Un algoritmo en tiempo real para etiquetado de componentes conectados en imágenes

Elisa Calvo
Departamento de Electrónica y Electromagnetismo
Universidad de Sevilla
Sevilla, España
calvo@imse-cnm.csic.es

Piedad Brox, Santiago Sánchez-Solano
Instituto de Microelectrónica de Sevilla (IMSE-CNM)
Consejo Superior de Investigaciones Científicas
Sevilla, España
brox, santiago@imse-cnm.csic.es

***Resumen*—Esta comunicación presenta un algoritmo de dos pasadas para el etiquetado en tiempo real de los componentes conexos en una imagen. El algoritmo propuesto es una buena opción frente a otras alternativas de dos y múltiples pasadas ya que ha sido diseñado considerando que su implementación en FPGAs ofrezca un buen compromiso entre recursos ocupados y velocidad de operación. Se describen dos implementaciones hardware de este algoritmo, cuyo desarrollo se ha llevado a cabo siguiendo un flujo de diseño basado en la herramienta System Generator de Xilinx.**

I. INTRODUCCIÓN

Cada vez con más frecuencia es necesaria la integración de sistemas de visión por computador en dispositivos específicos y sistemas empotrados (móviles, PDAs, redes de sensores, etc.). Las limitaciones que presentan estos dispositivos en cuanto a consumo de potencia y capacidad de cálculo y almacenamiento, así como la necesidad de operación en tiempo real de las aplicaciones donde se utilizan, obligan a llevar a cabo una revisión de los algoritmos existentes con el fin de adaptar su implementación a las características de las plataformas emergentes.

Este proceso de adaptación es especialmente importante en los algoritmos de etiquetado de componentes conexos o algoritmos CCL (*Connected Component Labeling*), ya que estos procedimientos constituyen un paso intermedio entre las tareas de tratamiento de imágenes a bajo y alto nivel, como se ilustra en el sistema de visión genérico mostrado en Fig. 1.

En este trabajo se describe el desarrollo de un nuevo algoritmo CCL para el etiquetado de imágenes correctamente preprocesadas, así como su implementación eficiente sobre FPGAs (en términos de recursos y/o velocidad). En el proceso de diseño se ha seguido una metodología basada en System Generator, una herramienta de Xilinx integrada en el entorno Matlab/Simulink que, al facilitar la realización de todas las etapas de diseño bajo un mismo marco de referencia, ha permitido acortar el tiempo de desarrollo de las distintas implementaciones.

La estructura de la comunicación es la siguiente. En la Sección II se revisan algunos conceptos básicos y los principales tipos de algoritmos CCL existentes en la literatura. La Sección III se centra en el análisis de un tipo concreto de algoritmo, los denominados algoritmos CCL de dos pasadas, y los problemas que presenta su implementación. La Sección IV recoge la propuesta de un nuevo algoritmo que minimiza el principal problema de los métodos anteriores. En la Sección V se muestran los resultados obtenidos en simulación con 250 imágenes usadas habitualmente en aplicaciones de visión y se comparan con los proporcionados por un algoritmo clásico. La Sección VI describe algunos detalles de la implementación y la metodología de diseño seguida. Las conclusiones del trabajo se resumen en la Sección VII.

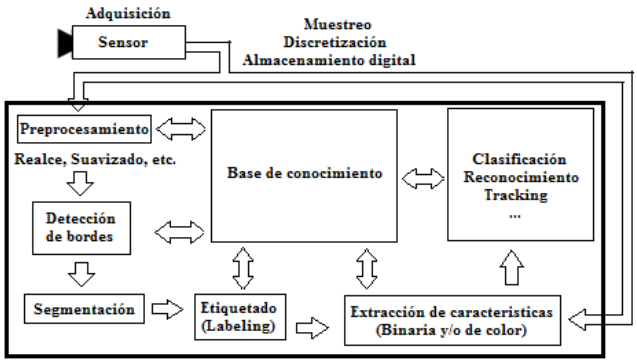

Figura 1. Sistema de visión genérico

II. ESTADO DEL ARTE DE LOS ALGORITMOS CCL

Los algoritmos CCL realizan la asignación de un identificador único a cada conjunto conexo de píxeles con unas mismas propiedades dentro de la imagen. Este trabajo se centra en algoritmos que, partiendo de una imagen binaria (B), van analizando conectividades entre píxeles situados en un entorno formado por cuatro (N_4) u ocho vecinos (N_8), de forma que, al final, dos píxeles, p y q, pertenecerán a un mismo componente cuando ambos se consideren parte del primer plano (o 'Foreground') o del fondo (también llamado 'Background') y exista un camino de píxeles del mismo tipo entre ellos, es decir, cuando se verifique (1), donde S es un subconjunto de píxeles de la imagen binaria B.

Este trabajo ha sido financiado por los proyectos: MOBY-DIC FP7-IST-248858 (www.mobydic-project.eu) de la Comunidad Económica Europea, TEC2008-04920 del Ministerio de Ciencia e Innovación de España y P08-TIC-03674 de la Junta de Andalucía (con soporte FEDER).

$$p\ conectado\ a\ q\ \Leftrightarrow\ \exists\ \{s_i \in S \mid s_1 = p,\ s_{n+1} = q, s_{i+1} \in N(s_i), i = 1, \dots, n\} \quad (1)$$

La Fig. 2 muestra un ejemplo de etiquetado de una imagen donde puede comprobarse cómo el número de componentes encontrados varía en función de la conectividad considerada.

Como consecuencia de la importancia de los algoritmos de CCL, desde la década de los 60 se han invertido muchos esfuerzos en el avance y desarrollo de los mismos. Aunque existen varios criterios que permiten catalogar los distintos procedimientos, como la regularidad en el acceso a memoria o la forma de representación de la imagen, resulta habitual representar la imagen como un array bidimensional y establecer una clasificación atendiendo al número de barridos que se realizan de la misma, entendiendo como tal la exploración de todos los píxeles independientemente del orden que se siga para ello. De este modo, podemos encontrar algoritmos:

- De una pasada (one-scan): Son algoritmos en los que se recorre la imagen una sola vez. Los accesos irregulares y aleatorios a las estructuras de datos que almacenan la imagen o las etiquetas asignadas, y la dificultad para predecir los tiempos de duración de los métodos, son los principales inconvenientes de este tipo de algoritmos, entre los que se encuentran el de trazado de contorno presentado por Chang en 2003 [1] o el de raster-scan presentado por Bailey en 2007 [2].
- De múltiples pasadas (multi-scan): Realizan varios barridos de la imagen (normalmente alternos, de arriba abajo y de izquierda a derecha, y de abajo a arriba y de derecha a izquierda), accediendo a memoria de forma regular. El tiempo de ejecución del procedimiento dependerá de la disposición de los píxeles en cada imagen, por lo que no es posible establecer, a priori, una duración del método en cada caso. Su implementación software y hardware es más sencilla que la de algoritmos de otros grupos. Entre ellos se pueden destacar los trabajos presentados por Haralick en 1981 [3] y Suzuki en 2000-2003 [4], [5].
- De dos pasadas (doble-scan): Son los métodos que llevan a cabo dos barridos de la imagen. Normalmente el primero permite un etiquetado temporal de la misma, mientras que el segundo posibilita la asignación de las etiquetas definitivas a cada píxel. Suelen acceder a memoria de forma regular y usar una o varias tablas para almacenar las equivalencias entre etiquetas distintas asignadas de forma temporal a un mismo componente. De hecho, las estructuras de datos usadas para almacenar estas equivalencias entre etiquetas (que han evolucionado desde matrices de adyacencia o estructuras de n-tuplas [6] hasta estructuras vectoriales [7] y tablas asociativas [8]), los algoritmos empleados para resolver las equivalencias y el instante en el cual tendrá lugar esa resolución, caracterizan las distintas alternativas propuestas en la literatura.

  Muchos trabajos publicados en esta línea se centran también en buscar una solución óptima al problema de exploración de vecinos, con el fin de minimizar el número de accesos a memoria [8]-[10].

  Este tipo de algoritmos son, en general, más difíciles de implementar en hardware que los de múltiples pasadas, su tiempo de ejecución es también elevado y dependiente de la complejidad de la imagen a tratar y pueden presentar, con determinadas estructuras de memoria, solapamientos y pérdidas de etiquetas. Sin embargo, frente a aquellos, presentan la clara ventaja de que permiten establecer la duración concreta del método.

Es importante mencionar que muchos de los algoritmos publicados en los últimos años introducen algún tipo de paralelismo en las estructura de datos y/o en los elementos de procesado. Estas técnicas, basadas en investigaciones iniciadas durante los 70 y los 80 en las cuales se diseñaron e implementaron algoritmos sobre arquitecturas masivamente paralelas (ej. [11]- [16]) permiten acelerar la ejecución a costa de un incremento del área ocupada del dispositivo, lo que se traduce en la imposibilidad de trabajar con tamaños de imagen mayores y en el uso, en ocasiones, de memorias externas (ej. [17]- [20]).

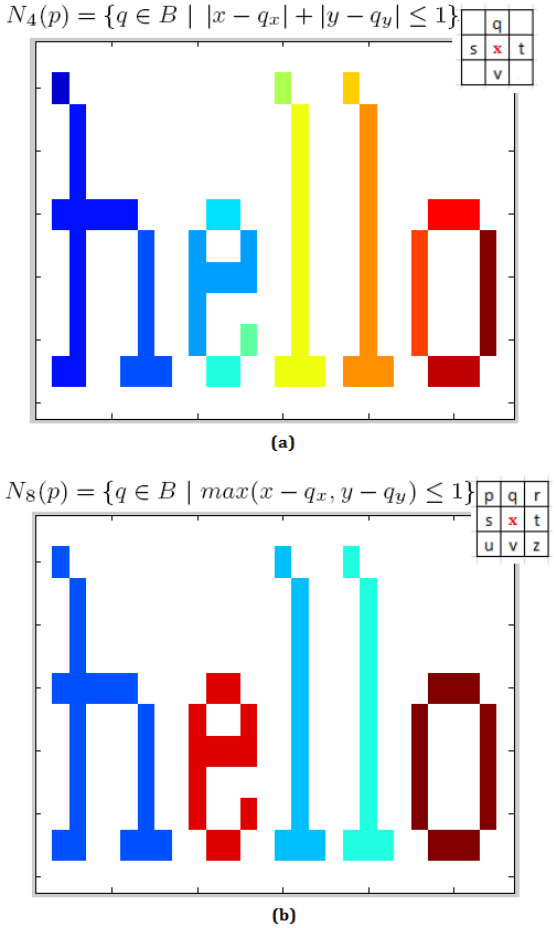

Figura 2. Concepto de conectividad cuando se considera un entorno de vecindad de 4 píxeles (a) u 8 píxeles (b). El array de píxeles considerados en cada caso (también llamado máscara) se muestra en la esquina superior derecha de cada ejemplo

III. ALGORITMOS CCL DE DOS PASADAS

Los algoritmos que se adaptan mejor a una implementación hardware en la que la imagen va a llegar como un flujo continuo de píxeles son los algoritmos de dos pasadas del tipo raster-scan, ya que la forma de acceso a memoria suele ser regular y establecen una duración finita del procedimiento.

A. Algoritmo genérico

Este tipo de algoritmos comienzan realizando un primer barrido de la imagen, en el que se asignan etiquetas temporales a los píxeles y se identifican las posibles equivalencias que puedan producirse. La expresión matemática que permite describir la etiqueta temporal asignada al píxel (x,y) durante el primer barrido es la siguiente:

$$g(x,y) = \begin{cases} F_B & si \quad b(x,y) = F_B \\ m & si \quad \forall\{(i,j) \in M\}\, g(i,j) = F_B \\ L & en\ otro\ caso \end{cases} \tag{2}$$

donde:

- F_B son los píxeles que constituyen el fondo de la imagen binaria (píxeles en negro con valor '0') y F_O son los correspondientes al primer plano (píxeles en blanco con valor '1'),
- M es la ventana que establece el criterio de conectividad entre píxeles,
- m incrementará su valor (m=m+1) cada vez que se verifique la condición $\forall\{(i,j) \in M\}\, g(i,j) = F_B$,
- L es la etiqueta temporal asignada finalmente al píxel (x,y). Esta etiqueta fue asignada ya a alguno de los píxeles vecinos analizados. La forma en que se determina su valor es diferente en los distintos algoritmos propuestos.

Una vez se concluye el primer barrido, o solapado con este total o parcialmente, se realizará la resolución de la tabla de equivalencias. En la segunda pasada se utiliza la información de la tabla de equivalencias para llevar a cabo la sustitución de las etiquetas temporales por permanentes.

En cuanto a recursos, a diferencia de los algoritmos multi-scan, la implementación hardware de un algoritmo de este tipo requiere únicamente una memoria para almacenar las equivalencias entre etiquetas, ya que las etiquetas temporales asignadas a cada píxel durante el primer barrido, que serán necesarias de nuevo en la fase de sustitución, podrán volver a ser calculadas durante esta fase utilizando la misma circuitería empleada en la fase anterior [18].

Como se ha comentado al establecer la clasificación de los algoritmos CCL, existen diferentes propuestas para la implementación de la memoria de equivalencias. De todas ellas, las estructuras vectoriales son las que permiten alcanzar un compromiso mejor entre el consumo de memoria y el coste de procesamiento. Sin embargo, presentan un inconveniente: la posible pérdida de equivalencias.

B. Problema de pérdida de equivalencias y su evaluación

Se producen pérdidas de equivalencias cuando se sobrescriben las posiciones de memoria de la tabla de equivalencias.

El algoritmo de doble-scan que se describe a continuación corresponde a una implementación clásica utilizando este tipo de estructura de memoria. Su análisis permite explicar el problema de pérdidas de equivalencia y será usado en este trabajo para establecer comparaciones con la solución propuesta. Dicho algoritmo, al cual llamaremos 'algoritmo 1', presenta las siguientes características:

- Etiqueta asignada en la primera pasada, si el píxel es blanco y no es una nueva etiqueta:

$$L = min[\{g(i,j)|(i,j) \in M\}] \tag{3}$$

- Actualización de la tabla de equivalencia con cada equivalencia encontrada:

$$T([\{g(i,j)|(i,j) \in M\}]) = min[\{g(i,j)|(i,j) \in M\}] \tag{4}$$

- Modo de resolución de la tabla: Tras el primer scan, se recorre la tabla de equivalencias desde la primera posición hasta la última. Para cada entrada, si la dirección de entrada es diferente de su equivalencia se actualiza dicha entrada en la tabla de acuerdo con:

$$si\ T(Label_i) \neq Label_i \rightarrow T(Label_i) = T\big(T(Label_i)\big) \tag{5}$$

En este método, considerando una conectividad 4, se distinguen dos tipos de pérdidas:

- Simples: Se producen cuando existen dos pares equivalentes que comparten uno de los elementos del par siendo el elemento compartido el de más alto valor de entre las tres etiquetas. Pueden darse en una fila o en filas diferentes. Varias equivalencias entre etiquetas pueden enlazarse (formando una cadena) lo que impide que puedan evitarse estas pérdidas de una forma sencilla.
- Múltiples: Se producen cuando existen más de dos pares equivalentes que comparten uno de los elementos del par, siendo el elemento compartido el de más alto valor de entre los considerados.

La Fig. 3 ilustra casos de pérdidas simples (en una fila (a) o entre filas (b)) y múltiples.

Para analizar con mayor profundidad con qué frecuencia se producen pérdidas de equivalencias, se crearon baterías de imágenes patrón con concavidades, convexidades y escaleras sucesivas y diferente orden de aparición de las etiquetas asignadas. Las simulaciones realizadas con estas imágenes muestran cómo el porcentaje de casos en los que existen pérdidas es muy elevado. Concretamente, con patrones con concavidades en una misma fila (Fig. 4 (a)) hay pérdidas en un 66,7 % de los casos, con patrones escalera creciente hay pérdidas en un 66,7% de los casos (Fig. 4 (b)) y con patrones

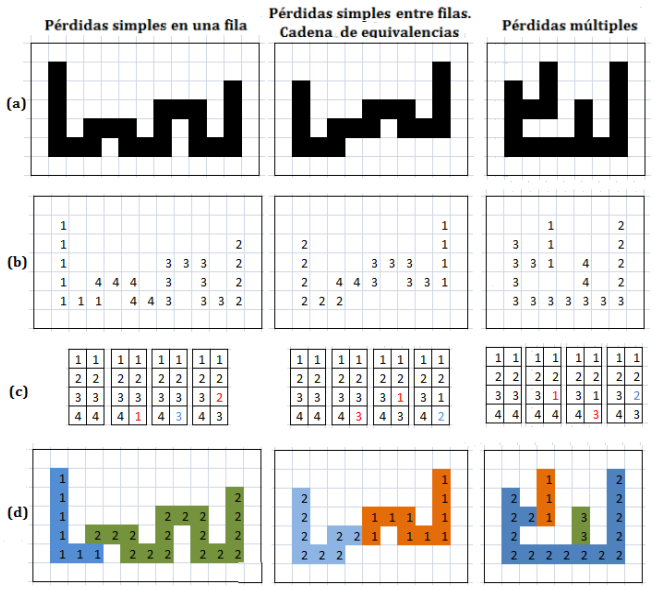

Figura 3. Ejemplos de pérdidas simples y múltiples (a) Imagen. (b) Etiquetado temporal. (c) Evolución de la tabla de equivalencias durante el barrido inicial (izq. inicial, der. final). En rojo, los cambios en cada ciclo. En azul, los casos en los que se sobreescriben equivalencias entre etiquetas. (d) Etiquetado final

con convexidades que comparten una columna (Fig. 4 (c)) ocurren pérdidas en un 50% de los casos.

IV. ALGORITMO PROPUESTO

Con el objetivo de reducir las pérdidas de equivalencias y aprovechar las ventajas que proporciona este tipo de algoritmos de dos pasadas, se ha propuesto una modificación sobre el 'Algoritmo 1'. La etiqueta asignada en la primera pasada (si el píxel es blanco y no es una nueva etiqueta) y la actualización de la tabla de equivalencias se hará de acuerdo a:

$$L = min[\{T(g(i,j))|(i,j) \in M\}] \quad (6)$$

$$T([\{g(i,j)|(i,j) \in M\}]) = min[\{T(g(i,j))|(i,j) \in M\}] \quad (7)$$

Es decir, en lugar de asignar la etiqueta mínima de entre las vecinas, se asigna el mínimo de entre las equivalencias de las etiquetas vecinas, al igual que se propone en [5]. Las entradas de la tabla de las etiquetas del entorno de vecindad se actualizan también con ese mismo valor. La forma en la que se lleva a cabo la resolución de la tabla de equivalencias se mantiene con respecto al 'Algoritmo 1'.

Al analizar la salida de este algoritmo con las distintas imágenes de las baterías patrón, se comprueba que con este método se eliminan las pérdidas múltiples y las simples que tienen lugar en una misma fila, además de reducirse las pérdidas simples que tienen lugar entre filas (el porcentaje de error desciende del 66,7% al 33%). Esto es debido a que se evitan las pérdidas que se producen cuando la equivalencia del elemento compartido sea menor, al producirse la sobrescritura, que la del otro elemento del segundo par equivalente (tanto si se ha modificado su valor directamente como si se ha hecho a través de una cadena con otros pares).

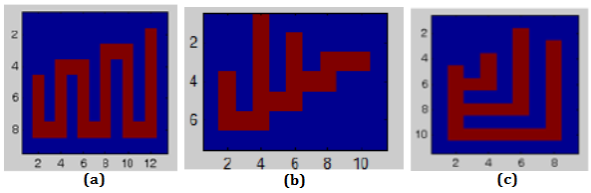

Figura 4. Ejemplos de patrones analizados en el estudio de pérdidas de equivalencias (a) Concavidades sucesivas en una fila (b) Escalera creciente (c) Convexidades sucesivas en una vertical

V. RESULTADOS DE SIMULACIÓN CON IMÁGENES REALES

A. *Imágenes*

Para evaluar la bondad y aplicabilidad del método propuesto se han realizado una serie de simulaciones con varios grupos de imágenes reales utilizando las herramientas del entorno Matlab. Las imágenes, tomadas de las bases de datos de la USC-SIPI [21] y el Berkeley Computer Vision Group [22], han sido seleccionadas intentando abarcar el mayor rango posible en cuanto a temas, para comprobar la aplicabilidad del mismo en distintos campos: medioambiente (animales, paisajes), seguridad (personas), medicina (células), etc.), y dificultad, con el fin de verificar la calidad del etiquetado (para ello se han seleccionado imágenes de texturas e imágenes aéreas).

Para aplicar los algoritmos sobre estas imágenes, fue necesaria la realización de distintas operaciones de preprocesado sobre las mismas: conversiones de color (modelo RGB) a niveles de gris, umbralizaciones mediante el método de Otsu y dilataciones de bordes (Fig. 5).

B. *Medidas realizadas*

La "calidad" del algoritmo se midió mediante el cálculo de los siguientes valores:

- Error absoluto ($Error_A$): Diferencia en el número de componentes conexos encontrados con respecto a los resultados proporcionados por la instrucción 'bwlabel' de Matlab.
- Error relativo ($Error_R$): Error absoluto cometido en cada imagen dividido por el número total de componentes etiquetados en cada caso.

C. *Resultados obtenidos*

Los resultados obtenidos, resumidos en la Tabla I, reflejan que el porcentaje de imágenes en las cuales se cometió errores en el etiquetado descendió de un 55-75% con el 'Algoritmo 1' a un 6-10% con el algoritmo propuesto. Este dato, unido a que el error medio que se comete es en el peor de los casos de 14 etiquetas en el algoritmo propuesto frente a 120 etiquetas en el 'Algoritmo 1', y a que el error relativo máximo cometido con el algoritmo propuesto es de 1,07% (en imágenes de texturas), permite afirmar que la calidad del algoritmo propuesto es buena y mejor que la de un algoritmo típico de este tipo. Además, hay que tener en cuenta que las imágenes con las que

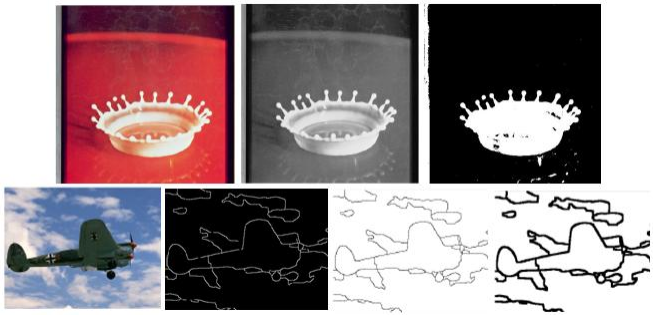

Figura 5. Ejemplo de las operaciones de preprocesamiento realizadas a las imágenes usadas como entrada

se cometen más errores han sido seleccionadas por su dificultad pero normalmente los algoritmos de etiquetado no se aplican a esas imágenes. Con las imágenes que habitualmente serán entrada de estos algoritmos (suelen tener disposiciones más sencillas) el algoritmo propuesto no comete errores.

TABLA I. ANÁLISIS DE LA BONDAD DEL ALGORITMO PROPUESTO: VALORES MEDIOS DE ERROR

N(p)	Imágenes	Algoritmo 1		Algoritmo propuesto	
		$Error_A$	$Error_R$	$Error_A$	$Error_R$
4	Texturas	119,22	8,89	13,88	1,07
	Aéreas	76,11	3.46	11,55	0,5
	Misceláneas	19,33	2.83	3,6	0,3
	Miscelaneas con un preprocesamiento de detección de bordes	0,9	7,4	0	0
8	Texturas	85,22	13,84	1,55	0,18
	Aéreas	88	7,99	1,88	0,14
	Misceláneas	25,8	6.34	0,73	0,2
	Miscelaneas con un preprocesamiento de detección de bordes	6,74	47,83	0	0

VI. IMPLEMENTACIÓN DEL ALGORITMO PROPUESTO

A. *Herramientas y metodología de diseño*

El proceso seguido, así como las herramientas usadas en él, se muestran en la Fig. 6. Se ha partido de una implementación software en Matlab (fichero .m) que ha sido traducida a un modelo Simulink (.mdl). Dicho modelo, compuesto por bloques tanto de la librería básica de Simulink como del Xilinx Blockset, ha sido verificado a nivel lógico y funcional desde Matlab, tras lo cual se ha compilado, generándose en el proceso los ficheros de un proyecto ISE. Desde el entorno ISE, se han realizado estimaciones de área y tiempo y otras operaciones de test. Por último, para comprobar que el funcionamiento real del sistema es el deseado, se ha llevado a cabo una co-simulación HW/SW, cerrando el lazo en el flujo de diseño.

B. *Implementación*

Se han realizado dos implementaciones sobre FPGA del algoritmo propuesto, considerando, en ambos casos, la conectividad en un entorno de 4 vecinos. Una de ellas permite reducir el número de recursos usados en el dispositivo a costa

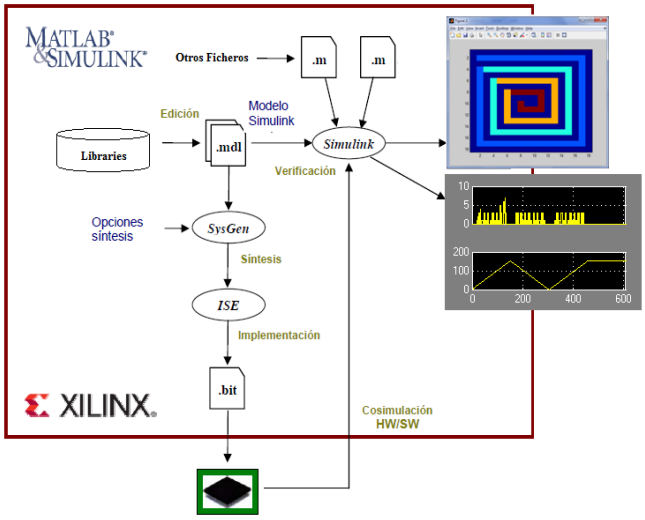

Figura 6. Herramientas usadas en el flujo de diseño

de la utilización de dos ciclos en el procesamiento de cada píxel en cada fase, mientras que la otra minimiza el número de ciclos invertido en procesar cada píxel haciendo uso de una cantidad mayor de memoria.

En ambos diseños, la implementación de la fase inicial del método emplea como buffer una memoria de dos puertos que permite almacenar las etiquetas temporales asignadas a los píxeles de dos filas consecutivas. De esta forma, en el etiquetado de cada píxel, se dispone de las etiquetas de los píxeles de la máscara situados en la fila anterior (en la Fig. 2 (b) los píxeles p, q y r en el etiquetado de x), las cuales fueron almacenadas con anterioridad, y de espacio suficiente para guardar la etiqueta que acaba de ser asignada (cada una de estas dos operaciones, el acceso a r para obtener el equivalente al pixel q en cada ciclo y la escritura de x, se llevan a cabo por un puerto diferente de la memoria). Las etiquetas de los pixeles del entorno del píxel x son a su vez entrada de dirección de una memoria o tabla de equivalencias que proporciona, en cada instante, los valores con los cuales se calcula el mínimo que va a ser almacenado. Además, para cada píxel será necesario comprobar si hay equivalencia entre las etiquetas vecinas. Si la hay, esta es almacenada por un segundo puerto de la memoria de equivalencias.

El hecho de que los diseños traten de optimizar el área ocupada sustituyendo una memoria de etiquetas temporales por este buffer conlleva, además, que sea necesaria la lectura de la imagen en dos ocasiones, así como la inicialización de este buffer de etiquetas entre la fase inicial y la fase de sustitución para que no existan errores en el etiquetado al reutilizar los bloques hardware (la fase de sustitución se realizará de la misma forma que la fase inicial, con la salvedad de que no se modificará la tabla de equivalencias, y los bloques usados en una fase estarán disponibles en la otra).

La implementación de la fase de resolución es diferente en los dos diseños. En el que invierte dos ciclos en procesar cada entrada de la tabla, por un puerto se accede siempre en lectura al equivalente de una etiqueta, o lo que es lo mismo, al valor

de esa entrada de la tabla. Por el otro puerto, en un ciclo se accede en escritura para almacenar el nuevo valor de la entrada anterior a la que está siendo considerada, mientras que en el otro se accede en lectura para obtener el equivalente del equivalente obtenido del primer puerto. En el diseño que invierte tan solo un ciclo en procesar cada píxel son necesarias dos memorias de equivalencias iguales para poder acceder, en un mismo ciclo, al equivalente de una etiqueta ($T(Label_i)$) y al equivalente del equivalente de la etiqueta anterior ($T(T(Label_{i-1}))$), el cual será almacenado por el otro puerto de las memorias en la entrada correspondiente a la $Label_{i-1}$.

Puesto que las memorias son recursos compartidos por las distintas fases del procedimiento, es necesario acceder a ellas mediante bancos de multiplexores controlados mediante señales de fase.

C. Resultados de la implementación

Tras sintetizar los diseños para una placa SPARTAN 3A DSP (XC3SD1800A), se comprueba que la mayor imagen que es posible sintetizar sin el uso de memoria externa considerando una tabla de equivalencias máxima es de 330x240 en el caso del diseño de un ciclo y de 400x370 en el caso del diseño de dos ciclos. No obstante, en la mayoría de los casos, el número de etiquetas asignadas en la primera pasada no excede del 20% de las que se pueden asignar. Si se considerase una tabla de equivalencias con el 30% del tamaño máximo, sería posible sintetizar imágenes con resoluciones de hasta 500x460 y 670x600 respectivamente. En ambos casos, el recurso que limita el tamaño máximo de imagen con el cual es posible trabajar son los bloques de memoria RAM de doble puerto de 16 kb de datos, ya que los porcentajes de utilización del resto de recursos son muy bajos (ej. sólo se usan un 3% de los "Slice Flip Flops" y "4 inputs LUTs" disponibles en la placa). La frecuencia máxima de reloj obtenida es aproximadamente de 32 MHz. Con estos datos, se estima que con el primer diseño se puede trabajar en tiempo real con el tamaño de imagen máximo sintetizable (es decir, si nos ajustamos a los estándares de vídeo existentes, es posible trabajar con un estándar WCIF) siendo la RAM disponible el recurso que impide trabajar con dimensiones mayores. El segundo diseño, por el contrario, está limitado por la velocidad a la cual es posible trabajar. Con él, se puede trabajar en tiempo real con resoluciones de hasta 640x480 (VGA).

VII. CONCLUSIONES

Como consecuencia de los requerimientos de las nuevas plataformas utilizadas por los sistemas de visión, es necesario llevar a cabo un proceso de rediseño de los algoritmos de tratamiento de imágenes que han venido implementándose mediante software en procesadores de propósito general. En este artículo se propone un nuevo algoritmo CCL que, al haber sido diseñado teniendo en cuenta su implementación hardware, constituye una buena opción para integraciones en dispositivos empotrados, alcanzándose con él un compromiso adecuado en consumo de recursos, velocidad de operación y calidad en el etiquetado. Además, en su diseño se ha seguido una metodología novedosa, que ha permitido acortar los tiempos de desarrollo y facilitar la realización de pruebas de integración del algoritmo en sistemas complejos.

BIBLIOGRAFÍA

[1] F.Chang, C. Chen,"A component-labeling algorithm using contour tracing technique", Proc. Int. Conf. Document Anal. Recog., págs. 741–745, 2003

[2] D.G. Bailey, C.T.Johnston, "Single Pass Connected Components Analysis", Proceedings of Image and Vision Computing, New Zeland, págs. 282-287, 2007.

[3] Haralick, R.M. "Some neighborhood operations". [aut. libro] M Onoe, K. Jr. Preston y A Rosenfeld. "Real Time Parallel Computer Image Analysis". New York : Plenum Press, págs. 11-35, 1981.

[4] K.Suzuki, I.Horiba, N.Sugie, "Fast connected-component labeling based on sequential local operarations in the course of forward raster scan followed by backward raster scan", Proc. of 15th International Conference on Pattern Recognition, Barcelona, Spain, págs. 434-437, Sept 3-7 2000.

[5] K.Suzuki, I.Horiba, N.Sugie, "Linear-time connected-component labeling based on sequential local operations". Computer Vision and Image Understanding 89, págs. 1-23, 2003.

[6] A.Rosenfeld, J.L Plazt, "Sequential operator in digital pictures processing", Journal of ACM,vol 13,4, págs. 471-494,1966.

[7] R.Lumia, L.Shapiro, O.Zungia, "A new connected components algorithm for virtual memory computers", Comput. Vision, Graphics, and Image Process. 22 (2), págs. 287–300, 1983.

[8] L.He, Y.Chao, l.Suzuki, "A run-based two-scan labeling algorithm", IEEE Trans. Image Process., vol. 17, no. 5,, págs. 749–756, 2008.

[9] K.Wu, E.Otoo, A. Shoshani , "Optimizing connected component labeling algorithms", Proc. SPIE Conf. Med. Imag, vol 5747, págs. 1965–1976, 2005.

[10] K.Wu, E.Otoo, K.Suzuki, "Optimizing two-pass connected-component labeling algorithms", Pattern. Anal. Applic. 12, págs. 117-135, 2009.

[11] R.Miller, Q.F. Stout, "Varying diameterand problem size on mesh-connected computer", Proc. Intl. Conf. Par. Proc, págs. 679-699, 1985.

[12] D. Nassini, S.Sahni, "Finding connected components and connected ones on a mesh-connected parallel computer" SIAM J.Comput, vol. 9, no 4, , págs. 744-757, 1980

[13] A. Agrawal, L. Nekludova, W.Lim, "A parallel O(log(N)) algorithm for finding connected components in planar images", Proc.Int.Conf.Parallel Processing, págs. 783-786, 1987.

[14] R.Cypher, J.L.C Sanz, L. Snyder "Algorithms for image component labeling on SIMD mesh connected computers", IEEE Trans. Comput, vol.39, no.2, págs. 276-281, 1990.

[15] H.M. Alnuweiri, V.K. Prasanna, "Fast image labeling using local operators on mesh-connected computers" IEEE Trans. Patt.Anal. Machine Intell, vol 13, no 2, págs. 202-207, 1991.

[16] H. Shi, G.X. Ritter, "O(n)-Time and O(logn)-Space Image Component Labeling with Local Operators on SIMD Mesh Conected Computers", International Conference on Parallel Processing, págs. 98-101, 1993.

[17] S.-W Yang et al.,"Vlsi architecture design for a fast parallel label assignment in binary image", Circuits and Systems, ISCAS 2005. IEEE International Symposium, vol 3, págs. 2393–2396, 2005.

[18] H. Flatt et al., "A Parallel Hardware Architecture for Connected Component Labeling Based on Fast Label Merging". Application-Specific Systems, Architectures and Processors, 2008. International Conference, págs. 144 - 149, 2008.

[19] S.-W Yang et al ,"Parallel 3-Pixel Labeling Method and its Hardware Architecture Design", Fifth International Conference on Information Assurance and Security, 2009.

[20] D.K. Kim et al. "Real-Time Component Labeling and Boundary Tracing System Based on FPGA". International Conference of Robotics and Biomimetics, Proceedings of the 2007 IEEE, Sanya, China, págs. 15-18, 2007.

[21] USC-SIPI(University of Southern California- Signal and Image Processing Institute. [En línea] http://sipi.usc.edu/database/.

[22] Berkeley Computer Vision (University of California). [En línea] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html.