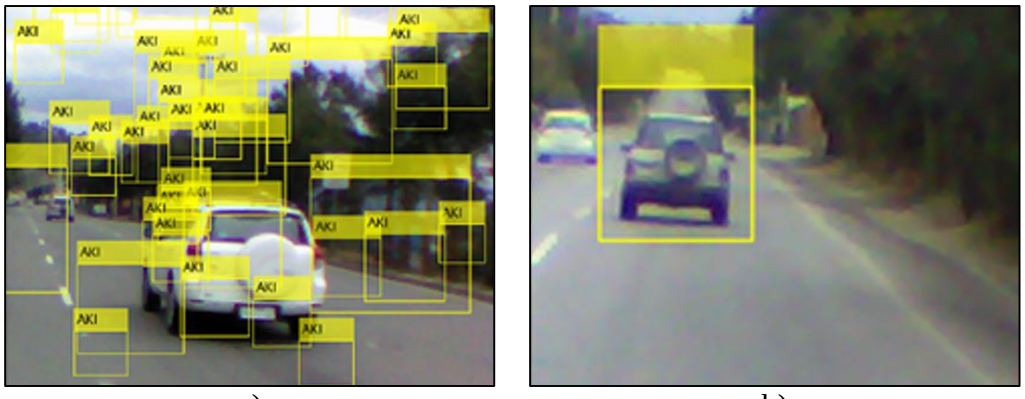
4.1.2.1.2 Verdadera Detecciones.

Es la tasa de detecciones verdaderas que va a generar el clasificador, si este es demasiado alto, se la opción a que el clasificador deje pasar una detección verdadera.

4.1.2.1.3 Etapas.

Si tenemos pocas etapas es más seguro que existan demasiadas falsas detecciones, en cambio si las etapas son demasiadas existe la posibilidad que el clasificador no identifique algunos objetos que deseemos.

Otro parámetro que influencia en la calidad de clasificador, es el número de imágenes positivas y negativas con la que entrenamos el clasificador; ya que si contamos con pocas imágenes positivas los clasificadores tendrán pocas características para comparar, al igual si tenemos pocas imágenes negativas el número de falsas detecciones va a tener un alto porcentaje; debido a que las imágenes negativas son las primeras en ser comparadas con la escena captada por la cámara, estás son descartadas de manera inmediata. Así también se logra mejorar el tiempo computacional del clasificador. En la Figura 4.11 podemos observar las diferencias de los resultados de los clasificadores Rastreador 1 y Rastreador 12 de la Tabla 4.1 [19], [21].



a) b)

Figura 4.11: a) Resultado Clasificador rastreador 1, b) Resultado Clasificador rastreador 12.
Fuente: Autores.

En la Figura 4.11, parte izquierda (a) observamos un clasificador con alto índice de falsas detecciones, mientras en la parte derecha (b) el clasificador obtiene bajo índice de falsas detecciones. En la siguiente Tabla 4.1 podemos observar algunas modificaciones de los parámetros mostrados y sus resultados obtenidos.

La Tabla 4.1 hemos colocado las modificaciones que hemos hecho para obtener al clasificador que mejor identifique los vehículos que deseamos reconocer, este clasificador va a reconocer vehículos como: autos, camionetas, 4X4, y vehículos semipesados. Para los vehículos pesados realizamos otro clasificador ya que las características entre estos grupo de vehículos presentan algunas diferencias en cuanto a su tamaño y forma. Si se genera un clasificador para detectar vehículos livianos y pesados a la vez la posibilidad de falsas detecciones es alta, por lo mismo hemos visto la necesidad de crear los clasificadores por separado para vehículos livianos y otro para vehículos pesados.

Tabla 4-1: Propiedades para Entrenar Clasificadores para Vehículos livianos y Semipesados.
Fuente: Autores.

Nombre	# Img. Positiva	# Img. Negativa	Etapas	Tipo Característica	% Falsas detecciones	% Verdaderas detecciones	Autos	4X4	Semipesado	Falsas Detecciones
rastreador_1	82	10	5	HOG	0.000000954	0.9049	×	×	×	Alto
rastreador_2	156	58	10	HOG	0.000000954	0.9049	+/-	×	×	Alto
rastreador_3	156	100	10	Haar	0.000000954	0.9553	+/-	×	×	Medio
rastreador_4	247	185	20	Haar	0.000000954	0.9049	+/-	×	+/-	Medio
rastreador_5	247	185	18	Haar	0.000000954	0.959	✓	×	×	Medio
rastreador_6	243	146	15	HOG	0.00000755	0.9362	✓	+/-	+/-	Bajo
rastreador_7	243	146	18	HOG	0.00003175	0.9524	+/-	+/-	+/-	Bajo
rastreador_8	243	146	20	HOG	0.000007622	0.95323	+/-	+/-	✓	Bajo
rastreador_9	310	131	20	Haar	0.00001138	0.9504	✓	+/-	+/-	Medio
rastreador_10	310	131	20	HOG	0.00001138	0.9504	✓	✓	×	Medio
rastreador_11	310	131	20	LBP*****	0.00001138	0.9504	✓	+/-	×	Medio
rastreador_12	310	131	27	HOG	0.00000231	0.9507	✓	✓	+/-	Bajo

*****El tipo de característica LBP no fue muy utilizado debido a que el tiempo computacional para generar un clasificador con este tipo de características es demasiado en este caso se demoró alrededor de 24 horas y los resultados fueron similares al de un clasificador con características Haar que en este caso se genera en un tiempo aproximado de 30 minutos.

Tabla 4-2: Tabla de Propiedades para Entrenar Clasificadores para Vehículos Pesados.
Fuente: Autores.

Nombre	# Img. Positiva	# Img. Negativa	Etapas	Tipo Característica	% Falsas detecciones	% Verdaderas detecciones	Buses	Volquetas	Vehículos Livianos	Falsas Detecciones
Pesado_1	104	79	20	HOG	0.000002141	0.9045	✓	×	+/-	Alto
Pesado_2	104	79	30	HOG	0.000002141	0.99778	+/-	×	✓	Alto
Pesado_3	104	104	40	HOG	0.000002141	0.99778	+/-	×	+/-	Medio
Pesado_4	104	104	40	Haar	0.000002141	0.99778	+/-	×	+/-	Medio
Pesado_5	104	104	37	Haar	0.000002383	0.9985	✓	×	+/-	Medio
Pesado_6	104	104	37	HOG	0.000002383	0.9985	✓	+/-	+/-	Medio

4.1.3 PROCESO DE PROGRAMACIÓN PARA LA DETECCIÓN DE VEHÍCULOS.

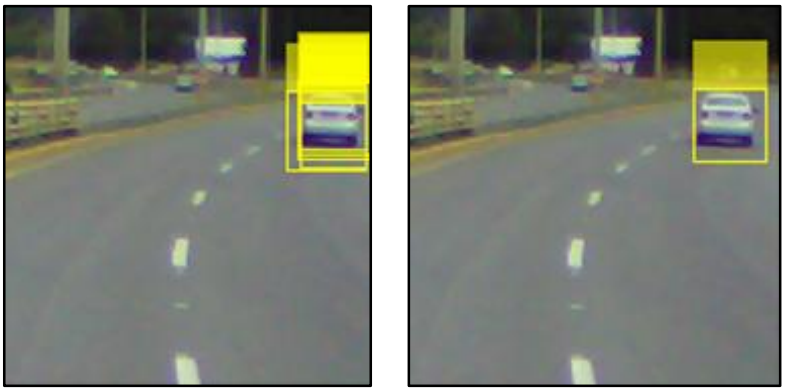
Una vez generado los clasificadores para los vehículos, podemos realizar la programación para reconocer los vehículos de las escenas que son captada por la cámara en tiempo real, para ello como primer paso debemos utilizar el comando para llamar al clasificador que previamente entrenamos, en este caso llamamos a nuestro clasificador rastreador6.xml por intermedio del comando **(vision.CascadeObjectDetector)** como vemos en la línea de programación 18, entre los paréntesis colocar comillas y el nombre de nuestro clasificador.

```
16 -    vid.ReturnedColorspace = 'rgb';
17
18 -    detector = vision.CascadeObjectDetector('rastreado6.xml');
19
20 -    detector.MinSize=[30 30];
21
22 -    detector.MaxSize=[100 90];
23
24 -    detector.MergeThreshold=2;
25
26 -    detector.ScaleFactor=1.2;
27
28 -    start(vid);
```

Existen algunos comandos que permiten modificar algunas propiedades de clasificador para mejorar el tiempo computo en el momento que comience a identificar los vehículos, como por ejemplo podemos delimitar con **MinSize** el tamaño mínimo del objeto que deseamos identificar como indicamos en la línea 20, mientras con **MaxSize** se limita el máximo tamaño del objeto que deseamos identificar como indicamos en la línea 22, en nuestro caso el tamaño de los vehículos que vamos a identificar va a encontrarse en un rango mínimo de 30 x 30 (ancho x alto), y el máximo tamaño a identificar es de 90 x 90 (ancho x alto), así permite que nuestro detector descarte de manera inmediata los objetos que no se encuentran entre este rango.

Otra propiedad que podemos modificar es **MergeThreshold** el cual crea un umbral para controlar que sobre un mismo objeto detectado vuelva a detectar otras veces, ya que el clasificador fue entrenado con varios ROIs de las imágenes positivas al

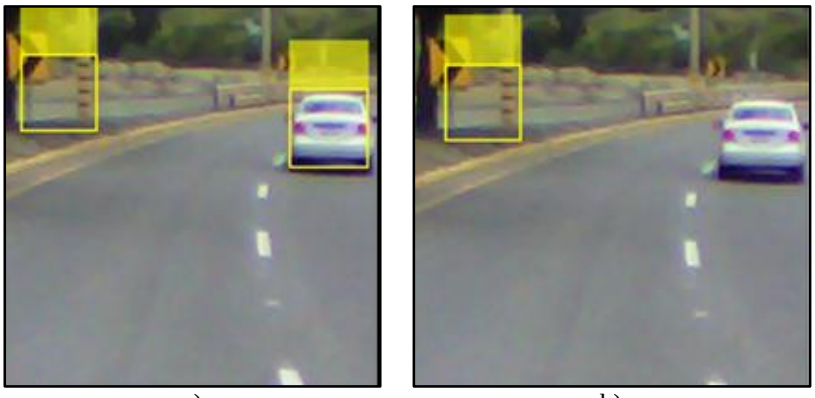
momento de identificar un vehículo es común que más de un clasificador haya aprobado el objeto, cuando el **MergeThreshold** sea igual a **0** como se muestra en la parte izquierda (a) de la Figura 4.12 el clasificador va a reconocer más de una vez al mismo objeto, si aumentamos el valor a 3 el número de detecciones sobre el mismo objeto va a disminuir como se muestra en la parte derecha (b) de la Figura 4.12.



a) b)

Figura 4.12: a) MergeThreshold =0, **b)** MergeThreshold =3.
Fuente: Autores.

Otra propiedad que nos ayuda a mejorar el tiempo de computo es **ScaleFactor** esta propiedad permite incrementar la región de búsqueda dentro de la escena de forma escalar, el incremento escalar se va a efectuar en el rango entre **MinSize** y **MaxSize** ya que las características que van a usar los clasificadores se encuentran entre este rango. En la Figura 4.13 se muestra los resultados cuando se modifica la propiedad **ScaleFactor.**



a) b)

Figura 4.13: a) ScaleFactor 1.005, b) ScaleFactor 1.07.
Fuente: Autores.

Si modificamos el *ScaleFactor* a 1.005 observamos que el clasificador identifica un vehículo además una falsa detección, si aumentamos el valor del *ScaleFactor* la falsa

detección va a permanecer puesto que al aumentar la escala de búsqueda el vehículo quedaría fuera de este rango de búsqueda, por lo que en este caso debemos de recurrir a realizar otros métodos como recortar el video solo de la región de interés Figura. 4.15 para así evitar que el clasificador analice regiones innecesarias que puedan contener potenciales falsas detecciones.

Figura 4.14: ScaleFactor = 1.02 solo de la Región de Interés.
Fuente: Autores.

4.1.4 DETECCIÓN DE LOS VEHÍCULOS EN EL VIDEO.

Una vez que ya está realizada el indexado del detector y modificado sus propiedades, procedemos a detectar los vehículos en el video que las cámaras estén captando, para ello dentro de la programación de retorno de pantalla introducimos los siguientes comandos.

```
37
38 -      bbox=step(detector,contraste);
39 -    if (size(bbox)~=0)
40
41 -       v=bbox(1,4);
42
43 -           if (v<=39)
44 -              disp('18m-16m de distancia');
45 -           end
46
47 -            if (40<=v)&&(v<=49)
48 -                disp('14m-10m de distancia');
49 -            end
50
51 -                if (50<=v)
52 -                  disp('-8m PELIGRO');
53 -                end
54 -    end
55 -   cuadro=insertObjectAnnotation(videocor,'rectangle',bbox,' ');
56
57 -   imshow(cuadro);
58
59 -   end
```

En la línea 38 utilizamos el comando *step* que permite obtener las coordenadas del objeto identificado por el clasificador de nombre detector en el video.

```
38 -      bbox=step(detector,contraste);
```

Para la visualización de los objetos que el clasificador este detectando, usamos el comando **insertObjectAnnotation** como en la línea de comando 55, dentro los paréntesis primero debemos de colocar el nombre de la variable donde se encuentra el video que deseamos ver en este caso es **videocor**, después entre comillas colocamos "**rectangle**" que permite encerrar el objeto detectado en un rectángulo, a continuación colocamos las coordenadas que deseamos delimitar en el cuadrado en este caso las coordenadas se encontraran en la variable **bbox**, por último entre dos comillas, se puede colocar texto sobre el objeto delimitado.

Por último en la línea 57 colocamos el comando **imshow** y entre paréntesis la variable (cuadro), para poder visualizar el video cuando detecta un vehículo en el caso de que deseemos observar el video captado por la cámara colocamos **imshow** y entre paréntesis (vid), la diferencia se muestra en la Figura 4.16.

a) b)

Figura 4.15: a) Imshow (vid), b) Imshow (cuadro).
Fuente: Autores.

4.1.5 RECONOCIMIENTO DE DISTANCIAS.

Al obtener un algoritmo sólido para el reconocimiento de los vehículos, ahora realizamos un programa para estimar la distancia que existe entre el vehículo detectado y el vehículo que lleva el sistema de adaptación de velocidad.

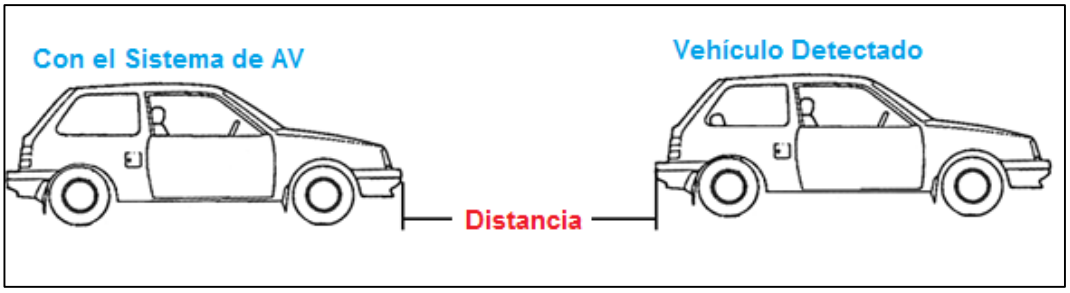


Figura 4.16: Aproximación de Distancia entre vehículos.
Fuente: Autores.

Para realizar un programa que detecte las distancia entre los vehículos debemos de tener en cuenta cual es la distancia máxima y mínima a la cual va a reconocer un vehículo el programa de detección de vehículos, para ello procedimos a colocar un vehículo a una distancia de 4m frente del vehículo que va a tener el sistema de Adaptación de Velocidad (AV) como muestra la Figura 4.17 , cada vez retrocedemos dos metros el vehículo que tiene el sistema de AV hasta que el programa deje de detectarnos el vehículo de enfrente, la distancia máxima que el programa de detección de vehículos reconoce es de 22m , la distancia mínima en este caso se da a 4m.

Al realizar esta verificación del rango de funcionamiento del programa de detección de vehículos, pudimos observar que mientras más lejos el vehículo se encuentra el cuadro que lo delimita se va haciendo más pequeño y cuando un vehículo está a

distancias más cortas el cuadro que lo delimita es más grandes, como podemos observar en la Figura 4.18 esta variación del tamaño del cuadro que lo delimita es proporcional a la distancia por lo cual nos fue de gran ayuda para realizar el programa para estimar la distancia entre vehículos.

a) b)

Figura 4.17: Diferencia de Ancho de Cuadro con Respecto a la Distancia.
Fuente: Autores.

Esta diferencia de las dimensiones del cuadrado delimitador, permite obtener la siguiente información como vemos en la Tabla 4.3 para cada distancia existen los cuadrados delimitadores obtiene un rango especifico que ocupamos para el programa de estimación de la distancia en la cual se encuentra el vehículo identificado.

Tabla 4-3: Rango del cuadro delimitador con respecto a la Distancia.
Fuente: Autores.

Distancia	Ancho	Alto
4m	[108 116]	[90 98]
5m	[101 107]	[85 89]
6m	[92 102]	[77 86]
7m	[88 91]	[74 76]
8m	[80 87]	[67 73]
9m	[77 79]	[65 66]
10m	[61 76]	[52 64]
12m	[52 59]	[44 50]
14m	[48 50]	[40 43]
16m	[45 48]	[37 39]
18m	[43 46]	[34 36]
20m	[43 35]	[33 32]
22m	[36 38]	[30 32]

Las medidas del alto del cuadro permiten realizar mediante condiciones lógicas la siguiente programación.

```
19 -    videocor=imcrop(image, [250 150 169 169]);
20
21 -    bbox=[0 0 0 0];
22 -    H=0;
23
24 -    bbox=step(detector,videocor);
25
26 -    if (size(bbox)~=0)
27
28 -        H=bbox(1,4);
29
30 -       switch H
31
32 -       case {74, 75, 76}
33
34 -                   disp('7m');
35
36 -       case {67, 68, 69, 70, 71, 72, 73}
37
38 -                   disp('8m');
39
40 -       case {65, 66}
41
42 -                     disp('9m');
43
44 -        case {51, 52, 53, 54,55, 56, 57, 58, 59,60, 61, 62, 63, 64}
45
46 -                     disp('10m');
```

```
47
48 -        case {44, 45, 46, 47, 48, 49,50}
49
50 -                    disp('12m');
51
52 -        case {40, 41, 42, 43}
53
54 -                    disp('14m');
55
56 -        case {37, 38, 39}
57
58 -                    disp('16m');
59
60 -        case {33, 34, 35, 36}
61
62 -                    disp('18m');
63
64 -        case {30, 31, 32}
65
66 -                    disp('20m');
67 -           otherwise
68 -               disp('')
69 -       end
70
71
72 -    end
73 - cuadro=insertObjectAnnotation(videocor,'rectangle',bbox,'carro ')
```

Creamos en la línea de comando 21 una matriz 1x4 de valor cero para cuando el detector no reconozca ningún vehículo el video se siga en su reproducción.

En la línea de comando 22 se crea la variable H=0 que cambiara al valor de **bbox** (1,4) cuando la condición lógica *if* se cumple. En este caso la condición para que el valor de H sea **bbox** (1,4) es que toda la matriz **bbox** debe de ser diferente de cero como lo expresamos en la línea de comando 26.

```
26 -    if (size(bbox)~=0)
27
28 -        H=bbox(1,4);
```

4.1.6 TRANSFERENCIA DEL PROGRAMA DE DETECCIÓN DE VEHÍCULOS DE MATLAB® A SIMULINK®.

El algoritmo para la detección de vehículos fue necesario realizar en el programa Matlab® ya que el mismo cuenta con el comando **vision.CascadeObjectDetector( )** el cual nos permite realizar la detección de los objetos que deseamos reconocer, lo

cual en el entorno de Simulink® no lo es fácil de realizarlo. Para poder unir el algoritmo de detección de la trayectoria de la vía y la de detección de vehículos procederemos a llamar el programa de detección de vehículos del entorno de Matlab® al entorno de Simulink®.

Para hacer posible que un programa realizado en Matlab® pueda correr en Simulink® se utiliza el block **Interpreted Matlab Function** Figura 4.19 este bloque podemos encontrar en la librería de Simulink® que tiene el mismo nombre "*Simulink®"* en "*User-Defined Functions*" y por ultimo obtenemos el "*Interpreted MATLAB Function*".

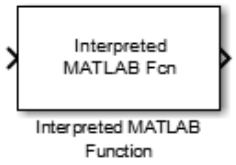


Figura 4.18: Block Interpreted MATLAB Function.
Fuente: Simulink®.

Con este bloque podemos hacer correr desde Simulink® un archivo .m realizado en Matlab Editor, para ello debemos de dar doble clic en el bloque para modificar las opciones Figura 4.19 en la opción *MATLAB® function*: colocamos el nombre del archivo .m en nuestro caso el nombre es videofn este archivo de nombre *videofn.m* es el que tendrá el algoritmo de Viola-Jones que permitirá la detección de vehículos, además debemos de desactivar la opción *Collapse 2-D results to1-D* una vez cambiada estas opciones damos clic en *Apply* y luego en OK.

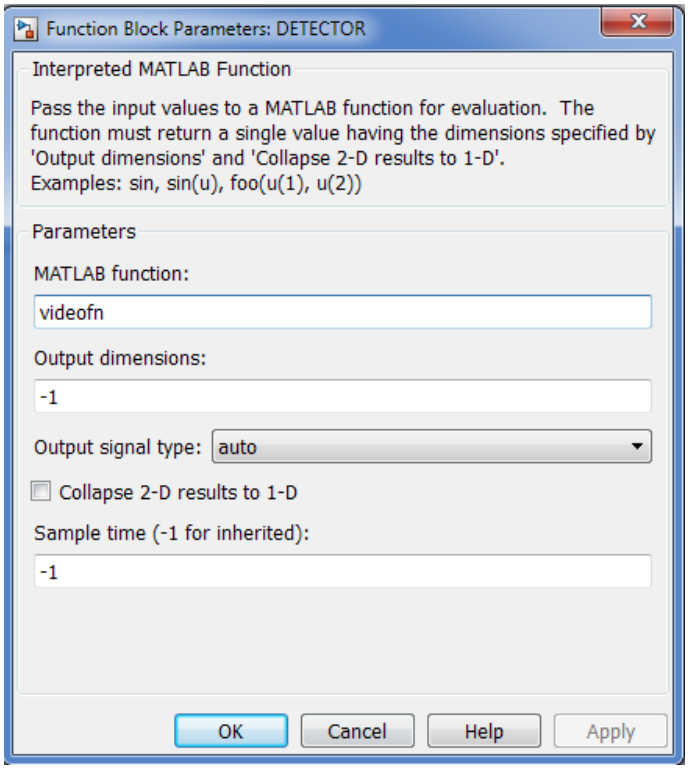


Figura 4.19: Propiedades Block Interpreted MATLAB Function.
Fuente: Simulink®.

El archivo de nombre *videofn.m* debe de ser programado en el Matlab Editor para ello debemos de declarar una función ya que el block debe de tener una entrada y una salida, en este caso la primera línea de comando colocamos el comando *function* entre paréntesis [y] que es la salida de nuestra función y este va a ser igual a *videofn(x)* donde x es la entrada del block como vemos en la línea de comando 1.

```
1    function [y]=videofn(x)
```

Para las siguientes líneas colocamos el comando **vision.CascadeObjectDetector ( )** en los paréntesis colocamos el clasificador de vehículos rastreador5.xml y a continuación las líneas de comando que permite modificar las propiedades del clasificador.

```
2 -      detector = vision.CascadeObjectDetector('rastreador5.xml');
3 -      detector.MinSize=[30 30];
4 -      detector.MaxSize=[75 75];
5 -      detector.MergeThreshold=2;
6 -      detector.ScaleFactor=1.2;
```

El siguiente procedimiento es realizar la detección de los vehículos en el video, para ello consideramos que la variable **x** de la función va a ser el video entonces creamos

una variable de nombre **bbox** la cual va a obtener las coordenadas de los vehículos detectados por el clasificador.

```
7
8 -     bbox=step(detector,x);
9
```

Las coordenadas que se obtiene al detectar un objeto van a representarse en forma de una matriz de 1X4 del siguiente modo [(x, y) alto y ancho] donde los dos primeros van a ser las coordenadas en (x, y) de la parte superior izquierda del objeto a detectar y las dos siguiente es el alto y el ancho del cuadrado delimitador del objeto detectado.

Cuando el clasificador no detecta vehículos esta matriz no contiene numero alguno, va a ser representada como una matriz vacía **bbox**=[ ] por lo cual genera un error al momento de correr el programa, para solucionar este inconveniente utilizamos un operador lógico ***if*** el cual nos permita decir que si la matriz **bbox** es diferente que cero la salida [y] de nuestra función va a ser las coordenadas dadas en **bbox** de la siguiente manera y=[bbox(1,1) bbox(1,2) bbox(1,3) bbox(1,4)], y si no se da esta condición el valor de [y] va a ser igual a y = [0 0 0 0], no debemos de olvidarnos que después de utilizar el operador lógico ***if*** debemos de cerrar la condición con end.

```
9
10 -    if (size(bbox)~=0)
11
12 -        y=[bbox(1,1) bbox(1,2) bbox(1,3) bbox(1,4)];
13 -    else
14 -        y=[0 0 0 0];
15
16 -    end
```

Esta condición nos permite eliminar el error cuando el clasificador no encuentra algún vehículo, además cuando el clasificador reconoce dos o más vehículos en un mismo instante la matriz va a ser de mayor filas, pero la que se encuentra más cerca va a ser la que ocupe la primera fila de la matriz por ello es que elegimos la misma como observamos en la línea de comando 12, ya que siempre debemos de realizar el análisis con el vehículo más próximo al nuestro.

Una vez terminado la programación de la función debemos de guardar el archivo Figura 4.20, para ello es recomendable guardar el archivo con el mismo nombre de la función en este caso *videofn* el cual colocamos en la línea 1 del comando.

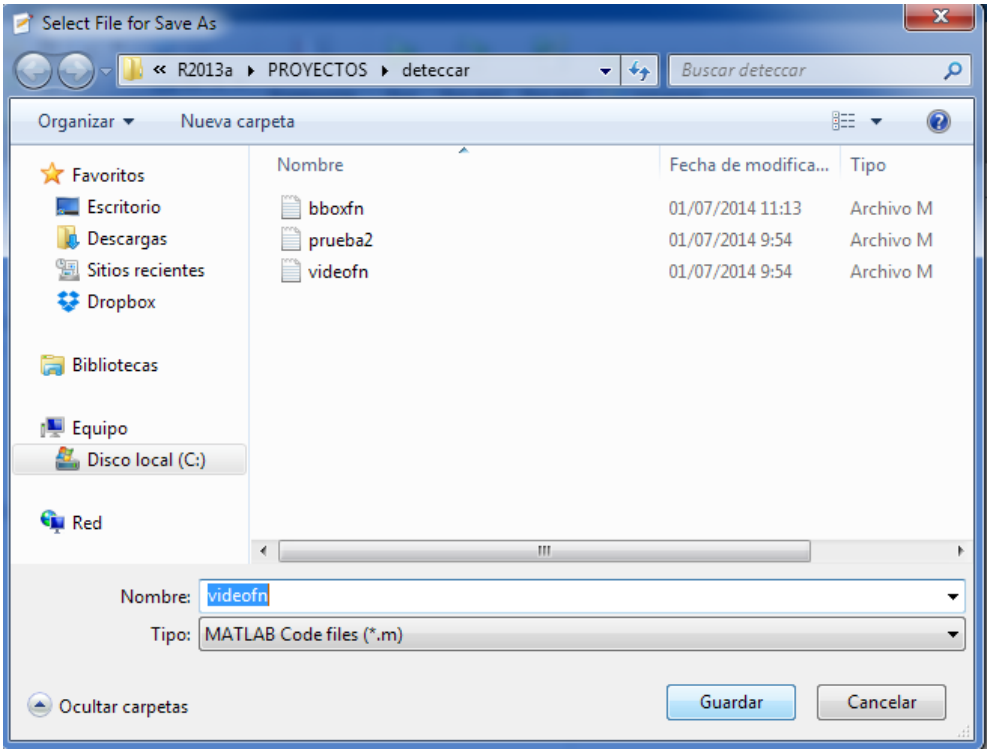


Figura 4.20: Guardar el Programa como función.
Fuente: Autores.

Ya con la programación echa en Matlab Editor es posible proseguir en Simulink® como la entrada del block interpreted MATLAB function es un video debemos de ocupar el block From Multimedia File que se encuentra en la librería "*Computer Vision System Toolbox*" en "*Sources*" y por último en "*From Multimedia File*" Figura 4.21.

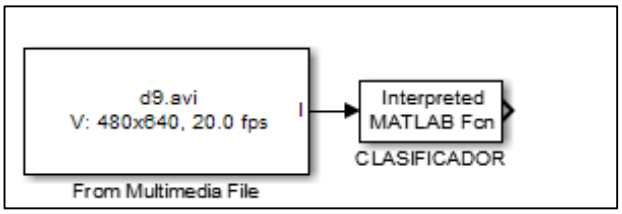


Figura 4.21: Conexión del Block From Multimedia File.
Fuente: Simulink®.

El bloque From Multimedia File permite trabajar con videos pregrabados, es recomendable utilizar el mismo cuando recién comenzamos a programar, damos doble clic en el block para cambiar sus propiedades además para seleccionar el video

con el que trabajaremos Figura 4.22 en la opción *File name* colocamos la dirección donde está el video que deseamos, después en la opción *Ouput color formal* elegimos la opción de *Intensity*, otra opción que debemos de cambiar es el tipo de dato con el que debemos de trabajar para eso damos click en la pestaña de nombre *Data Types* y elegimos el tipo de dato *doble*.

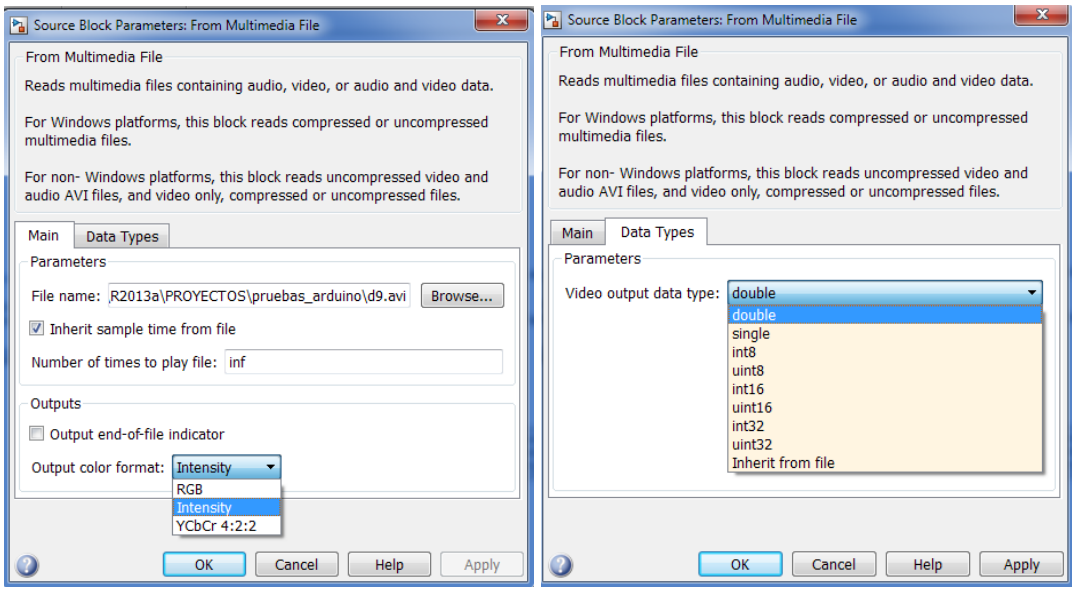


Figura 4.22: Propiedades Block From Multimedia File.
Fuente: Simulink®.

En caso de trabajar con una cámara en tiempo real utilizamos el bloque "*From Video Device*" que se encuentra en la librería "*ImagenActisition Toolbox*" Figura 4.23.

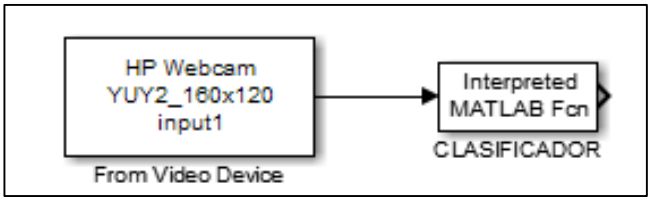


Figura 4.23: Conexión del Block From Video Device.
Fuente: Autores.

Del mismo modo que con el block From Multimedia File debemos que cambiar las propiedades dando doble click sobre el block en este caso los parámetros que cambiamos es el *Device*: el cual permite elegir la cámara con la que deseamos trabajar que en nuestro caso elegimos *winvideo 2*, el formato del video le cambiamos en el parámetro *Video format* escogemos YUY2_160x120, el color de salida debe de ser *intensity* o *grayscale* esto lo cambiamos en el parámetro Output color space, por

último cambiamos el tipo de dato en el parámetro *Data type* de single a *double* como vemos en la Figura 4.24.

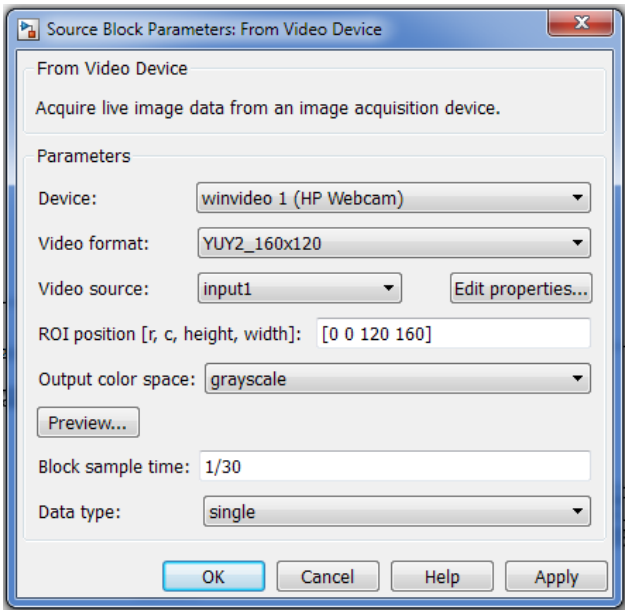


Figura 4.24: Propiedades Block From Video Device.
Fuente: Simulink®.

Después de que el video sea analizado por el clasificador del block *interpreted MATLAB function*, usamos el block *Draw Shapes* que encontramos en la librería de "*Computer Vision System Toolbox"* en "*Text & Graphics*" y por último el "*Draw Markers"* para enmarcar el objeto que el clasificador ha detectado Figura 4.25

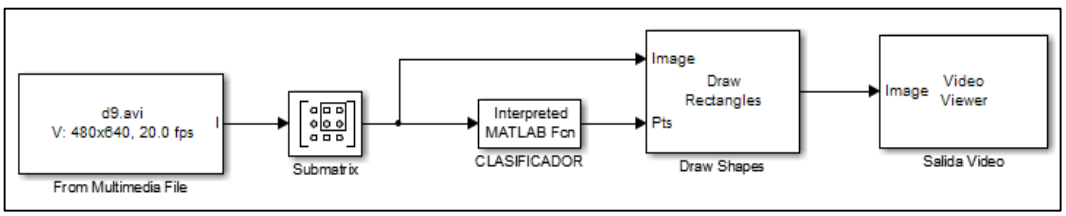


Figura 4.25: Diagrama de Detección de Vehículos.
Fuente: Autores.

Para la determinación de la distancia entre vehículos utilizamos el block *Interpreted MATLAB Fuction* la programación va a ser la siguiente.

```
function [W]=drangosfn(x)

            W=0;

  if (x~=0)

     switch x

     case {47, 48, 49,50, 51, 52, 53, 54, 55, 56, 57, 58,59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73}

            W=2.5; %8-15

     case {30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46}

            W=4.5; %16-22

     end

  end
```

Este programa va a permitir enviar una señal de 2.5 cuando el sistema detecte un vehículo a una distancia entre 8-15 m y cuando este entre 16-20m envía una señal de 4.5, con esta programación obtendremos las señales para compararlas con la velocidad que ingresa.

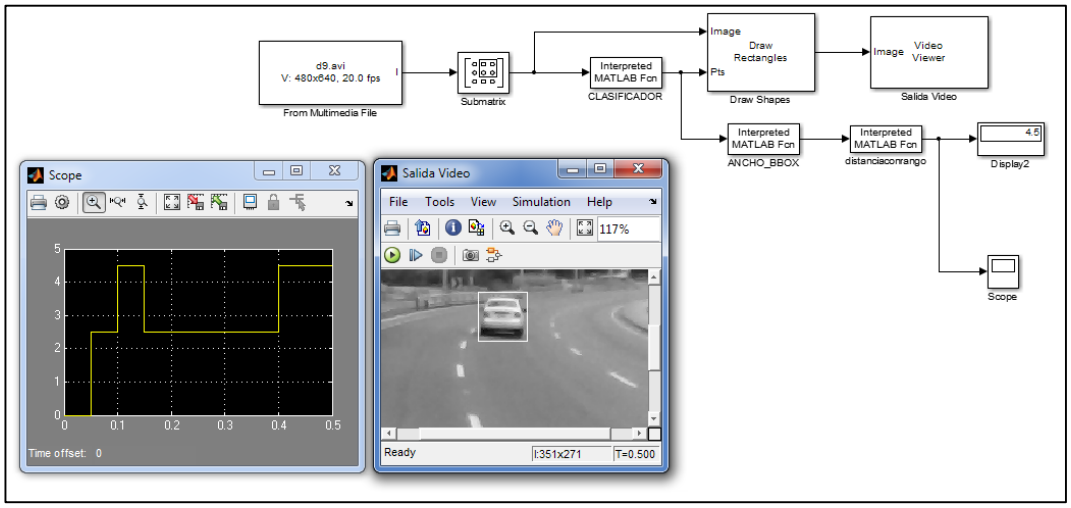


Figura 4.26: Diagrama de Detección de Vehículos con detector de Distancias.
Fuente: Autores.

El bloque ANCHO_BBOX Figura 4.27 es el que permite obtener solo el ancho del cuadro del vehículo detectado, la programación va de la siguiente manera.

```
function [y]=bboxfn(x)

    y=x(1,4);
```

4.2 DETECCIÓN DEL TRAZADO DE LA VIA.

Para la detección de curvas del trazado de la vía se utiliza la Transformada de Hough, que fue patentada por Paul Hough en los Estados Unidos de América, la transformada nos permitirá la detección de contornos cuyas formas básicas se pueden representar como una curva paramétrica, que pueden ser líneas, círculos, elipses, cónica, etc.[11]-[22].

Para la construcción de este algoritmo se utiliza diferentes tipos de bloques que se encuentran dentro de la librería de Simulink® en "*Computer Vision System Toolbox*":

Como se explica a continuación sobre cómo se obtiene la transformada de Hough.

4.2.1 TRANSFERENCIA DE HOUGH PARA DETECCIÓN DE LÍNEAS.

La transformada de Hough es muy utilizada para el sistema de visión artificial, el método se ha convertido en el más utilizado para la detección de líneas rectas y curvas, y nos permitirá la detección de la calzada de la vía[23]. Para ello se utiliza la ecuación de la recta misma hay que parametrizar o convertirla en polar una vez que la obtengamos.

$$y = mx + b \qquad (1)$$

Dónde: ***m*** nos representa la pendiente y ***b*** el punto del eje *y* donde la línea lo intercepta.

4.2.1.1 El espacio de parámetro.

Las líneas Lp que pasa por el punto $p_0 = (x_0, y_0)$, tiene la siguiente ecuación.

$$L_p: y_0 = Kx_0 + d \quad (2)$$

Los valores **K** y **d** son variados para trazar todas las posibles líneas que tienen en común x_0 e y_0. El conjunto de soluciones para **K** y **d**, son la cantidad infinita de rectas que pasa por el punto $\mathbf{x_0}$ e $\mathbf{y_0}$.

$$d = y_0 - kx_0 \quad (3)$$

Esta ecuación describe los parámetros de todas las posibles rectas $\mathbf{L_p}$ las cuales pasan a través del punto $\mathbf{p_0 = (x_0, y_0)}$.

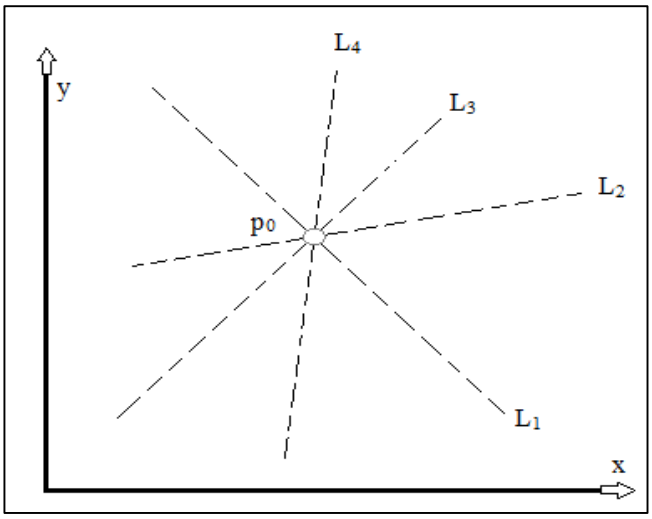


Figura 4.27: Conjunto de rectas que pasan por el punto p_0 en común.
Fuente: [11].

Para un determinado píxel de la imagen $\mathbf{p_i = (x_i, y_i)}$, corresponde a un conjunto de rectas definidas por:

$$R_i: d = y_i - \ kx_i \qquad (4)$$

Los parámetros variables son **K** y **d** como también lo llaman espacio de Hough, y $\mathbf{x_i}$ y $\mathbf{y_i}$ son los parámetros fijos llamados también el espacio de parámetros de la imagen. Cada punto p_i en el espacio de parámetros de la imagen corresponde a una recta en el espacio de parámetros de Hough. Los valores **K** y **d** que representa la recta del espacio de la imagen que pasa por los puntos que formularon las rectas en el espacio de parámetros de Hough como se puede observar en la Figura 4.27. Las rectas $\mathbf{R_1}$ y $\mathbf{R_2}$ se cortan en el punto $\mathbf{q = (k_{12}, d_{12})}$ en el espacio de parámetros de Hough, que representan a los puntos p_1 y p_2 en los espacios de los parámetros de la imagen. Si en los parámetros de Hough se cortan más rectas en el espacio que se corten significa que las líneas en el espacio de la imagen estará formada por ese número de puntos [11].

Si NR es el número de rectas que se interceptan en (K,d) del espacio de parámetros de Hough, entonces habrá NR puntos que se encuentran sobre la línea definida por y = kx + d en el espacio de la imagen [11].

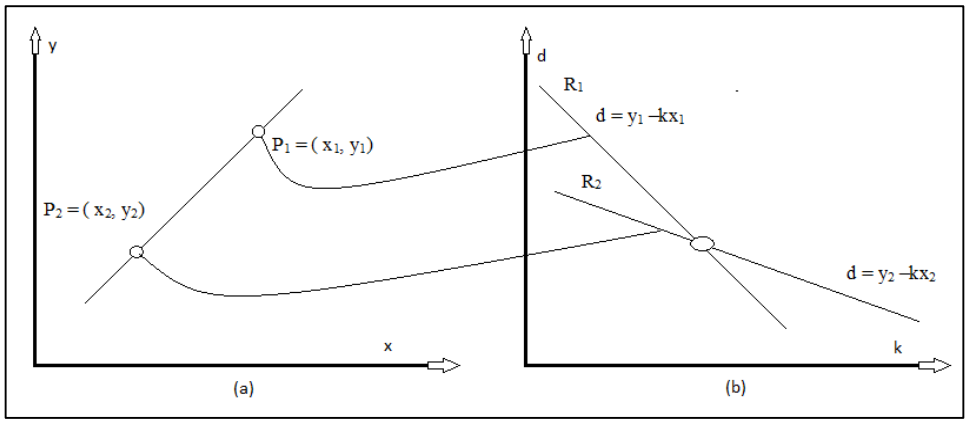


Figura 4.28: (a) Espacio de Parámetro de la Imagen y (b) Espacio de Parámetro de Hough.
Fuente: [11].

4.2.1.2 Transformada de Hough o Paramétrico.

La recta de la ecuación (1) no se puede utilizar ya que tiene un error computacional ocasionado en las líneas verticales donde **k = ∞.** Para ello tenemos la ecuación[11].

$$xcos(\theta) + ysen(\theta) = r \qquad (5)$$

De acuerdo a esta ecuación podemos obtener tanto el ángulo y el radio (Θ,r).

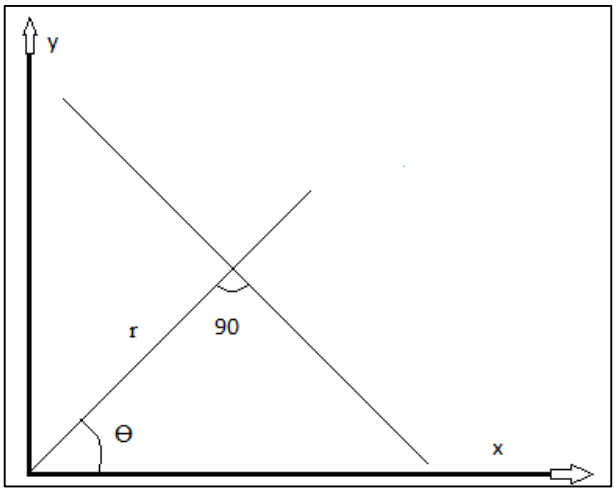


Figura 4.29: Representación de la recta y de Θ, r.
Fuente: [23].

4.2.2 PROGRAMACIÓN GRAFICA MEDIANTE BLOQUES EN SIMULINK.

En el Capítulo 3 se realizó la adquisición, procesamiento y la segmentación, continuando con ello tenemos la detección de la línea en el trazado de la vía, con la utilización de la Transformada de Hough.

Se puede observar a continuación los diferentes bloques que se utilizaron para el algoritmo de detección del trazado de la vía.

4.2.2.1 Hough Transform.

El bloque lo podemos encontrar en la librería de "*Computer Vison System Toolbox*" en "*Transforms*".

Con dicho bloque podemos encontrar las líneas dentro de una imagen mediante el cálculo de la matriz, con la siguiente ecuación.

$$rho = x*cos(theta) + y*sin(theta).$$

Este bloque recibe como entrada una imagen binaria que entra a BW que corresponden a los bordes de la imagen, y de salida se obtiene tres diferentes salidas, la matriz de registro Hough y dos vectores **Rho** y **Theta**. Donde **Rho** representa el radio en pixeles y **Theta** representa el ángulo de dicha línea en radianes ver Figura 4.30.

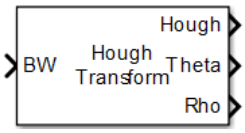


Figura 4.30: Bloque Hough Transform.
Fuente: Simulink®.

En la siguiente figura podemos cambiar tanto las resoluciones de **Rho** y **Theta**. El "*Output data type*" especifica los datos de señal de salida en este caso se escogió "*double*", ya que no se puede escoger de tipo "*single*" porque nos presenta un error dentro del algoritmo el mismo que se trata de un dato solo.

Todos los cálculos numéricos desempeñados por Matlab se realizan considerando que los números son del tipo double (o números de forma flotante), por lo que muchas de las operaciones realizadas sobre imagen tiene este formato [11].

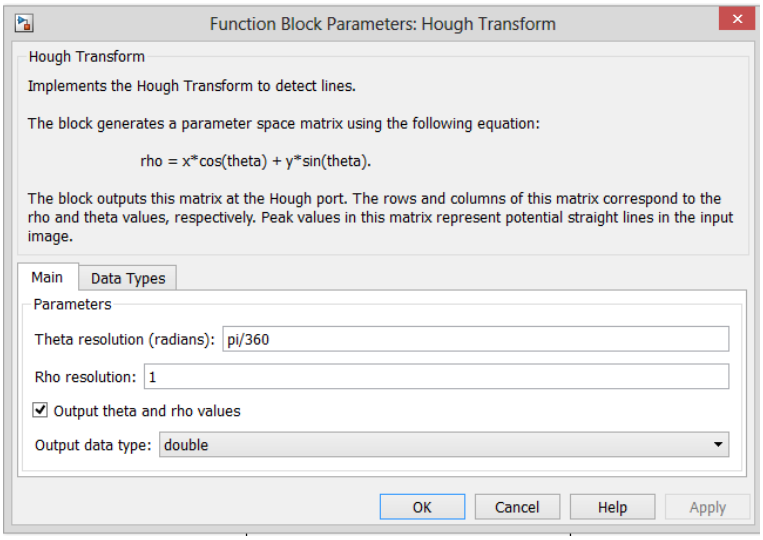


Figura 4.31: Cambio de parámetros en el bloque Hough Transform.
Fuente: Simulink®.

4.2.2.2 Find Local Maxima (Encontrar máximos locales).

Se lo puede encontrar en la librería de "*Computer Vison System Toolbox*" en "*Statistics*". Con dicho bloque podemos encontrar los máximos locales que se encuentran en una matriz de una determinada vecindad, ver Figura 4.32. Cuya entrada es una matriz Hough. La cual nos devuelve un **Idx** donde se encontraran los puntos máximos, en nuestro Algoritmo no se ocupara **Count** ya que solo nos señaliza el número máximos válidos.

Figura 4.32: Find Local Máxima.
Fuente: Simulink®.

Se realizó los cambios de los siguientes parámetros dentro de la venta de "*Find Local Máxima*", El número totales de máximos locales se dejó en 1 ya que si se ponía 2 o más el mismo nos representaba un erro de anchura de dimensiones del puerto de salida 1. La región de vecindad "*Nieghborhood size*" se lo dejo en lo que nos presenta la ventana en una matriz de [5 7] ya que nos representa el tamaño de la vecindad alrededor de los máximos. El umbral (*Threshold*) se lo dejo en 10. Se debe escoger el ingreso de la matriz Hough para los rangos de theta. Los tipos de datos de salida son *unit8* se presenta en el intervalo de [0,255], si la imagen es del tipo *double* los datos que la constituye son del tipo flotante y se encuentran en intervalos de [0,1].

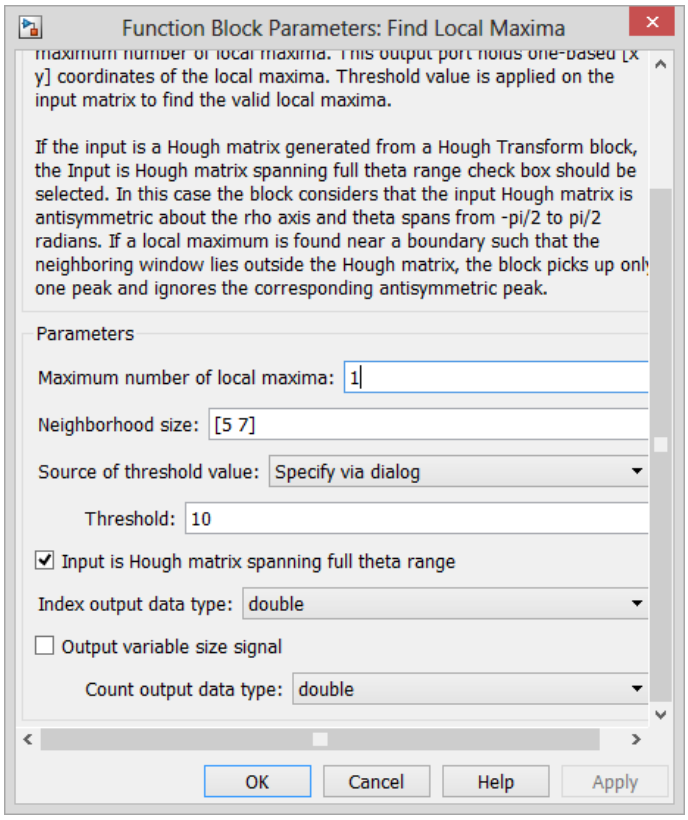


Figura 4.33: Cambio de parámetros en Find Local Maxima.
Fuente: Simulink®.

4.2.2.3 Selector.

El bloque lo podemos encontrar en la librería de Simulink® en "*Routing*", tiene la función que cuando ingresa una matriz se encarga de seleccionar o cambiar los elementos específicos de una señal de entrada multidimensional. En nuestro algoritmo se utilizó dos selectores para separar los índices de **Rho** y **Theta**, que son de salida del puerto **Idx** que se asocian con el valor máximo de la matriz de Hough.

Figura 4.34: Selector.
Fuente: Simulink®.

Los parámetros que se cambiaron son el "*Index mode*" que es el modo del índice que se escogió de base Zero porque estamos utilizando un Índice de vector de cero. El tamaño de puerto de entrada es de dos ya que es una matriz de (2*2).

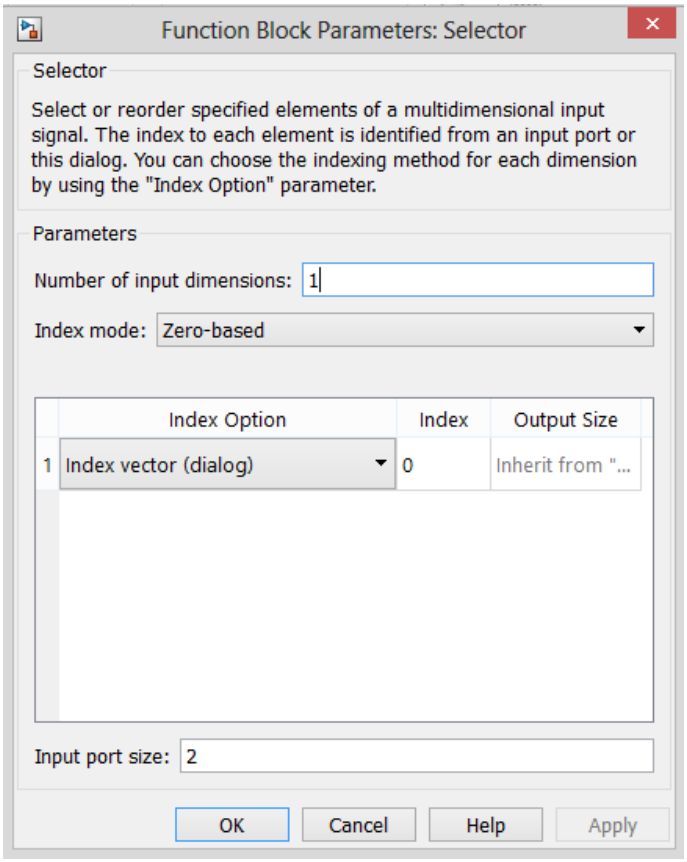


Figura 4.35: Fuction Block Parameters: Selector.
Fuente: Simulink®.

4.2.2.4 Variable Selector (Selector de Variables).

Se la puede obtener en la librería de DSP "*System Toolbox*" por consiguiente en el "*Signal Management*" la cual nos lleva a la siguiente librería "*Indexing*" donde se encuentra el bloque "*Variable Selector*".

Nos permite extraer un renglón o una columna o un conjunto de renglones como de columnas de una matriz. Y si ingresa una señal también nos devuelve una señal de salida que representa una matriz.

Como tenemos dos salidas de la transformada de Hough es necesario de la utilización de dos "Variable Selector" pero en este caso que sean de "*Columns*" (columnas), ya que en nuestro algoritmo solo se quiere extraer columnas tanto de *Rho* como de *Theta*. Donde *Idx* es la entrada de señal de los dos selectores y *ln1* en la entrada de señal para él un selector *Theta* y para el otro *Rho*.

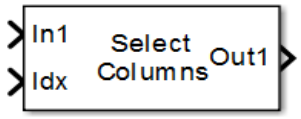


Figura 4.36: Variable Selector.
Fuente: Simulink®.

En la Figura 4.37 se puede observar los parámetros que se modificaron. "*Number of imput signals*" (Número de entras de señal) ya que solo necesitamos una, en "*Select*" seleccionamos solo columnas, el "*Selector mode*" nos da la opción de que sea fijo (Fixed) o (Variable) se selección variable ya que nos permite utilizar otra variable y aparecerá una entrada del bloque *Idx***.** El "*Index Mode*" en base Zero. El "*Ivalid Index*" especifica cómo el bloque maneja un valor de índice válido, nosotros escogimos la siguiente opción "*Index Clip*" para el valor del índice válido más cercano y no emitir una alerta. En "*Fill empy in outputs (For logical indexing*)" que significa llenar los espacios vacíos en las salidas (para la indexación lógica), cuando los elementos de la indexación del vector son del tipo de datos *Boolean*, el bloque realiza la indexación lógica. Esto puede causar espacios vacíos en la salida. Seleccione este parámetro para designar valores que se agregan a la salida en el parámetro de los valores de relleno.

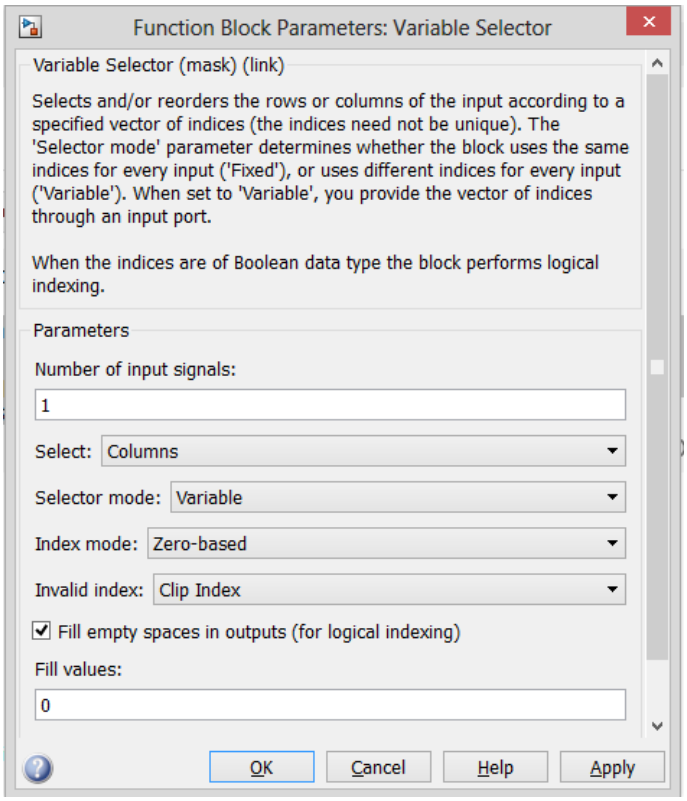


Figura 4.37: Fuction Block Parameters: Variable selector.
Fuente: Simulink®.

4.2.2.5 Hough Line (Obtención de Línea de Hough).

Lo podemos encontrar en la librería "*Computer Vision System Toolbox*" en "*Transform*", el bloque localiza las líneas que describe la parametrización de **Theta** y **Rho,** En el cual ingresan los valores **Theta** y **Rho**, define los vectores que se mapearon de la ecuación (5), la cual también entra la imagen que sale de la *Submatrix* a la **Ref** 1, de la cual solo son importantes las dimensiones, se calculan las coordenadas de las líneas definidas. La salida de este bloque es una matriz cuyos índices empiezan en (0,0) en las coordenadas de la intersección de las líneas con las fronteras de la imagen de referencia [11].

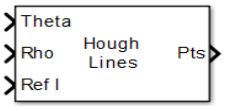


Figura 4.38: Hough Linea.
Fuente: Simulink®.

Los parámetros que se modificaron fue el "*Sine value computation method*" Método del valor sine, el cual nos permite escoger entre tablas o funciones trigonométricas,

en nuestro caso se optó por utilizar las funciones trigonométricas para realizar los cálculos trigonométricos.

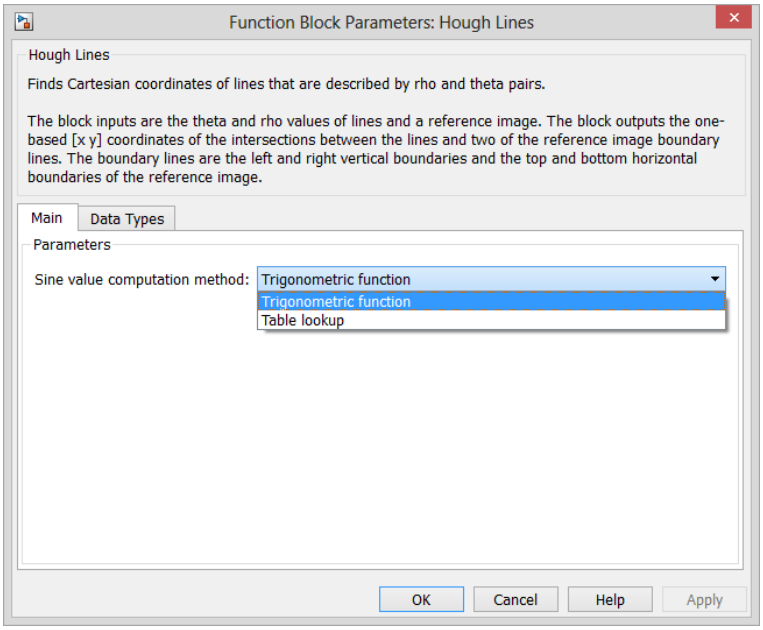


Figura 4.39 : Fuction Block Parameters: Hough Lines.
Fuente: Simulink®.

4.2.2.6 Draw Shapes (Dibujar Figuras).

El bloque se encuentra en la librería "*Computer Vision System Toolbox*" en "*Text & Graphics*". Se encarga de dibujar figuras como, rectángulos, líneas, polígonos y círculos. De la que se recibe como salida la línea en la imagen segmentada en un bloque de *Video Viewer*.

Entra una matriz de valores de intensidad a la Imagen, y **Pts** que utiliza valores enteros para definir las coordenadas de la forma de base uno, si se utiliza valores no enteros, el bloque se redondea al entero más cercano. Y recibe la señal de salida de *Hough Lines*.

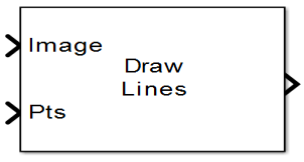


Figura 4.40: Draw Shapes.
Fuente: Simulink®.

Los parámetros a cambiar dentro del "*Draw Shapes*" la forma esta quedarían en líneas que es lo que se necesita para nuestro algoritmo, la fuente de color del borde

queda en "*Specify via dialog*" ya que este nos permite obtener el color de borde que se dejaría en negro y una señal multidimensional.

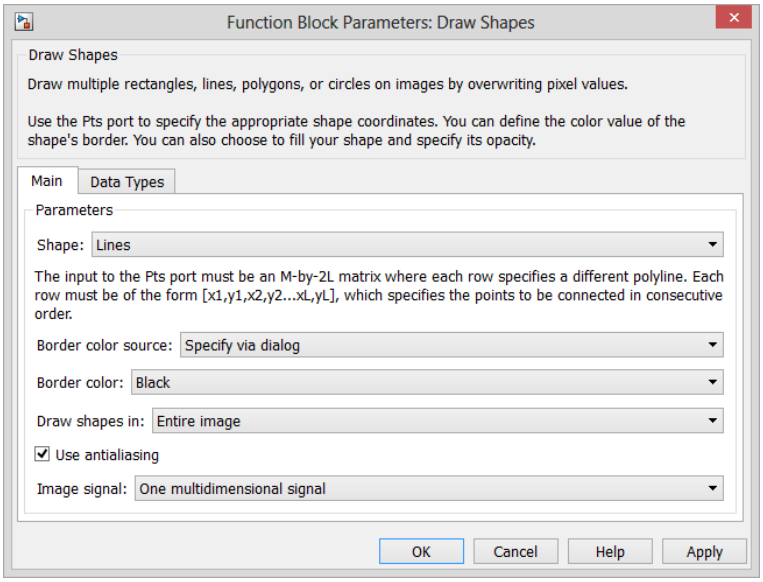


Figura 4.41: Fuction Block Parameters: Draw Shapes.
Fuente: Simulink®.

Diagrama de bloques del algoritmo para la detección de líneas del trazado de la vía, el mismo que es utilizado tanto par le lado derecho como para el lado izquierdo, se presenta en la Figura 4.42.

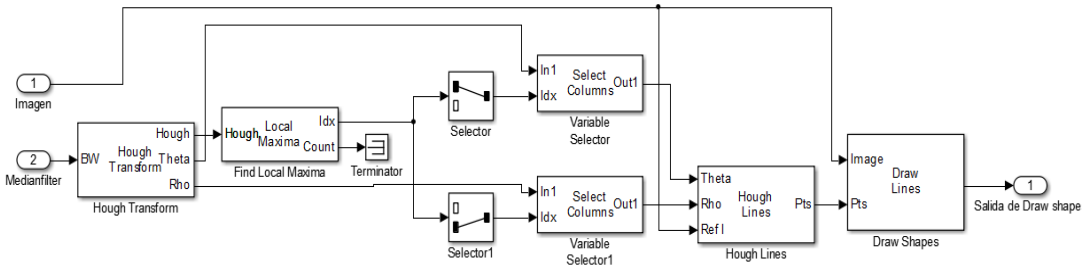


Figura 4.42: Diagrama de bloques del Algoritmo de Control.
Fuente: Autores.

En la Figura 4.43 se puede observar la detección de las líneas mediante la utilización de la transformada de Hough.

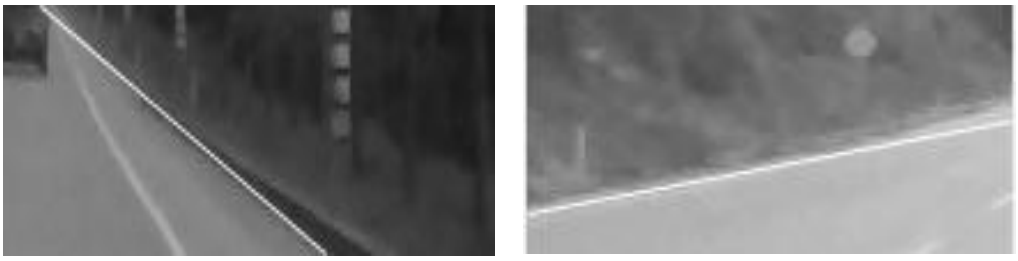


Figura 4.43: Detección de líneas de la parte Derecha e Izquierda.
Fuente: Autores.

4.2.3 CONDICIONES DE THETA.

En la Figura 4.44 nos presenta el vector Theta y Rho donde Theta es el ángulo de la recta y Rho el radio. Ya que desde el bloque de "*Hough Transform*" nos presenta los dos vectores que se dirigen a la "Variable Selector" para escoger solo los valores que nos presenta en columnas de la salida de estos valores nos presenta el vector Theta en radianes y Rho en pixeles.

En nuestro caso solo se utiliza el vector Theta para adquirir los valores de la inclinación de la recta cuando se presenta de aproximación a la curva, para ello se utilizó una transformación de radianes a grados como se observa en la Figura 4.44, esta transformada se utiliza tanto para el lado derecho como para el izquierdo.

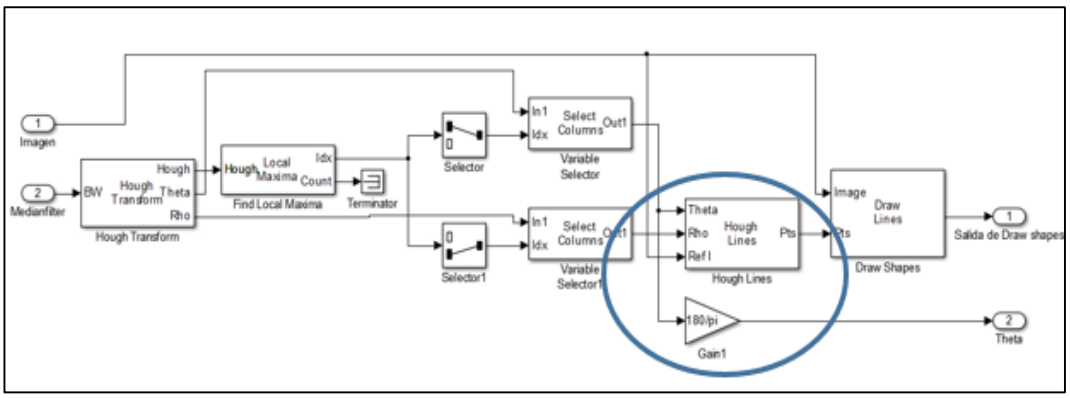


Figura 4.44: Transformar de Radianes a Grados.
Fuente: Autores.

4.2.3.1 Determinación de Valores de la Recta Cuando el Vehículo Ingresa a la Curva Mediante la Webcam Derecha.

En la Figura 4.45 se puede observar los pasos para determinar el trazado de la vía, en nuestro caso la inclinación de la recta. Para ello se utiliza el filtro pasa bajo que se lo puede observar en (1) ya que el mismo nos permite obtener valores menores a los que está ingresando de la conversión de radianes a grados. Luego se utiliza una

ecuación de primer grado (2) la misma nos permite obtener valores positivos. De ahí se determinó los valores cuando el vehículo se aproxima a la curva como se observa en (3) que nos da un valor de 1 a 14 ya que en 14 el vehículo se aproxima a una curva y mientras va reduciendo su valor nos determina que el vehículo sigue estando en la curva y el conductor debe reducir su velocidad hasta la permitida por la ley de tránsito en circulación en curva, estos datos funciona cuando la línea se encuentra en el lado derecho como se puede observar en la Figura 4.46.

Cuando la línea se encuentra al lado izquierdo tenemos diferentes valores que están entre 360 a 160 como se observa en (4) de la Figura 4.47 Estos valores tanto del (3) como del (5) ingresan a un operador lógico del tipo AND que nos permite trabajar en esos rangos de valores y nos da 1 cuando está en ese rango y 0 cuando no se encuentra en el mismo y por último se utiliza la compuerta OR en (6) que trabaja con los datos que nos da las dos compuertas AND y cuando la condición sea ceros en ambos lados el resultado será cero, si cualesquiera de las condiciones es 1 el resultado será 1 y si ambos son 1 el resultado también será 1, este valor será que nos indique que está ingresando a una curva, estos resultados se observa en la Figura 4.46.

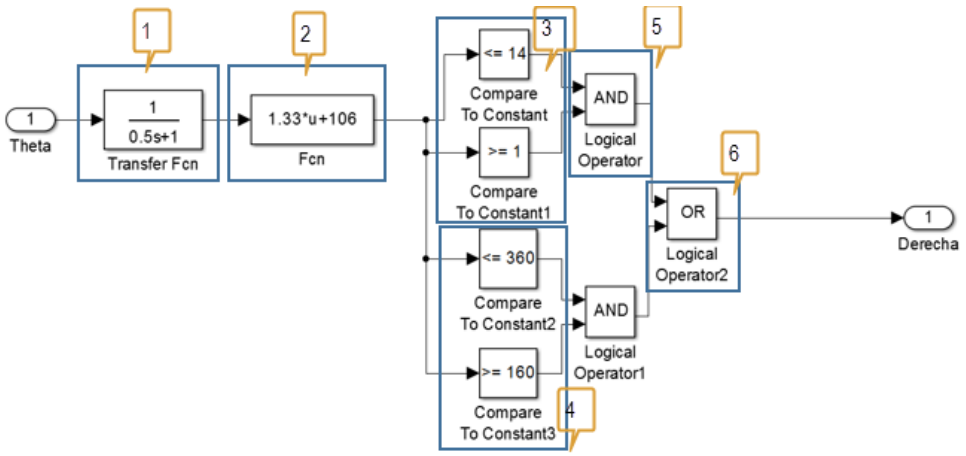


Figura 4.45: Condiciones para la Señalización de Curvas del Lado Derecho.
Fuente: Autores.

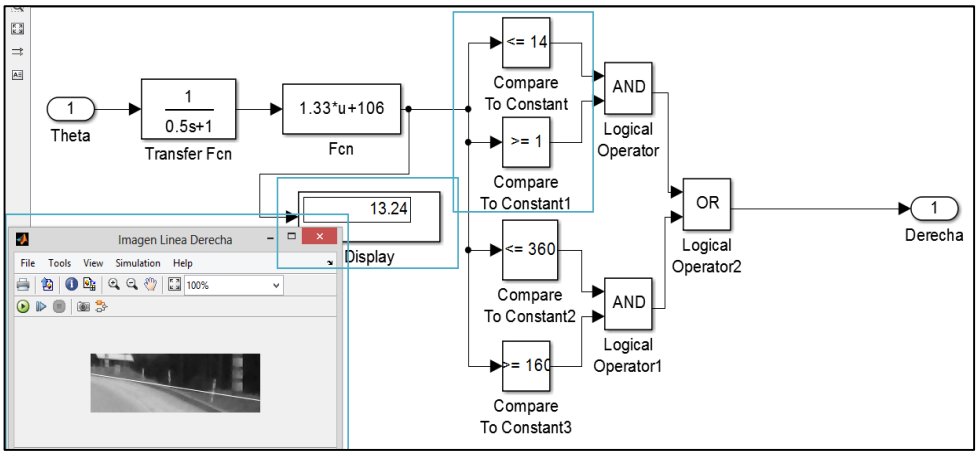


Figura 4.46: Valores del lado derecho de la vía utilizando la Cámara Derecha.
Fuente: Autores.

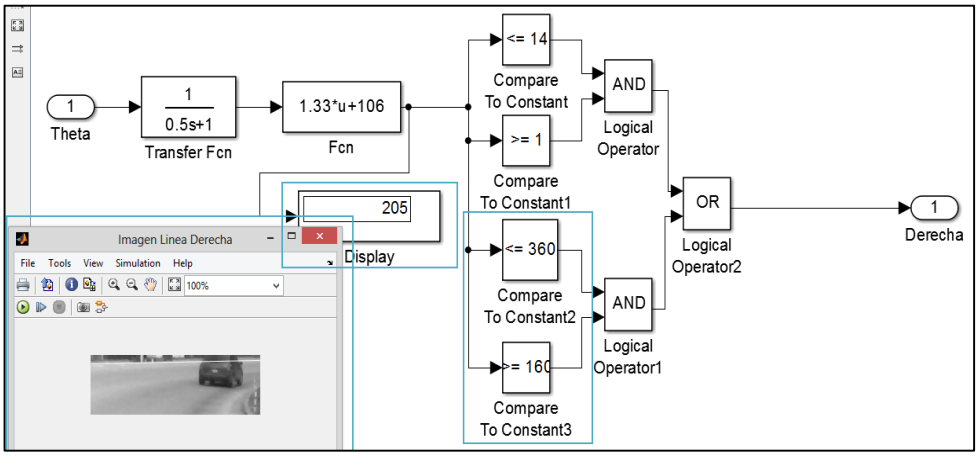


Figura 4.47: Valores del lado Izquierdo de la vía Utilizando la Cámara Derecha.
Fuente: Autores.

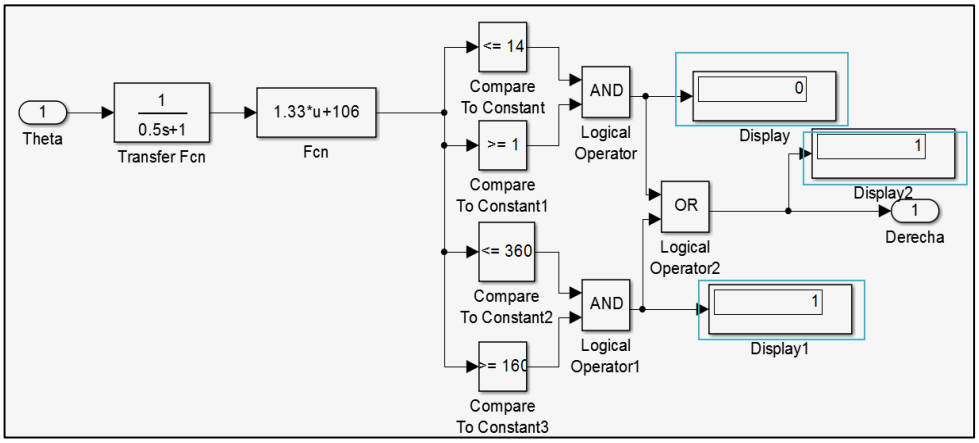


Figura 4.48: Compuerta OR y sus Valores Cámara Derecha.
Fuente: Autores.

4.2.3.2 Determinación de Valores de la Recta Cuando el Vehículo Ingresa a la Curva Mediante la Cámara Izquierda.

En la Figura 4.49 lo único que cambia son los valores del (3) y (4) ya que se encuentran en el lado izquierdo y sus valores cambian para el (3) se encuentra en 10 a 1 y para el (4) de 200 a 300. Y los demás son igual a lo explicado en el literal anterior.

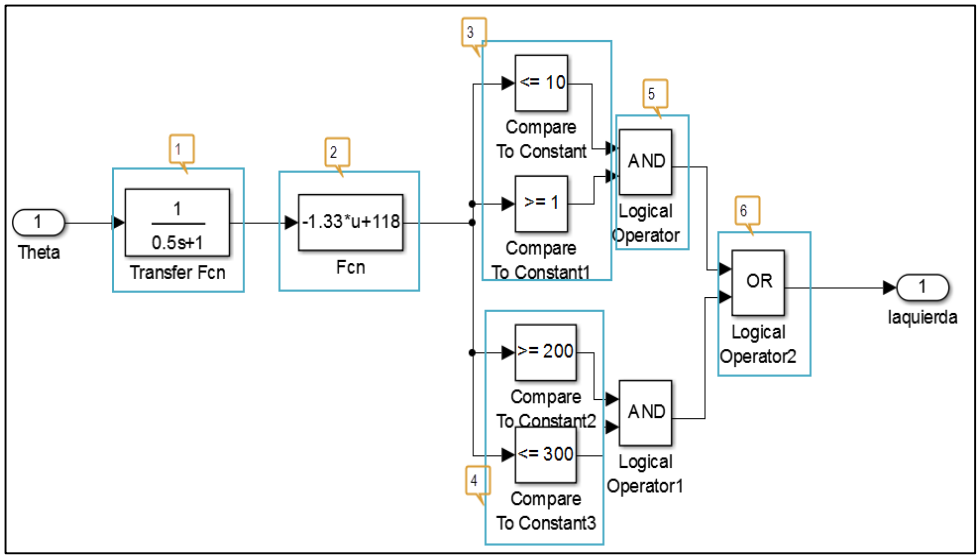


Figura 4.49: Condiciones para la Señalización de Curvas del lado Izquierdo.
Fuente: Autores.

Los datos que se obtuvieron tanto de la parte derecha como de la parte izquierda de nuevo ingresan a un comparador OR y de ahí a un SCOPE donde nos presenta los valores de cero cuando no se encuentra en una curva y 1 cuando se aproxima a una curva como se observar en la Figura 4.51.

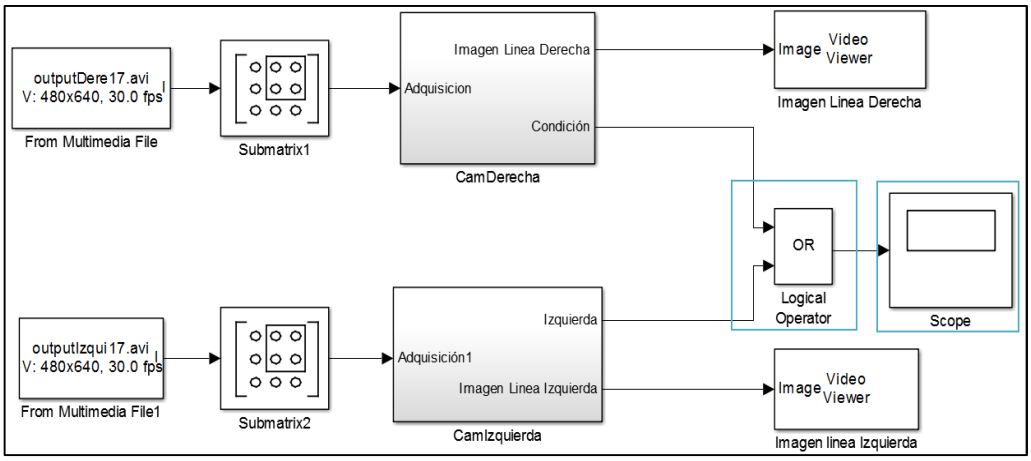


Figura 4.50: Utilización del comparador OR y el ESCOPE.
Fuente: Autores.

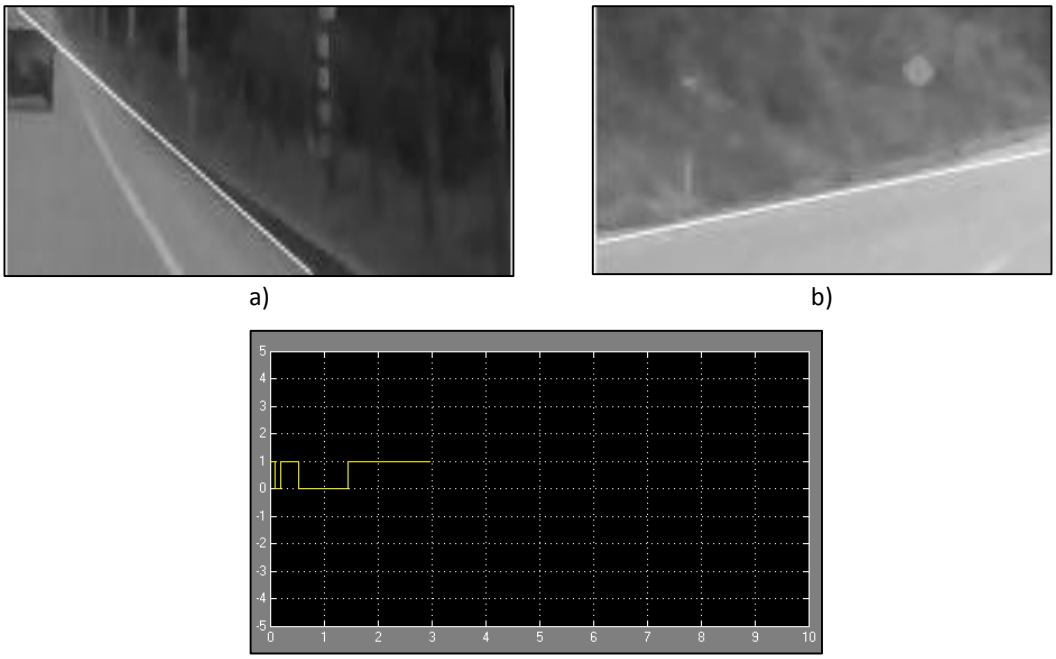


Figura 4.51: Utilización del comparador OR y el ESCOPE en (a) se puede observar la línea del lado Derecho, (b) la línea de la cámara Izquierda y en (c) se visualiza el ESCOPE donde nos presenta en cero cuando es una recta y uno cuando se aproxima a una curva.
Fuente: Autores.

4.3 ADAPTACIÓN DE VELOCIDAD.

La adaptación de velocidad del automóvil, toma en cuenta cada uno de los aspectos explicados anteriormente, como la detección de vehículos y el reconocimiento del trazado de la vía, esto en conjunto con la velocidad del automóvil, trabajará para dar un aviso al conductor para la adaptación de velocidad. En este caso se presentara cada uno de los aspectos a tomar, para el funcionamiento del algoritmo.

Al tener cada una de las señales como: la detección de vehículos en zona de advertencia y peligro, al reconocer una curva, la velocidad del vehículo en que está montado el prototipo y la señal de activación de freno. Todas las señales se entran al software Simulink® para proceder a la adaptación de velocidad, teniendo en cuenta que la captación de señales y la salida de señales de advertencia, peligro y aviso acústico se utilizó la tarjeta de procesamiento Arduino® Mega 2560 como se observa en la Figura 4.52.

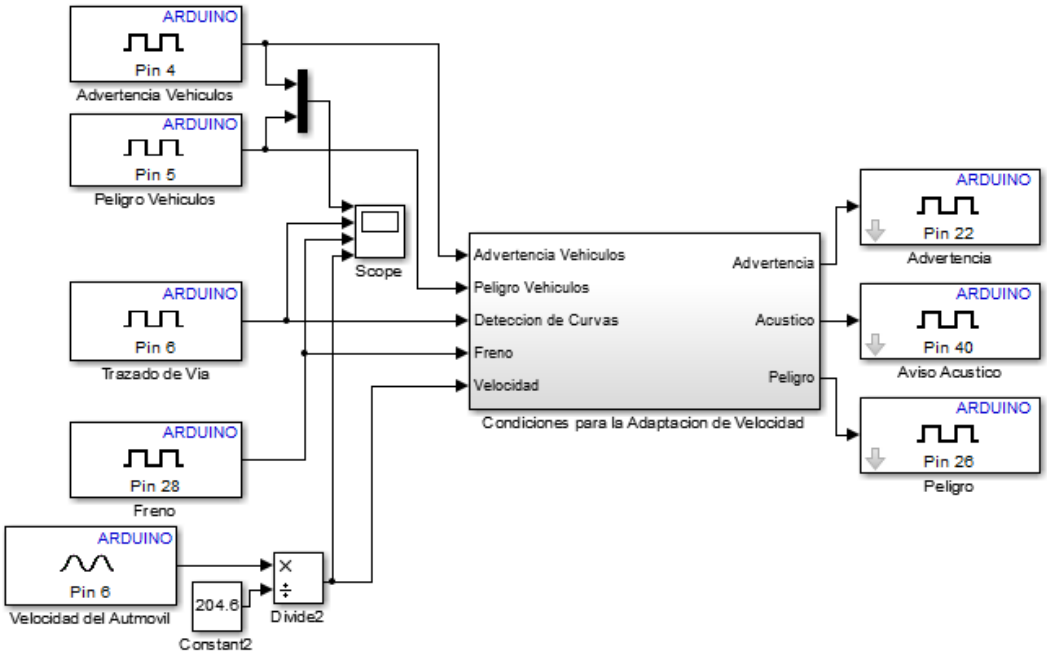


Figura 4.52: Programación en Simulink para la Adaptación de Velocidad.
Fuente: Autores.

A continuación veremos cada uno de los procesos y bloques ocupados para la adaptación de velocidad.

4.3.1 VELOCIDAD DEL VEHÍCULO.

Al ser obtenida la velocidad del vehículo en una señal analógica que varía de 0 a 5 voltios, como se explicara en el Capítulo 5. Esta señal es introducida al Simulink® por medio de la tarjeta de adquisición de datos Arduino® UNO debemos de recordar que esta señal ingresa en el valor de 1024 bits, entonces es necesario transformarla a 5 voltios dividiendo para 204.6, así conseguimos una señal analógica de 0 a 5 voltios. La señal de velocidad se estableció que 5 voltios este alrededor de 110km/h, esto se puede ver en el siguiente Capítulo 5.

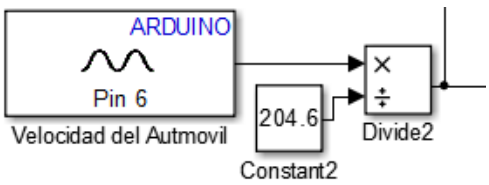


Figura 4.53: Transformación de Señal de Velocidad
Fuente: Autores.

Los valores establecidos de voltaje por medio de la velocidad del vehículo, se observan en la siguiente Tabla 4.4.

Tabla 4-4: Voltaje de Acuerdo a la Velocidad del Vehículo.
Fuente: Autores.

Velocidad (Km/h)	Voltaje (V)
20	0.8 – 0.9
25	1.13 – 1.15
30	1.3 – 1.4
35	1.47 – 1.55
40	1.8 – 1.9
45	2.0 -2.1
50	2.2 -2.4
55	2.4 -2.5
60	2.6 – 2.7
65	2.8 – 2.9
70	3.1 – 3.3
75	3.3 -3.4
80	3.6 – 3.7
85	3.85 – 3.9
90	4.0 – 4.2
95	4.3 – 4.4
100	4.4 – 4.5
105	4.8 – 4.9
110	5

La velocidad del vehículo es la encargada de activar el sistema y enviar las señales al conductor. La señal analógica de 5 voltios al entrar al bloque de condiciones de adaptación de velocidad, será la encargada de comandar la activación de los avisos lumínicos de advertencia, peligro y el aviso acústico, estos avisos serán los encargados de informar al conductor que debe disminuir la velocidad, como se observa en la Figura 4.54 que es el interior del bloque de condiciones para la adaptación de velocidad.

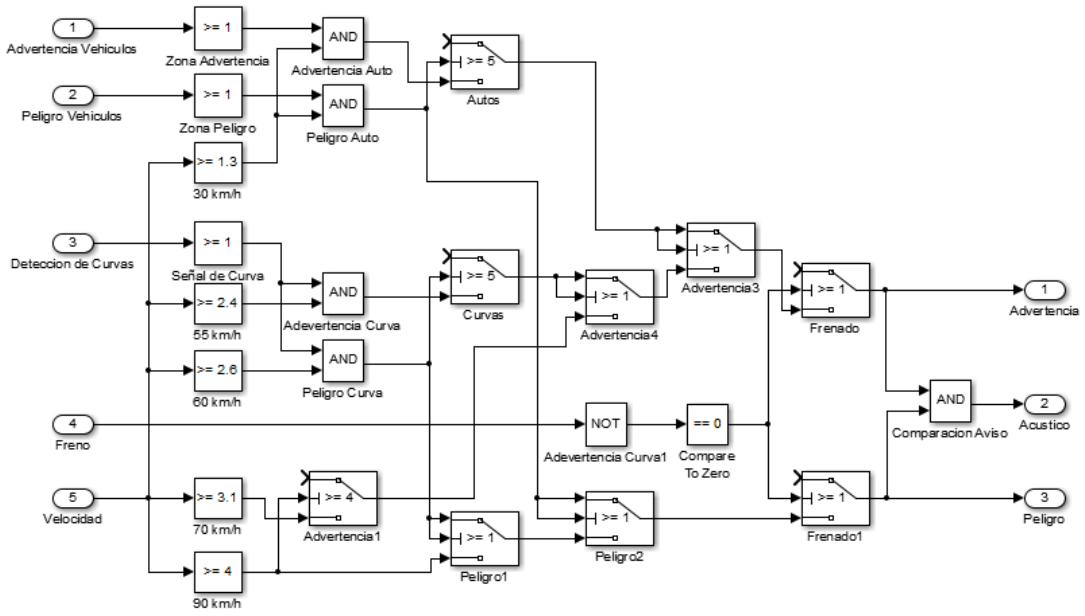


Figura 4.54: Condiciones para Adaptación de Velocidad
Fuente: Autores.

4.3.1.1 Adaptación de Velocidad con Proximidad entre Vehículos.

Al observar la parte de detección de vehículos, envía dos señales digitales, una cuando el vehículo detectado está dentro de una distancia de conducción de advertencia, y otra cuando el vehículo entra en una distancia de peligro.

Las señales de advertencia y peligro, dan el paso de señal mediante compuertas AND, si el vehículo, donde está montado el dispositivo, está a más de 30 km/h, esta velocidad se considera ya que al realizar los cálculos dinámicos en el Capítulo 2, se calcula la distancia de frenado a 30km/h es de 16m aproximadamente, considerando que esta distancia de frenado depende de la velocidad. Al cumplir con las condiciones para la detección de vehículos, los interruptores dejan pasar la señal primero de advertencia y luego peligro; dependiendo del cumplimiento de las condiciones para que se activen los interruptores, así se puede observar en las Figuras 4.55 y 4.56 los colores de amarillo para advertencia y rojo para peligro.

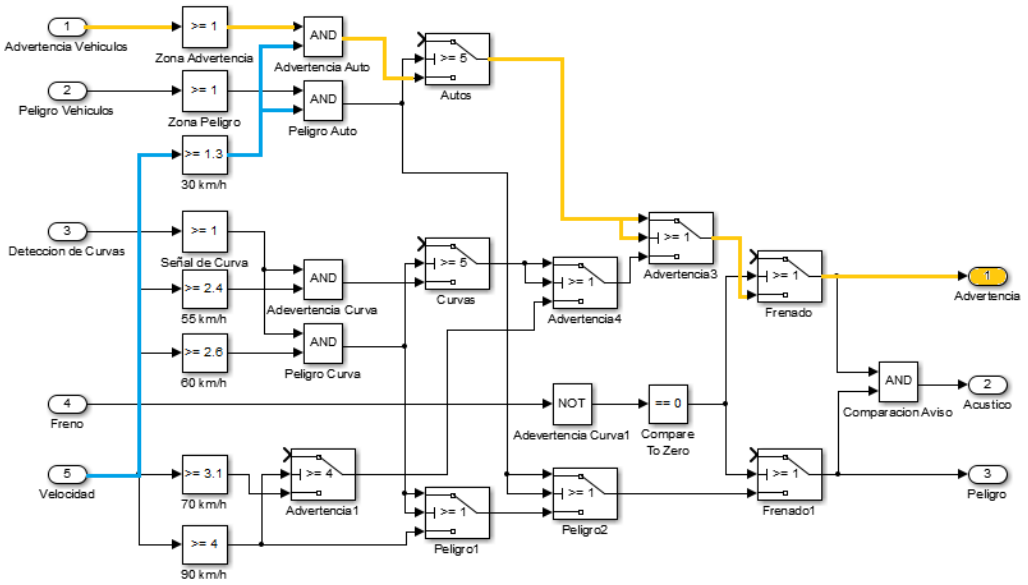


Figura 4.55: Zona Advertencia con Proximidad de Vehículos.
Fuente: Autores.

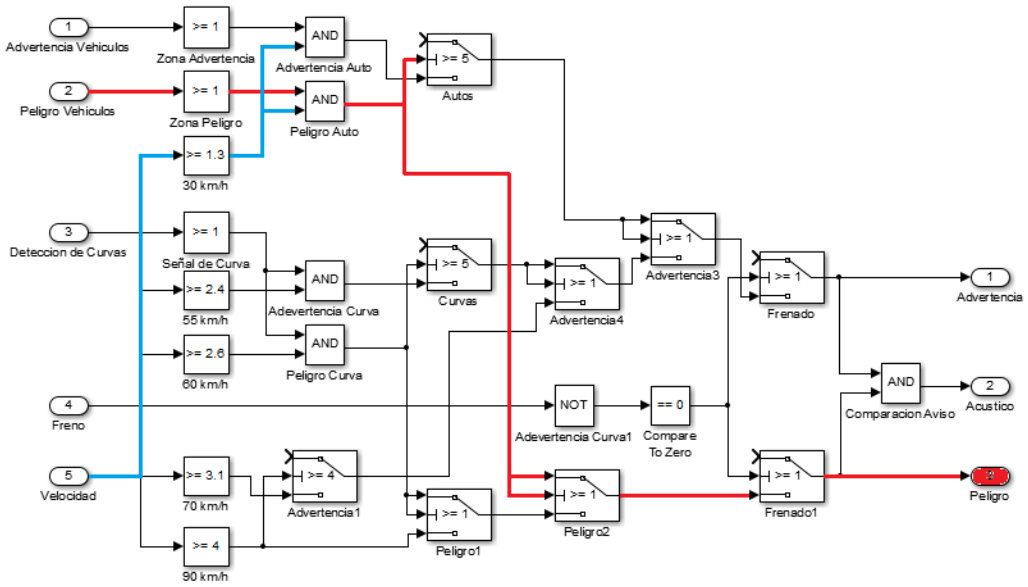


Figura 4.56: Zona Peligro con Proximidad de Vehículos.
Fuente: Autores.

4.3.1.2 Adaptación de Velocidad con el Trazado de la Vía.

El reconocimiento del trazado de la vía, es el encargado de detectar curvas en la vía, por consiguiente si detecta una curva mandará una señal digital de 1 y si es recta 0. Debemos de tener en cuenta para la circulación en curvas se establecen velocidades de derrape y vuelco como se vio en el Capítulo 2, al reconocer una curva y a una

velocidad de 55km/h envía una señal de advertencia, al llegar a 60km/h o más se enviara una señal de peligro.

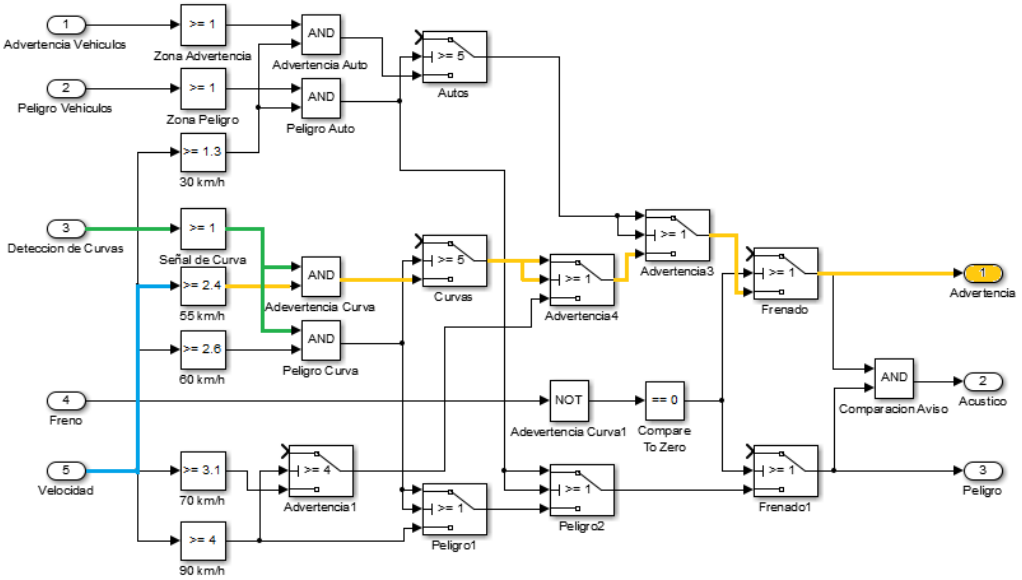


Figura 4.57: Advertencia con Detección de Curva.
Fuente: Autores.

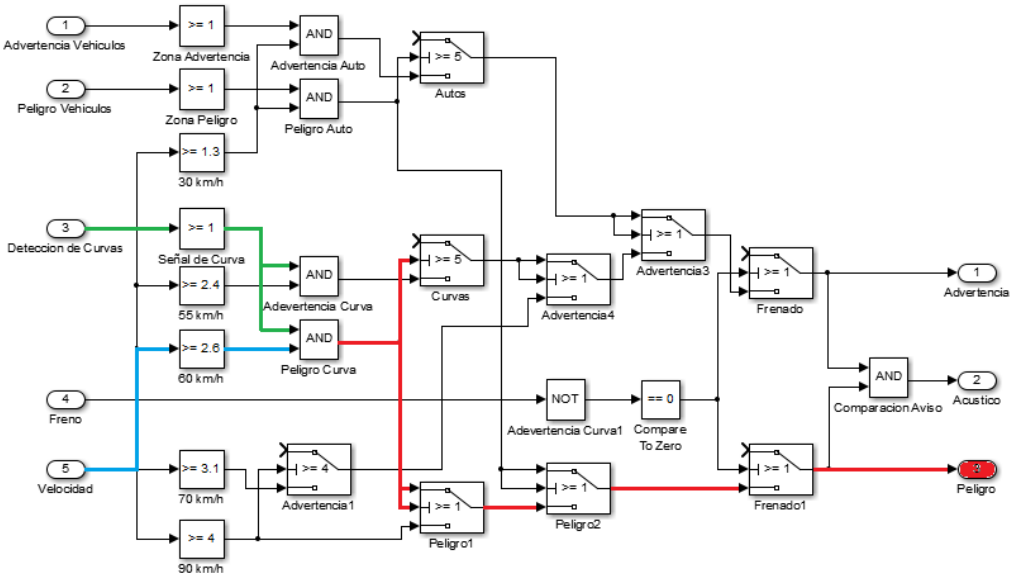


Figura 4.58: Peligro con Detección de Curva.
Fuente: Autores.

4.3.1.3 Adaptación de Velocidad con Límites Establecidos.

Los límites velocidad establecidos por la Ley de Transportes Terrestre, establece que la velocidad de 90km/h es para circular en vías perimetrales, considerando nuestro medio donde la mayoría de vías rápidas no hay como circular a más de 90km/h, por lo cual cuando llega a esta velocidad enviara un aviso de peligro, considerando que antes se enviara una señal de advertencia dando señal que está a una velocidad de 70km/h como observamos en las siguientes Figuras.

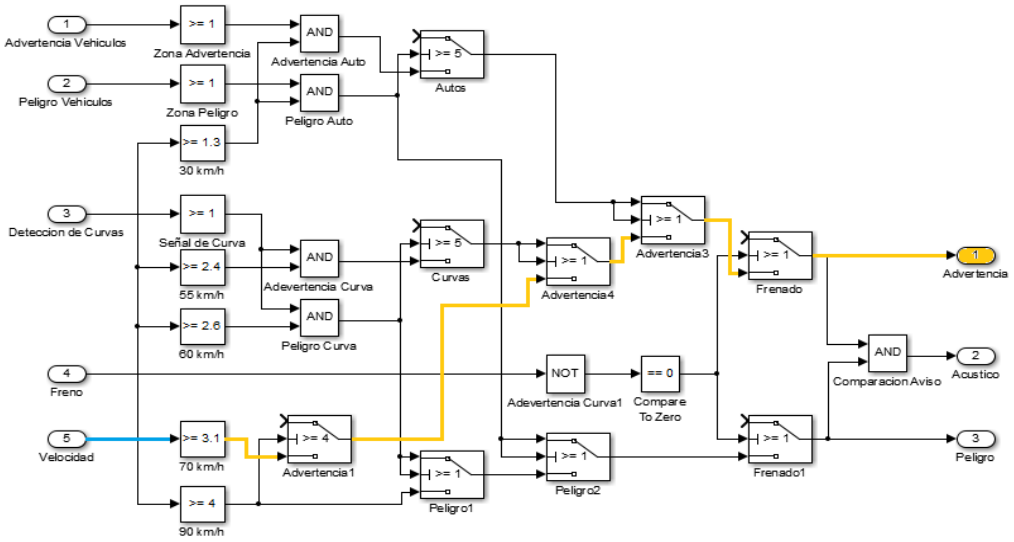


Figura 4.59: Advertencia con Velocidad de 70km/h.
Fuente: Autores.

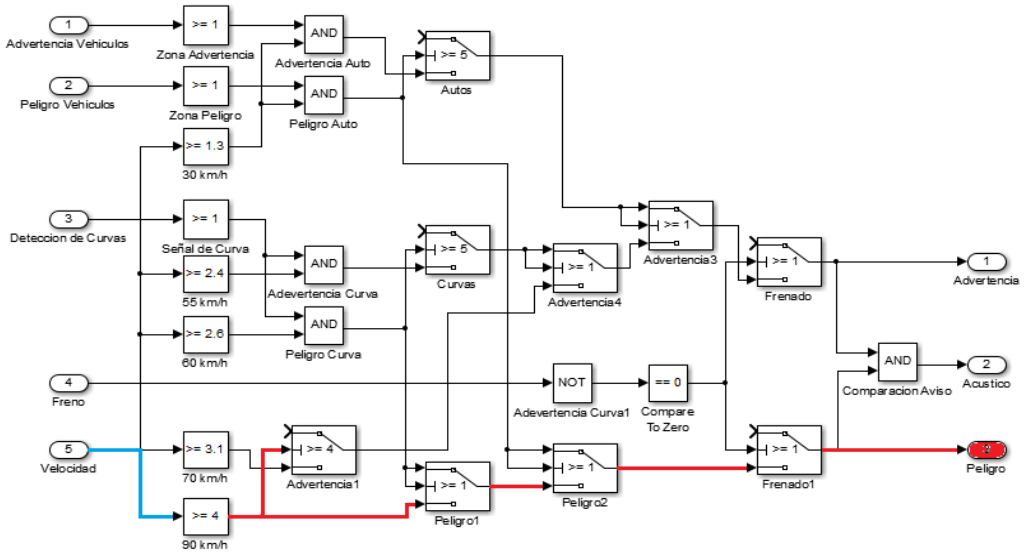


Figura 4.60: Peligro con Velocidad de 90km/h.
Fuente: Autores.

4.4 REDUCCIÓN DE VELOCIDAD.

La reducción de velocidad es controlada por el conductor del vehículo, al recibir cada una de las señales el tendrá la libertad de adaptar la velocidad dependiendo las situaciones de manejo. Al no obedecer a las indicaciones del algoritmo, tendremos situaciones donde se encenderá la luz de advertencia y peligro, en este caso se enviara una señal acústica. Los tres indicadores que llegan al conductor se desactivan en el momento que se activa el pedal de freno; consiguiendo así la reducción de velocidad.

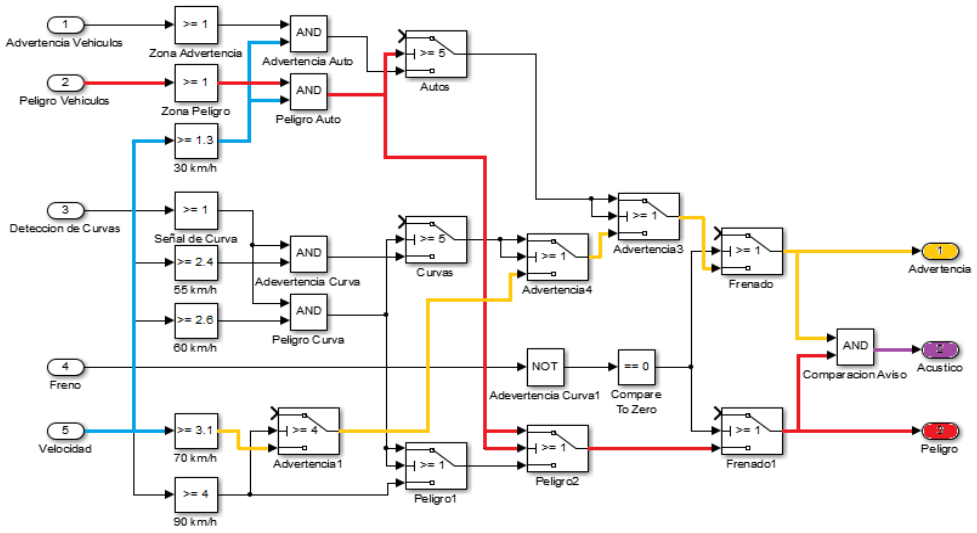


Figura 4.61: Indicador Acústico, Zona de Peligro a una Velocidad de 70km/h.
Fuente: Autores.

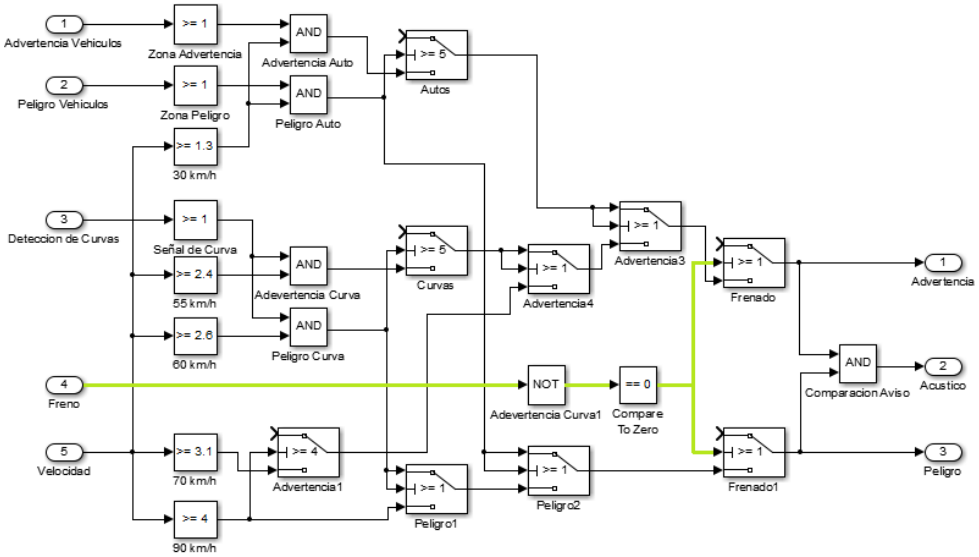


Figura 4.62: Desactivado de los Indicadores por Medio del Pedal de Freno.
Fuente: Autores.

Dentro de la programación de adaptación de velocidad en el software Simulink®. Se considera dar prioridad a la proximidad de vehículos ante la detección del trazado de la vía y los límites de velocidad, ya que al probar el dispositivo en situaciones reales de manejo, es recomendable mantener la distancia de conducción entre vehículos. Por lo cual, si el vehículo circula en una recta y se aproxima a una curva, el sistema bloquea el control de límites de velocidad priorizando la detección de curvas, censando a la velocidad a la que circula el vehículo. De igual manera en el caso de estar en una recta o curva y detecta la proximidad de un vehículo, bloquea los límites de velocidad y la detección del trazado de la vía y entra a trabajar la proximidad de vehículos.

CAPITULO 5

IMPLEMENTACIÓN DEL SISTEMA DE ADAPTACIÓN INTELIGENTE DE LA VELOCIDAD EN UN VEHÍCULO AUTOMOTOR.

5. IMPLEMENTACIÓN DEL SISTEMA DE ADAPTACIÓN INTELIGENTE DE LA VELOCIDAD EN UN VEHÍCULO AUTOMOTOR.

Para realizar la adaptación inteligente de velocidad con respecto a la proximidad de vehículos y el trazado de la vía, es necesario obtener variables que permiten que la advertencia se dé, como vimos en el capítulo anterior se ocupó la velocidad de circulación del vehículo además de la posición de accionamiento del pedal de frenado. Estas variables permiten retroalimentar al sistema para mantener la velocidad adecuada de circulación.

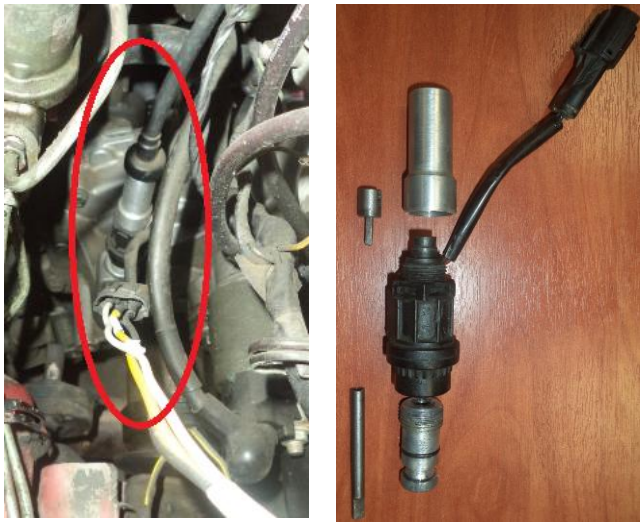
En este capítulo se tratara sobre la obtención de la señal de velocidad de circulación del vehículo que se obtendrá con la adaptación de un sensor digital al vehículo, además de la adaptación de un indicador que permita conocer cuando el conductor este accionando el freno. Como último se explicara cómo funcionan los indicadores de advertencia que dará el sistema de adaptación inteligente de velocidad al conductor.

5.1 ADAPTACIÓN DEL SENSOR DE VELOCIDAD.

Para la obtención de la señal de velocidad utilizamos un sensor de velocidad tipo Hall Figura. 5.1 el cual permite obtener una señal digital, esta varía su frecuencia de forma proporcional a la velocidad a la que el vehículo circula. La alimentación del sensor se realiza por el cable de color rojo con 5v, la señal digital enviada por el sensor es por el cable de color azul, mientras el cable de color negro es conexión a masa.

Figura 5.1: Sensor de Velocidad Daewoo.
Fuente: www.brajovic.com.

El vehículo Suzuki Forsa 1, donde está montado el prototipo, no cuenta con sensor de velocidad por lo cual se realiza una adaptación del sensor de velocidad al velocímetro de la caja de cambios.



a) b)

Figura 5.2: a) Sensor de Velocidad instalado en el Vehículo, b) Sensor de Velocidad con Adaptaciones
Fuente: Autores.

5.1.1 CONVERSIÓN DE FRECUENCIA A VOLTAJE.

La señal obtenida del sensor de velocidad es una onda cuadrada, que va a tener una frecuencia proporcional a la velocidad de circulación del vehículo, esta variación de frecuencia es de difícil interpretación en el entorno de Simulink®. Por lo que es necesario convertir la variación de frecuencia a una señal de fácil interpretación para Simulink®, Es conveniente convertir la variación de frecuencia en variación de voltaje (señal analógica), que de igual forma es proporcional el voltaje con la velocidad.

Por lo cual conseguimos la frecuencia que envía el sensor, a cada una de las velocidades de circulación. Como se observa en la Tabla 5.1.

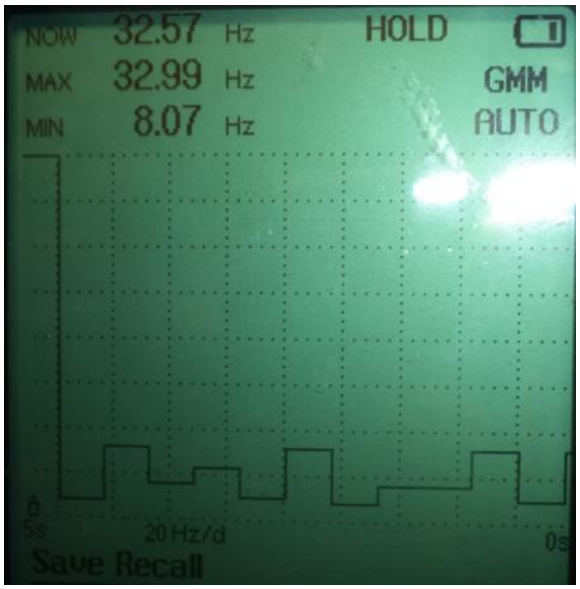


Figura 5.3: Frecuencia del Sensor de 32.57 Hz a 30km/h.
Fuente: Autores.

Tabla 5-1: Frecuencia del Sensor de Velocidad.
Fuente: Autores.

Frecuencia de Vss (Hz)	***Velocidad (km/h)***
23.27	20
32.57	30
38.93	40
48.34	50
54.94	60
61.20	70
61.16	80
78.74	90
85.13	100

Para la conversión de frecuencia a voltaje hemos ocupado el integrado LM2907, el mismo permite trabajar con un voltaje de hasta 28 voltios de alimentación, dentro del integrado posee un transistor que puede absorber una corriente de hasta 0.50mA permitiendo comandar directamente leds o relés sin circuitos complementarios; la configuración del circuito recomendada por el fabricante para la conversión de frecuencia a voltaje se muestra en la Figura 5.4 [20].

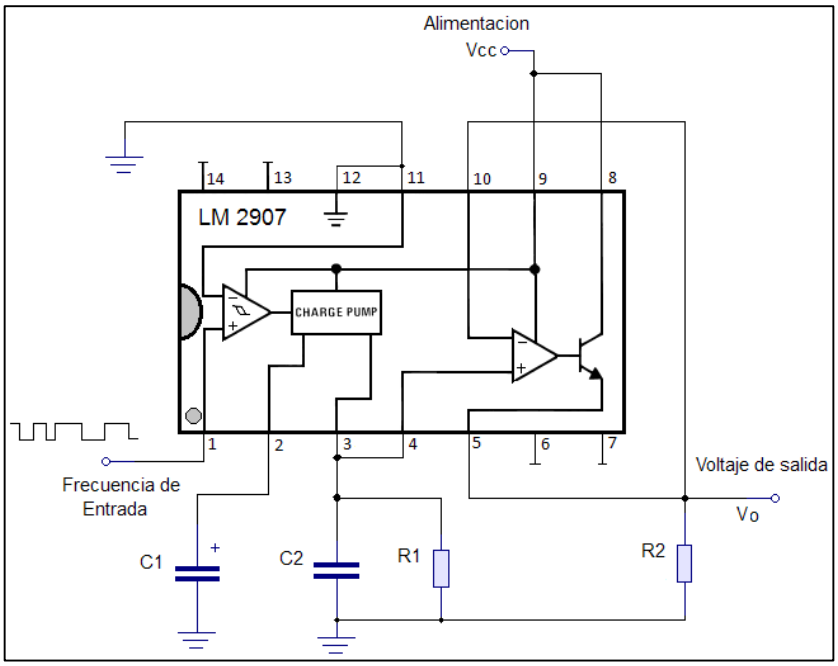


Figura 5.4: Circuito de Transformada de Frecuencia a Voltaje recomendada por el Fabricante.
Fuente: Autores.

Para determinar los valores de las resistencias y condensadores a utilizar, el fabricante nos proporciona las siguientes consideraciones:

- C_1 debe de ser mayor a 100 pF para mantener la compensación interna en la bomba de carga.
- R_2 y C_2 son valores que el fabricante recomienda para el circuito de conversión de frecuencia a voltaje, los valores son de resistencia 10kΩ y condensador electrolítico de 1µf.
- Para el valor de R_1 los valores son calculados con la formula dada que permite determinar la resistencia según el voltaje de salida y la frecuencia de entrada [24].

$$\boldsymbol{V_o} = V_{cc} \times f_{in} \times C_1 \times R_1 \times K$$

Donde cada componente de la ecuación es:

$$\boldsymbol{Vo} = \textit{Voltaje de Salida.}$$
$$\boldsymbol{Vcc} = \textit{Voltaje de Alimentación.}$$
$$\boldsymbol{fin} = \textit{Frecuencia de Entrada.}$$
$$\boldsymbol{C1} = \textit{Valor del condensador } \#1.$$

$$\boldsymbol{R1} = Valor\ Resistivo\ \#1.$$

$$\boldsymbol{K} = \ Constante\ (com\acute{u}nmente\ K = 1).$$

Para el cálculo debemos imponernos el valor del condensador C_1 y el Voltaje de salida (V_o), para despejar el valor de la resistencia que necesitamos y la constante k=1 para obtener los siguientes valores:

$$Vo = \ 4V$$

$$Vcc = \ 14.4V\ (Batería)$$

$$fin = \ Frecuencia\ a\ 90\ km/h = 78.74\ Hz$$

$$C1 = \ 0.01\mu f.$$

$$R1 = ?$$

$$R1 = \frac{Vo}{Vcc * fin * C1 * K}$$

Una vez calculado el valor de la resistencia R_1= 352.7kΩ procedemos a armar el circuito con los valores de los componentes como muestra la Figura. 5.5.

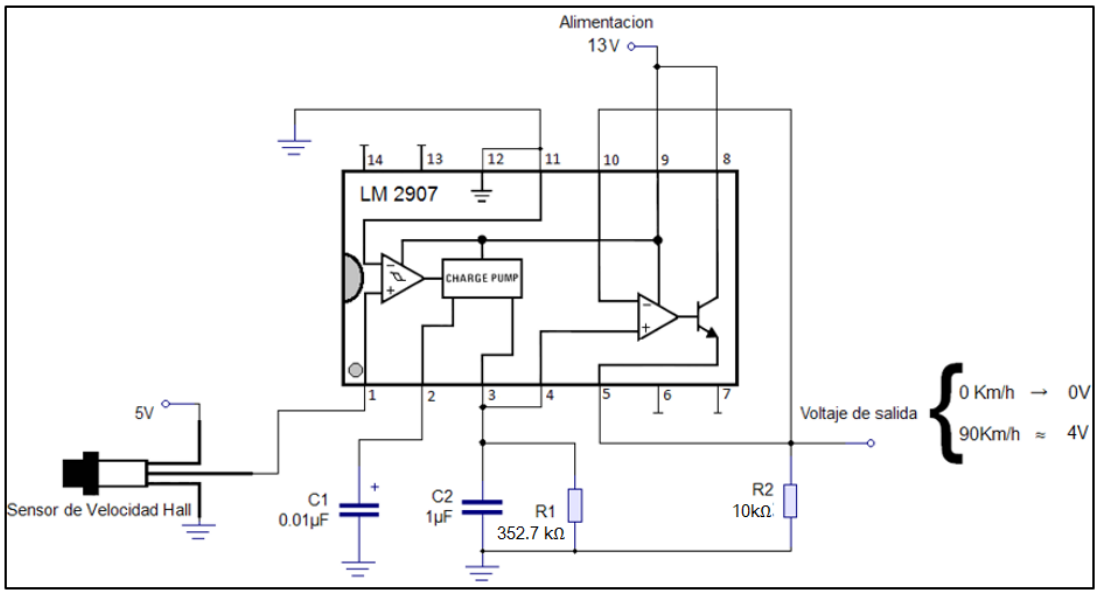


Figura 5.5: Circuito de Transformada de Frecuencia a Voltaje con Valores Reales.
Fuente: Autores.

Para el circuito soldado en la placa debemos de utilizar la R_1 tipo trimmer multi-vueltas para calibrar la resistencia calculada, se observa en la siguiente Figura el circuito armado con todos los componentes, la placa está montada en la tarjeta de adquisición de datos Arduino®.

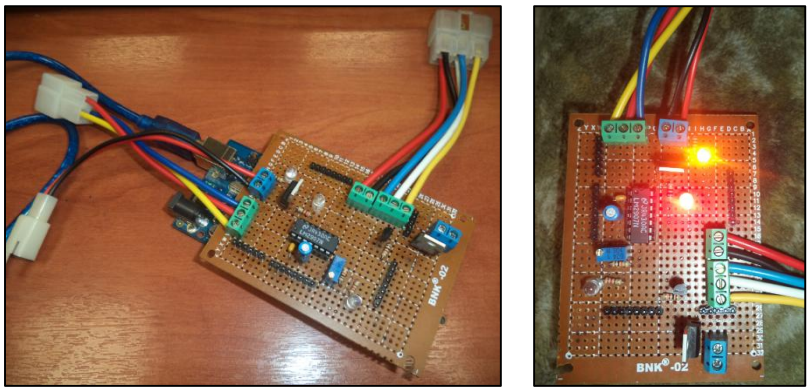
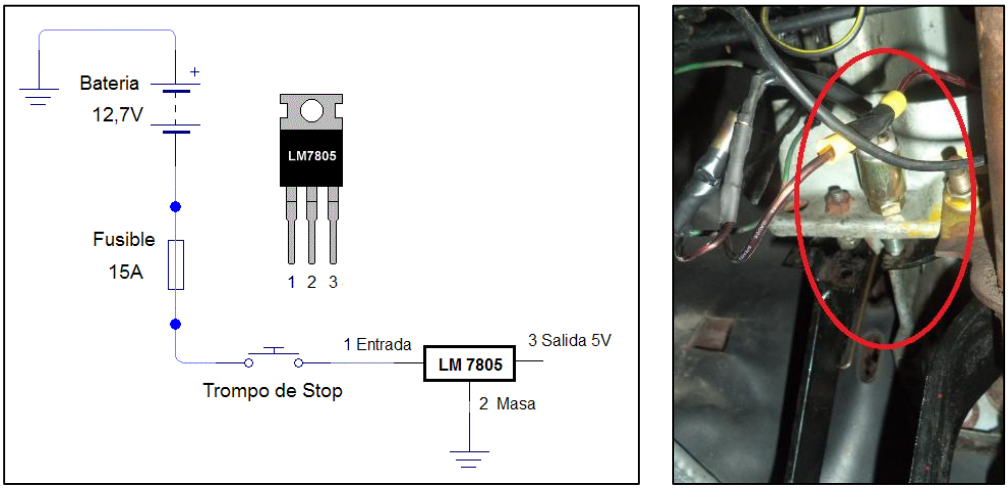


Figura 5.6: Circuito montado para captación de Voltaje y Enviar las Señales.
Fuente: Autores.

5.2 OBTENCIÓN DE LA SEÑAL DE FRENADO.

La señal de frenado es una de las señales importantes para el funcionamiento del algoritmo como se mencionó en el capítulo 4, al activarse el sistema de frenos no debe de funcionar los indicadores, hasta cumplir los aspectos de seguridad. Para ello hemos obtenido la señal desde el trompo de stop, el mismo envía una señal de 12V cuando el pedal de freno es accionado, debemos de tener en cuenta que esta señal debe de ingresar a Simulink® por intermedio de Arduino Mega 2560, para ello es necesario transformar la señal de 12V a 5V debido a que el Arduino Mega 2560 trabaja con un máximo de 5V, transformándola a una señal digital.

Para reducir el voltaje de 12V a 5V utilizamos un integrado 7805 como se muestra en la Figura. 5.7.



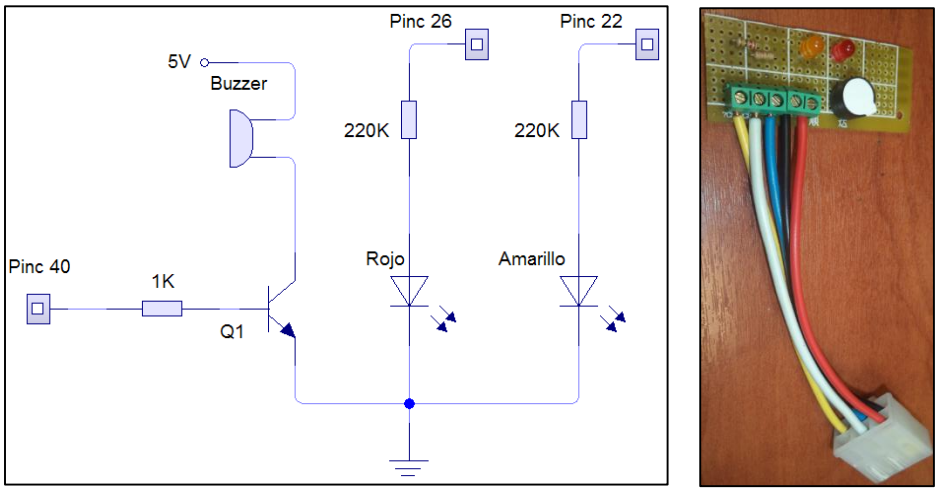
a) b)

Figura 5.7: a) Circuito de Accionamiento del Pedal de Freno, b) Trompo de Stop.
Fuente: Autores.

5.3 CIRCUITO DE ADVERTENCIA.

Para dar advertencia al conductor cuando está fuera de los rangos de velocidad establecidos en el capítulo 2 en el instante que se encuentra en una curva o próximo a otro vehículo, se utiliza un circuito de un led de color amarillo para advertencia, otro de color rojo para peligro y un buzzer para señal acústica, cuando no cumple con las señales visuales y el vehículo sigue aumentando la velocidad.

Las señales para la activación de cada componente son enviadas desde el entorno de Simulink® por intermedio del Arduino® Mega 2560 como se muestra en la Figura 5.9. Debe tomarse en cuente que para la activación del buzzer se utiliza un transistor para que el voltaje enviado por el pin 40 del Arduino excite al transistor y permita alimentar con 5 voltios al buzzer. Las señales de accionamiento para el led amarillo y rojo salen respectivamente de los pines 22 y 26.



a) b)

Figura 5.8: a) Circuito de Advertencia al Conductor, b) Indicadores.
Fuente: Autores.

CAPITULO 6

ANALISIS DE RESULTADOS.

6. ANÁLISIS DE RESULTADOS.

Procedemos a verificar el funcionamiento del prototipo de adaptación de velocidad, obteniendo resultados de cada uno de los sistemas, como: proximidad de vehículos y la detección del trazado de la vía. Estos procesos serán probados a diferentes velocidades, en el caso de proximidad con vehículos se realizará con vehículos estacionados en la vía, para comparar con los datos obtenidos anteriormente en el Capítulo 2; detección del trazado de la vía se comprobara con la detección de curvas, en carreteras sin tráfico.

Los datos serán obtenidos en unidad de tiempo, estos datos serán el tiempo para la reacción del conductor ante un evento de conducción, estas serán a diferentes velocidades.

6.1 PROCEDIMIENTO PARA LA OBTENCIÓN DE DATOS.

Las pruebas se realizaron en vías rápidas y perimetrales a la ciudad, ya que se pueden alcanzar velocidades hasta los 90 km/h sin romper los límites de velocidad, los sistemas fueron probados independientemente, la proximidad de vehículos se probó desde 30 km/h hasta 90km/h, teniendo en cuenta la programación de proximidad de vehículos se activa desde 30km/h. Para las pruebas de detección de curvas desde 50 km/h hasta 90km/h, considerando la programación para la detección de curvas se realiza desde los 55km/h para advertencia y 60km/h en adelante para la señal de peligro.

Las pruebas se realizan a 6 conductores a diferentes velocidades. Obteniendo así el tiempo y por consiguiente la distancia desde que se acciona el pedal hasta levantarlo en una situación de detección de curvas y la proximidad de un vehículo estacionado.

a) b)

Figura 6.1: a) Vehículo Estacionado, b) Detección de Curvas.
Fuente: Autores.

6.2 DATOS DE DETECCIÓN DEL TRAZADO DE LA VÍA.

Se tomó datos de seis conductores para probar el Sistema de Adaptación de Velocidad (S.A.V.), las pruebas fueron con y sin sistema. Se analizó a cuatro velocidades; el primer dato es sin sistema, se toma el tiempo en el momento que el conductor activa el pedal de frenos hasta que reduce la velocidad y el conductor desactiva el pedal de frenos; en base al tiempo y la velocidad a la que viaja el vehículo encontramos la distancia. Como se observa en la siguiente Figura 6.2.

Figura 6.2: Esquema para Obtener la Distancia de Frenado sin Sistema.
Fuente: Autores.

En la siguiente Figura 6.3 se muestra el esquema con la utilización del Prototipo.

Figura 6.3: Esquema para Obtener la Distancia de Frenado con Sistema.
Fuente: Autores.

Los datos obtenidos se observan en la siguiente Tabla 6-1.

Tabla 6-1: Distancias para Detección de Curvas.
Fuente: Autores.

Detección de Curvas (m)								
Conductores	**60 km/h**		**70 km/h**		**80 km/h**		**90 km/h**	
	Sin Sistema	Con Sistema	Sin Sistema	Con Sistema	Sin Sistema	Con Sistema	Sin Sistema	Con Sistema
1	4,16	55,86	10,18	56,69	13,77	51,08	22,97	47,34
2	2,77	55,97	10,21	54,06	14,96	50,92	22,55	47,82
3	5,20	55,32	11,28	56,58	12,78	52,10	23,24	48,67
4	3,36	56,43	11,11	55,48	14,53	52,51	22,73	47,98
5	4,34	56,71	9,40	54,72	14,95	52,74	22,23	48,59
6	4,68	55,57	10,94	55,77	14,88	51,76	23,85	47,60
Media de Muestras								
	4,08	55,97	10,52	55,55	14,31	51,85	22,93	48,00

6.3 DATOS DE PROXIMIDAD DE VEHÍCULOS.

La obtención de datos se realizó de la misma forma con detección de curvas, en este caso, las velocidades van a cambiar ya que el sistema comienza detectar desde 30 km/h, hasta los 60 km/h, ya que al realizar pruebas el sistema a mayores velocidades tiene el inconveniente de no enviar señal antes del conductor, esto será sustentado en el análisis estadístico. La programación envía una distancia entre proximidad de vehículos, por lo cual podemos verificar la distancia. De igual manera obtenemos el tiempo en el cual el conductor activa el pedal de freno en una conducción normal, además el tiempo cuando el sistema envía una la señal para el frenado. En las Figuras 6.4 y 6.5 se muestra un esquema sin y con sistema.

Figura 6.4: Esquema para Obtener la Distancia de Frenado sin Sistema.
Fuente: Autores.

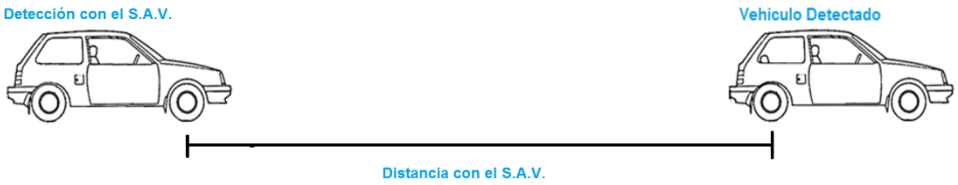


Figura 6.5: Esquema para Obtener la Distancia de Frenado con Sistema.
Fuente: Autores.

Los datos obtenidos se observan en la siguiente Tabla 6-2.

Tabla 6-2: Distancias para Detección de Vehículos.
Fuente: Autores.

Detección de Vehículos (m)								
Conductores	**30 km/h**		**40 km/h**		**50 km/h**		**60 km/h**	
	Sin Sistema	Con Sistema	Sin Sistema	Con Sistema	Sin Sistema	Con Sistema	Sin Sistema	Con Sistema
1	3,79	33,25	5,24	30,90	11,13	26,07	20,25	18,35
2	2,05	35,11	6,26	30,73	10,52	26,26	20,35	19,19
3	2,18	32,17	3,86	29,17	9,87	25,72	19,33	18,45
4	2,33	32,72	3,76	29,93	12,10	25,34	18,49	19,66
5	2,02	34,47	4,30	29,47	10,19	25,32	18,97	18,99
6	2,80	35,31	2,46	28,58	10,14	27,06	20,25	18,75
Media de Muestras								
	2,53	33,84	4,31	29,80	10,66	25,96	19,61	18,90

6.4 ANÁLISIS ESTADÍSTICO.

Para el análisis estadístico a través del software Minitab®, una vez obtenidos los datos se procede utilizar la prueba ANOVA, con un valor significativo de α=0,05, ya que para este caso para cada velocidad fueron dos comprobaciones una con el Sistema de Adaptación Inteligente de Velocidad y otra sin el Sistema Adaptación Inteligente de Velocidad, ya que permite determinar si existe o no un aumento en la distancia frenado ante la proximidad a una curva o un vehículo, al momento de aumentar esta distancia de frenado se dirá que el tiempo para la reacción del conductor aumentó, puesto que el aviso se dio antes de que el conductor active el freno.

$$Hipótesis\ (H1): \mu A\ <\ \mu B$$

Dónde:

$$\boldsymbol{\mu A} =\ Valor\ de\ la\ media\ sin\ Sistema.$$

$$\boldsymbol{\mu B} =\ Valor\ de\ la\ media\ con\ Sistema.$$

La tabla ANOVA permite obtener el valor de la significancia P este debe de ser menor a α=0,05, para que se cumpla H1, esto establece que existe diferencia de distancias de frenado de acuerdo a la velocidad de circulación cuando este activado.

Además se procede a generar la prueba de Tukey que hace un análisis de las medias de los tratamientos. Con esta prueba se aprecia las medidas correspondientes son significativamente diferentes, ratificando la aceptación de la hipótesis H_1.

6.4.1 ANÁLISIS ESTADÍSTICO PARA DETECCION DE AUTOS.

A continuación se tomaran dos muestras diferentes, una a 30 km/h que es la velocidad de inicio para el reconocimiento y a 60 km/h donde no existe un aumento en el tiempo para que reaccione el conductor.

6.4.1.1 Análisis de Distancias de Reacción a la Velocidad de 30km/h.

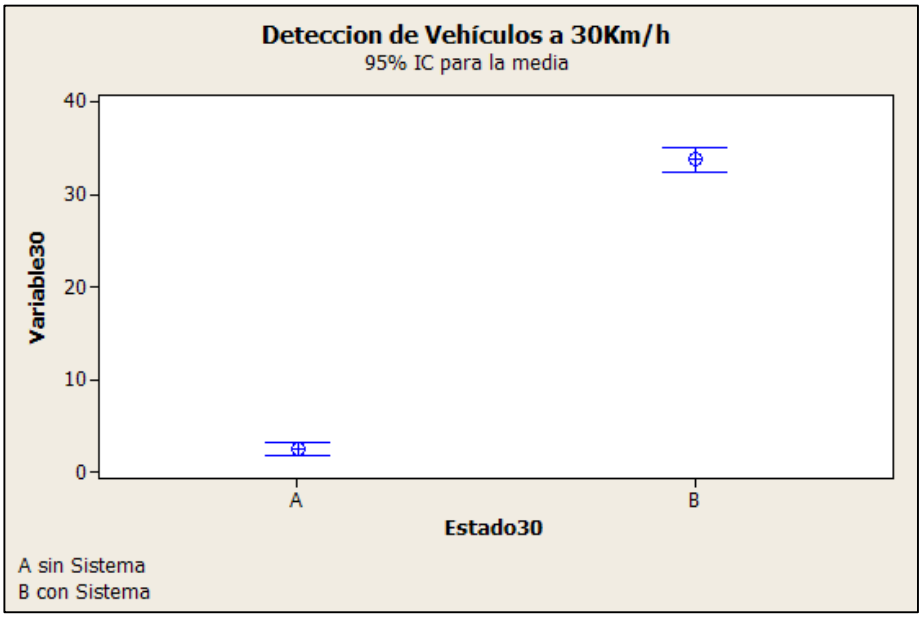


Figura 6.6: Intervalo de Medias para la Distancia de Reacción entre Vehículos a 30Km/h.
Fuente: Autores.

Tenemos:

Fuente	GL	SC	CM	F	P
Estado30	1	2928,65	2928,65	2700,01	0,000
Error	10	10,85	1,08		
Total	11	2939,50			

Donde $\rho < 0{,}05$, entonces en cuanto al aumento de la distancia de reacción de frenado es altamente significativa.

Agrupar información utilizando el método de Tukey

Estado30	N	Media	Agrupación
B	6	33,839	A
A	6	2,595	B

Las medias que no comparten una letra son significativamente diferentes.

Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado50

6.4.1.2 Análisis de Distancia a una Velocidad de 60km/h.

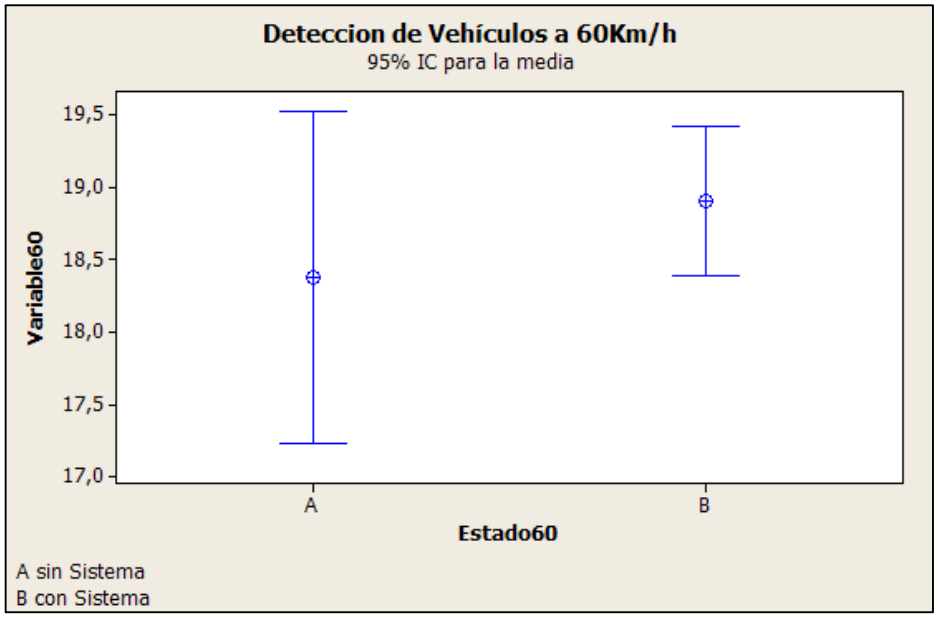


Figura 6.7: Intervalo de Medias para la Distancia de Reacción entre Vehículos a 60Km/h.
Fuente: Autores.

Tenemos:

Fuente	GL	SC	CM	F	P
Estado60	1	0,813	0,813	1,14	0,311
Error	10	7,145	0,714		
Total	11	7,958			

Donde $\rho > 0{,}05$, entonces el aumento de la distancia de reacción de frenado será insignificante debido a que la diferencia de medias es baja.

Agrupar información utilizando el método de Tukey

Estado60	N	Media	Agrupación
B	6	18,8984	A
A	6	18,3778	A

Las medias que no comparten una letra son significativamente diferentes.

Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado60

6.4.2 ANÁLISIS ESTADÍSTICO PARA DETECCION DE CURVAS.

En esta parte se analizará solo una velocidad a 60km/h, ya que desde esta velocidad, es la actúa el sistema, teniendo en cuenta que el análisis de las demás curvas y proximidad de vehículos se encuentra en Anexos.

6.4.2.1 Análisis de Distancia de Reacción a una Velocidad de 60km/h.

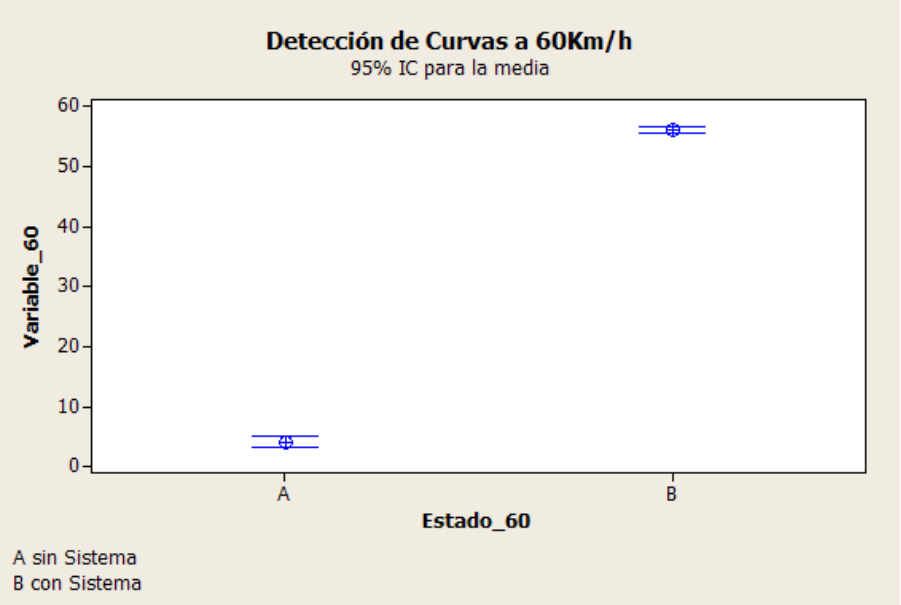


Figura 6.8: Intervalo de Medias para la Distancia de Reacción en Curva a 60Km/h.
Fuente: Autores.

Tenemos:

Fuente	GL	SC	CM	F	P
Estado_60	1	8078,230	8078,230	15369,21	0,000
Error	10	5,256	0,526		
Total	11	8083,487			

Donde $\rho < 0{,}05$, entonces el aumento de la distancia de reacción de frenado será significativa en relación de conducir con o sin el sistema.

Agrupar información utilizando el método de Tukey

Estado_60	N	Media	Agrupación
B	6	55,975	A
A	6	4,083	B

Las medias que no comparten una letra son significativamente diferentes.

Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado_60

6.4.3 ANÁLISIS DE GRÁFICOS LINEALES.

La validación del prototipo, se la realiza por medio de la media o promedio de la distancia de frenado en cada velocidad, teniendo en cuenta que esta distancias en metros se la puede transformar a unidad de tiempo, comprobando así si existe o no el aumento de tiempo para que reaccione el conductor para la acción de frenado. A continuación se analizar por medio de diagramas, como se ve en las Figuras 6.9 y 6.10.

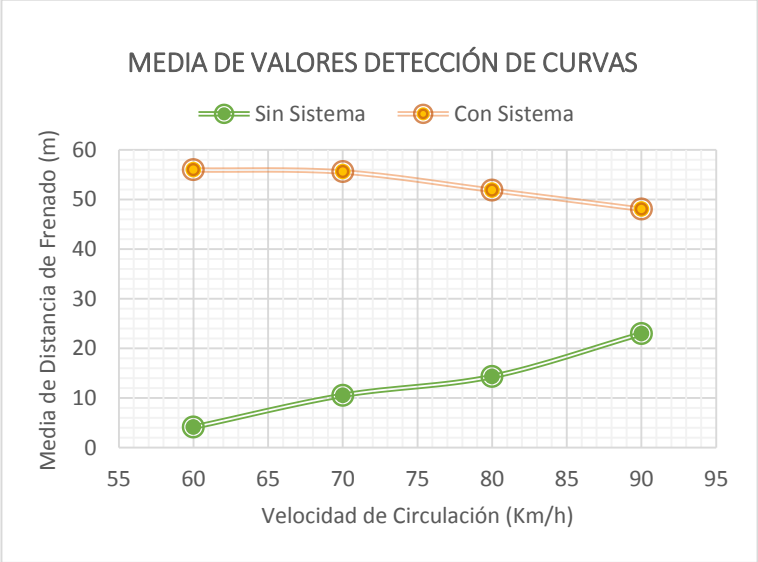


Figura 6.9: Diagrama de Detección de Curvas.
Fuente: Autores.

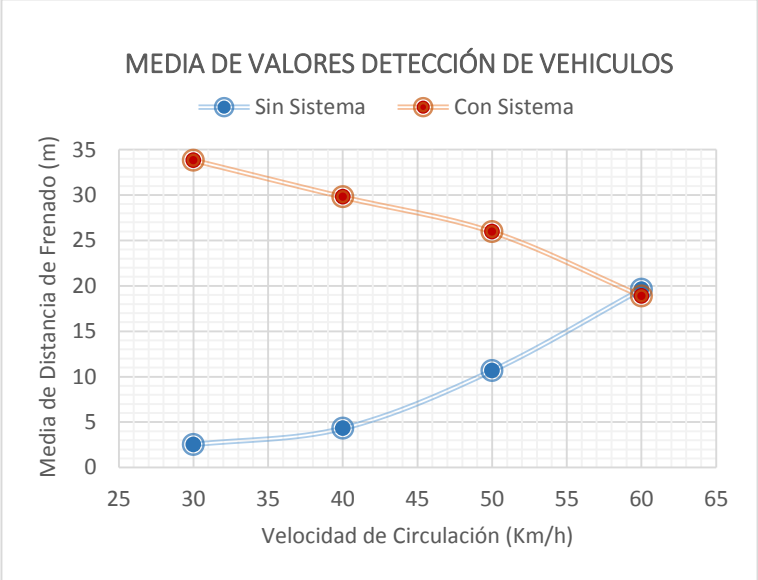


Figura 6.10: Diagrama de Detección de Vehículos.
Fuente: Autores.

Observando los diagramas tanto de detección de curvas y proximidad de vehículos. Vemos la diferencia de las media de cada muestra. Se determinaron los porcentajes en los cuales aumentan cada valor de media, en la siguiente tabla 6-3. Observamos el porcentaje aumentado.

Tabla 6-3: Porcentaje en la Distancia para Detección de Curvas.
Fuente: Autores.

Detección de Curvas				
Velocidad (km/h)	60	70	80	90
Porcentaje Aumentado (%)	1270,77	428,06	262,33	109,36

Tabla 6-4: Porcentaje en la Distancia para Detección de Vehículos.
Fuente: Autores.

Detección de Vehículos				
Velocidad (km/h)	30	40	50	60
Porcentaje Aumentado (%)	1239,08	590,68	143,57	-3,62

El porcentaje en los dos análisis en velocidades bajas son muy grandes, pero en el caso de detección de curvas, la aumentar la velocidad va disminuyendo el porcentaje, de igual forma se puede notar en el diagrama correspondiente. Con esto se podría decir que el sistema funciona mejor en velocidades bajas, pero en el caso de reconocimiento de curvas no serviría de nada ya que al circular velocidades bajas podríamos generar situaciones de peligro en el caso de vías perimetrales a la ciudad.

Para el reconocimiento de vehículos a velocidades bajas tiene un funcionamiento correcto, pero al aumentar la velocidad este baja su nivel de funcionamiento ya que no reconoce con tiempo el vehículo estacionado, por lo cual a mayores velocidades el conductor reacciona con anterioridad el sistema, esto lo comprobamos con un porcentaje negativo.

El porcentaje encontrado es el mismo para el aumento en el tiempo para la reacción del conductor ya que se relaciona la distancia, tiempo y velocidad.

7. CONCLUCIONES Y RECOMENDACIONES:

- El desarrollo del algoritmo de adaptación inteligente de velocidad, demostró el aumento del tiempo para la reacción del conductor, al aumentar este tiempo el conductor activara el sistema de frenos con anticipación; comparando la misma situación sin el sistema. Este tiempo tiende a disminuir en altas velocidades ya que la percepción del auto o de la curva, no es la misma que abajas velocidades, por lo que es recomendable para bajas velocidades en la parte de reconocimiento de vehículos, en cambio para la detección de curvas según datos, funciona en altas velocidades.
- El funcionamiento del prototipo funciona independientemente de las condiciones del conductor, esto quiere decir que el sistema no depende de la experiencia del conductor o habilidades de manejo. Esto influencia directamente en el tiempo de reacción del conductor, este tiempo se ha encontrado que varía dependiendo del conocimiento del conductor ante una situación de peligro, es decir, una situación inesperada o una situación esperada. Ante situaciones esperadas el tiempo de reacción es de 0.54 segundos, en cambio en situaciones inesperadas el tiempo es de 1.31 segundos en promedio. Por lo que el Sistema de Adaptación Inteligente de Velocidad, transforma una situación inesperada en esperada, por lo que ayuda mucho al conductor, que no es necesario la experiencia, habilidades, etc. Del mismo para un correcto manejo del vehículo, ya que el sistema avisara con anticipación cuando debería actuar el conductor para adaptar la velocidad de circulación.
- Según la experiencia de los conductores de prueba, el sistema le informara cuando modificar la velocidad, independientemente del estado del conductor; es decir el conductor puede estar distraído, en estados de ánimos diferentes, entre otros, el sistema igual informara cuando adaptar la velocidad del vehículo, así entonces evitar que un conductor distraído o estresado puede accidentarse.
- El sistema al proporcionar información el conductor, del momento de reducir la velocidad, mantener la distancia de seguridad entre vehículos en

circulación, anticipar el frenado para entrar a una curva, o prevenir de cuando frenar ante un vehículo detenido. Creará una educación a los conductores, ya que al percibir las señales de advertencia, peligro y acústica. Los conductores sea acostumbrarán a no encender ninguna de las señales de aviso, entonces así creando una conducción más homogénea para evitar accidentes.

- Al realizar el sistema de detección de la vía se debe de presentar una perfecta señalización de líneas y sus bordes deben estar a la vista ya que de ellos depende que el sistema funcione correctamente para que determine en que momento el vehículo va a ingresar a una curva.
- Las ubicaciones de las cámaras para la detección del trazado de la vía, es muy importante ya que se puede conseguir una óptica de mayor calidad como de menor, de acuerdo a su altura, realizando las diferentes pruebas de la ubicación de la cámara en el vehículo ya que en primera instancia se la ubicó en la mascarilla del vehículo a una altura…….. pero nos presentaba el problema de que se observa el trazado de la vía a poca distancia, y mientras que se sigue cambiando su altura a una de mayor nos presenta una mejor observación del trazado de la vía por lo que se decidió poner las cámaras en la parte más alta del vehículo. Como en los vehículos de mayor tamaño nos presenta una mayor facilidad de visión del trazado de la vía a largas distancias.
- En el trazado de la vía nos encontramos con peraltes diferentes, en donde tenemos un mayor peralte en la vía Tarqui y en la autopista uno menor, y este es una gran inconveniente en el recorte de la imagen (Submatrix) ya que se tiene que cambiar los parámetros dentro de la submatrix y lo mejor sería de que se regulen automáticamente dependiendo si se encuentra en la autopista o en vías de un sólo carril.
- La imagen adquirida del entorno tiene efectos de visualización, ya que se puede encontrar los efectos climáticos como son la lluvia, la neblina, como también las sombras que producen ciertos objetos de los cuales tenemos los árboles, y esto es un impedimento para la detección de los bordes y del trazado de la vía, lo que deberíamos indagar mucho en el filtrado para que

corrija automáticamente todos estos efectos y nos presente líneas continuas en todo los tramos de las vía.

- Para él envió de datos tanto de la detección de vehículos como del reconocimiento del trazado de la vía realizados en Matlab y Simulink®, a una tarjeta de adquisición de datos, se utilizó en primera instancia la tarjeta Arduino Mega 2560 pero la misma presenta problemas ya que las herramientas utilizadas en Simulink® de la librería "Computer Visión Systema" tienen un gasto computacional muy alto en la Arduino por la cual no se puede procesar el algoritmo descrito. Ya que se debería de utilizar una tarjeta que tenga la capacidad de procesar señales de video realizada con los bloques de la librería "Computer Visión Systema".

  Otra opción se puede realizar la programación en el Matlab Editor, para no utilizar Simulink®, ya que de esta manera el programa de la detección del trazado de la vía no tiene un cargo computacional muy alto y se la puede enviar los datos al Arduino.
- En el momento en que la vía presenta una pendiente mayor, y al momento de ascender el vehículo por ella el sistema no puede detectar, ya que las cámaras estarían enfocadas al horizonte, y como puede ser que al llegar hasta la cima de la pendiente se encuentre una curva y en este caso no la puede detectar. Pero como también al momento que se desciende de la pendiente el sistema puede observar a mayor longitud la curva.
- Al momento de adquirir las imágenes positivas y negativas para entrenar al clasificador en cascada es necesario obtener imágenes de igual o menor resolución con respecto a la resolución de la cámara de video que va a trabajar con el sistema, debido a que si se ocupa imágenes con resoluciones significativamente altas o bajas a la resolución del video, estas van a generar clasificadores con características que posean mayores pixeles a los que se desea reconocer, produciéndonos falsas detecciones en el momento de funcionar el clasificador.
- El algoritmo para la detección de vehículos tiene un alcance limitado para vehículos livianos y semi-pesados, para futuras investigaciones se recomienda realizar una implementación del algoritmo para reconocer

vehículos pesados o cualquier tipo de vehículo con los que se pueda encontrar en las vías.

- La programación para determinar la distancia de proximidad entre vehículos presenta un error de ± 2 metros con respecto a la distancia real, por lo que es recomendable que para el mejoramiento del sistema en futuras investigaciones se pueda reducir el error de distancia de proximidad con el complemento de otros sistemas o el mejoramiento del algoritmo.
- Si se desea trabajar con Simulink® para el procesamiento de imágenes es recomendable utilizar mini ordenadores como: Raspberry ®, PandaBoard, Gumstix Over entre otros debido a que el paquete de herramientas para el procesamiento de imágenes que presenta Simulik® que genera una carga significativa de información para tarjetas de adquisición de señales como es el caso de Arduino.

8. BIBLIOGRAFÍA:

[1] A. González Marcos, F. J. Marrtínez de Pisón Ascacíbar, A. V. Pernía Espinoza, F. Alba Elías, M. Castejón Limas, J. Ordieres Meré, and E. Vergara Gonzáles, *Técnicas y Algoritmos Básicos de Visión Artificial*, 1ª ed. Uniiversidad de La Rioja: Servicio de Publicaciones, 2006.

[2] L. Rossi, *Pixel Detectors: From Fundamentals to Applications*. Springer, 2006.

[3] R. Molina, *Introducción al Procesamiento y Análisis de Imágenes Digitales*. Granada, España.

[4] E. Cuevas, D. Zaldívar, and M. Pérez-Cisneros, *Procesamiento digital de imágenes usando MatLAB & Simulink*. Alfaomega, 2010.

[5] Víctor Alonso Mendieta, "DETECCIÓN Y RECONOCIMIENTO DE SEMÁFOROS POR VISIÓN ARTIFICIAL," UNIVERSIDAD CARLOS III DE MADRID, 2013.

[6] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Sixth International Conference on Computer Vision, 1998*, 1998, pp. 555–562.

[7] P. Viola and M. Jones, "Robust real-time face detection," in *Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Proceedings*, 2001, vol. 2, pp. 747–747.

[8] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," in , *Proceedings of the 12th IAPR International Conference on Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision amp; Image Processing*, 1994, vol. 1, pp. 582–585 vol.1.

[9] Oliver Moll Javier, "Detección de Personas en secuencias de vídeo en tiempo real," Universitat Politécnica de Valéncia., 2011.

[10] F. Cárdenas and R. Llerena, "AUTOMATIZACIÓN DE UN SISTEMA DE CENTRADO DE COMPONENTES UTILIZANDO VISIÓN ARTIFICIAL," UNIVERSIDAD POLITÉCNICA SALESIANA SEDE CUENCA, 2012.

[11] E. Cuevas, D. Zaldívar, and M. Pérez-Cisneros, *Procesamiento digital de imágenes usando MatLAB & Simulink*. Alfaomega, 2010.

[12] Vélez Serrano JoséF., Moreno Díaz Ana Belén, Calle Ángel Sánchez, and Sánchez Marín José L., *VISIÓN POR COMPUTADOR*, 2nd ed. .

[13] M. I. A. Galipienso, M. A. C. Quevedo, O. C. Pardo, F. E. Ruiz, and M. A. L. Ortega, *Inteligencia artificial: modelos, técnicas y áreas de aplicación*. Editorial Paraninfo, 2003.

[14] C. Vera, V. D. López, and F. Aparicio, *Teoría de los vehículos automóviles*. Escuela Técnica Superior de Industriales, Sección de Publicaciones, 2001.

[15] E. A. SOBRADO MALPARTIDA, "SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO Y MANIPULACIÓN DE OBJETOS UTILIZANDO UN BRAZO ROBOT," PONTIFICIA UNIVERSIDA CATÓLICA DEL PERÚ, 2003.

[16] E. A. VEGA AQUINO, "DETECCIÓN Y SEGUIMIENTO DE ROSTROS," UNIVRSIDAD CARLOS III DE MADRID, 2011.

[17] P. Viola and M. Jones, "Robust real-time face detection," in *Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Proceedings*, 2001, vol. 2, pp. 747–747.

[18] "Cascade Training GUI: Specify Ground Truth - File Exchange - MATLAB Central." [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/file_infos/39627-cascade-training-gui--specify-ground-truth. [Accessed: 27-May-2014].

[19] "Detect objects using the Viola-Jones algorithm - MATLAB." [Online]. Available: zotero://attachment/82/. [Accessed: 04-Jun-2014].

[20] MathWorks, "Label Images for Classification Model Training - MATLAB & Simulink." [Online]. Available: http://www.mathworks.com/help/vision/ug/label-images-for-classification-model-training.html. [Accessed: 16-Jul-2014].

[21] "Train a Cascade Object Detector - MATLAB & Simulink." [Online]. Available: http://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html. [Accessed: 19-Nov-2013].

[22] E. Sucar and G. Gómez, "Visi on Computacional."

[23] R. Molina, "INTRODUCCIÓN AL PROCESAMIENTO Y ANÁLISIS DE IMÁGENES.," Universidad de Granada, 1998.

[24] Texas Instruments Incorporated, "LM2907/LM2917 Frequency to Voltage Converter." Texas Instruments, 2013-2000.

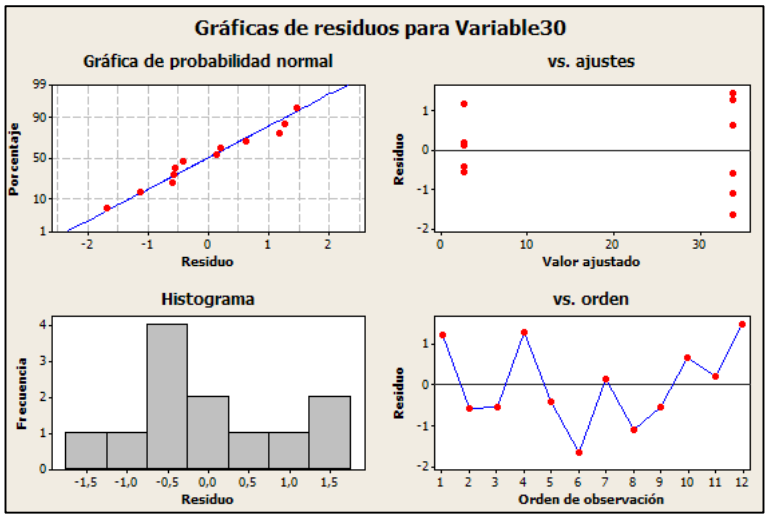
9. ANEXOS

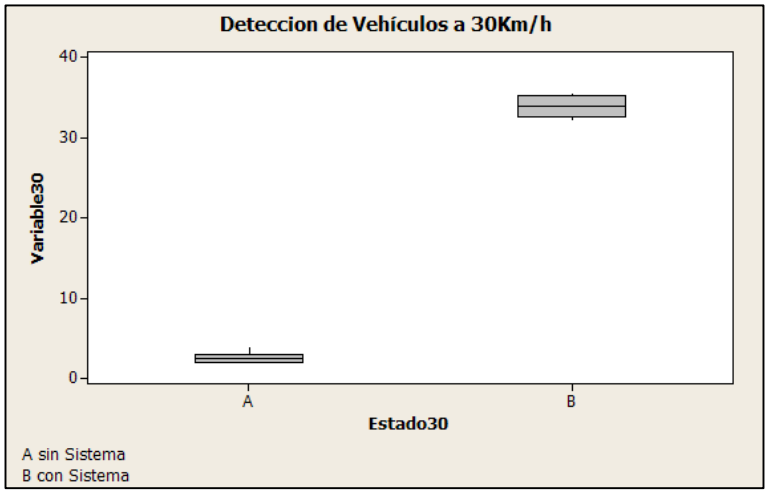
DETECCIÓN DE VEHÍCULOS

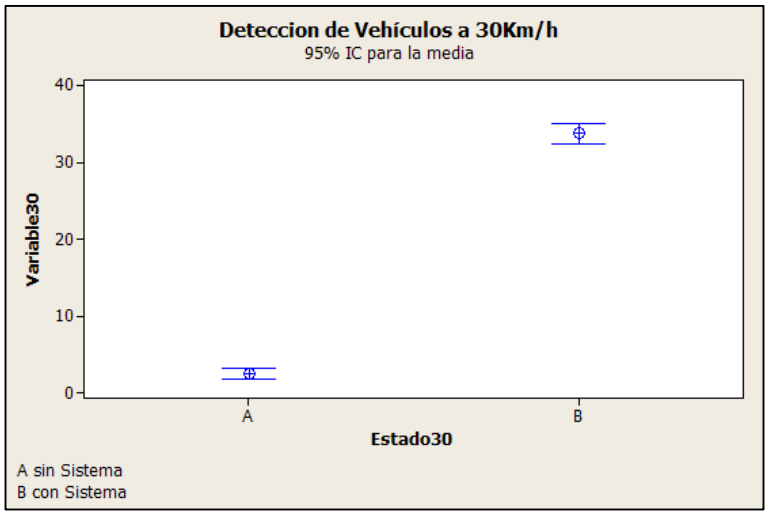
30km/h

ANOVA unidireccional: Variable30 vs. Estado30

```
Fuente    GL       SC       CM        F      P
Estado30   1  2928,65  2928,65  2700,01  0,000
Error     10    10,85     1,08
Total     11  2939,50

S = 1,041   R-cuad. = 99,63%   R-cuad.(ajustado) = 99,59%


                            ICs de 95% individuales para la media
                            basados en Desv.Est. agrupada
Nivel  N   Media  Desv.Est.  --------+---------+---------+---------+-
A      6   2,595      0,676  (*)
B      6  33,839      1,308                                (*)
                            --------+---------+---------+---------+-
                                   10        20        30        40

Desv.Est. agrupada = 1,041


Agrupar información utilizando el método de Tukey

Estado30  N   Media  Agrupación
B         6  33,839  A
A         6   2,595    B

Las medias que no comparten una letra son significativamente diferentes.


Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado30

Nivel de confianza individual = 95,00%


Estado30 = A restado de:

Estado30  Inferior  Centro  Superior  -+---------+---------+---------+------
--
B           29,905  31,244    32,584                                (*-)
                                     -+---------+---------+---------+------
--
                                      0        10        20        30
```


Gráficas de residuos para Variable30
Gráfica de probabilidad normal
Porcentaje
Residuo
vs. ajustes
Residuo
Valor ajustado
Histograma
Frecuencia
Residuo
vs. orden
Residuo
Orden de observación

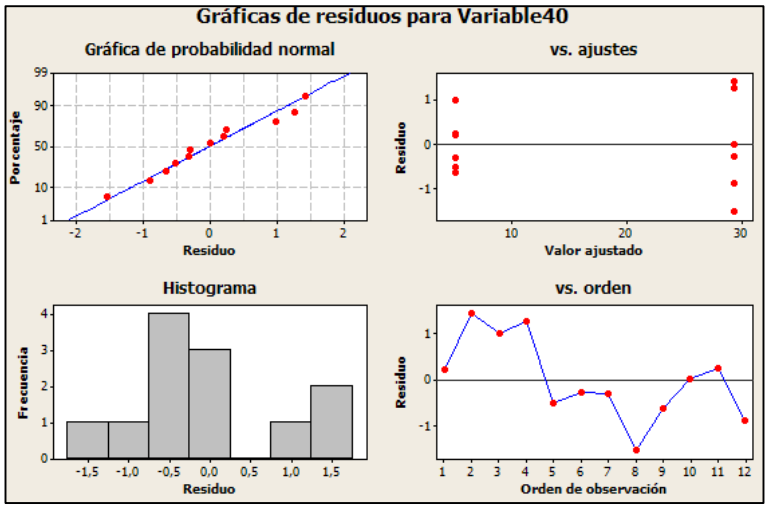

Deteccion de Vehículos a 30Km/h
Variable30
Estado30
A
B
A sin Sistema
B con Sistema

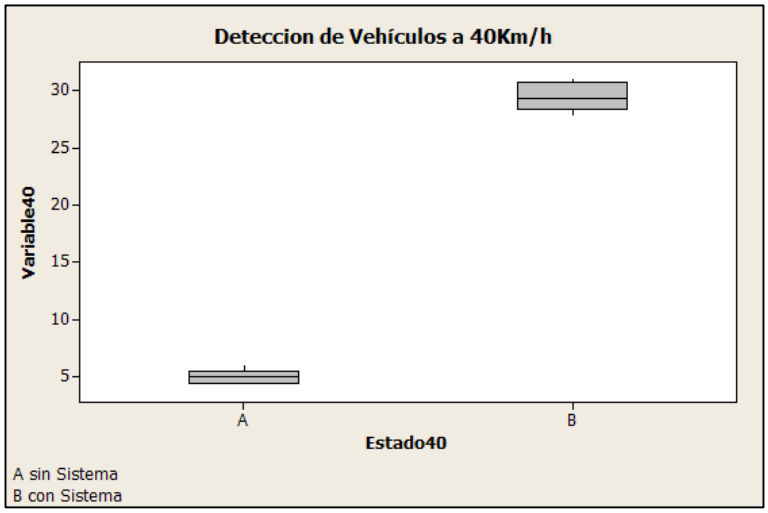

Deteccion de Vehículos a 30Km/h
95% IC para la media
Variable30
Estado30
A
B
A sin Sistema
B con Sistema

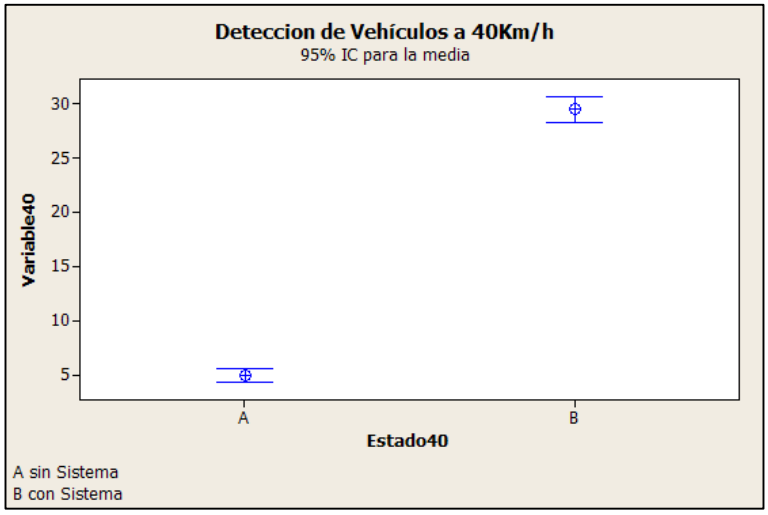
40Km/h

ANOVA unidireccional: Variable40 vs. Estado40

```
Fuente    GL        SC        CM        F      P
Estado40   1  1803,145  1803,145  2059,24  0,000
Error     10     8,756     0,876
Total     11  1811,902

S = 0,9358   R-cuad. = 99,52%   R-cuad.(ajustado) = 99,47%


                            ICs de 95% individuales para la media
                            basados en Desv.Est. agrupada
Nivel  N   Media  Desv.Est.  ----+---------+---------+---------+-----
A      6   4,948      0,610  (*)
B      6  29,464      1,175                                    (*)
                            ----+---------+---------+---------+-----
                              7,0      14,0      21,0      28,0

Desv.Est. agrupada = 0,936


Agrupar información utilizando el método de Tukey

Estado40  N   Media  Agrupación
B         6  29,464  A
A         6   4,948    B

Las medias que no comparten una letra son significativamente diferentes.


Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado40

Nivel de confianza individual = 95,00%


Estado40 = A restado de:

Estado40  Inferior  Centro  Superior  --+---------+---------+---------+-----
--
B           23,313  24,516    25,720                                    (-
*-)
                                      --+---------+---------+---------+-----
--
                                      0,0       7,0      14,0      21,0
```


Gráficas de residuos para Variable40
Gráfica de probabilidad normal
Porcentaje
Residuo
vs. ajustes
Residuo
Valor ajustado
Histograma
Frecuencia
Residuo
vs. orden
Residuo
Orden de observación

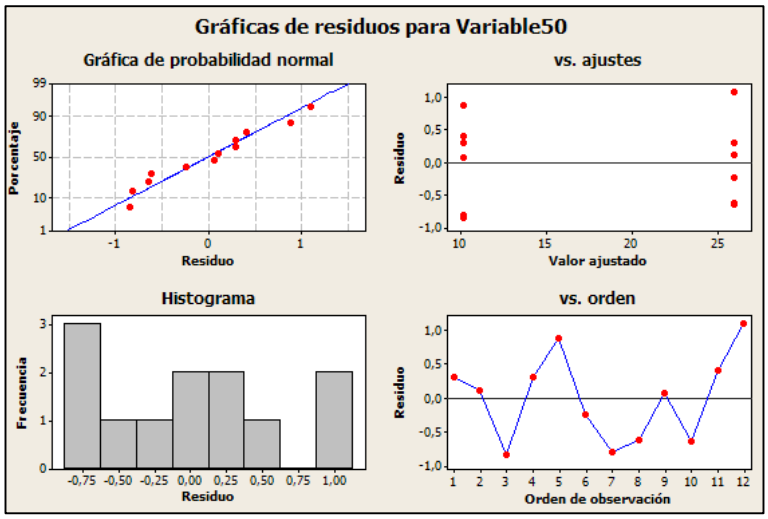

Deteccion de Vehículos a 40Km/h
Variable40
Estado40
A
B
A sin Sistema
B con Sistema

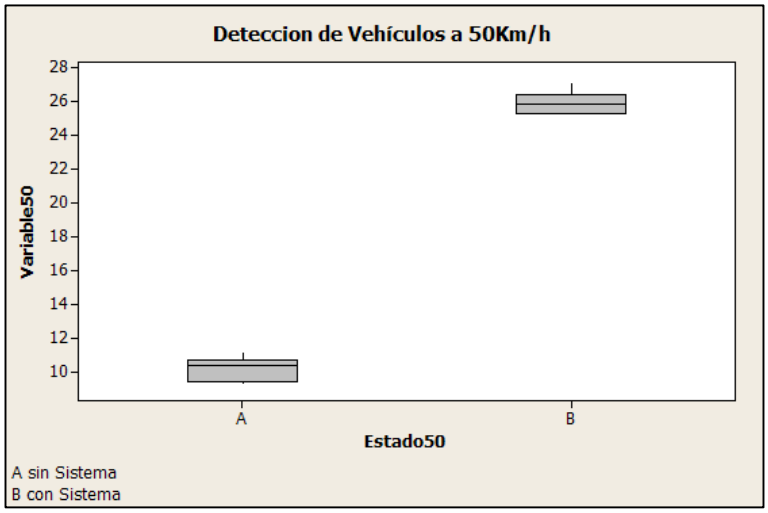

Deteccion de Vehículos a 40Km/h
95% IC para la media
Variable40
Estado40
A
B
A sin Sistema
B con Sistema

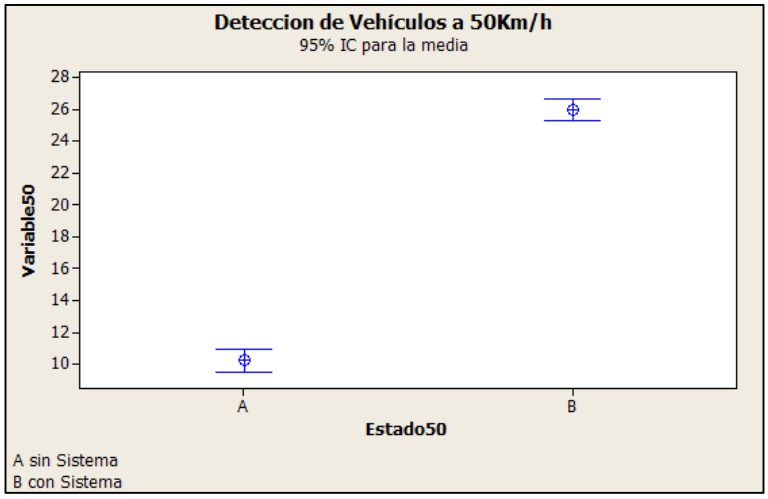
50Km/h

ANOVA unidireccional: Variable50 vs. Estado50

```
Fuente    GL       SC       CM        F      P
Estado50   1  743,229  743,229  1628,48  0,000
Error     10    4,564    0,456
Total     11  747,793

S = 0,6756   R-cuad. = 99,39%   R-cuad.(ajustado) = 99,33%


                            ICs de 95% individuales para la media
                            basados en Desv.Est. agrupada
Nivel  N   Media  Desv.Est.   -+---------+---------+---------+--------
A      6  10,220      0,694   (*-)
B      6  25,959      0,656                                  (*)
                              -+---------+---------+---------+--------
                            10,0      15,0      20,0      25,0

Desv.Est. agrupada = 0,676


Agrupar información utilizando el método de Tukey

Estado50  N   Media  Agrupación
B         6  25,959  A
A         6  10,220    B

Las medias que no comparten una letra son significativamente diferentes.


Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado50

Nivel de confianza individual = 95,00%


Estado50 = A restado de:

Estado50  Inferior  Centro  Superior  --+---------+---------+---------+-----
--
B           14,871  15,740    16,609                                 (*-)
                                    --+---------+---------+---------+-----
--
                                    0,0       5,0      10,0      15,0
```


Gráficas de residuos para Variable50
Gráfica de probabilidad normal
Porcentaje
Residuo
vs. ajustes
Residuo
Valor ajustado
Histograma
Frecuencia
Residuo
vs. orden
Residuo
Orden de observación

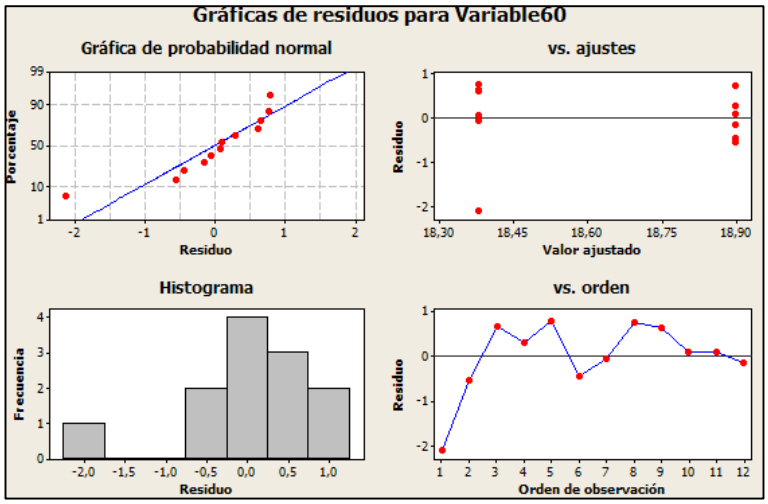

Deteccion de Vehículos a 50Km/h
Variable50
Estado50
A
B
A sin Sistema
B con Sistema

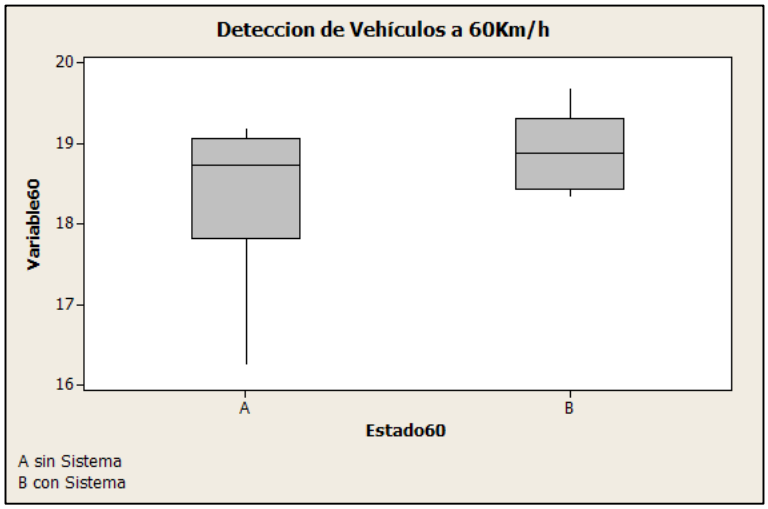

Deteccion de Vehículos a 50Km/h
95% IC para la media
Variable50
Estado50
A
B
A sin Sistema
B con Sistema

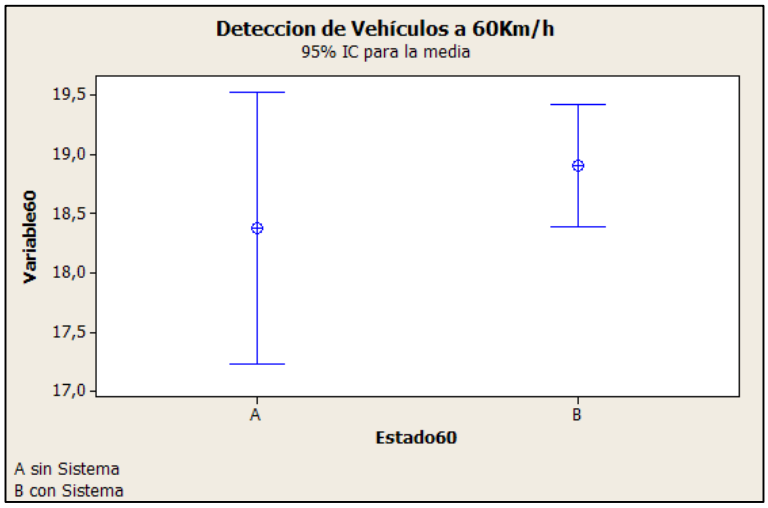
60Km/h

ANOVA unidireccional: Variable60 vs. Estado60

```
Fuente    GL     SC     CM     F      P
Estado60   1  0,813  0,813  1,14  0,311
Error     10  7,145  0,714
Total     11  7,958

S = 0,8453   R-cuad. = 10,22%   R-cuad.(ajustado) = 1,24%


                             ICs de 95% individuales para la media
                             basados en Desv.Est. agrupada
Nivel  N   Media  Desv.Est.  -------+---------+---------+---------+--
A      6  18,378      1,090  (------------*------------)
B      6  18,898      0,491           (------------*------------)
                             -------+---------+---------+---------+--
                                 18,00     18,60     19,20     19,80

Desv.Est. agrupada = 0,845


Agrupar información utilizando el método de Tukey

Estado60  N    Media  Agrupación
B         6  18,8984  A
A         6  18,3778  A

Las medias que no comparten una letra son significativamente diferentes.


Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado60

Nivel de confianza individual = 95,00%


Estado60 = A restado de:

Estado60  Inferior  Centro  Superior  ------+---------+---------+---------+-
--
B          -0,5667  0,5206    1,6080          (--------------*--------------
-)
                                     ------+---------+---------+---------+-
--
                                        -0,70      0,00      0,70      1,40
```


Gráficas de residuos para Variable60
Gráfica de probabilidad normal
Porcentaje
Residuo
vs. ajustes
Residuo
Valor ajustado
Histograma
Frecuencia
Residuo
vs. orden
Residuo
Orden de observación

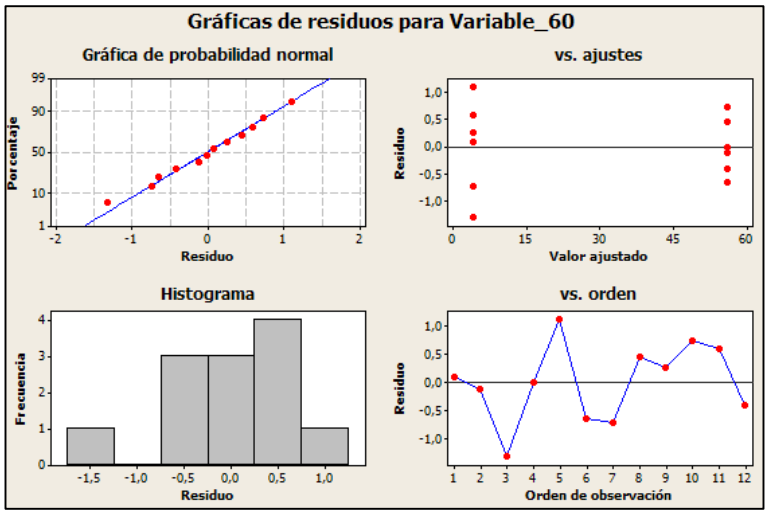

Deteccion de Vehículos a 60Km/h
Variable60
Estado60
A
B
A sin Sistema
B con Sistema

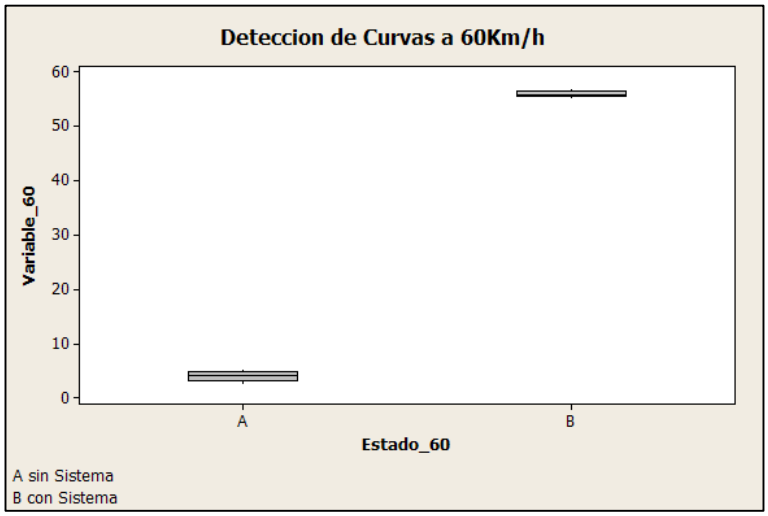

Deteccion de Vehículos a 60Km/h
95% IC para la media
Variable60
Estado60
A
B
A sin Sistema
B con Sistema

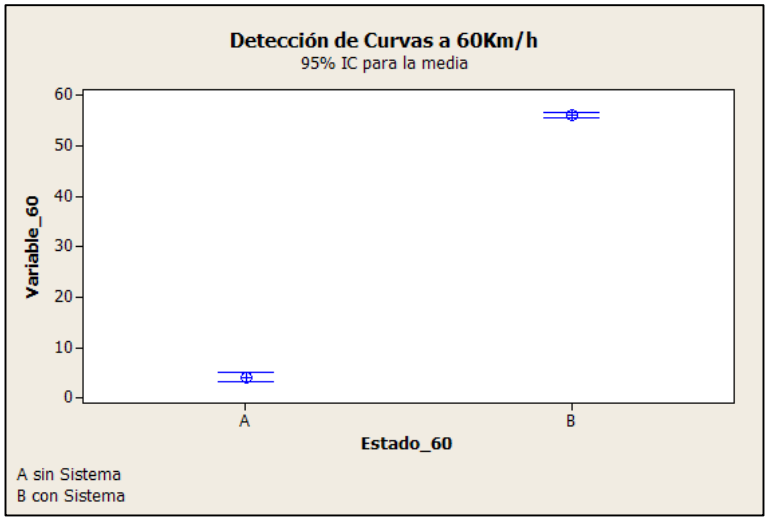
- **DETECCIÓN DE CURVAS**

60Km/h

```
Fuente     GL        SC        CM         F      P
Estado_60   1  8078,230  8078,230  15369,21  0,000
Error      10     5,256     0,526
Total      11  8083,487

S = 0,7250   R-cuad. = 99,93%   R-cuad.(ajustado) = 99,93%


                            ICs de 95% individuales para la media
                            basados en Desv.Est. agrupada
Nivel  N   Media  Desv.Est. --------+---------+---------+---------+-
A      6   4,083      0,885  (*
B      6  55,975      0,518                                  *)
                            --------+---------+---------+---------+-
                                   15        30        45        60

Desv.Est. agrupada = 0,725


Agrupar información utilizando el método de Tukey

Estado_60  N   Media  Agrupación
B          6  55,975  A
A          6   4,083    B

Las medias que no comparten una letra son significativamente diferentes.


Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado_60

Nivel de confianza individual = 95,00%


Estado_60 = A restado de:

Estado_60  Inferior  Centro  Superior  -+---------+---------+---------+-----
---
B            50,959  51,892    52,824                                    (*
                                     -+---------+---------+---------+-----
---
                                      0        15        30        45
```


Gráficas de residuos para Variable_60
Gráfica de probabilidad normal
Porcentaje
Residuo
vs. ajustes
Residuo
Valor ajustado
Histograma
Frecuencia
Residuo
vs. orden
Residuo
Orden de observación

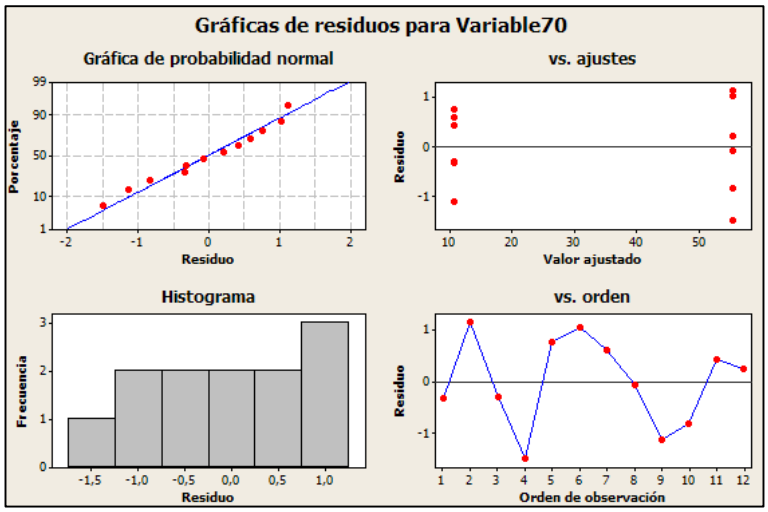

Deteccion de Curvas a 60Km/h
Variable_60
Estado_60
A
B
A sin Sistema
B con Sistema

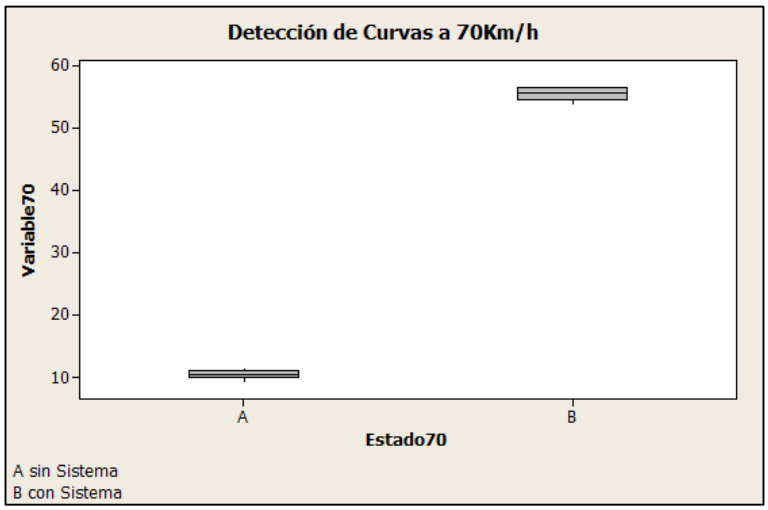

Detección de Curvas a 60Km/h
95% IC para la media
Variable_60
Estado_60
A
B
A sin Sistema
B con Sistema

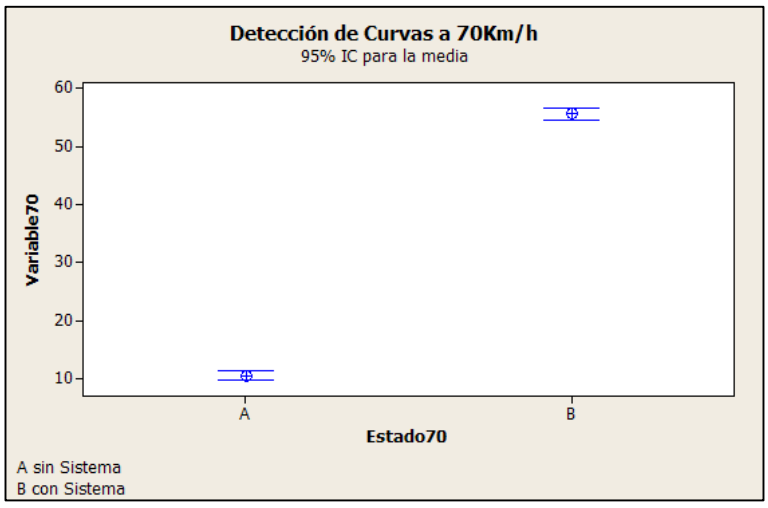
70Km/h

ANOVA unidireccional: Variable70 vs. Estado70

```
Fuente    GL        SC        CM        F      P
Estado70   1  6083,317  6083,317  7698,90  0,000
Error     10     7,902     0,790
Total     11  6091,218

S = 0,8889   R-cuad. = 99,87%   R-cuad.(ajustado) = 99,86%


                            ICs de 95% individuales para la media
                            basados en Desv.Est. agrupada
Nivel  N   Media  Desv.Est.  --+---------+---------+---------+-------
A      6  10,520      0,717  (*
B      6  55,551      1,032                                       *)
                            --+---------+---------+---------+-------
                             12        24        36        48

Desv.Est. agrupada = 0,889


Agrupar información utilizando el método de Tukey

Estado70  N   Media  Agrupación
B         6  55,551  A
A         6  10,520    B

Las medias que no comparten una letra son significativamente diferentes.


Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado70

Nivel de confianza individual = 95,00%


Estado70 = A restado de:

Estado70  Inferior  Centro  Superior  -+---------+---------+---------+------
--
B           43,887  45,031    46,174
(*
                                     -+---------+---------+---------+------
--
                                      0        12        24        36
```


Gráficas de residuos para Variable70
Gráfica de probabilidad normal
Porcentaje
Residuo
vs. ajustes
Residuo
Valor ajustado
Histograma
Frecuencia
Residuo
vs. orden
Residuo
Orden de observación

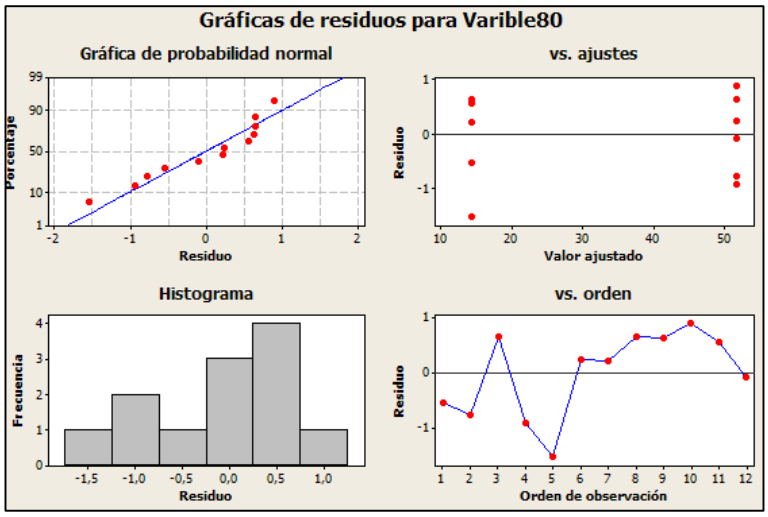

Detección de Curvas a 70Km/h
Variable70
Estado70
A
B
A sin Sistema
B con Sistema

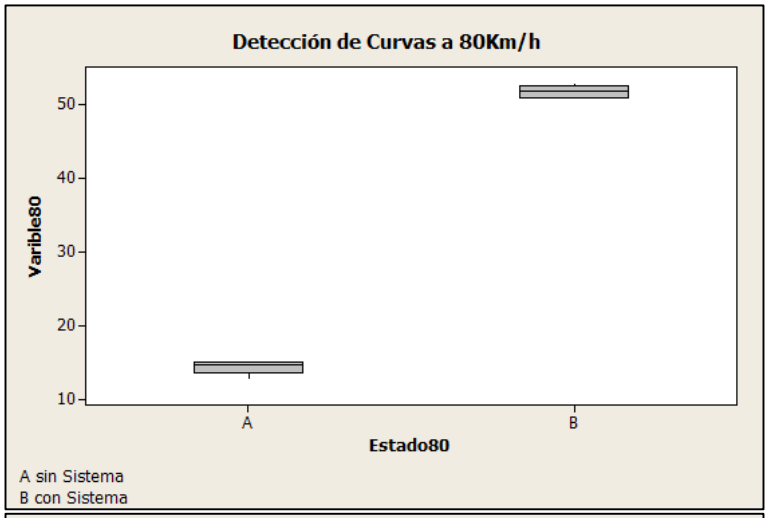

Detección de Curvas a 70Km/h
95% IC para la media
Variable70
Estado70
A
B
A sin Sistema
B con Sistema

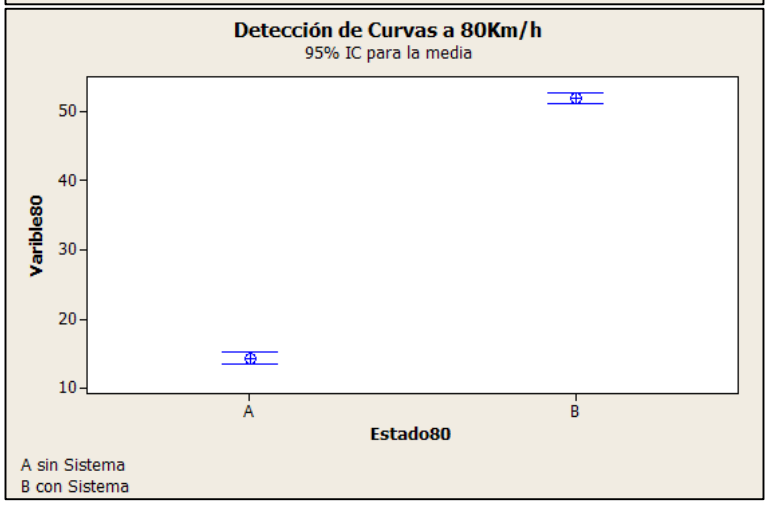
80Km/h

ANOVA unidireccional: Varible80 vs. Estado80

```
Fuente    GL        SC        CM        F      P
Estado80   1  4228,023  4228,023  6407,98  0,000
Error     10     6,598     0,660
Total     11  4234,621

S = 0,8123   R-cuad. = 99,84%   R-cuad.(ajustado) = 99,83%


                             ICs de 95% individuales para la media
                             basados en Desv.Est. agrupada
Nivel  N   Media  Desv.Est.  ------+---------+---------+---------+---
A      6  14,311      0,877  *)
B      6  51,852      0,742                                      (*)
                             ------+---------+---------+---------+---
                                  20        30        40        50

Desv.Est. agrupada = 0,812


Agrupar información utilizando el método de Tukey

Estado80  N   Media  Agrupación
B         6  51,852  A
A         6  14,311    B

Las medias que no comparten una letra son significativamente diferentes.


Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado80

Nivel de confianza individual = 95,00%


Estado80 = A restado de:

Estado80  Inferior  Centro  Superior  -+---------+---------+---------+------
--
B           36,496  37,541    38,586
(-*)
                                    -+---------+---------+---------+------
--
                                     0        10        20        30
```


Gráficas de residuos para Varible80
Gráfica de probabilidad normal
Porcentaje
Residuo
vs. ajustes
Residuo
Valor ajustado
Histograma
Frecuencia
Residuo
vs. orden
Residuo
Orden de observación

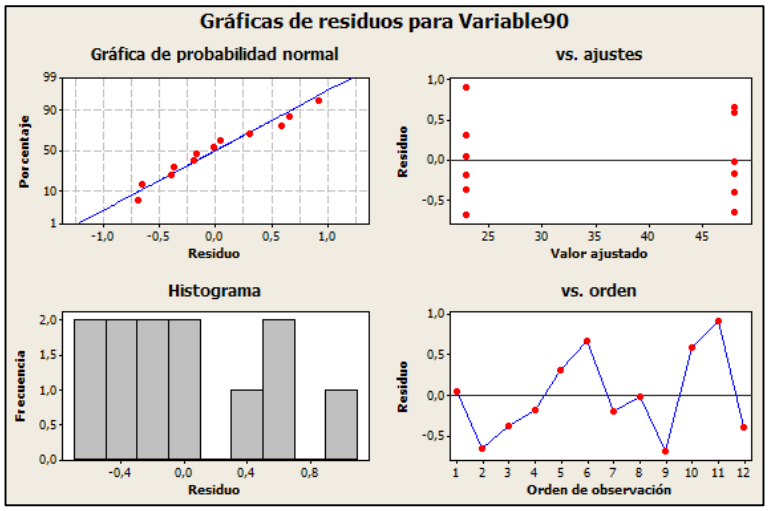

Detección de Curvas a 80Km/h
Varible80
Estado80
A
B
A sin Sistema
B con Sistema

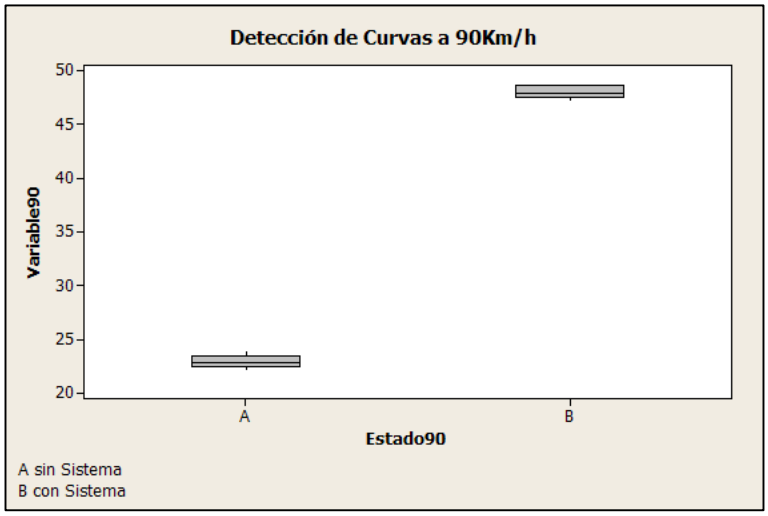

Detección de Curvas a 80Km/h
95% IC para la media
Varible80
Estado80
A
B
A sin Sistema
B con Sistema

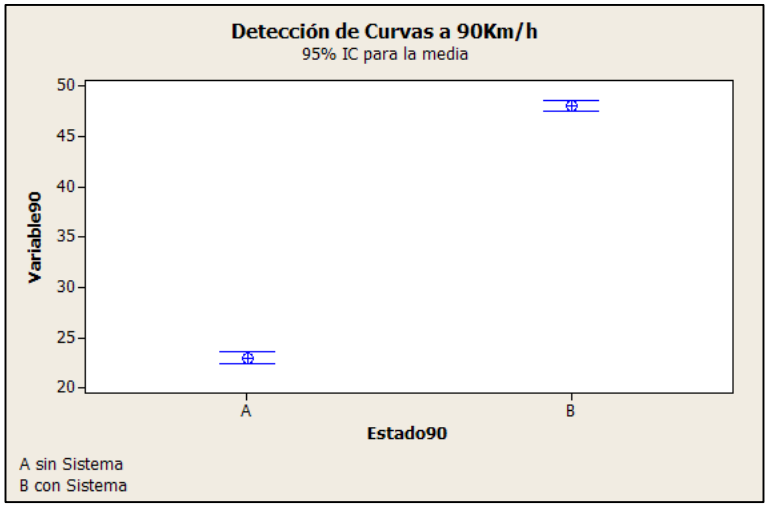
90Km/h

ANOVA unidireccional: Variable90 vs. Estado90

```
Fuente    GL        SC        CM        F      P
Estado90   1  1886,046  1886,046  6223,37  0,000
Error     10     3,031     0,303
Total     11  1889,076

S = 0,5505   R-cuad. = 99,84%   R-cuad.(ajustado) = 99,82%


                             ICs de 95% individuales para la media
                             basados en Desv.Est. agrupada
Nivel  N   Media  Desv.Est.  --------+---------+---------+---------+-
A      6  22,928      0,568  (*
B      6  48,002      0,533                                  (*
                             --------+---------+---------+---------+-
                                  28,0      35,0      42,0      49,0

Desv.Est. agrupada = 0,551


Agrupar información utilizando el método de Tukey

Estado90  N   Media  Agrupación
B         6  48,002  A
A         6  22,928    B

Las medias que no comparten una letra son significativamente diferentes.


Intervalos de confianza simultáneos de Tukey del 95%
Todas las comparaciones de dos a dos entre los niveles de Estado90

Nivel de confianza individual = 95,00%


Estado90 = A restado de:

Estado90  Inferior  Centro  Superior   -+---------+---------+---------+-----
---
B           24,365  25,074    25,782
(*)
                                      -+---------+---------+---------+-----
---
                                     0,0       7,0      14,0      21,0
```


Gráficas de residuos para Variable90
Gráfica de probabilidad normal
Porcentaje
Residuo
vs. ajustes
Residuo
Valor ajustado
Histograma
Frecuencia
Residuo
vs. orden
Residuo
Orden de observación


Detección de Curvas a 90Km/h
Variable90
Estado90
A
B
A sin Sistema
B con Sistema


Detección de Curvas a 90Km/h
95% IC para la media
Variable90
Estado90
A
B
A sin Sistema
B con Sistema