Evaluación de algoritmos de correspondencia estereoscópica y su implementación en FPGA

Carlos Colodro, Javier Toledo, J. Javier Martínez, Javier Garrigós y J. Manuel Ferrández[1]

Resumen— **Este trabajo presenta una evaluación de algoritmos para el cálculo de distancias en sistemas de visión estereoscópica y su implementación hardware sobre FPGA. Se han estudiado las principales soluciones basadas en área para el cálculo del mapa de disparidad. Los resultados muestran el compromiso entre calidad de dicho mapa y consumo de recursos hardware que ofrece cada solución, con el fin de servir de guía en la implementación de algoritmos de correspondencia estéreo en sistemas de procesamiento en tiempo real.**

Palabras clave— **visión estéreo, SAD, Census, Rank, filtrado mediana**

I. Introducción

La visión binocular permite a los seres humanos desenvolvernos en el mundo tridimensional que nos rodea. Las pequeñas diferencias en las imágenes adquiridas por nuestros ojos, debidas a la separación entre ellos, son procesadas por el cerebro para percibir el volumen de los objetos y la distancia a ellos. Inspirados en ella, los sistemas de visión estereoscópica intentan emular esta capacidad. La gran cantidad de campos de aplicación, desde la robótica y la interacción hombre-máquina hasta la arquitectura, en los que la información de volúmenes y distancia resulta de interés y conlleva un salto cualitativo en las prestaciones, hace que la visión estereoscópica sea actualmente una de las grandes líneas de trabajo de la visión por computador.

El trabajo en este campo se centra en obtener mapas de disparidad de máxima calidad, para lo que habitualmente se proponen algoritmos optimizados para lograr alta precisión a costa de una complejidad computacional que limita su integración en sistema embebidos de tiempo real. La necesidad de encontrar un equilibrio entre el tiempo de ejecución y la calidad del mapa ha sido considerada recientemente por [1], pero no incluye en su análisis la implementación sobre hardware reconfigurable. Sin embargo, los dispositivos FPGA constituyen la plataforma hardware ideal para explotar al máximo los diferentes niveles de paralelismo de las operaciones involucradas en los algoritmos de correspondencia estéreo, especialmente de los basados en área, como demuestran diferentes implementaciones descritas en la literatura (p.ej. [2], [3], [4], [5]). En estos trabajos, la comparación de prestaciones resulta complicada por las diferentes configuraciones específicas para su aplicación adoptadas por los autores.

El presente trabajo realiza una evaluación de algoritmos para el cálculo de la correspondencia estéreo entre dos imágenes y su implementación sobre FPGA. El objetivo es determinar la solución más apropiada para ser integrada en un sistema de reconocimiento de la mano [6] en aplicaciones embebidas en entornos no controlados donde tanto la cámara como la escena pueden estar en movimiento.

Este artículo está organizado de la siguiente manera. El apartado 2 describe los conceptos básicos de algoritmos de correspondencia en visión estereoscópica. El apartado 3 presenta los resultados obtenidos en la evaluación software de diferentes algoritmos, mientras que en el apartado 4 se evalúa el coste en recursos de una determinada tasa de aciertos en cada algoritmo y se presentan los resultados de implementación para diferentes configuraciones. Finalmente, en el apartado 5 se exponen las conclusiones.

II. Visión estereoscópica

Un sistema básico de visión estereoscópica está formado por dos cámaras separadas una determinada distancia sobre un eje situado en un plano horizontal, y cuyos ejes ópticos son paralelos entre sí y al plano horizontal, y perpendiculares a dicho eje. Dada esta disposición, las imágenes I e I' capturadas por las cámaras presentan, debido a la distancia entre éstas, pequeñas diferencias que se traducen en que un mismo punto P en el espacio ocupa posiciones $I(x_1, y_1)$ e $I'(x_2, y_2)$ con coordenadas diferentes en cada imagen. A la diferencia relativa en la posición de un mismo objeto en una y otra imagen, que por estar en el mismo plano horizontal los centros ópticos de las cámaras sólo puede producirse en dirección horizontal ($y_1 = y_2$), se la denomina *disparidad* $d = x_1 - x_2$. El valor de disparidad del punto P está directamente relacionado con la distancia a la que se encuentre dicho punto de las cámaras. El objetivo de los algoritmos de correspondencia de visión estereoscópica es estimar un mapa de disparidad a partir de las imágenes adquiridas con un sistema de cámaras de geometría determinada. Para ello, consideran un punto en una de las imágenes y buscan su posición en la otra. Esta búsqueda se realiza, como se ha comentado, a lo largo de una línea horizontal, y se acota a un valor de distancia límite denominado *disparidad máxima*. Los valores de disparidad estimados en una serie de puntos forman el mapa de disparidad.

Existen básicamente dos aproximaciones al cálculo de un mapa de disparidad: basada en características y basada en área. En la primera, los puntos para los cuales se determina la disparidad corresponden a la posición de determinadas características, como

[1]Dpto. Electrónica y Tecnología de Computadores, Univ. Politécnica de Cartagena
e-mail: javier.toledo@upct.es

bordes o esquinas, en las imágenes. En este caso, el mapa de disparidad está formado por la diferencia en las posiciones en una y otra imagen de cada característica. El número de correspondencias establecidas determina la *densidad* del mapa de disparidad, y depende por tanto del número de características encontradas en la imagen que son resueltas satisfactoriamente.

Por su parte, los métodos basados en área consideran una determinada *ventana* alrededor de un píxel de una de las imágenes y resuelven la correspondencia buscando en la otra imagen la ventana del mismo tamaño más parecida. El grado de similitud de las ventanas en una y otra imagen es estimado por una determinada función de coste, que para cada ventana centrada en $I(x_1, y_1)$ se evalúa en todo el rango $I'(x_1 + d, y_1)$ acotado por la disparidad máxima $d \in [0, d_{\max}]$. Calculadas todas las funciones de coste de $I(x_1, y_1)$, la solución más extendida es aplicar el algoritmo WTA (*Winner Take All*) y establecer como valor de disparidad la diferencia entre la posición del punto de la imagen de referencia y el centro de la ventana en la otra imagen que minimiza la función de coste. De esta manera, el mapa de disparidad toma valor para cada píxel donde se considere centrada una ventana. En el caso ideal, es posible determinar una correspondencia para cada píxel de la imagen, obteniéndose un mapa de disparidad del mismo tamaño que las imágenes y, en consecuencia, de mucha mayor densidad que el generado por los métodos basados en características.

En la actualidad, las aplicaciones y los trabajos sobre métodos basados en área predominan sobre los de características. Por un lado, la mayor densidad de sus mapas de disparidad es considerada más interesante en muchos campos [7]. Por otro, las tareas de procesamiento que requieren son más fácilmente paralelizables, lo cual se traduce en una ventaja computacional que posibilita su implementación en sistemas embebidos de tiempo real. Por ambos motivos, en este trabajo se ha optado por evaluar métodos basados en área para el cálculo del mapa de disparidad.

III. Evaluación de algoritmos de correspondencia estéreo basados en área

Los algoritmos utilizados para determinar la similitud entre ventanas y para transformar las imágenes de entrada dan lugar a muy diferentes soluciones basadas en área. El gran interés que la aplicación de la visión estéreo despierta en numerosos campos ha propiciado una considerable cantidad de trabajos que proponen muy diferentes técnicas. No obstante, la revisión bibliográfica realizada [1], [7], [8] ha revelado que las funciones de coste más extendidas son SAD (*Sum of Absolute Differences*), SSD (*Sum of Squared Differences*) y NCC (*Normalized Cross Correlation*), y también que es frecuente manipular la información contenida en las imágenes como paso previo a la evaluación de la función de búsqueda sobre las ventanas por medio de las transformaciones no paramétricas Census y Rank.

La transformada Census [9] persigue proporcionar robustez frente a variaciones de iluminación. Para ello, considera una ventana $M \times N$ centrada en un píxel $I(x, y)$ de la imagen y codifica la información de la ventana en una cadena de $MN - 1$ bits en la que asigna valor '1' al bit asociado a cada píxel con valor mayor que el central y valor '0' en otro caso. En su aplicación a visión estereoscópica, una vez transformadas las ventanas centradas en $I(x_1, y_1)$ e $I'(x_2, y_2)$ en las correspondientes cadenas, el algoritmo Census utiliza la distancia Hamming como función de coste para evaluar el grado de parecido entre ellas.

Con el mismo propósito de mejorar la fiabilidad frente a variaciones de iluminación, la transformada Rank [9] considera una ventana de $P \times Q$ píxeles centrada en $I(x, y)$ y le asigna a éste un valor igual al número de píxeles que en dicha ventana tienen un valor menor que el propio píxel central. Realizada la transformación, la evaluación del grado de similitud entre ventanas de diferentes imágenes se puede realizar con cualquiera de las funciones de coste mencionadas, siendo SAD la más habitual [10]. El tamaño de ventana de la transformada Rank es independiente de la que define el área sobre la que se estima la correspondencia con la otra ventana.

Para una primera estimación de la utilidad de la información de distancia en la aplicación mencionada, se ha optado por estudiar estos métodos. Para ello, se han utilizado las imágenes *Tsukuba*, *Venus*, *Teddy* y *Cones* disponibles en la web dedicada a la visión estéreo en la Universidad de Middlebury [11], que actualmente constituyen una referencia para la evaluación de algoritmos [8]. Las Figuras 1, 2 y 3 recogen las tasas de acierto en función del tamaño de ventana. Estos porcentajes han sido calculados considerando una tolerancia del 5% en la disparidad estimada sobre la real. La Figura 4 muestra los mapas de disparidad obtenidos con diferentes soluciones para la imagen *Cones* de la mencionada base de datos.

La Figura 1 muestra el porcentaje de aciertos para SAD, SSD y NCC en función del tamaño N de ventana. En ella se observa que SAD, SSD y NCC presentan un comportamiento muy similar, no apreciándose mejoras significativas en el mapa de la Figura 4.4 frente al de la 4.3. Por tanto, el mayor coste computacional del cálculo del cuadrado de la diferencia en SSD o de la normalización en NCC no se traduce en mejoras significativas en los resultados con respecto a la diferencia absoluta utilizada en SAD. Además, se detecta que los resultados tienden a empeorar a partir de un determinado tamaño de ventana, debido al excesivo suavizado de los bordes tal y como puede apreciarse en la Figura 4.6.

El impacto de la transformación previa de las imágenes de entrada por medio de Rank se muestra en la Figura 2. En todos los casos, son posibles tasas de acierto superiores a las del algoritmo SAD original a partir de un determinado valor de ventana

N, que se reduce a $N = 5$ a partir de ventanas Rank de 7×7. Esta mejora puede apreciarse visualmente comparando el mapa de la Figura 4.5 con el de la 4.3. Se comprueba también que a partir de 7×7 el incremento del tamaño de la transformada no se traduce en mejores resultados.

De acuerdo con los resultados en la Figura 1, Census requiere un tamaño de ventana de al menos 17×17 para aproximarse a los resultados ofrecidos por SAD. Al visualizar los mapas de disparidad obtenidos para configuraciones con tasas de aciertos similares, se observa que las imágenes de Census presentan un serie de errores que se asemejan al ruido impulsional. Es el caso de los mapas generados por SAD 11×11 y Census 21×21 en las Figuras 4.3 y 4.7, respectivamente. Para tratar de mejorar los resultados, es posible completar la transformada Census original con información adicional, de manera que la cadena de bits resultante no dependa exclusivamente del valor del píxel central. Para ello se ha considerado información del gradiente en direcciones horizontal y vertical, calculado por medio del operador Sobel. A esta modificación se la ha denominado CensusE (Census Extendido), y para 21×21 corresponde al mapa de la Figura 4.8. Como se aprecia visualmente y muestra la Figura 1, CensusE mejora a Census para cualquier tamaño de ventana y, a partir $N = 11$, también a SAD.

No obstante, los mapas de disparidad producidos por CensusE también presentan, aunque en menor medida que Census, lo que podría equivaler a ruido impulsional, con píxeles de valor muy diferente a sus vecinos. Para tratar de reducirlo, se propone el post-procesamiento del mapa de disparidad mediante un filtrado de mediana. La Figura 3 muestra la mejora que la aplicación de tal filtrado con diferentes tamaños de máscara conlleva en las tasas de acierto tanto en Census como en CensusE. Visualmente, esta mejora se aprecia en los mapas 4.9 y 4.10.

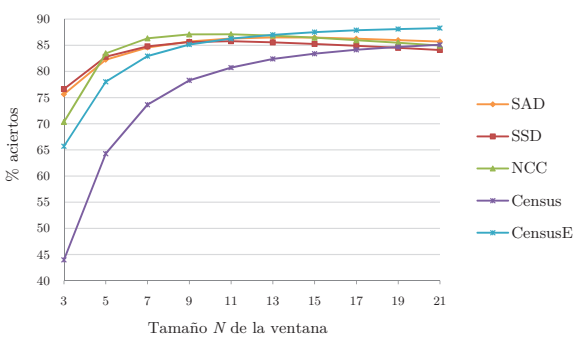

Fig. 1. Tasas de acierto de los algoritmos SAD, SSD, NCC, Census y CensusE, para diferentes tamaños de ventana N.

Finalmente, la Figura 3 también compara estas configuraciones de Census y CensusE con SAD sobre la transformada Rank, comprobándose que existe un tamaño de máscara del filtro para el cual los dos primeros ofrecen mejores resultados.

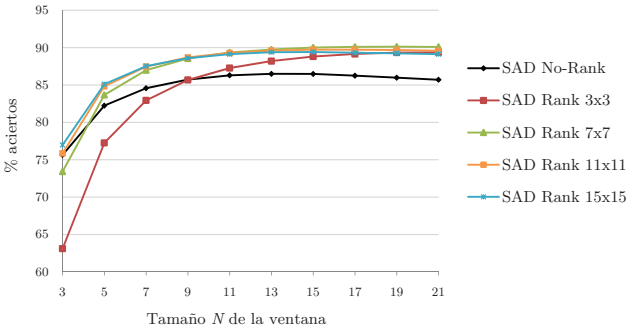

Fig. 2. Tasas de acierto del algoritmo SAD con transformada Rank previa de diferentes tamaños, para diferentes tamaños de ventana N.

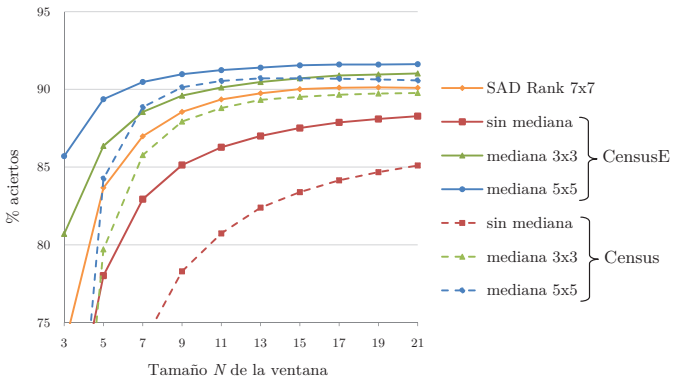

Fig. 3. Tasas de acierto del algoritmo Census con filtrado de mediana de diferentes tamaños, para diferentes tamaños de ventana N.

IV. Estimación de los recursos para implementación en FPGA

Del anterior estudio es posible extraer las conclusiones ya comentadas sobre las características de los algoritmos evaluados y sus comportamientos. Sin embargo, a la hora de tomar una decisión sobre qué algoritmo implementar deben tenerse presente los requisitos de la aplicación a la que va destinado y valorarse el coste de cada uno de ellos.

El coste computacional de los algoritmos de correspondencia estéreo basados en área está determinado fundamentalmente por dos parámetros: el tamaño de ventana y la disparidad máxima. El tamaño $M \times N$ de ventana determina cuantas operaciones hay que realizar para estimar mediante la correspondiente función de coste C el grado de similitud de dos ventanas. Para SAD, requiere calcular la suma de $M \times N$ diferencias absolutas:

$$\mathrm{C_{SAD}}(x, y, d) = \sum_{i=-w}^{i=w} \sum_{j=-w}^{j=w} \mathrm{AD}(x + i, y + j, d)$$

con

$$\mathrm{AD}(x, y, d) = |I(x, y) - I'(x + d, y)|$$

suponiendo ventanas cuadradas $M = N = 2w + 1$. Para Census supone calcular la distancia Hamming de dos cadenas de $MN - 1$ bits:

$$\mathrm{C_{CENSUS}}(x, y, d) = \mathrm{d_{HAMMING}}(t(x, y), t(x + d, y))$$

siendo $t(x, y)$ la transformada Census en una ventana de $M \times N$ píxeles de cada imagen.

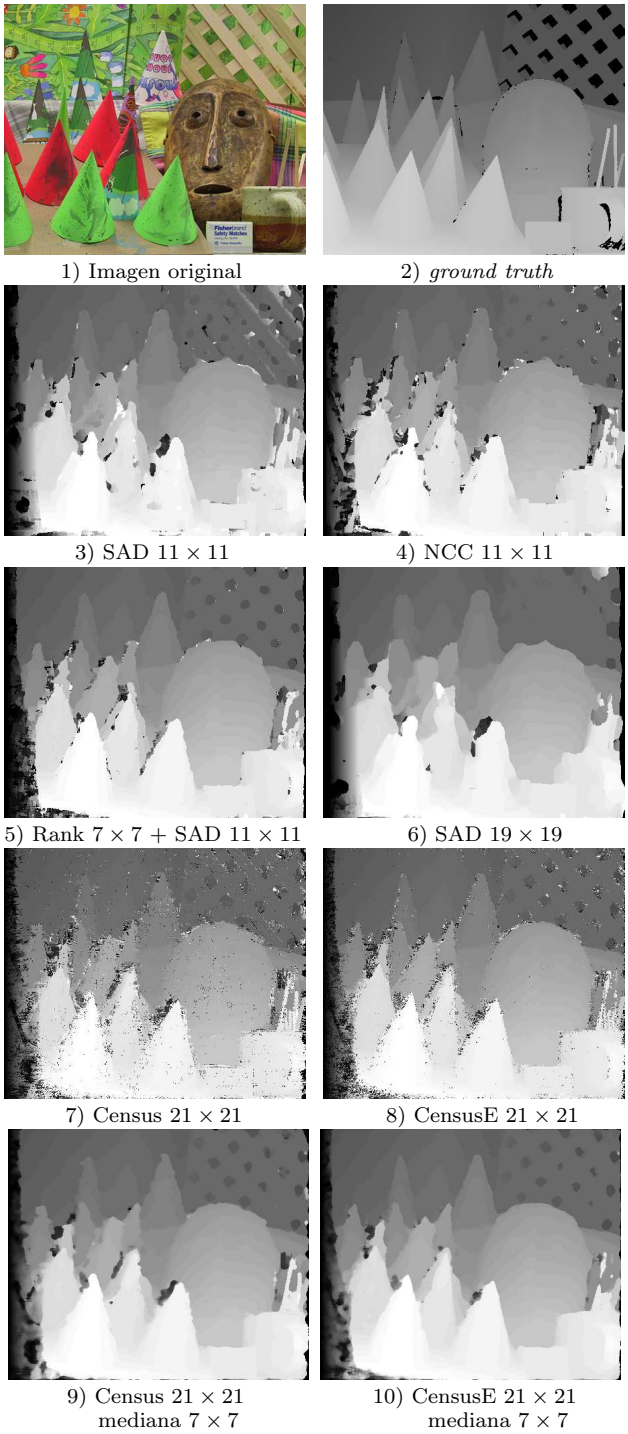

Fig. 4. Ejemplos de mapas de disparidad resultantes para *Cones*

Por su parte, la disparidad máxima $d_{\max}$ determina la evaluación de la función de coste C en $d_{\max}+1$ píxeles $I'(x+i,y)$ con $i \in [0, d_{\max}]$ para cada $I(x,y)$. Finalmente, la disparidad estimada d para cada $I(x,y)$ es el valor i que minimiza $\{C(x,y,i)\}$.

En la implementación hardware sobre FPGA, este coste computacional se traduce en el número de píxeles $I(x,y)$ cuya disparidad puede ser estimada por unidad de tiempo (*throughput*) y en el área consumida para realizar las operaciones necesarias. Ambos valores dependen de la cantidad y complejidad de las operaciones que deba realizar el algoritmo. Además, para cada algoritmo el *throughput* está directamente relacionado con los recursos consumidos

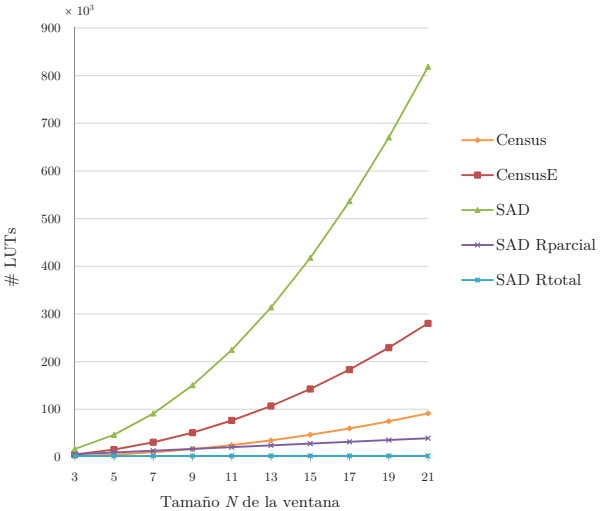

Fig. 5. Estimación de LUTs necesarias en función de N para algoritmos evaluados

por medio del grado de paralelismo de su implementación. Con el fin de poder determinar el algoritmo más eficiente, se ha evaluado el coste en recursos de una determinada tasa de aciertos, de manera que sea posible alcanzar el mejor compromiso entre calidad de los resultados y coste de la implementación. Para ello, se ha estimado el número de recursos necesarios utilizando Xilinx System Generator. El *throughput* viene impuesto por la aplicación, que para vídeo es típicamente de 30 *frames* por segundo (fps). Para asegurar esta tasa, los algoritmos han sido diseñados para maximizar el *throughput* explotando al máximo el nivel de paralelismo, de manera que todas las operaciones de la función de coste se realizan en paralelo y a su vez las $d_{\max}+1$ funciones de coste también son calculadas en paralelo. Esto asegura que el cálculo de la disparidad de un píxel requiere de un único ciclo de reloj.

La Figura 5 muestra la estimación del número de LUTs de 4 entradas necesarias para implementar Census, CensusE y SAD con diferentes tamaños de ventana. La adopción de esquemas que maximizan el paralelismo conlleva el aumento exponencial de los recursos con el tamaño de ventana. Este comportamiento es especialmente destacado en SAD, consecuencia del mayor consumo de recursos de sus operaciones aritméticas con respecto al de las operaciones sobre cadenas binarias de ambas versiones de Census. También se observa que la inclusión del gradiente en dirección horizontal y vertical hace que CensusE triplique los recursos de Census.

El estudio detallado de las operaciones que realiza el algoritmo SAD en cada píxel permite comprobar que es posible reutilizar resultados intermedios que han sido calculados en el cómputo de la suma de diferencias absolutas en algún píxel anterior. Para un tamaño de ventana 5×5, la Figura 6 permite comprobar que tanto en dirección horizontal, de (x,y) a $(x+1,y)$, como en dirección vertical, de (x,y) a $(x,y+1)$, el cálculo de SAD repite operaciones realizadas anteriormente. Ejemplificado en dicha figura para $d=0$, este análisis se puede extender a todo el rango de disparidad hasta $d_{\max}$. Generalizando para

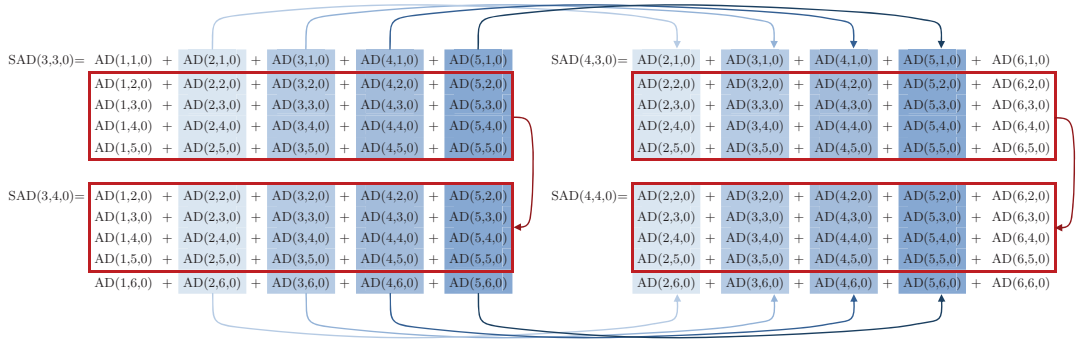

Fig. 6. Esquema de reutilización de datos intermedios en SAD

cualquier (x, y), se comprueba que:

$$\begin{aligned} \mathrm{C}_{\mathrm{SAD}}(x,y,d) &= \mathrm{C}_{\mathrm{SAD}}(x-1,y,d) \\ &+ \mathrm{C}_{\mathrm{SAD}_y}(x+w,y,d) \\ &- \mathrm{C}_{\mathrm{SAD}_y}(x-w-1,y,d) \end{aligned}$$

con $\mathrm{C}_{\mathrm{SAD}_y}$ dado por:

$$\mathrm{C}_{\mathrm{SAD}_y}(x,y,d) = \sum_{j=-w}^{j=w} |I(x,y+j) - I'(x+d,y+j)|$$

Extendido el análisis en dirección vertical, en cada columna y se pueden reutilizar datos anteriores, puesto que:

$$\begin{aligned} \mathrm{C}_{\mathrm{SAD}_y}(x,y,d) &= \mathrm{C}_{\mathrm{SAD}_y}(x,y-1,d) \\ &+ \mathrm{AD}(x,y+w,d) \\ &- \mathrm{AD}(x,y-w-1,d) \end{aligned}$$

La aplicación de la primera solución permite reducir de $M \times N$ a M la cantidad de módulos AD necesarios a cambio de memoria donde almacenar la suma de sus respectivas salidas durante los N cálculos anteriores, dando lugar a una implementación que se ha denominado R_{Parcial}. La aplicación conjunta de los dos soluciones, llamada R_{Total} sólo necesita un cálculo de diferencia absoluta a cambio de memoria donde almacenar los citados resultados parciales para M líneas de la imagen. De esta manera, es posible ocupar menos recursos para implementar operaciones aritmético-lógicas a cambio de memoria donde guardar resultados parciales anteriores.

La Figura 5 muestra las LUTs necesarias para las configuraciones R_{Parcial} y R_{Total}. Como cabría esperar, si el cálculo de SAD en cada instante se reduce a una única diferencia absoluta, cuyo resultado es sumado y restado, según corresponda, con resultados anteriores para dar el valor $\mathrm{C}_{\mathrm{SAD}}(x,y,d)$, el número de LUTs es el menor posible y permanece prácticamente constante frente al cambio del tamaño de ventana. Como contrapartida, la cantidad de recursos de memoria necesarios aumenta exponencialmente, como se refleja en la Figura 7. Por tanto, esta configuración representa el extremo opuesto en el espacio de diseño a la configuración SAD original y ahora son los recursos específicos de memoria *Block-RAM* los que acotan el tamaño máximo de SAD.

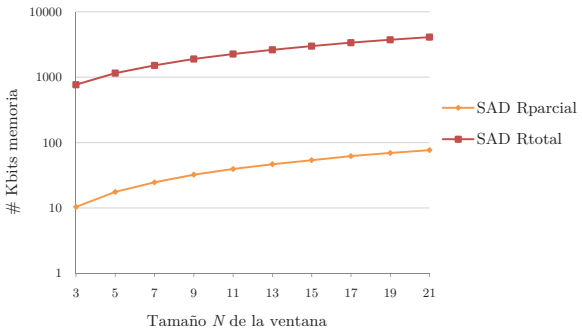

Fig. 7. Estimación de Kbits de memoria necesarios para guardar resultados parciales con R_{Parcial} y R_{Total} en función de N.

La configuración con reutilización parcial ofrece un compromiso entre ambos extremos. Las Figuras 5 y 7 muestran que los requisitos de memoria y de LUTs se reducen aproximadamente en dos y un orden de magnitud, respectivamente, con respecto a los peores casos. Naturalmente, la tasa de aciertos es independiente del grado de reutilización de resultados intermedios.

Finalmente, la Figura 8 representa la tasa de aciertos de cada algoritmo en función del número de LUTs estimado para su implementación. Para SAD únicamente se representa la configuración R_{Parcial}. De acuerdo con los resultados de la Figura 3, se ha considerado filtrado de mediana 5×5 para Census y CensusE, y Rank 7×7 para SAD. Ambos, al igual que el filtrado con Sobel 3×3 para el cálculo del gradiente que requiere CensusE, han sido diseñados maximizando el paralelismo y se ejecutan en un único ciclo de reloj. Estas gráficas conducen a la conclusión de que Census, CensusE y SAD R_{Parcial} con Rank presentan un comportamiento superior al resto y muy similar entre sí, con tasas de acierto prácticamente idénticas para el mismo consumo de recursos. La implementación de Rank y el filtrado de mediana contribuye a mejorar los resultados con un coste mínimo en recursos. Census y CensusE permiten mejorar la tasa de aciertos, pero con un gran coste en recursos para una mejora mínima.

Naturalmente, estos resultados de ocupación dependen de la disparidad máxima. En este estudio, su valor ha sido fijado de acuerdo con el rango de distancias de los objetos en la escena y es, para todos los casos, $d_{\max} = 63$. Por evaluarse todas las

	Rank 7 × 7	Mediana 5 × 5	Census 9 × 9	CensusE 5 × 5	SAD 11 × 11 R_{Parcial}
Flip-flops	585	1976	18247	17347	16965
LUTs 4-input	533	2747	18384	17204	24558
Area (*slices*)	555	1468	13804	12988	15334
BlockRAM	6	4	16	30	22
T_{min} (ns)	6.70	4.60	9.99	9.85	9.91

TABLA I

RESULTADOS DE IMPLEMENTACIÓN PARA DIFERENTES CONFIGURACIONES DE ALGORITMOS DE CORRESPONDENCIA SOBRE VIRTEX-4

C_i funciones de coste en paralelo, $d_{\max}$ afecta linealmente a los recursos necesarios pero no altera las conclusiones extraídas.

La Tabla I resume los resultados de implementación para SAD 11 × 11 con Rank 7 × 7, de Census 9 × 9 con mediana 5 × 5 y de CensusE 5 × 5 con el mismo filtro de mediana. Estas son las configuraciones que en las gráficas de la Figura 8 alcanzan el 90% de tasa de aciertos con un menor número de recursos. Los diseños han sido realizados con Xilinx System Generator. La tabla recoge de manera independiente los resultados de los algoritmos de correspondencia, siendo posible comprobar que el reducido impacto de Rank y del filtrado en los recursos ocupados frente a los requeridos por cada algoritmo se traduce, no obstante, en una mejora considerable de los mapas generados. El periodo mínimo reportado por el *Place&Route* realizado con Xilinx ISE indica el tiempo necesario para calcular la disparidad en un píxel. Para las imágenes de 352 × 288 consideradas, esto supone aproximadamente 1 milisegundo por *frame*, en cualquiera de las configuraciones implementadas. Ni Rank ni la mediana comprometen estas frecuencias máximas de funcionamiento. Un mayor tamaño de imagen implicaría más recursos de BlockRAM para almacenar los datos de cada línea necesarios, y considerar un valor mayor de disparidad máxima para mantener el rango de distancia medible. Con los resultados obtenidos, en la familia Virtex-4 es posible el procesamiento de 100 fps de vídeo de alta definición 1280 × 720 con $d_{\max} = 200$.

V. CONCLUSIONES

El presente trabajo realiza una evaluación de algoritmos para el cálculo del mapa de disparidad en sistemas de visión estereoscópica y su implementación hardware sobre FPGA. Siendo habitualmente necesaria su inclusión en sistemas embebidos que procesan vídeo en tiempo real, una solución eficiente requiere un compromiso entre la calidad de los resultados necesaria y las prestaciones del diseño. Para ello, se ha estimado la tasa de aciertos de cada algoritmo en función de su consumo de recursos. En todos los casos considerados, los algoritmos han sido diseñados con el máximo grado de paralelismo posible, lo cual asegura procesamiento en tiempo real.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por la Fundación Séneca de la Región de Murcia a través del proyecto 15419/PI/10.

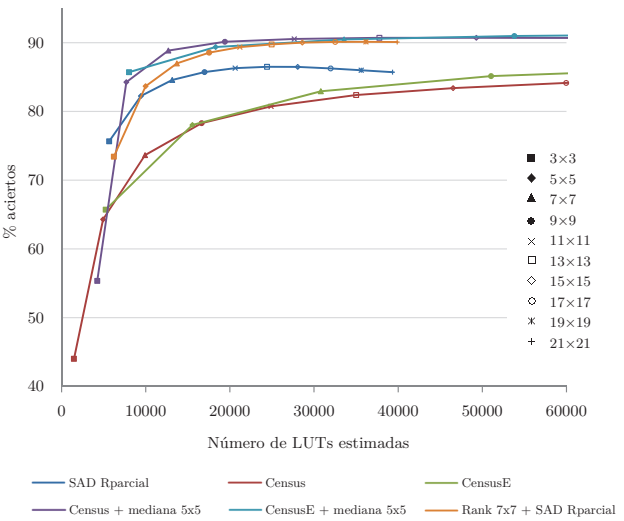

Fig. 8. Tasa de aciertos de cada algoritmo en función de las LUTs estimadas necesarias para su implementación.

REFERENCIAS

[1] M. Humenberger, C. Zinner, M. Weber, W. Kubinger y M. Vincze *A fast stereo matching algorithm suitable for embedded real-time systems*, Computer Vision and Image Understanding 114, pp 1180–1202, Elsevier, 2010.

[2] K. Ambrosch, W. Kubinger, M. Humenberger y A. Steininger, *Hardware implementation of an SAD based stereo vision algorithm*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–6, 2007.

[3] C. Murphy, D. Lindquist, A.M. Rynning, T. Cecil, S. Leavitt y M. Chang *Low cost stereo vision on an FPGA*, Int. Symposium on Field-Programmable Custom Computing Machines, pp. 333–334, 2007.

[4] D.S. Kim, S.S. Lee y B.H. Choi, *A real time stereo depth extraction hardware for intelligent home assitant robot*, IEEE Transactions on Consumer Electronics, 56 (3), pp. 1782–1787, 2010.

[5] K. Ambrosch, C. Zinner y W. Kubinger *Accurate hardware based stereo vision*, Computer Vision and Image Understanding 114, pp. 1303–1316,Elsevier, 2010.

[6] J. Toledo, J. Martínez y J.M. Ferrández, *A Hand-based interface for augmented reality*, IEEE Symposium on Field-Programmable Custom Computing Machines, pp. 291–292, 2007.

[7] D. Scharstein y R. Szeliski, *A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms*, Int. Journal Computer Vision, 47 (1-3), pp. 7–42, 2002.

[8] H. Hirschmuller y D. Scharstein, *Evaluation of Stereo Matching Costs on Images with Radiometric Differences*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 31 (9), pp 1582–1599, 2009.

[9] R. Zabih y J. Woodfill, *Non-parametric local transforms for computing visual correspondence*, Int. Conference on Computer Vision ECCV '94, pp. 151–158, 1994.

[10] J. Banks y M. Bennamoun *Reliability analysis of the rank transform for stereo matching*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 31 (6), pp 870–880, 2001,

[11] *Middlebury College Stereo Vision Research Page*, http://vision.middlebury.edu/stereo