DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN PARA LA DETECCIÓN Y RECONOCIMIENTO DE SEÑALES DE LIMITACIÓN DE VELOCIDAD

DESIGN AND IMPLEMENTATION OF AN APPLICATION FOR THE DETECTION AND RECOGNITION OF SPEED LIMIT SIGNS

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN PARA LA DETECCIÓN Y RECONOCIMIENTO DE SEÑALES DE LIMITACIÓN DE VELOCIDAD

DESIGN AND IMPLEMENTATION OF AN APPLICATION FOR THE DETECTION AND RECOGNITION OF SPEED LIMIT SIGNS

Roberto Alejandro Larrea Luzuriaga[1],
Cristina Alejandra Orozco Cazco[2],

[1] Instituto Superior Universitario Carlos Cisneros, Ecuador, roberto.larrea@istcarloscisneros.edu.ec
[2] Instituto Superior Universitario Carlos Cisneros, Ecuador, cristina.orozco@istcarloscisneros.edu.ec

RESUMEN

Se realizó un análisis a la aplicación diseñada para brindar una solución a nivel de asistencia en la conducción, respecto a la detección oportuna de señales de tránsito, específicamente a las señales de velocidad, que permitan una respuesta oportuna por parte del conductor. El objetivo del artículo es presentar el diseño, la implementación y la validación de una aplicación funcional capaz de detectar y reconocer señales de limitación de velocidad a partir archivos de imagen, de video o por medio de un flujo de video en tiempo real. La metodología utilizada consiste en 4 procesos que establecen la adquisición de la imagen, procesamiento de la imagen y video, detección y control de la señal y velocidad, y visualización de resultados. El desarrollo de la aplicación se lo realizó en Eclipse en código C++, utilizando librerías para el tratamiento de imágenes y video de CImg y OpenCV. Con base a los resultados obtenidos de la aplicación implementada, se ha logrado óptimas prestaciones, es decir, se tiene una certeza de casi el 90% y un error alrededor del 1%. Además, se ha analizado dificultades y errores de detección, así como también se estableció estadísticas de tiempo de procesamiento en relación a las características del hardware utilizado.

Palabras clave: Tratamiento de imagen y video, Señal velocidad, Visión Artificial, CImg, OpenCV.

ABSTRACT

An analysis was conducted on the application designed to provide a driving assistance solution for the timely detection of traffic signs, specifically speed signs, enabling a timely response by the driver. The objective of this article is to present the design, implementation, and validation of a functional application capable of detecting and recognizing speed limit signs from image or video files or a real-time video stream. The methodology used consists of four processes: image acquisition, image and video processing, sign and speed detection and control, and results visualization. The application was developed in Eclipse using C++ code, using Cimg and OpenCV image and video processing libraries. Based on the results obtained from the implemented application, optimal performance was achieved, with an accuracy of almost 90% and an error rate of around 1%. In addition, detection difficulties and errors were analyzed, and processing time statistics were established in relation to the characteristics of the hardware used.

Keywords: *Image and video processing, Velocity signal, Machine Vision, Cimg, OpenCV*

Recibido: Agosto 2025 Aceptado: Diciembre 2025
Received: August 2025 *Accepted: December 2025*

1. INTRODUCCIÓN

La creciente demanda por Sistemas Avanzados de Asistencia al Conductor (ADAS, por sus siglas en inglés) ha impulsado el desarrollo de tecnologías capaces de reconocer señales de tránsito en tiempo real, con el propósito de mejorar la seguridad vehicular y reducir accidentes causados por exceso de velocidad. En este contexto, esta investigación aborda el diseño e implementación de una aplicación usando procesamiento de imágenes con CImg, visión por computadora con OpenCV y programación en C++ para detectar y reconocer señales de limitación de velocidad, presentándolas anticipadamente al conductor como un asistente visual.

La seguridad vial representa uno de los mayores desafíos en las sociedades modernas, donde el incremento del parque automotor y la complejidad de las redes de carreteras exigen el desarrollo de tecnologías innovadoras para la prevención de accidentes. Una de las principales causas de siniestralidad vial es el exceso de velocidad, a menudo producto de la distracción o del desconocimiento de los límites específicos en un tramo determinado [1]. En este contexto, los Sistemas ADAS han emergido como una solución tecnológica fundamental para mitigar los riesgos asociados a la conducción [2]. Estos sistemas, que integran diversas tecnologías como sensores, cámaras y algoritmos de procesamiento, buscan mejorar la interacción entre el conductor, el vehículo y su entorno.

La presente investigación se ha realizado con el propósito de explorar y contribuir al desarrollo de aplicaciones en tiempo real que permitan asistir activamente a la conducción vehicular. Específicamente, se enfoca en el reconocimiento automático de señales de tránsito, con particular interés en las de limitación de velocidad, mediante la aplicación de técnicas de procesamiento digital de imágenes y video. Para ello, se utiliza la biblioteca de procesamiento de imágenes CImg (Digital Image Processing), y de la biblioteca de visión por computador OpenCV (Open Source Computer Vision Library), ambas de código abierto, las cuales ofrecen un robusto conjunto de herramientas para el análisis de imágenes en tiempo real, siendo ampliamente adoptadas tanto en la industria como en la academia [3], [4].

El problema central que se pretende resolver es la reducción de la incidencia de accidentes de tránsito causados por el exceso de velocidad en calles y carreteras urbanas. Se busca abordar esta problemática mediante el diseño y la implementación de una herramienta de asistencia visual que permita el procesamiento en tiempo real de imágenes y vídeo capturados por una cámara frontal, identificando señales de límite de velocidad y notificando al conductor oportunamente. Esto se fundamenta en la premisa de que una intervención oportuna puede fomentar comportamientos de conducción más seguros y reducir la incidencia de accidentes por velocidad excesiva. A diferencia de los sistemas basados en GPS, que pueden tener información desactualizada o imprecisa, un sistema basado en visión por computador ofrece información contextual y dinámica, directamente extraída del entorno vial [5].

El estado del arte en el reconocimiento de señales de tránsito (TSR, por sus siglas en inglés) ha experimentado avances significativos en la última década. Las investigaciones iniciales se centraron en métodos basados en color y forma para la segmentación y detección de señales [6]. Trabajos como el de Stallkamp et al. [7] sentaron las bases al crear benchmarks para la comparación de algoritmos de clasificación. Posteriormente, el enfoque ha migrado hacia el uso de técnicas de aprendizaje automático y, más recientemente, de aprendizaje profundo (deep learning). Arquitecturas como las Redes Neuronales Convolucionales (CNN) han demostrado una precisión sobresaliente en la clasificación de señales de tránsito, incluso en condiciones de iluminación variable, oclusión parcial o deformación de la señal [8], [9]. No obstante, la implementación de estos modelos en sistemas embebidos o de bajo costo para aplicaciones en tiempo real sigue siendo un desafío activo debido a su alta demanda computacional [10]. Por ello, la optimización de algoritmos clásicos de visión artificial, como los que ofrece OpenCV para la detección de características (ej. Haar Cascades, HOG) y el reconocimiento de formas, sigue siendo un área de investigación relevante y con aplicabilidad práctica directa en sistemas de asistencia al conductor [4], [5].

El objetivo de este artículo es, por lo tanto, presentar el diseño, la implementación y la validación de una aplicación funcional capaz de detectar y reconocer señales de limitación de velocidad a partir de un flujo de video en tiempo real. El sistema propuesto se

basa en una cascada de algoritmos de procesamiento de imágenes implementados en C++ con el apoyo de las bibliotecas de CImg y OpenCV, buscando un equilibrio entre la precisión del reconocimiento y la eficiencia computacional necesaria para su operación en un entorno de conducción real.

2. METODOLOGÍA Y MATERIALES

La metodología utilizada en el desarrollo de la aplicación se basó en cuatro procesos (Fig.1), estableciéndose varias etapas en los procesos 2 al 4, para su diseño y funcionamiento.

1. Adquisición de la imagen
2. Procesamiento y tratamiento de la imagen.
3. Detección y control de señal de velocidad.
4. Visualización de resultados.

La aplicación fue desarrollada en la plataforma Eclipse en código C++, instalada sobre una máquina virtual en Virtual Box con sistema operativo Linux Ubuntu 12.04.

Fig.1: Diagrama de bloques de la aplicación desarrollada

1.- Adquisición de la imagen

El proceso de adquisición de imágenes estableció la utilización de una cámara de video, que permitió en tiempo real la captura de imágenes, procesadas por la aplicación para la detección y visualización de señales de velocidad. Sin embargo, en este estudio se utilizó imágenes de entrenamiento y videos que permitieron ajustar parámetros los demás procesos y establecer cierta información útil en relación a su funcionamiento.

2.- Procesamiento y tratamiento de la imagen

Etapa 1.- Seleccionar componente Roja de Imagen

Para la detección se eligió la componente roja de la imagen RGB, de modo que la circunferencia que encierra la señal de velocidad pueda ser eliminada en las siguientes etapas.

Etapa 2.- Realizar un Cierre de la imagen

Para la ejecución del cierre que contempló realizar una dilatación y luego un cierre, se utilizó un elemento estructurante horizontal de tamaño inicial de largo 26 píxeles y ancho 1 pixel, de modo que se mantenga la mayor cantidad de elementos menores al tamaño del elemento estructurante.

Etapa 3.- Realizar una diferencia del cierre y la imagen en gris

Con la diferencia del cierre y la imagen en gris se eliminaron la mayor cantidad de elementos, de tamaño mayor al elemento estructurante.

Etapa 4.- Aplicar un umbral para la binarización de la imagen

Para la obtención del umbral se sacó un promedio de pixeles de la imagen y se le sumó una constante de valor 25 para efectos de corrección. Posterior con dicho umbral se realizó la binarización de la imagen obteniendo una imagen en blanco y negro.

Etapa 5.- Aplicar una segmentación

Sobre la imagen binaria se realizó la segmentación, de este modo, se detectó y etiquetó a cada uno de los elementos que constituyen un objeto en la imagen.

Etapa 6.- Aplicar un filtrado

Para la eliminación de la mayor cantidad objetos que no constituyen un valor numérico, se ha obtenido el BoundingBox, así como también la orientación y elipticidad de cada objeto obtenido en la etapa anterior. Dichos elementos obtenidos de cada objeto fueron necesarios para la implementación del filtro. De esta manera, los objetos que pasaron el filtrado son solo los objetos que posean las características de un número de señal de velocidad. Para el filtro se utilizó criterios de relación aspecto, tamaño, elipticidad, razón de altura por anchura de los posibles números, espaciado de carácter y carácter, y ángulo del objeto con respecto a su ubicación sobre la imagen.

3.- Detección y control de la señal de velocidad

Etapa 7.- Seleccionar los objetos que constituyen una señal de velocidad

Después del filtrado se eliminó un 95% de objetos que no constituyen un posible número. En esta fase se seleccionó los elementos que constituyen una cantidad de 2 o 3 cifras, con el criterio de proximidad de las unidades (posibles números) y búsqueda de izquierda a derecha. Por medio de una función de detección de números de señales de velocidad se realizó la detección de cada unidad, construyendo la cantidad de velocidad detectada. Aquí también, se realizó un control de los posibles valores que se debería obtener (que existen a nivel de señales de tránsito), ya que en caso de detectar un valor que no esté acorde a una cantidad valida de señal de velocidad, se descartó.

Etapa 8.- Seleccionar la cantidad de la señal de velocidad mayor

Después de la fase de detección se obtuvo un vector con varias cantidades de velocidad, se aplicó un algoritmo para la detección del mayor valor.

Etapa 9.- Localizar la señal de velocidad detectada en la imagen

Para la señalización de la imagen de velocidad, o extracción, luego de obtener la cantidad de mayor velocidad, y utilizando los componentes (etiquetas, BoundingBox, Orientación, Centro de gravedad) de cada uno de los números que constituyen la cifra en decenas o centenas, se calculó los vértices de un rectángulo que encierre la cantidad de la señal de velocidad.

4.- Visualización de resultados

Etapa 10.- Detección de velocidad para una imagen, un conjunto de imágenes, cámara de video y video.

Para cada una de las opciones de detección se utilizó las funciones creadas, de modo que se visualicen los resultados.

Para las opciones de video y cámara, se utilizó un efecto de degradación d la última imagen detectada hasta que se detecte una nueva.

Para la opción de múltiples imágenes se obtuvo un fichero con los valores de velocidad detectados de cada imagen procesada.

Para cada imagen se procesará hasta con tres valores posibles de elemento estructurado de modo que la señal de velocidad pueda ser detectada a lo lejos y muy cerca.

Esquematización de la aplicación, funciones implementadas y paso de variables de entrada y salida

En la Tabla 1 se describen las funciones implementadas, su operación dentro de las etapas antes mencionadas, así como también los parámetros de entrada y salida establecidos.

Tabla 1. Descripción de funciones implementadas y su operación

Función	`void ocr_number(cimg_library::CImg<float> & vectores)`
Operaciones que realiza	Carga en memoria el entrenador para la detección de los números
Parámetros de Entrada	-
Parámetros de Salida	Vectores
Función	`int number(const cimg_library::CImg<int> & seg, cimg_library::CImg<float> & vectores, int label)`
Operaciones que realiza	Detecta el posible valor (número) del posible objeto señal de velocidad.
Parámetros de Entrada	Imagen segmentada seg, vectores.
Parámetros de Salida	Valor numérico entero.
Función	`void tpimgvelocidad(const cimg_library::CImg<unsigned char> & input, int ee, cimg_library::CImg<int> &output)`
Operaciones que realiza	1. Seleccionar componente roja de la imagen 2. Realizar un cierre de la imagen 3. Realizar un diferencial del cierre y la imagen en gris 4. Aplicar un umbral para la binarización de la imagen 5. Segmentación 6. Filtrado
Parámetros de Entrada	Imagen a detectar input, Tamaño elemento estructurante ee.

Parámetros de Salida	Imagen filtrada y segmentada output
Función	`void detectvelocidad(const cimg_library::CImg<int> & segm, cimg_library::CImg<float> & vectores, int &vmax, int &ax, int &ay, int &a1x, int &a1y, int &bx, int &by, int &b1x, int &b1y )`
Operaciones que realiza	7. Selección de los objetos que constituyen una señal de velocidad, Detección y Control. 8. Selección de Cantidad de Señal de velocidad mayor 9. localización de la señal de velocidad detectada en la imagen
Parámetros de Entrada	Imagen filtrada segmentada segm, Vectores. Valor de velocidad máximo detectado vmax,
Parámetros de Salida	Coordenadas de los vértices del rectángulo que localiza la señal de velocidad detectada en la imagen de entrada a(ax,ay), b(bx,by), a1(a1x,a1y), b1(b1x,b1y).
Aplicación	**speed**
Operaciones que realiza	10. Detección de velocidad para una imagen, un conjunto de imágenes, cámara de video y video.
Parámetros de Entrada	Opciones: - Imagen - Archivo .txt con path de conjunto de imágenes - 0 (Cámara de video) - Video
Parámetros de Salida	-Imagen que resalta la señal de velocidad detectada con recuadro de la misma, valor numérico en ventana. -Archivo .txt con valor numérico de la señal de velocidad detectada de cada una de las imágenes del archivo de entrada del conjunto de imágenes. -Para cámara y video, Imagen que resalta la señal de velocidad detectada con recuadro de la misma, valor numérico en ventana. El recuadro de la señal de velocidad detectada presenta u efecto de difumina miento de la última señal detectada.

3. RESULTADOS

Posterior al desarrollo de la aplicación y evaluación de su funcionamiento, los resultados paso a paso del proceso establecido para la detección y reconocimiento de señales de velocidad, se describe cada resultado obtenido en cada proceso desarrollado y ejecutado en etapas, con relación al procesamiento de una imagen desde su adquisición, posterior procesamiento, detección y obtención de la velocidad máxima en pantalla.

1. Adquisición de la imagen

En la Fig. 2 se observa una imagen de archivo fotográfico utilizada en la implementación de la aplicación, que constituye una carretera con señalización de límite de velocidad.

Fig.2: Archivo fotográfico para la detección de señales de velocidad en carretera.

2. Procesamiento y tratamiento de la imagen

En la Fig. 3 se observa el resultado de aplicar la selección de la componente Roja de la Imagen.

Fig.3. Selección de la componente roja de la imagen.

En la Fig. 4 se observa el resultado de realizar el cierre de la imagen con un elemento estructurante de tamaño 26X1 pixeles.

Fig.4. Cierre de la imagen

En la Fig. 5 se observa el resultado de realizar una diferencia del cierre y la imagen en gris.

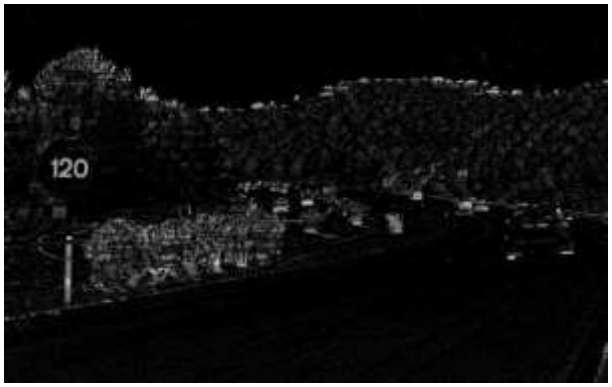

Fig.5. Diferencia del cierre y la imagen en gris

En la Fig. 6 se observa el resultado de aplicar un Umbral = 25, para la binarización de la imagen.

Fig.6. Binarización de la imagen aplicado un umbral

En la Fig. 7 se observa la realización de segmentación de la imagen estableciéndose 2112 objetos.

Fig.7. Segmentación de la imagen

En la Fig. 8 se observa la realización de filtrado de la imagen reduciéndose el número de objetos a 8.

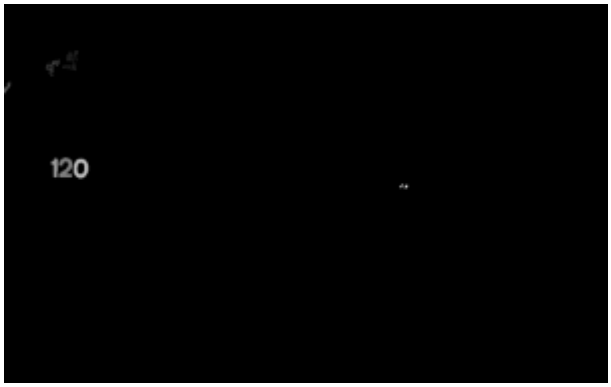

Fig.8. Filtrado de la imagen

3. Detección y control de la señal de velocidad

En la Fig. 9 se observa el resultado del proceso de detección y control de la señal de velocidad, en este proceso con el vector de objetos resultantes, de cada uno se aplica una selección sobre los objetos que cumple ciertas condiciones para considerarse números, cantidades válidas en relación a la detección y validación de izquierda a derecha, que constituyen una señal de velocidad. De todas las cantidades de velocidad, detectadas, validadas y almacenadas en un vector, se establece como resultado la de mayor valor. Además, posterior a su detección, se procede a ubicar la señal de velocidad en la imagen con una figura rectangular de color rojo.

Fig.9. Selección, validación, detección de objetos como señal de mayor velocidad y ubicación de la misma en la imagen.

4. Visualización de resultados

En la Fig. 10 se observa el resultado de la aplicación, detección de velocidad para una imagen, un conjunto de imágenes, cámara de video y video, mostrándose en recuadro la mayor velocidad detectada, dicho recuadro se ha recortado y se muestra a la derecha de la imagen, además, de incorporar una leyenda con la velocidad máxima de 120Km/h.

Velocidad Máxima= 120Km/h

Fig.10. Detección de velocidad para una imagen, un conjunto de imágenes, cámara de video y video.

Evaluación de Prestaciones

Como parte de los resultados se estableció una evaluación de las prestaciones del funcionamiento de la aplicación, teniendo en cuenta que la misma se ejecutó sobre una máquina virtual en Virtual Box con sistema operativo Linux Ubuntu 12.04.

Las especificaciones de la máquina anfitrión:

- Sistema Operativo: Windows 8.1 de 64 bits
- Procesador: Intel Core i7 de 1.8Ghz con turbo boost hasta 3Ghz
- Memoria RAM: 16GB

Las especificaciones de la máquina huésped:

- Sistema Operativo: Linux Ubuntu 12.04
- Procesador: 3 Núcleos
- Memoria RAM: 3GB
-

En la misma aplicación se desarrolló un algoritmo, que permitió medir el tiempo de procesamiento de cada imagen, estableciéndose las siguientes estadísticas en relación a la resolución de la imagen, como se observa en la Tabla 2. Como era de esperarse el menor tiempo de procesamiento para la detección de la señal de velocidad es para la imagen con menor resolución de 800x600 pixeles, siendo de 0.1 segundos.

Tabla 2. Tiempo de procesamiento y detección en relación a la resolución de la imagen

Imagen	Tiempo de procesamiento y detección
Tipo: jpeg (The JPEG image format) Resolución: 800X600 Tamaño: 207.6KB	0.1 segundos
Tipo: jpeg (The JPEG image format) Resolución: 1024X642 Tamaño: 88.4KB	0.39 segundos
Tipo: jpeg (The JPEG image format) Resolución: 3008X2000 Tamaño: 3.8MB	21.26 segundos

Además, con un banco de imágenes se estableció una estadística a nivel de precisión en la detección de la señal de velocidad, tal como se observa en la Tabla 3. Para este cálculo se utilizó 58 imágenes de entrenamiento de velocidad, dentro de las cuales se añadió imágenes de velocidad hasta 3 dígitos. Los resultados de esta estadística muestran un porcentaje de aciertos del 89.65%, en relación a errores con un porcentaje de 1.72% y con un porcentaje de 8.62% de no detección.

Tabla 3. Precisión en la detección de la señal de velocidad

Número de Imágenes	Detecta Bien	No Detecta	Detecta Mal
58	52	5	1
100%	89.65%	8.62%	1.72%

4. DISCUSIÓN (O ANÁLISIS DE RESULTADOS)

En esta sección se establece un análisis en relación a las imágenes que no fueron detectadas por la aplicación, como dificultades, y también a las detecciones erróneas.

Dificultades

En la Fig. 11 se analiza y se encuentra que la dificultad, es la perspectiva de la señal, lo que hace que la elipticidad sea diferente a los rangos aplicados en el filtro.

Fig.11. Imagen de señal de velocidad en perspectiva

En la Fig. 12 la señal de velocidad está muy alejada, al segmentar la parte del disco de la señal de velocidad está unida al dígito 6, lo que hace que en el filtro se elimine y no se detecte nada.

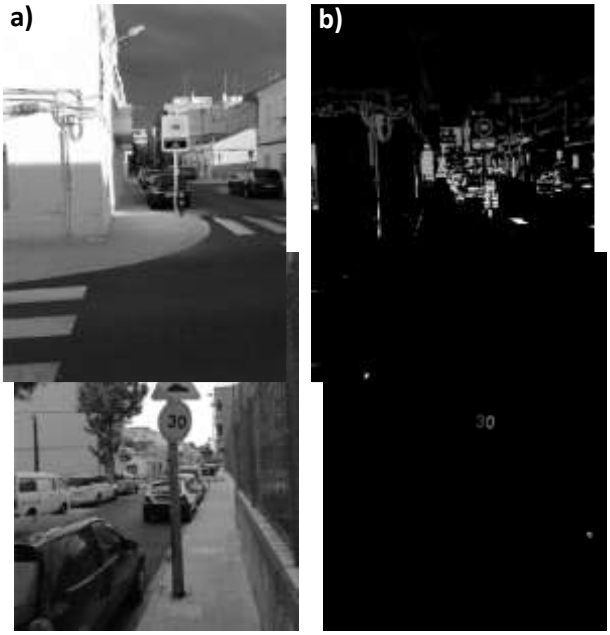

Fig.12. a) Imagen de señal de velocidad muy alejada. b) Segmentación de la imagen

En la Fig. 13 la señal presenta una leve inclinación hacia la derecha lo que hace que en fase de detección y control se elimine por elipticidad. La segmentación es correcta.

Fig.13. a) Imagen de señal de velocidad inclinada hacia la derecha. b) Segmentación de la imagen

En la Fig. 14 la señal se encuentra bastante lejos de manera que la separación de los dígitos de la señal es bastante pequeña, lo que produce que, al momento de segmentar los dos dígitos constituyen un mismo objeto.

Fig.14. a) Imagen de señal de muy alejada. b) Segmentación de la imagen

Detecciones erróneas

En la Fig. 15 el error de detección en esta imagen se debe a que se pasa dos letras de un cartel, al aumentar el valor del elemento estructurante, de modo que el detector encuentra una cifra mayor a la de la señal de velocidad existente también detectada, pasa del filtro la letra O y U detectándose un 90 mayor a los 40 de la señal de velocidad (Fig. 16). Cambiando los valores del filtro (eliminar objetos con valores mínimos) se logra liminar el error, pero con la consecuencia de que en otras imágenes dejan de ser detectadas, por lo que se mantiene los valores del filtro.

Fig.15. Imagen con error de detección

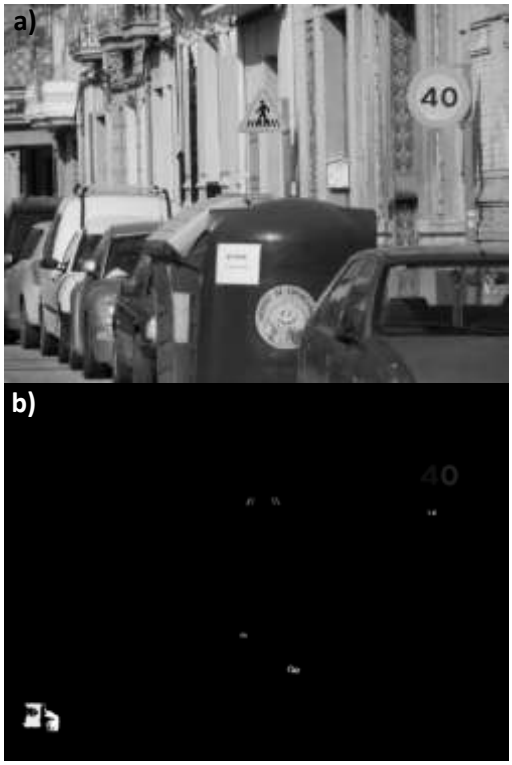

Fig.16. a) Imagen con carteles y señal de velocidad. b) Segmentación de la imagen.

5. CONCLUSIÓN

El tratamiento y procesamiento de imágenes y video por medio de la utilización de librerías de código abierto disponibles tanto en CImg como en OpenCV, permiten la implementación de aplicaciones de detección que constituyen un elemento muy importante a la hora de desarrollar sistemas que evalúan en tiempo real imágenes y extraer y visualizar información relevante o de interés.

El campo de acción de un sistema de detección de señales de velocidad, en el área automovilista incrementa las opciones de dar inteligencia de reacción o información adicional e importante al conductor, o en un sistema más autónomo de control de velocidad.

Las herramientas utilizadas en el diseño e implementación de la aplicación, da múltiples posibilidades para resolver problemas, innovar y crear nuevas aplicaciones que puedan ser de utilidad.

Con respecto al sistema implementado, se obtuvo prestaciones óptimas respecto al 58 imágenes analizadas, es decir, detección idónea de las imágenes es de alrededor del 90%, un 8% no las identifica y un error alrededor del 2% que corresponde a una mala detección de las imágenes.

Finalmente se puede concluir que se ha diseñado un prototipo de aplicación que puede ser mejorado y optimizado. Los próximos pasos a seguir, consiste en mejorar el prototipo y experimentar otras opciones que permitan aumentar su eficiencia y uso útil en un ambiente real.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] World Health Organization, *Global status report on road safety 2018*. Geneva, Switzerland: WHO, 2018.

[2] A. Ziebinski, R. Cupek, D. Grzejszczak, and M. Podpora, "A Survey of Advanced Driver-Assistance Systems (ADAS)," *2016 International Conference on Signals and Electronic Systems (ICSES)*, Krakow, Poland, 2016, pp. 1-6. doi: 10.1109/ICSES.2016.7594341.

[3] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA, USA: O'Reilly Media, 2008.

[4] J. A. H. R. Putra, D. A. D. Dewantara, and T. H. N. T. Putri, "Real-time traffic sign recognition system using OpenCV," *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, Malang, Indonesia, 2017, pp. 199-203. doi: 10.1109/SIET.2017.8304126.

[5] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484-1497, Dec. 2012. doi: 10.1109/TITS.2012.2209421.

[6] F. Zaklouta and B. Stanciulescu, "Real-time traffic sign recognition in three stages," *Robotics and Autonomous Systems*, vol. 62, no. 1, pp. 16-24, Jan. 2014.

[7] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," *The 2011 International Joint Conference on Neural Networks*, San Jose, CA, USA, 2011, pp. 1453-1460. doi: 10.1109/IJCNN.2011.6033389.

[8] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333-338, Aug. 2012.

[9] Y. Li, S. Liu, and H. Wang, "Traffic Sign Recognition Based on Convolutional Neural Network," *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)*, Wuxi, China, 2019, pp. 496-500. doi: 10.1109/SIPROCESS.2019.8868623.

[10] S. S. Ardiansyah, E. A. Suprayitno, and D. P. H. Saptomo, "A real-time speed limit sign recognition on embedded system," *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, Yogyakarta, Indonesia, 2017, pp. 1-6. doi: 10.1109/EECSI.2017.8239129.