

**PROTOTYPE OF HARDWARE ACCELERATED COMPUTER VISION
SYSTEM FOR VEHICLE COUNTING IN INTELLIGENT TRANSPORTATION
SYSTEMS**

**PROTOTIPO DE SISTEMA DE VISIÓN ARTIFICIAL CON ACELERACIÓN
POR HARDWARE PARA CONTEO DE VEHÍCULOS EN SISTEMAS
INTELIGENTES DE TRANSPORTE**

**Julian Uribe-Rios, MSc. Luis Castano-Londono,
MSc. David Marquez-Viloria**

Instituto Tecnológico Metropolitano

Facultad de Ingenierías

Calle 54A No. 30 – 01, Campus Fraternidad, Medellín, Antioquia, Colombia

Tel.: +57 4 460 07 27

E-mail: julianuribe209085@correo.itm.edu.co; {luiscastano, davidmarquez}@itm.edu.co

Abstract: Computer vision systems are used for the acquisition of information in transportation systems. However, the integration of these systems can be limited by cost and ease of implementation or maintenance. Additionally, some applications store or transmit video to a central unit for analysis. In intelligent vision systems, the processing is done locally with lower costs and greater effectiveness. In an IoT scenario, these systems send meta-data with relevant information decreasing the bandwidth demand. This work proposes developing a smart camera that performs the counting of vehicles using a computer vision system with hardware acceleration, through video processing using heterogeneous architectures of low cost and low power consumption. With these features, the developed system is easy to install and maintain for integration into intelligent transport systems.

Keywords: Computer vision, FPGA, IoT, intelligent transport system, vehicle counting.

Resumen: Los sistemas de visión artificial son utilizados para la adquisición de información en sistemas de transporte. Sin embargo, la integración estos sistemas puede verse limitada por aspectos como los costos y facilidad de implementación o mantenimiento. Adicionalmente, algunas aplicaciones almacenan o transmiten el video a una unidad central para su análisis. En los sistemas de visión inteligentes, el procesamiento es realizado localmente con menores costos y mayor efectividad. En un escenario IoT, estos sistemas envían metadatos con información relevante disminuyendo la demanda de ancho de banda. En este trabajo, se propone desarrollar una cámara inteligente que realice el conteo de vehículos usando un sistema de visión artificial con aceleración por hardware, mediante el procesamiento de video utilizando arquitecturas heterogéneas de bajo costo y bajo consumo de potencia. Con estas características se busca que el sistema desarrollado sea de fácil instalación y mantenimiento para su integración en sistemas de transporte inteligente.

Palabras clave: Visión artificial, FPGA, IoT, sistema de transporte inteligente, conteo de vehículos

1. INTRODUCCION

Los sistemas de visión por computadora son utilizados en sistemas inteligentes de transporte para el monitoreo o vigilancia de tráfico, teniendo como ventajas que no son intrusivos y aportan una cantidad importante de información (Liang *et al.*, 2019). Estos sistemas son utilizados para la detección, reconocimiento y seguimiento de vehículos a partir de las características de apariencia visual, generando información sobre diferentes eventos y parámetros del flujo del tráfico (Abdulrahim y Salam, 2016). Dentro de las tareas que realizan los sistemas basados en visión se puede encontrar el conteo de vehículos y la clasificación del tipo de vehículo de forma simultánea, las cuales pueden ser utilizadas en la medición de la densidad del tráfico, el flujo de tráfico o emisiones de contaminantes y gases de efecto invernadero (Liang *et al.*, 2019).

En las aplicaciones de video vigilancia de tráfico se presentan diferentes retos relacionados con las características de las vías, la variedad de los tipos y posturas de los vehículos, las condiciones de iluminación y clima, la posición de la cámara y la congestión del tráfico, entre otros, frente a lo cual se han desarrollado diferentes tipos de técnicas (Abdulrahim y Salam, 2016).

En este contexto, se han presentado diferentes aplicaciones de las cámaras inteligentes con sus respectivas arquitecturas de hardware. En (Leeser *et al.*, 2004) utilizan una cámara inteligente para procesamiento de imágenes médicas y dinámica de fluidos computacional. Consiste en una cámara de video de alta calidad y una grabadora de imágenes conectada directamente a una placa de procesamiento FPGA. Las cámaras inteligentes usan procesamiento integrado para salvar el sistema host de la carga de procesamiento de bajo nivel y para reducir los flujos de comunicación y los gastos generales (Dias *et al.*, 2007). Los autores describieron una plataforma de cámara inteligente basada en un sensor de imagen CMOS y un dispositivo FPGA. En (Bourrasset *et al.*, 2013b) se presenta una plataforma de cámara inteligente basada en FPGA, llamada DreamCam, que puede intercambiar de manera autónoma la información procesada en una red Ethernet. En (Birem y Berry, 2014) se describe otra aplicación con la cámara DreamCam proponiendo una nueva arquitectura de cámara inteligente y varios módulos (IP) para extraer y clasificar eficientemente las características visuales en tiempo real.

En (Holzer, 2012) se describe una cámara inteligente de un sistema de monitoreo distribuido de bajo costo y fácil mantenimiento para el mejoramiento de la calidad del servicio de una vía

concesionada. Para mejorar el desarrollo y el manejo del sistema se utilizan varios núcleos IP específicos del proveedor y se conectan a través de buses locales de procesador (PLB). Una cámara estéreo basada en FPGA se presenta en (Kim y Yoo, 2019), la cámara está diseñada para una implementación de reconocimiento de carretera y carril en tiempo real. En (Bourrasset *et al.*, 2013a), los autores describen una plataforma basada en FPGA que admite la cooperación entre nodos y el procesamiento de imágenes en tiempo de ejecución usando una arquitectura basada en un procesador softcore.

En (Bhowmik *et al.*, 2017) se presenta un diseño de flujo de datos de una arquitectura de cámara basada en la prominencia visual dirigida a una plataforma heterogénea CPU + FPGA para proponer una cámara inteligente en infraestructura de red. El flujo de diseño propuesto abarca la implementación del algoritmo de procesamiento de imágenes, la integración de hardware y software, además de la conectividad de red. El diseño logró un rendimiento en tiempo de ejecución en tiempo real y un consumo de energía de 0.25 vatios. El uso de recursos en una plataforma Xilinx Zynq sigue siendo significativamente bajo. Otra cámara inteligente usando Xilinx Zynq fue descrita en (Liu *et al.*, 2019). En este caso, los algoritmos de procesamiento de imágenes fueron implementados usando Vivado HLS. La visualización se basa en el sistema Linux, OpenCV, y una interfaz gráfica.

Las cámaras comerciales que integran sistemas inteligentes para aplicaciones como el conteo automático de vehículos son costosas. Por esta razón, en muchos casos estas aplicaciones son realizadas con cámaras que no generan la información en tiempo real y se requiere el envío de las tramas de video a centros de operación remotos para un procesamiento posterior. Con el desarrollo de este prototipo de cámara inteligente se busca obtener un sistema de bajo costo, que permita el conteo de vehículos asociados al tráfico vehicular en tiempo real. Este sistema realizaría el procesamiento de video en el sitio utilizando un equipo de cómputo basado en una arquitectura heterogénea (CPU+FPGA) para evitar la transferencia de la trama de video, reduciendo las necesidades de comunicación y almacenamiento de los datos.

2. METODOLOGÍA

2.1 Prototipo del sistema de visión artificial inteligente

Las arquitecturas de cámaras inteligentes son una tecnología emergente y una base de los marcos de

seguridad en los sistemas de visión modernos (Bhowmik *et al.*, 2017). Las cámaras inteligentes tienen un procesamiento de imágenes en tiempo real integrado de alto rendimiento. Para construirlas, los sensores de imagen se interconectan directamente con circuitos integrados de aplicación específica (ASIC) o FPGA para permitir altas velocidades de datos y rendimiento a través del procesamiento masivo en paralelo.

Los FPGA presentan un interés especial para el diseño de cámaras inteligentes debido a la flexibilidad, la configurabilidad y la capacidad de procesamiento en paralelo. Estas cámaras tienen aplicaciones en videovigilancia, seguridad, visión artificial, HCI (Interacción Persona-Computadora), entre otros (Ham y Shi, 2009).

Una cámara inteligente debe contar con las siguientes características (Belbachir, 2010):

- Integración de las funciones clave del dispositivo como óptica, iluminación, un sistema de adquisición y procesamiento de imágenes.
- Uso de una unidad informática y software para implementar algo de inteligencia.
- La capacidad de realizar funciones asignadas sin acción externa.

El sistema está compuesto por una fuente de video, que puede ser una cámara o un archivo de video. Para realizar pruebas del algoritmo se hizo uso de videos con una resolución de 1280x720. El video es enviado y procesado en un dispositivo Ultra96 mediante cable USB. La Ultra96 es un sistema de desarrollo basado en arquitectura heterogénea y las principales características son:

- Xilinx Zynq UltraScale+ MPSoC ZU3EG A484
- Memoria LPDDR4 2GB RAM
- 16 GB microSD
- Wi-Fi 802.11b/g/n
- Bluetooth 5
- Un puerto USB 3.0 tipo Micro-B upstream
- Dos puertos USB 3.0 y uno USB 2.0 de tipo A downstream
- Mini DisplayPort
- Administración de energía de alta eficiencia

Entre las aplicaciones en las que se emplea esta la inteligencia artificial, el aprendizaje automático, conectividad IOT, computación integrada, robótica.

2.2 Aplicación de conteo de vehículos sobre el sistema de visión artificial

El conteo de vehículos fue realizado usando el método *frame difference*. Este método es una solución sencilla para la detección de objetos y tiene un bajo costo computacional. La implementación del algoritmo se realizó en el lenguaje Python utilizando la librería OpenCV.

El proceso de desarrollo empieza con la implementación de los algoritmos en un PC para verificar el funcionamiento. Luego, se realiza una prueba sobre el dispositivo Ultra96 en el sistema de procesamiento (PS) para definir sobre el sistema embebido que partes del sistema se ejecutarán en software y en hardware. Por último, el algoritmo es acelerado sobre la lógica programable (PL) para aprovechar el paralelismo inherente de los FPGA.

El diagrama de bloques de la Fig. 1 muestra los pasos que componen el algoritmo de detección. El primer paso es convertir la imagen a escala de grises para simplificar el procesamiento con el uso de imágenes en este formato. Además, el ruido es minimizado aplicando un suavizado a la imagen con un filtro de desenfoque Gaussiano, ver Fig. 2. Este ruido puede ser provocado por la propia cámara o por la iluminación.

La sustracción entre los frames es realizada aplicando la Ecuación (1). Donde $h(x,y)$ es el resultado de la resta entre los frames $f(x,y)$ y $g(x,y)$.

$$h(x,y) = |f(x,y) - g(x,y)| \quad (1)$$

El valor absoluto en esta expresión garantiza que al realizar la resta no se obtengan valores negativos. La Fig. 3 muestra el resultado de la resta entre dos frames.

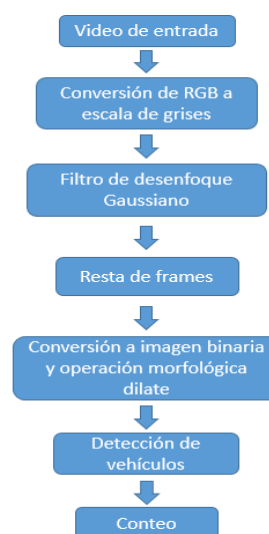


Fig. 1. Diagrama de bloques para el algoritmo de conteo de vehículos

En el siguiente paso la imagen que resulta de la resta es convertida en binario. Para esto, se aplicó un umbral a la imagen para tomar los pixeles que superen el umbral como blancos y los que no lo superen serán negros, como se muestra en la Ecuación (2). T es el umbral usado, $f(x,y)$ es el frame que se quiere convertir a binario, y $g(x,y)$ la imagen resultante.

$$g(x,y) = \begin{cases} 255, & \text{if } f(x,y) \geq T \\ 0, & \text{if } f(x,y) < T \end{cases} \quad (2)$$

En este proceso algunas imágenes pueden terminar con agujeros no deseados. Para rellenar estos agujeros se utilizó la función morfológica de dilatación utilizando la función *dilate* de OpenCV. La Fig. 4 muestra el proceso de aplicación de umbral y dilatación.

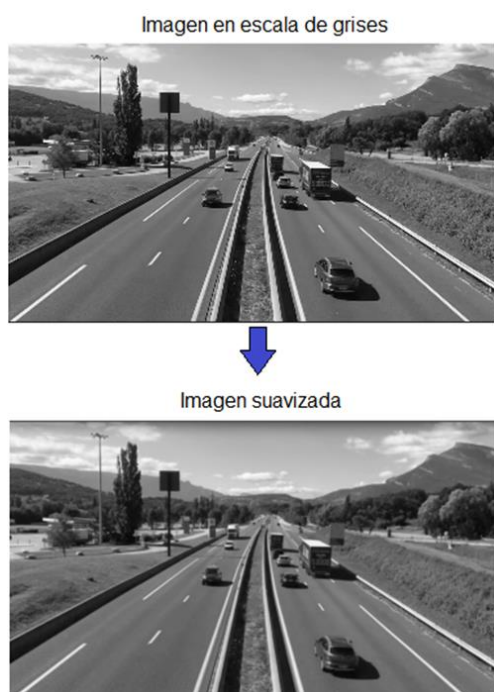


Fig. 2. Conversión a escala de grises y suavizado

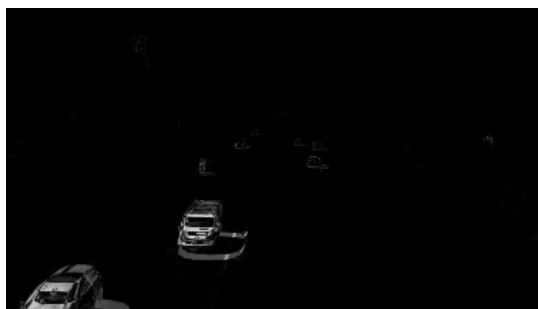


Fig. 3. Resta de frames

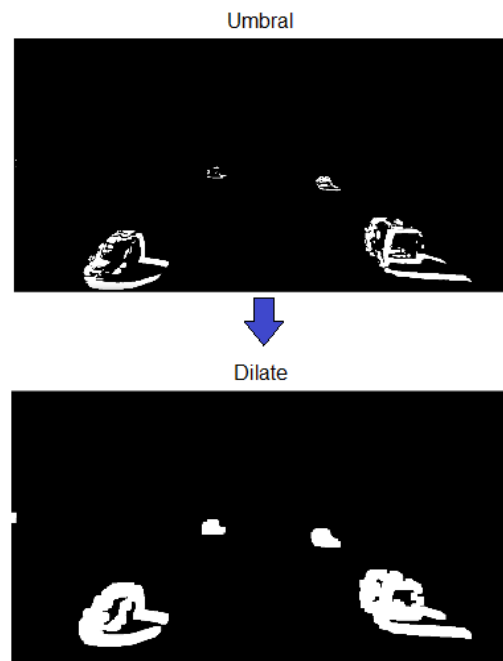


Fig. 4. Umbral y operación morfológica dilate

Para encontrar los objetos en movimiento se detectaron los contornos en la imagen binaria, los cuales corresponden a los vehículos que transitan en la vía, ver Fig. 5.



Fig. 5. Detección

PYNQ ofrece overlays o librerías de hardware que se ejecutan en el PL y se controlan desde el PS usando Python. Estos overlays contienen diseños programables y configurables para arquitecturas heterogéneas basadas en FPGA, que permiten extender una aplicación desde el PS hacia el PL logrando una aceleración en hardware.

Existen overlays para OpenCV disponibles para Ultra96, los cuales traen métodos que se ejecutan en el PL. Para utilizar estos overlays se debe descargar el repositorio *computer vision* de PYNQ con la Ultra96 conectada a internet.

En el algoritmo propuesto se pueden remplazar tres métodos de OpenCV que se ejecutan en el PS usando los overlays disponibles para realizar la aceleración en el PL. Estos métodos son: *GaussianBlur*, *absdiff*, y *dilate*.

El método *GaussianBlur* fue cambiado por el método *2Dfilter*, debido a que no existe el mismo método en los overlays, además, este último puede ser usado para aplicar un filtro en la etapa de suavizado de la imagen. Los métodos *absdiff* y *dilate* se reemplazaron con métodos homónimos que traen los overlays. En este trabajo nosotros analizamos el desempeño en hardware de estos métodos para determinar el efecto que tiene la aceleración de estas funciones sobre hardware. También, se calcularon los consumos de potencia para la implementación con mejor desempeño sobre hardware y los comparamos con el consumo sin aceleración en hardware. Con esto podemos determinar el efecto de las aceleraciones en cuanto al consumo de potencia del dispositivo.

3. RESULTADOS Y DISCUSIÓN

En la Tabla 1 se muestran los FPS obtenidos, para cada método remplazado. Los resultados muestran que al implementar el algoritmo de conteo de vehículos usando el método *GaussianBlur* sin aceleración en hardware, obtenemos un mejor desempeño que el algoritmo *2Dfilter* sin aceleración. Sin embargo, al pasar el método *2Dfilter* al PL usando el overlay alcanzamos un mejor desempeño.

Cuando seleccionamos el método *absdiff* para aceleración en hardware, se obtiene el mismo desempeño con respecto a la implementación en software. Teniendo en cuenta que en este caso estamos usando el método *2Dfilter* en reemplazo de *GaussianBlur*. El mejor desempeño se obtiene al acelerar en hardware el método *dilate* en este caso mejoramos en 1.8 veces el algoritmo en software usando *GaussianBlur* y 2.475 veces usando *2Dfilter*.

Tabla 1: Resultados de desempeño del algoritmo para diferentes aceleraciones en Hardware utilizando overlays de PYNQ-OpenCV

Implementación	Aceleración en Hardware	FPS
Algoritmo sin overlays con método <i>GaussianBlur</i>	NO	5.5
Algoritmo sin overlays con método <i>2Dfilter</i>	NO	4
Algoritmo con método <i>2Dfilter</i> en el PL	SI	6
Algoritmo con método <i>absdiff</i> en el PL	SI	5.5
Algoritmo con método <i>dilate</i> en el PL	SI	9.9

La Tabla 2 muestra la cantidad de vehículos contados en el video de prueba por el algoritmo propuesto. Los resultados indican que el algoritmo cometió 6 falsos positivos y 1 vehículo no fue contado.

Tabla 2: Cantidad total de vehículos en el video de prueba versus el conteo del algoritmo

Cantidad de Vehículos	
Total	28
Contados	33
Falsos Positivos	6
No contados	1

En la fila 1 y 3 de la Fig. 6 se observan los resultados obtenidos de tiempo de ejecución por frame y los frames por segundo para el video de pruebas para los métodos de OpenCV implementados en software, y en la fila 2 y 4 los resultados para los métodos implementados en hardware. Para los métodos *Dilate* y *Filter2D* se logra reducir el tiempo de ejecución por frame gracias a la implementación en hardware, y esto a su vez hace que el tiempo de ejecución por frame de todo el algoritmo disminuya de 175ms a 100ms. Para el método *absdiff* el tiempo de ejecución por frame fue el mismo, tanto en la implementación por software como por hardware.

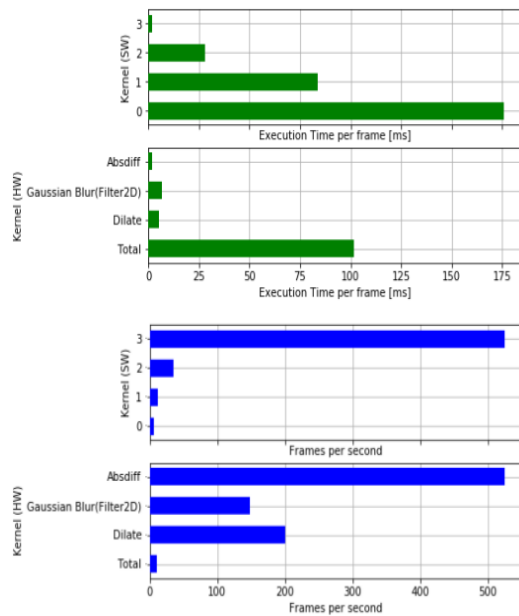


Fig. 6. Resultados de frames por segundo y tiempo de ejecución por frame

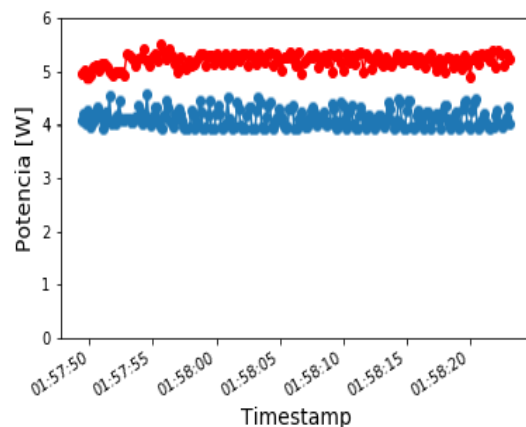


Fig. 7. Consumo potencia de implementación en software versus la aceleración de mejor desempeño

La Fig. 7 muestra el consumo de potencia de la implementación del algoritmo. La trama en color azul representa el consumo del algoritmo en software, mientras que la roja es el consumo al aplicar la aceleración del método dilate que es la implementación que obtuvo el mejor desempeño en las pruebas.

Un problema que afecta el conteo son las sombras, ya que éstas también están en movimiento y por lo tanto el algoritmo las detecta. Esto provoca que se pueda contar dos vehículos como uno solo, si la sombra generada por un vehículo se acerca a otro vehículo, como se observa en la Fig. 8.

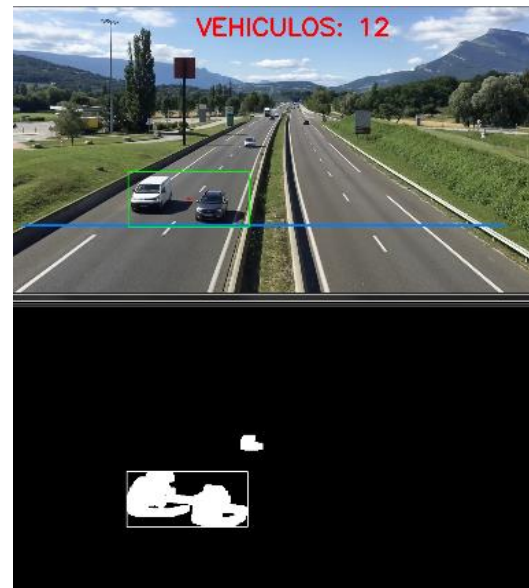


Fig. 8. Problema de sombras

Otro problema que ocurre principalmente con los vehículos de gran tamaño es que se detecten falsos positivos debido a que el contorno lo divide en varios contornos más pequeños como se muestra en la Fig. 9.

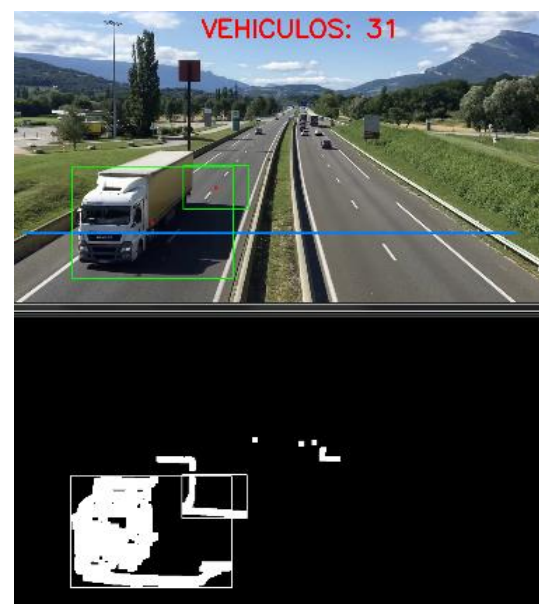


Fig. 9. Falsos positivos

4. RECONOCIMIENTO

Este estudio fue apoyado por el grupo de investigación AE&CC COL0053581, en el Laboratorio de Sistemas de Control y Robótica, adscrito al Instituto Tecnológico Metropolitano. Este trabajo se enmarca en el proyecto “Sistema de visión artificial inteligente con aceleración por hardware para aplicaciones IoT industriales” con código P2023, financiado por el Instituto Tecnológico Metropolitano.

5. CONCLUSIONES

En este trabajo se presentó un prototipo de sistema de visión artificial con aceleración por hardware que busca ser una solución de bajo costo en sistemas inteligentes de transporte. En una primera aplicación presentamos la implementación de un algoritmo para el conteo de vehículos utilizando frame difference.

Una de las características deseadas en nuestro prototipo es permitir el desarrollo del algoritmo usando herramientas de diseño de hardware de alto nivel para disminuir el tiempo de desarrollo, brindar mayor flexibilidad, y facilitar la administración. Por esto, usamos la herramienta PYNQ en el diseño e implementación de las arquitecturas, incorporando lenguajes de alto nivel como Python junto con librerías de hardware para acelerar los algoritmos. Además, es posible adicionar bloques IP personalizados y tener fácil manejo de los periféricos.

Las medidas de desempeño obtenidas con los diferentes overlays probados mostraron que se puede obtener una aceleración en hardware en dispositivos basados en arquitectura heterogénea. Esto nos permite proponer sistemas de visión artificial de bajo costo.

Como trabajo futuro se propone la implementación de un algoritmo más robusto para conteo vehicular, explorar otras técnicas de aceleración, e implementar diferentes algoritmos para sistemas inteligentes de transporte.

REFERENCIAS

- Liang, M. (2015). Counting and classification of highway vehicles by regression analysis. *IEEE Transactions on Intelligent Transportation Systems*, 16(5), 2878-2888.
- Abdulrahim, K. y Salam, R. A. (2016). Traffic surveillance: A review of vision based vehicle detection, recognition and tracking. *International journal of applied engineering research*, 11(1), 713-726.
- Leeser, M. (2004, April). Smart camera based on reconfigurable hardware enables diverse real-time applications. In *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (pp. 147-155). IEEE.
- Dias, F. (2007, September). Hardware, design and implementation issues on a FPGA-based smart camera. In *2007 First ACM/IEEE International Conference on Distributed Smart Cameras* (pp. 20-26). IEEE.
- Bourrasset, C. (2013, October). DreamCAM: A FPGA-based platform for smart camera networks. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)* (pp. 1-2). IEEE.
- Birem, M. y Berry, F. (2014). Dreamcam: A modular fpga-based smart camera architecture. *Journal of Systems Architecture*, 60(6), 519-527.
- Holzer, M. (2012, January). Optimized hardware architecture of a smart camera with novel cyclic image line storage structures for morphological raster scan image processing. In *2012 IEEE International Conference on Emerging Signal Processing Applications* (pp. 83-86). IEEE.
- Kim, J. G. y Yoo, J. H. (2019). HW Implementation of Real-Time Road & Lane Detection in FPGA-Based Stereo Camera. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)* (pp. 1-4). IEEE.
- Bourrasset, C. (2013, October). Distributed FPGA-based smart camera architecture for computer vision applications. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)* (pp. 1-2). IEEE.
- Bhowmik, D. (2017). Power efficient dataflow design for a heterogeneous smart camera architecture.
- Liu, G. (2019, March). Embedded intelligent camera algorithm based on hardware IP. In *Tenth International Symposium on Precision Engineering Measurements and Instrumentation* (Vol. 11053, p. 110533T). International Society for Optics and Photonics.
- Ham, Y. C. y Shi, Y. (2009, May). Developing a smart camera for gesture recognition in HCI applications. In *2009 IEEE 13th International Symposium on Consumer Electronics* (pp. 994-998). IEEE.
- Belbachir, A. N. (Ed.). (2010). *Smart cameras* (Vol. 2). New York: Springer.