Sistema móvil de detección de colisión temprana

Mobile early collision detection system

Roderik Acevedo[1], Roy Henríquez[1], Eddie Pan[1], Carlos Tuñón[1], Elba Valderrama[2]*
[1] Licenciatura en Ingeniería de Software - Campus Víctor Levi Sasso - Universidad Tecnológica de Panamá
[2]Facultad de Ingeniería de Sistemas Computacionales - Campus Víctor Levi Sasso - Universidad Tecnológica de Panamá

Resumen Hoy en día los automóviles de última generación traen consigo sistemas que asisten al conductor en múltiples situaciones, como lo son las cámaras de retroceso, sistemas de frenado automático e inclusive, conducción autónoma. Pero este tipo de tecnología por lo general resulta ser muy costoso para una persona de clase media. Debido a esto, la presente investigación tiene el objetivo de presentar un sistema anticolisión automovilístico que sea económico y accesible para cualquier persona por medio de un dispositivo móvil.

Palabras clave Kotlin, red neuronal, *smartphone*, triangulación, vehículo, IDE, aprendizaje automático.

Abstract Today, the latest generation cars bring systems that assist the driver in multiple situations, such as recoil cameras, automatic braking systems and even autonomous driving. But this type of technology usually turns out to be very expensive for a middle-class person. Due to this, the present investigation has the objective of presenting an automobile anti-collision system that is economical and accessible to anyone through a mobile device.

Keywords Kotlin, neural network, smartphone, triangulation, vehicle, IDE, machine learning.

*** Corresponding author:** elba.valderrama@utp.ac.pa

1 Introducción

La población panameña sigue creciendo a medida que pasan los años, por consiguiente, la cantidad de vehículos en las calles aumenta también. Estadísticas por parte del Instituto Nacional de Estadística y Censo [1], desde el 2010 al 2016 se ha dado un aumento del 25% en la cantidad de accidentes automovilísticos en toda la república. Por esto es necesario buscar formas para mitigar esta tendencia.

Ya existen bastantes tecnologías, a la fecha, que ayudan a prevenir y hasta evitar colisiones. Están las cámaras de retroceso, radares indicadores de punto ciego, sistemas de frenado automático; inclusive, existen vehículos autónomos, por ejemplo, los desarrollados por la compañía Tesla. En Panamá, de los principales vendedores, solo pudimos encontrar mención de la venta de vehículos con sistemas de prevención de colisiones (Honda Sensing) por Honda con el vehículo Honda Accord 2017. El precio de venta inicial de este auto es de 22.000 dólares. Para poder adquirir estas características uno debe comprar un vehículo nuevo. Esto impone una gran barrera para cualquier ciudadano. Además, cualquier persona que ya tenga un vehículo funcional no tienen incentivo de comprar uno nuevo.

Gracias a los avances en la tecnología, la mayoría de las personas tienen acceso a un celular inteligente con gran cantidad de procesamiento para lo pequeño que son estos dispositivos. Estos dispositivos "generalmente" cuentan con una cámara fotográfica. Con la ayuda de las redes neuronales y el aprendizaje de máquinas buscamos reducir la barrera para la obtención de estas características a cualquier persona con un dispositivo móvil.

2 Antecedentes

Por medio de realización de una revisión en el Google Play Store se encontraron dos aplicaciones relacionadas a la nuestra. La más cercana a nuestro enfoque es la aplicación "DailyRoads Voyager" [2] creada por DailyRoads la cual consiste en una aplicación que grabe los sucesos que ocurran en el transcurso de la calle. La otra es Advanced Driver Assistance System (ADAS) - Ringo [3], la cual intenta detectar colisiones delanteras, utilizando el GPS y la cámara del dispositivo. Aunque ADAS detecta decentemente la velocidad del vehículo y proyecta la posición en un mapa de Google Maps, la misma no muestra claramente la imagen de la cámara, reconoce vehículos raramente y cuando los reconoce, lo hace a 100 metros de distancia del vehículo que maneja o donde se encuentra el usuario.

Con lo anterior expuesto se puede afirmar que la aplicación propuesta es de interés y pertinencia para cualquier persona que quiera tener un sistema para evitar colisiones. Lo cual

https://doi.org/10.33412/rev-ric.v5.2.2504

permitirá que con un auto normal pueda tener la función de un auto de lujo.

De los antecedentes detectados relacionados con la presente investigación, se tienen:

Analysis of the possibility of using video recorder for the assessment speed of vehicle before the accident: Desarrollada por Michal Mariusz Abramowsk en la cual demuestra por una serie de pruebas la posibilidad de utilizando una videocámara la posibilidad de evacuar antes del accidente [4].

YOLOv3: ***You Only Look Once***: ***Unified***, ***Real-Time Object Detection***: Desarrollada por Joseph Redmon y Ali Farhadi, la cual es un nuevo enfoque al problema de la detección de objetos en el área de la visión de computadoras a diferencia de los anteriores clasificadores modificados [5]. La arquitectura de YOLO toma el problema de la detección de objetos como un problema de regresión a cajas de posición y probabilidades de clases.

Sistema de comunicación inteligente para la disminución de accidentes automovilísticos por colisión: Desarrollado por Emmanuel Contreras Medina y Jorge Rafael Aguilar Cisneros, consiste en el desarrollo de un sistema que permita la comunicación entre dispositivos en la cual estas puedan disminuir los accidentes automovilísticos [6].

En comparación con los sistemas anteriormente expuestos, se puede decir que el sistema de detección de colisión temprana propuesta será desarrollado a un ambiente móvil, bajo la filosofía de una aplicación gratuita. Asi como también esta será una herramienta que podrá ser utilizada por cualquier persona. Además de brindar un servicio de seguridad preventiva.

3 Metodología

Se realizó la búsqueda de herramientas que permitan el desarrollo de esta aplicación. Se determinó que la aplicación será desarrollada para la plataforma Android. Debido a esto se utilizó la IDE Android Studio. El código de la aplicación reside en un repositorio en GitHub para mantener un control de versiones.

Después de crear las aplicaciones se realizaron diferentes pruebas para evaluar el rendimiento y exactitud de esta. Para evaluar el rendimiento se probó en diferentes dispositivos, el tiempo en que la red neural demora en evaluar un fotograma de la cámara. Llamamos a esto tiempo de reacción. Por último, se evaluó la exactitud de la medición de la distancia de la cámara al vehículo a distancia corta, distancia media y distancia larga.

Se analizaron los resultados de las pruebas para decidir si la aplicación puede llegar a ser utilizada como un detector de colisiones económico.

4 Sistema de detección de colisión temprana

El Sistema DTC (Detección de colisión temprana) utiliza la red neural YOLOv2 (You Only Look Once) para realizar la tarea de detectar objetos en una imagen. Se escogió esta red neuronal que ya cuenta con una implementación del modelo en Tensorflow, librería para el aprendizaje de máquinas. Esto es necesario, ya que el sistema se desarrollaría en Android y Tensorflow cuenta con soporte para este sistema y ejecutar los modelos. Otra razón es que a finales de 2015 en el ranking del PASCAL VOC en la competencia de detección de objetos, YOLO quedó de cuarto en precisión media. Además, en su momento era la única capaz de alcanzar las detecciones en tiempo real [7]. Esto es primordial debido a que los dispositivos móviles no cuentan con grandes cantidades de poder de procesamiento.

La red neuronal recibe como entrada una imagen y como resultado, devuelve las clases detectadas con sus respectivas probabilidades de certeza y su posición en la imagen. De los pesos disponibles para la red neuronal se utilizó Tiny-Yolo. Esta fue entrenada con el dataset VOC, él cuenta con imágenes de vehículos. Además, este es el modelo menos demanda de recursos tiene, pero a la vez es el menos preciso.

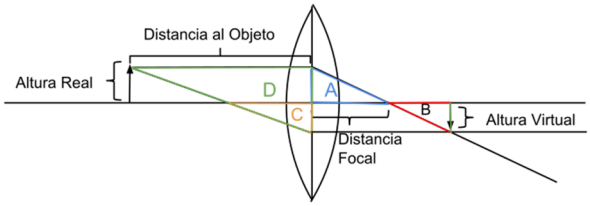

Figura 1. Representación gráfica del método de medición de distancia por medio de triangulación a través de una cámara.

Para realizar la estimación de la distancia de un objeto a la cámara se propone la utilización simple de los triángulos similares que se forman al tener lentes convexos. Haciendo referencia a la figura 1, lo que se quiere obtener es la distancia al objeto. Al ser los triángulos A y B opuestos por el vértice y además los lados opuestos son paralelos, estos son similares. Por la misma razón el triángulo C y A son similares, lo cual también lo hace similar al B. Por último, el triángulo D es similar al C debido a que comparten un ángulo y además dos de sus lados son paralelos. Al final tenemos los triángulos A, B, C y D como similares (ecuación 1).

Con esto podemos derivar la siguiente ecuación.

$$\frac{H_r}{H_v} = \frac{D_o}{D_f} (1)$$

Donde H_r es la altura real, H_v es la altura virtual, D_o es la distancia al objeto y D_f es la distancia focal del espejo.

Despejando la distancia al objeto resulta:

$$D_o = \frac{D_f * H_r}{H_v} \quad (2)$$

Para poder obtener la distancia de la cámara al vehículo se necesita la distancia focal del lente de la cámara, el alto o el ancho del vehículo a distancia y el alto o el ancho de la imagen virtual dependiendo de que se utilizó en el vehículo (ecuación

2). La gran mayoría de los vendedores indican la distancia focal del lente y esta información puede estar disponible desde el mismo sistema operativo. Para el ancho del vehículo se utilizó 1.69 metros. El ancho de la imagen virtual se obtiene contando los píxeles de ancho de la imagen.

Las posiciones que retorna YOLO es un cuadro que delimita la posición del objeto dentro de la imagen. Utilizando el ancho de este cuadro ya podemos calcular la distancia de la cámara al vehículo. YOLO con los parámetros del Tiny-YOLO retorna unos cuadros un poco más grandes de la dimensión del objeto en la imagen, ver figura 2.

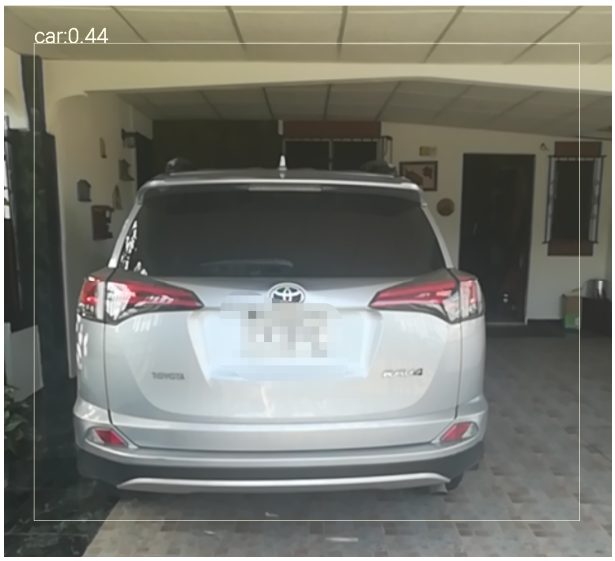

Figura 2. Detección de la localización del vehículo en la imagen por la red neural YOLO.

Para mejorar la exactitud de la medición del vehículo se implementó el algoritmo para la detección de bordes Canny, creado por John F. Canny en 1986 [5]. Este algoritmo está disponible en la librería de Código OpenCV. Después de aplicar el algoritmo se obtiene de resultado la figura 3. Para la aplicación del algoritmo, se recorta el área donde YOLO detectó el vehículo y esta se le pasa al algoritmo. Ya con los bordes detectados podemos tener un resultado igual o más exacto del que YOLO obtuvo del ancho del vehículo, no menos.

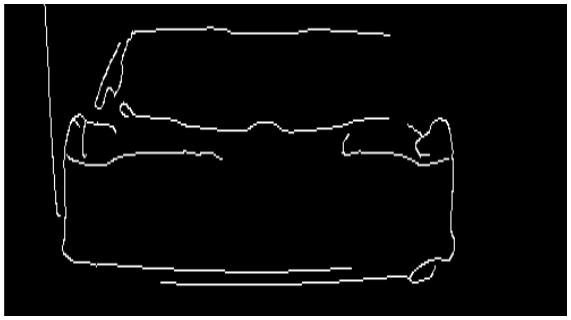

Figura 3. Imagen del vehículo pasada por el algoritmo de detección de bordes Canny.

5 Desarrollo del sistema DCT

La aplicación está desarrollada en el lenguaje de programación Kotlin utilizando la IDE (Integrated Development Environment) Android Studio. Las razones por la cual Kotlin fue escogido fue por ser una alternativa moderna a Java, ser soportado oficialmente en Android Studio, y su interoperabilidad con Java [8]. Gracias a esta característica podíamos utilizar Kotlin y aprovechar todas las librerías, tanto de Kotlin como de Java, y syntax que facilitan el desarrollo en Android, y utilizar Java sí Kotlin no cumplia con lo que necesitábamos para implementar ciertas funciones, esencialmente utilizando dos lenguajes como si fueran uno.

6 Análisis de resultados

El tiempo de respuesta de la red neural en todos los dispositivos es bastante alto como se puede ver en la tabla 1. Con todo y que se utilizó la versión Tiny-Yolo, los dispositivos que utilizamos no son lo suficientemente rápidos para tener un tiempo de respuesta aceptable. El OnePlus 5T, considerado dispositivo de gama alta, tuvo el tiempo de respuesta más bajo de todos y es de esperarse, su procesador es mucha más rápido. Esto al final quiere decir que hay que esperar medio segundo para obtener una medición de distancia. La situación empeora en los otros dos dispositivos donde la red neural demora segundo y medio en responder. Este tan poco desempeño es debido a que la implementación de Tensorflow para Android actualmente no tiene soporte para OpenGL ni CUDA, es decir, no es capaz de utilizar el poder de la GPU la cual aceleraría en gran cantidad la computación de la red neuronal. En su versión actual se está corriendo la aplicación puramente en el procesador.

Tabla 1. Tiempo de respuesta de la red neural en dispositivos distintos

Dispositivo	**Tiempo de respuesta promedio**
OnePlus 5	525 ms
Honor 6x	1640 ms
OnePlus 2	1622 ms

En cuanto a las pruebas con las mediciones de las distancias podemos ver en la tabla 2 que hubo un error bastante alto en las mediciones obtenidas. Además, hay una tendencia de aumento del error al aumentar la distancia entre la cámara y el dispositivo a medir. Sin embargo, es cierto que en todos los casos, alejar la cámara del vehículo incrementa la lectura de la distancia medida. Esto es debido a la utilización del modelo Tiny-YOLO el cual no puede definir con exactitud la posición del modelo. Como se mencionó anteriormente, Tiny-YOLO es necesario debido al poco poder de procesamiento.

Es cierto que existen chips integrados a menor costo y a mayor rendimiento, los cuales permitirían obtener mejores

resultados de rendimiento. Pero el hecho de que un dispositivo que todo mundo tiene en sus bolsillos, tiene todas las herramientas necesarias para realizar esta tarea, lo hace más preferible.

Tabla 2. Resultados de las mediciones con la aplicación en dispositivos distintos

	Distancias		
Dispositivo	**1m**	**3m**	**5m**
Honor 6x	0,981m	1,389m	2,304m
Error Relativo	0,0190	0,5370	0,5392
OnePlus 5T	0,999m	1,263m	1,800m
Error Relativo	0,001	0,5790	0,6400
OnePlus 2	1,389m	1,710	2,718m
Error relativo	0,3890	0.4300	0,5392

7 Conclusión

La problemática de los accidentes vehiculares es un problema que todavía existe, pero que se está mitigando con las nuevas tecnologías de seguridad vehicular. Nuestra aplicación, aunque no esté lista para utilizarse en la vía debido a problemas de rendimiento y exactitud, sirve como prototipo que demuestra la posibilidad de traer esta característica de detectar colisiones a vehículos antiguos de manera económica.

8 Trabajos futuros

Primero nos gustaría mejorar nuestro proceso de medición del sistema, ya que aspectos importantes como el consumo de batería o qué tan caliente se tornaba el dispositivo, no se le prestó atención. También no se hicieron suficientes pruebas con el sistema debido al tiempo.

Para mejorar la eficiencia de la aplicación se podrían aplicar del sistema. Uno sería implementando un detector Hair-Like entrenado con imágenes de vehículos en vez de utilizar la red neuronal YOLO. En el trabajo de Ren y compañía, utilizando este método, obtuvieron un desempeño de siete imágenes por segundo con un dispositivo móvil de gama alta de hace siete años [9]. También podría aplicar el procesado remoto, como lo hicieron Ran y compañía, donde el dispositivo móvil decidía si procesar las imágenes en un servidor remoto con más potencia [10]. En el caso de que la latencia fuera muy alta o el ancho de banda muy bajo, el dispositivo podría decidir si realizar las operaciones localmente.

REFERENCIAS

[1] Insituto Nacional de Estadistica y Censo (INEC) [2017], “Accidentes de Tránsito en la República, por Provincia y Comarca Indígena: Años 2006 -16” [base de datos en linea], Panamá, https://www.contraloria.gob.pa/inec/archivos/P8271451-01.pdf [Consultado: 15 de Julio del 2018]

[2] "DailyRoads Voyager - Aplicaciones en Google Play", Google. [En línea]. Disponible: https://play.google.com/store/apps/details?id=com.dailyroads.v&hl=en. [Consultado: 26-Sep-2018].

[3] “Advanced Driver Assistance Systems (ADAS) - Ringo - Aplicaciones en Google Play," Google. [En línea]. Disponible: https://play.google.com/store/apps/details?id=com.forforest.ringo&hl=en. [Consultado: 26-Sep-2018].

[4] M. M. Abramowski, “Analysis of the Possibility of Using Video Recorder for Speed Assessment of Vehicle Before the Accident”. [En línea]. Disponible en: http://www.zeszyty.waw.pl/artykuly/zn4(104)2015/087_097.pdf. [Consultado: 02-may-2018].

[5] J. Redmon y A. Faradic,“You Only Look Once: Unified, Real-Time Object Detection”. [Consultado: 08 de abril del 2018].

[6] E. C. Medina y J. R. A. Cisneros, “Sistema de comunicación inteligente para la disminución de accidentes automovilísticos por colisión / Intelligent communication system for the reduction of car accidents by collision”, RECI Revista Iberoamericana de las Ciencias Computacionales e Informática, vol. 6, núm. 12, pp. 123–139, nov. 2017 [Consultado: 08 de abril del 2018].

[7] J. Canny, “A Computational Approach to Edge Detection,” Readings in Computer Vision, pp. 184–203, 1987 [Consultado: 13 de Junio del 2018].

[8] “Kotlin Programming Language,” Kotlin. [Online]. Available: https://kotlinlang.org/. [Consultado: 13 de junio del 2018].

[9] Z. Ren, C. Wang, and J. He, “Vehicle Detection Using Android Smartphones,” Proceedings of the 7th International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design driving assessment 2013, 2013.

[10] X. Ran, H. Chen, Z. Liu, and J. Chen, “Delivering Deep Learning to Mobile Devices via Offloading,” Proceedings of the Workshop on Virtual Reality and Augmented Reality Network - VR/AR Network 17, 2017.