Aceleración del Análisis de Componentes Conectadas en Imágenes para Procesamiento de Altas Prestaciones en el Borde

José L. Mira, Jesús Barba, Julián Caba, José A. de la Torre,
Fernando Rincón, Soledad Escolar y Juan Carlos López[1],

Resumen— **En el procesamiento de imágenes, el algoritmo de Componentes Conectadas es un método utilizado para identificar y etiquetar los distintos objetos o regiones presentes en una imagen digital. Este algoritmo puede ser útil para diversas tareas de procesamiento de imágenes, como el reconocimiento de objetos, la segmentación de imágenes y la extracción de características. Este trabajo presenta la implementación de un algoritmo de paso único en un dispositivo basado en FPGA adecuado para aplicaciones de visión por computador de alto rendimiento en el edge, la tarjeta de computación Ultra96-V2. El diseño y la implementación del núcleo IP se han enfrentado a retos utilizando el flujo de trabajo y las herramientas HLS de AMD-Xilinx, que requieren un uso eficiente y optimizado de los recursos, así como la reingeniería del algoritmo para cumplir con los requisitos impuestos por el marco de desarrollo. El rendimiento del acelerador propuesto se ha analizado a fondo utilizando el marco de evaluación comparativa YACCLAB frente a una CPU de gama alta y otra de gama baja. Los resultados muestran una pérdida de rendimiento esperada debido a las limitaciones de memoria y frecuencia de reloj. Sin embargo, en lo que respecta a la eficiencia energética, la arquitectura multinúcleo de hardware supera a las alternativas de software con una mejora de entre dos y cinco veces, dependiendo del tamaño y la complejidad de las imágenes.**

Palabras clave— **Análisis de componentes conectadas, computación de altas prestaciones, procesamiento en el borde, visión por computador, FPGA**

I. Introducción

El análisis de componentes conectadas (CC) es un paso crucial en muchas aplicaciones de visión por computador, que consiste en asignar etiquetas únicas a cada región de una imagen que ha sido previamente segmentada aplicando un proceso de umbralización para diferenciar los objetos del fondo. A continuación, se extraen las características de cada región, como el área, el centro de gravedad, el cuadro delimitador y el valor del píxel, basándose en sus etiquetas con el objetivo de clasificar cada región en una de las múltiples clases.

La complejidad temporal y el uso intensivo de memoria son dos de los principales problemas a los que hay que hacer frente cuando se trata de desarrollar una solución válida para dispositivos integrados con una cantidad de recursos limitados o un consumo de energía restringido. De hecho, este es el contexto para el procesamiento de imágenes en el edge, un paradigma informático que realiza las tareas de procesamiento de imágenes localmente, cerca de la fuente de los datos, en lugar de enviar todos los datos a una nube o centro de datos para su procesamiento. El procesamiento de imágenes en el edge también puede permitir el análisis en tiempo real o casi real de datos de imágenes, lo que puede ser crítico en aplicaciones como vehículos autónomos, sistemas de vigilancia o imágenes médicas.

A pesar de que ha habido múltiples algoritmos de CC a lo largo del tiempo, sólo los algoritmos de paso único son realmente adecuados para implementaciones basadas en FPGA debido a la reducción del ancho de banda de memoria, que representa el principal cuello de botella para esta clase de dispositivos [1]. Esta clase de algoritmos CC también son adecuados para el procesamiento de imágenes en streaming.

En este trabajo, se revisa el modelado, diseño e implementación del algoritmo de análisis CC single-pass propuesto originalmente por Bailey et al. en [2] y posteriormente optimizado en [3] y adaptado al flujo de diseño Xilinx-AMD Vitis. El objetivo principal es evaluar el rendimiento y la eficiencia energética de la implementación FPGA frente a las alternativas de software más avanzadas que se ejecutan en plataformas de procesadores de gama alta y baja. Queda fuera del alcance de este trabajo comparar nuestra solución hardware en términos de uso de recursos (principalmente memoria BRAM) con otras propuestas debido al uso de diferentes arquitecturas FPGA y herramientas de síntesis. Además, el modelo HLS propuesto para el núcleo se ha generalizado para permitir su evaluación comparativa mediante el framework YACCLAB [4]. La arquitectura permite configuraciones multinúcleo, contador de etiquetas en el peor de los casos, así como resoluciones más allá de la resolución común de 640x640 reportada en la mayoría de la literatura [5][3], lo que aumenta la demanda de recursos.

II. El acelerador HW-CC

Aunque el algoritmo propuesto por Bailey et al. [2] está diseñado teniendo en cuenta los dispositivos de computación basados en FPGAs evitando, por ejemplo, la necesidad de almacenar toda la imagen en memorias BRAM, sigue presentando retos a la hora de empaquetarlo como un núcleo completamente funcional para arquitecturas FPA-SoCs como la Xilinx ZynQ-UltraScale+. Además, se ha realizado un análisis del rendimiento y los recursos, limitando la validación a un tamaño específico de imágenes de entrada y complejidad. Por lo tanto, falta una visión

[1]Escuela Superior de Informática. Dpto. de Tecnologías y Sistemas de Información. Universidad de Castilla-La Mancha, Ciudad Real. e-mail: joseluis.mira@uclm.es

completa del desarrollo real de las soluciones basadas en FPGA, lo que impide una comparación justa con las soluciones de software. Además, enfrentarse al modelado de la arquitectura y la lógica del algoritmo utilizando la tecnología de Síntesis de Alto Nivel (HLS) exige la reingeniería del diseño original para hacerlo compatible con las restricciones semánticas y sintácticas de las herramientas HLS.

La principal aportación de este trabajo es el desarrollo del acelerador HW-CC, un núcleo IP totalmente parametrizado y listo para su uso en la plataforma FPGA-SoC de Xilinx. La parametrización permite su uso para un amplio rango de imágenes de entrada y escenarios (ver III) haciendo que nuestra solución sea flexible para adaptarse a diferentes requisitos de aplicaciones y plataformas de destino (disponibilidad de memoria, latencia, etc.). Esta flexibilidad tiene un coste, principalmente debido a las grandes necesidades de memoria para almacenar datos intermedios cuando se trabaja con imágenes grandes; por ejemplo, la *data table*, una estructura de datos que contiene estadísticas de las regiones de etiquetas, depende de la anchura (W) y la altura (H) siendo $W/2 \times H/2$ el máximo teórico. Por este motivo, en la solución propuesta, se han tenido que desarrollar mecanismos para superar este hándicap, haciendo que una plataforma de gama baja basada en FPGA sea adecuada para los algoritmos CC.

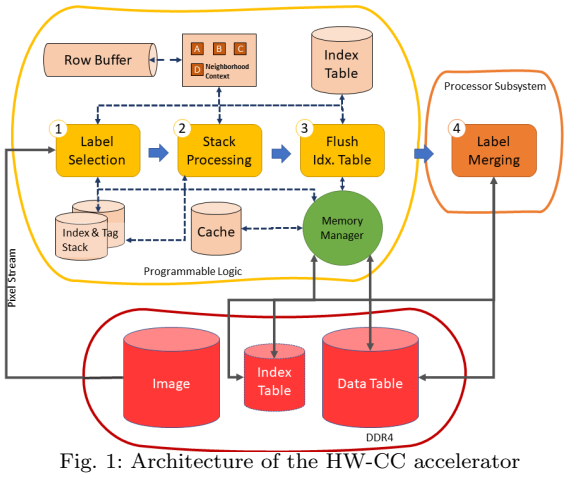

Fig. 1: Architecture of the HW-CC accelerator

Fig. 1 esboza una vista de alto nivel de la arquitectura HW-CC que se ha modelado en Vitis HLS C/C++. La interfaz de memoria utiliza puertos AXI-Stream y AXI-Memory para que el núcleo IP pueda integrarse fácilmente con el subsistema del procesador. Este subsistema es el encargado en ejecutar el firmware para la configuración de la plataforma y los periféricos, la gestión de la transferencia DMA y la ejecución del paso *Label Merging*.

El modelo HLS define el control y los elementos de memoria, junto con su mapeo a los recursos de la FPGA, que realiza el algoritmo de la forma más óptima mediante el uso de los pragmas y parámetros de modelo adecuados.

A. *Elementos de la memoria*

- **Neigbourhood Context + Row Buffer**: Estos dos elementos se encargan de almacenar temporalmente la información necesaria para determinar la creación y actualización de etiquetas asociadas a grupos de píxeles. El tamaño del Row Buffer es un parámetro del modelo.
- **Index Table**: Matriz unidimensional que contiene el valor de la etiqueta a la que apunta un índice determinado. El tamaño de esta matriz puede parametrizarse en el modelo para ajustarse a los requisitos típicos de la imagen de entrada. Otro parámetro del modelo es el número máximo de etiquetas, que determina el tamaño de la palabra para la tabla de índices.
- **Index & Tag Stack**: Se trata de una estructura de datos utilizada para fusionar dos regiones en función de los valores de sus etiquetas. Esta estrucura es accedida una vez al final de cada fila de la imagen. Para aprovechar que el acceso a la información de índices y etiquetas es independiente, se ha decidido aplicar el pragma *ARRAY PARTITION* para permitir el acceso paralelo a ambos valores. El tamaño de la pila también es un parámetro del modelo.
- **Memory Manager + Data Table + Cache**: La *data table* es la estructura en la que se almacenan las estadísticas relacionadas con las etiquetas. En la aplicación propuesta se proporcionan las coordenadas de la esquina superior izquierda, la anchura, la altura y el area de cada región. Los requisitos de memoria de la tabla de datos agotarían rápidamente las BRAM disponibles en la estructura de la FPGA, especialmente para altas resoluciones. Para que el acelerador HW-CC se adapte a una gran variedad de tamaños y complejidades, se ha decidido implementar un sencillo sistema de caché que permita al núcleo manejar grandes tamaños de imagen, sin sacrificar la velocidad de acceso a los datos. Este módulo también se encarga de comprobar, leer y escribir los bloques de la memoria caché con respecto a la RAM de a bordo cuando es necesario. Dado que durante el funcionamiento del algoritmo, éste va asignando nuevas etiquetas a las nuevas regiones que atraviesa. En este proceso, se ocupa espacio en la *data table* correspondiente a estas nuevas etiquetas. Al mismo tiempo que tiene lugar este proceso, también se produce la fusión de varias etiquetas asignadas anteriormente. De esta forma, el acceso a las entradas de la tabla se suele realizar de forma alterna. A la hora de implementar la caché, este patrón de acceso a la memoria supone un problema, ya que hay zonas de la imagen en las que se accede de forma consecutiva a entradas de la *data table* pertenecientes a distintos bloques de datos. Esta situación se traduce en un elevado número de fallos de caché, con el consiguiente tiempo de ejecución empleado en tener que reemplazar el bloque de datos varias veces.

Basándose en estas características de los patrones de acceso a la memoria, se decidió implementar un sistema de caché de 2 vías para evitar la sustitución innecesaria de bloques de datos.

B. Etapas del algoritmo

B.1 Label Selection (HW)

Esta etapa se realiza una vez para cada píxel de la imagen, donde se evalúa el Neighbourhood Context para determinar qué etiqueta se generará en un píxel dado. En este proceso pueden darse las siguientes situaciones:

- Generar una nueva etiqueta, para lo cual hay que crear un nuevo registro en la tabla de datos.
- Asignar una etiqueta existente, lo que requiere acceder a la tabla de datos para actualizar las estadísticas relacionadas con esa etiqueta.
- Hay dos posibles etiquetas en el Contexto de Vecindad por lo que se almacenan en el *Index & Tag Stack* para ser resueltas más tarde, la menor de las dos se asigna al píxel y las estadísticas se actualizan en la *Data Table*.

B.2 Stack Processing (HW)

Esta etapa se realiza para cada fila de la imagen que ha sido procesada por el algoritmo. Accede al *Index & Tag Stack* para realizar el proceso de fusión entre etiquetas, que accede a la tabla de índices para actualizar los índices correspondientes.

C. Flush Index Table (HW)

Una vez que el algoritmo ha terminado de recorrer la imagen, los bloques de datos almacenados en caché se actualizan en la DDR.

C.1 Label Merging (SW)

Una vez finalizado el componente FPGA, el procesador recorre la Tabla de Índices y la Tabla de Datos contenidas en la memoria RAM para realizar el proceso final de fusión entre las diferentes regiones con el fin de obtener el número final de etiquetas y sus estadísticas. Este paso se trasladó al ámbito del software debido a que se trata de una tarea intensiva de control con acceso irregular a patrones de memoria, un escenario en el que el procesador rinde mejor que la FPGA.

III. Resultados

En esta sección, se presentaran los resultados de nuestra evaluación de rendimiento de varios algoritmos de etiquetado de regiones utilizando el marco de referencia YACCLAB [4]. Nuestra evaluación tenía como objetivo comparar la velocidad y la eficiencia energética del acelerador HW-CC IP propuesto frente a diferentes algoritmos en un conjunto de imágenes de prueba proporcionadas por YACCLAB. Las pruebas de software se ejecutaron en: (1) un Intel(R) Core(TM) i7-12700K de 12ª generación (3,6GHz, 64GB DDR5) ejecutando una distribución Ubuntu 20.4 GNU/Linux ; y (2) un Intel(R) Celeron(R) N5095 (2,0GHz, 8GB DDR4) ejecutando una distribución Ubuntu 22.4 GNU/Linux. En ambos casos, el conjunto de pruebas se compiló con GCC 11.3. Al incorporar tanto un procesador de estación de trabajo de gama alta como una CPU de gama baja comúnmente presente en las soluciones comerciales de computación en el edge, la validación experimental puede proporcionar una mejor comprensión de las capacidades del acelerador propuesto.

El conjunto de pruebas incluyó los algoritmos de CC listos para usar que vienen con YACCLAB, a saber: Scan Array-based with Union Find [6] (SAUF), Block-Based with Decision Tree [7] (BBDT), Pixel Prediction [8] (PRED), Directed Rooted Acyclic Graph [9] (DRAG) y Spaghetti Labelling [10].

El conjunto de datos utilizado para la validación experimental comprendía una selección del subconjunto YACCLAB 2D, que consiste en imágenes binarias con conected components etiquetados. El conjunto de datos incluye tanto imágenes sintéticas como imágenes del mundo real procedentes de diversas fuentes. Las imágenes reales se utilizan para probar el rendimiento de los algoritmos en imágenes con variaciones naturales de forma e intensidad. En este trabajo, probamos el acelerador HW-CC para el procesamiento de imágenes en el edge sobre MIR-flickr (25K imágenes, tamaño medio y componentes conectados: 0,17 megapíxeles y 495, respectivamente), Hamlet (104 imágenes), Tobacco800 (1290 documentos, 150-300 PPP y resoluciones de 1200x1600 a 2500x3200), 3DPeS y subconjuntos de imágenes de ruido aleatorio sintético. Cabe destacar que este último permite comprobar objetivamente la escalabilidad y eficacia de distintos algoritmos. Este subconjunto proporciona diez imágenes para cada combinación de tamaño (32x32 hasta 4096x4096 en pasos de x2) y densidad (10 % a 90 %), lo que da un total de 720 imágenes.

La plataforma de prototipado Ultra96-v2 se ha utilizado para desplegar y probar el núcleo del acelerador de procesamiento. La placa Ultra96-v2 se basa en el MPSoC Xilinx Zynq UltraScale+, que integra lógica programable y un procesador Arm Cortex-A53 de 64 bits y cuatro núcleos. El chip ZU3EG integrado en la plataforma es una FPGA de gama baja con capacidades modestas y bajo consumo de energía, lo que la hace adecuada para aplicaciones de computación en el edge. Se ha utilizado la versión 2021.1 de la suite de herramientas de Xilinx (Vitis HLS, Vivado y Vitis Unified Software Platform) para modelar el núcleo IP HW-CC, diseñar y sintetizar la plataforma y desplegar toda la solución.

En primer lugar, evaluamos el rendimiento de la solución propuesta frente a los subconjuntos del mundo real seleccionados. Aunque la arquitectura del acelerador HW-CC es flexible y permite configurar múltiples parámetros, para este experimento se propuso una estrategia agresiva con el fin de obtener el máximo rendimiento. Por lo tanto, el tamaño del búfer de fila se estableció en 4K palabras, la memoria

de datos de etiquetas en 64K y la memoria de bloque de caché en 1K. El número de núcleos aceleradores instanciados en la FPGA es de cuatro, lo que eleva la utilización de los recursos BRAM al 83 % del total disponible en el dispositivo. Los resultados posteriores a la síntesis mostraron que era posible configurar un reloj de 175 MHz para controlar la lógica. Sin embargo, establecimos un enfoque conservador a este respecto para evitar comportamientos inesperados en la placa. Así, se estableció una configuración de reloj de 150MHz.

Tabla I: Tiempo medio de ejecución (ms) para el subconjunto de imágenes reales

	MIRFlickr	3DPes	Hamlet	Tobacco	Avg. Gain
HW-CC-150	1.71	2.73	19.77	36.92	-
Intel i7-12700K					
SAUF	0.44	0.61	4.96	7.65	**-4.29x**
BBDT	0.36	0.35	3.44	4.93	**-6.45x**
PRED	0.54	0.72	6.92	10.52	**-6.33x**
DRAG	0.36	0.35	3.44	4.91	**-6.46x**
Spaguetti	0.2	0.18	1.89	2.68	**-11.99x**
Intel Celeron N5095					
SAUF	1.46	2.76	22.19	37.33	**-1.01x**
BBDT	0.88	1.33	13.86	22.96	**-1.76x**
PRED	1.49	2.87	22.96	38.69	**+1.02x**
DRAG	0.87	1.33	13.86	23.03	**-1.76x**
Spaguetti	0.41	0.53	5.16	7.89	**-4.46x**

La Tabla I presenta los tiempos medios de ejecución para todos los subconjuntos de imágenes. Los resultados muestran una pérdida significativa de rendimiento en comparación con el procesador i7, y una disminución moderada del rendimiento en el caso del procesador Celeron.

La característica más atractiva de la arquitectura Zynq UltraScale+ para aplicaciones de computación en el edge es su eficiencia en términos de consumo de energía. El consumo medio de energía de nuestro HW-CC se ha medido utilizando la interfaz proporcionada por la *Platform Management Unit* integrada, lo que ha dado como resultado 3,74 W durante la ejecución de la prueba. En el caso del procesador i7, se ha utilizado la herramienta de línea de comandos *powerstat* para monitorizar el incremento en el consumo de energía a través de la interfaz RAPL (Running Average Power Limit); se observó una media de 31,4W. Sin embargo, en la plataforma informática basada en Celeron se utilizó un medidor de consumo de energía externo que registró un incremento sostenido de 10,3 W durante la ejecución de la prueba.

Tabla II: Ganancia en la eficiencia energética media (J) para el subconjunto real de imágenes. HW-CC vs Intel i7 & Celeron.

	SAUF	BBDT	PRED	DRAG	Spaghetti
Intel i7	1.87	1.24	2.56	1.24	0.67
Intel Celeron	2.87	1.75	2.97	1.76	0.63

La Tabla II muestra la comparación de la eficiencia, expresada como *Julios* consumidos, desglosada por subconjunto y algoritmo. Los resultados medios exponen una ganancia consistente entre plataformas en todos los casos excepto en el algoritmo Spaghetti, que supera al resto debido a su bajo tiempo de ejecución.

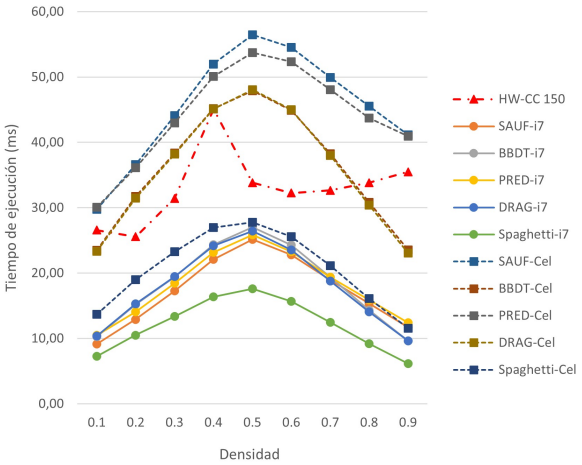

Fig. 2: Resultados por densidad de test

Para validar el rendimiento y la escalabilidad de una solución, YACCLAB incluye un conjunto de imágenes generadas aleatoriamente que permite la ejecución de pruebas de densidad y tamaño. Las Figuras 2 y 3 muestran una comparación de los tiempos de ejecución de cada algoritmo de software y la versión de 4 núcleos a 150 MHz del acelerador HW-CC propuesto.

El tiempo medio de ejecución es estable para HW-CC con valores ligeramente superiores a medida que aumenta la densidad, con un pico en las imágenes de densidad de primer plano de 0,4, como puede verse en la Fig. La comparación con el procesador i7 muestra una degradación del rendimiento de $\approx -2x$ para todos los algoritmos excepto Spaghetti que alcanza $-3,03x$ con una pérdida máxima de $5,76x$ para imágenes de densidad 0,9. Sin embargo, HW-CC es más rápido que el procesador Celeron en todos los casos - SAUF (35,9 %), BBDT (4,2 %), PRED (32,6 %), DRAG (3,4 %) - pero, de nuevo, Spaghetti ($1,72x$ más lento de media con una pérdida máxima de $3,07x$ para imágenes de densidad 0,9).

En cuanto a las pruebas de tamaño, el rendimiento del acelerador propuesto mantiene el mismo patrón en todos los procesadores y algoritmos. Como puede verse en la Fig. 3, el núcleo HW-CC mantiene la misma dependencia lineal del tiempo de ejecución que sus homólogos de software. La pérdida media de rendimiento frente al procesador i7 es de $-2,14x$. Teniendo en cuenta que el consumo de energía del procesador Intel i7 es aproximadamente diez veces la energía que necesita la FPGA, se obtiene una mejora de $\approx 5x$ en términos de eficiencia energética, al tiempo que se mantiene una capacidad de computación justa para un dispositivo que se va a desplegar en edge de la infraestructura informática. En cuanto a la CPU Celeron, la pérdida media de rendimiento es de $1,09$, muy penalizada por la eficiencia del algoritmo Spaghetti ($1,82$ mejor). En los demás casos, HW-CC es un 10 % más rápido. En conjunto, nuestra solución basada en FPGA es 2,5 veces más eficiente que la plataforma informática Celeron.

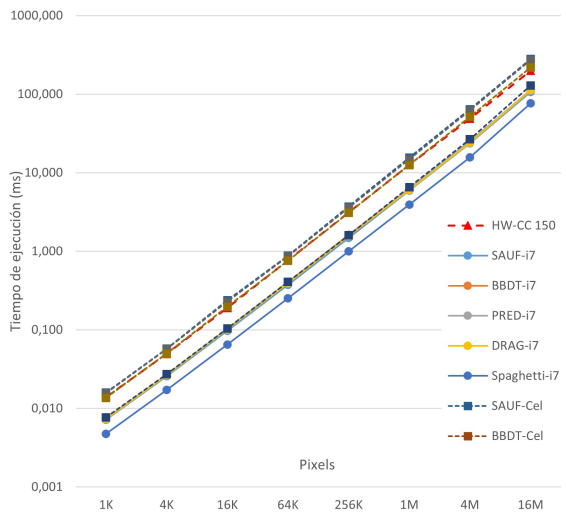

Fig. 3: Resultados por tamaño de test.

IV. Conclusiones

En este trabajo se ha realizado el diseño y desarrollo de la arquitectura de un acelerador hardware para el análisis y etiquetado de componentes en imágenes de una manera eficiente en términos de la relación potencia de cómputo versus consumo. Debido a que su despliegue tiene como objetivo plataformas de cómputo empotradas que estarán desplegadas en el borde de la infraestructura de adquisición, procesamiento y transmisión de la información (p.ej. en una plataforma IoT), el análisis de los resultados obtenidos ha tenido en cuenta, no sólo el rendimiento sino también las necesidades energéticas de la solución. En última instancia, el tipo de plataformas consideradas para la aplicación final consistirá en un dispositivo de procesamiento basado en un SoC-FPGA y estará alimentado por batería; de ahí la importancia de la dimensión energética.

La comparación con aportaciones previas en el estado del arte es bastante difícil debido a que los trabajos realizados en muchas ocasiones no proporcionan la información necesaria, tienen como objetivo dispositivos FPGA de generaciones previas y tampoco existe una estandarización respecto a las pruebas realizadas e imágenes utilizadas en las mismas (con impacto directo en la latencia de procesamiento que depende de la complejidad de la misma y su tamaño, por ejemplo). Como siguiente paso, trabajaremos en el desarrollo de un marco comparativo y recopilación de información que permita evaluar lo más objetivamente posible nuestra solución respecto a otros trabajos en el estado del arte.

Agradecimientos

Esta investigación ha sido financiada por el programa H2020 de la Unión Europea bajo el acuerdo de subvención nº 857159 (proyecto SHAPES) y por el Ministerio de Economía y Competitividad (MINECO) del Gobierno de España bajo el proyecto TALENT (PID2020-116417RB-C4, subproyecto 1) y el proyecto proyecto MIRATAR (TED2021-132149B-C41).

Referencias

[1] Robert Walczyk, Alistair Armitage, and T. David Binnie, "Comparative study on connected component labeling algorithms for embedded video processing systems," in *International Conference on Image Processing, Computer Vision, & Pattern Recognition*, 2010.

[2] Donald Bailey and Christopher Johnston, "Single pass connected components analysis," *Proceedings of Image and Vision Computing*, 01 2007.

[3] Ni Ma, Donald G. Bailey, and Christopher T. Johnston, "Optimised single pass connected components analysis," in *2008 International Conference on Field-Programmable Technology*, 2008, pp. 185–192.

[4] Costantino Grana, Federico Bolelli, Lorenzo Baraldi, and Roberto Vezzani, "Yacclab - yet another connected components labeling benchmark," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 3109–3114.

[5] Fanny Spagnolo, Fabio Frustaci, Stefania Perri, and Pasquale Corsonello, "An efficient connected component labeling architecture for embedded systems," *Journal of Low Power Electronics and Applications*, vol. 8, no. 1, 2018.

[6] Kesheng Wu, Ekow J. Otoo, and Kenji Suzuki, "Optimizing two-pass connected-component labeling algorithms," *Pattern Analysis and Applications*, vol. 12, pp. 117–135, 2009.

[7] Costantino Grana, Daniele Borghesani, and Rita Cucchiara, "Optimized block-based connected components labeling with decision trees," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1596–1609, 2010.

[8] Costantino Grana, Lorenzo Baraldi, and Federico Bolelli, "Optimized connected components labeling with pixel prediction," in *Advanced Concepts for Intelligent Vision Systems*, Jacques Blanc-Talon, Cosimo Distante, Wilfried Philips, Dan Popescu, and Paul Scheunders, Eds., Cham, 2016, pp. 431–440, Springer International Publishing.

[9] Federico Bolelli, Lorenzo Baraldi, Michele Cancilla, and Costantino Grana, "Connected components labeling on drags," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 121–126.

[10] Federico Bolelli, Stefano Allegretti, Lorenzo Baraldi, and Costantino Grana, "Spaghetti labeling: Directed acyclic graphs for block-based connected components labeling," *IEEE Transactions on Image Processing*, vol. 29, pp. 1999–2012, 2020.