Procesamiento de Imagen para Seguimiento de Objetos Basado en MicroBlaze

Martín-Ortega Alberto[1], Aguayo Estanislao, Gómez Arribas Francisco, Lopez-Buedo Sergio[2]

[1] alberto.martino@estudiante.uam.es
[2] sergio.lopez-buedo@ii.uam.es
Escuela Politécnica Superior, Madrid, Spain,
http://www.eps.uam.es

Abstract. En este artículo se presenta la implementación en una FPGA Xilinx Spartan-2E de un algoritmo de seguimiento de objetos, utilizando el microprocesador embebido MicroBlaze. La flexibilidad que ofrece esta alternativa ha permitido implementar el sistema empleando sólo una FPGA, memoria externa y un decodificador de vídeo, lo que permite reducir sus costes. Adicionalmente, el uso de las librerías estándar GNU permite que sea trivial prototipar los algoritmos en un PC para más tarde implementarlos en MicroBlaze.

1 Introducción

El uso de *soft-processors* implementados sobre FPGAs presenta una serie de ventajas sobre la alternativa convencional de emplear microcontroladores. En especial, al usar un circuito programable se pueden diseñar el controlador a medida, incluyendo sólo los periféricos que son necesarios, y además, aprovechando los recursos de la FPGA para implementar la *glue-logic* siempre necesaria en todo sistema. El resultado es que con sólo una FPGA, y probablemente unas memorias externas, se puede construir el sistema completo.

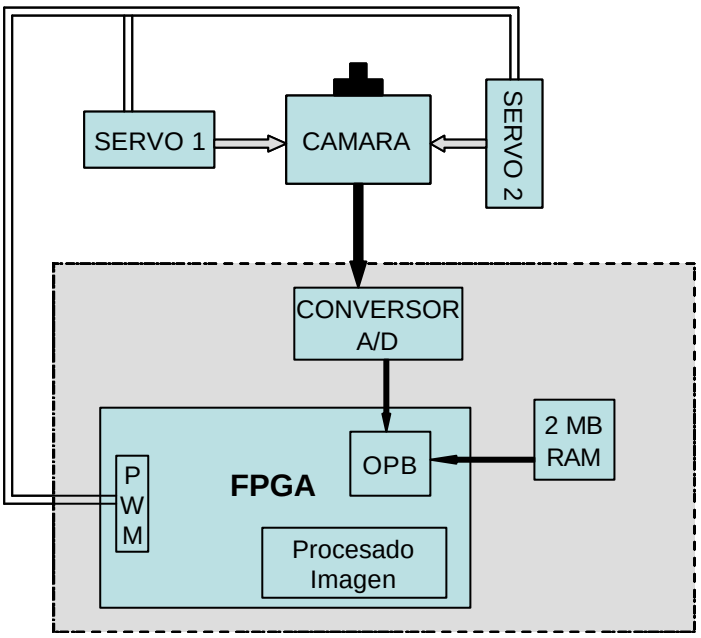
En este caso, se ha implementado el algoritmo empleando sólo una FPGA de bajo coste (Spartan-2E), memoria SRAM externa y un decodificador de vídeo (TVP5145) [6]. Una solución convencional hubiera añadido un microcontrolador adicional, del que probablemente muchos recursos no hubieran sido usados. Al mismo tiempo, tampoco habría podido prescindir de la lógica programable, pues hubiera sido necesaria para conectar el decodificar de video con el microcontrolador, sin tener en cuenta la lógica adicional que hubiera necesitado.

MicroBlaze es el encargado de establecer el protocolo y los buses de datos para la comunicación con el conversor, gestionando el buffer de almacenamiento de imágenes. Gracias al *soft-processor*, es posible utilizar lenguajes de más alto nivel (ie. C, C++), por lo que los algoritmos de procesamiento de imágenes son fácilmente implementables. El módulo PWM

implantado para el movimiento de los servos también realizado sobre MicroBlaze, muestra la salida del sistema posibilitando el seguimiento del objeto por todo el escenario de actuación.

2 Arquitectura

La entrada del sistema es una cámara analógica sujetada entre dos servos. Uno de ellos se encarga del movimiento horizontal, mientras que el otro del movimiento vertical, consiguiendo así una cobertura total en un espacio de 180º x 180º. La señal enviada por la cámara a la placa, es capturada por el conversor A/D, el cual envía la señal digital a la FPGA.



La comunicación entre MicroBlaze y el conversor A/D se realiza a través del Onchip Peripheral Bus (OPB), así como la comunicación con la memoria RAM externa. El código ejecutable del procesado de imagen, captura las imágenes almacenadas por MicroBlaze en un buffer, las procesa, y utilizando el modulo PWM comunica a los servos el movimiento relativo que deben realizar.

2.1 Implementación del sistema en la FPGA Sparta-2E

El sistema ha sido implementado en la placa de desarrollo de Van Microsistemas - XC2S400E.

En la siguiente tabla podemos observar cual es el grado de ocupación de la FPGAS con el sistema completo.

UTILIZACIÓN	DE LA	FPGA
Componente	Num	Porcentage
GCLKIOBs	1	25%
IOBs	92	51%
BLOCKRAMs	28	70%
SLICEs	2761	57%
BSCAN	1	100%
GCLKs	2	50%

Tabla 1.

4 Procesado de Imagen

Los problemas que comúnmente más afectan al procesado de imágenes mediante sistemas Hardware son su espacio y complejidad. Estos dos problemas están enfrentados, ya que el espacio que ocupa una imagen es indirectamente proporcional a la complejidad de esta. Es decir, se ha tenido que decidir si compensaba más tratar imágenes grandes fácilmente procesables, o bien imágenes más pequeñas pero más complejas de tratar. Al final se ha optado por escoger algoritmos donde solo necesitamos una imagen a la vez, minimizando así el espacio requerido, para poder utilizar imágenes sencillas.

El objetivo del procesado de imagen es conseguir hallar el desplazamiento del objeto dentro de esta, mediante la utilización de algoritmos de segmentación. Estos algoritmos son descritos en los puntos 4.1 y 4.2.

Las imágenes que se van a procesar dentro de MicroBlaze serán de 320x240 píxeles en escala de grises. Cada píxel estará definido por un número entre 0 y 255 indicando la luminosidad de este, donde 255 representa el color blanco, y el 0 el negro.

4.1 Algoritmos de búsqueda de Umbral

Una parte muy importante en el procesado de imágenes para el seguimiento de objetos es la utilización de técnicas de segmentación. La segmentación consiste en la búsqueda de un

umbral que permita distinguir entre el objeto y el entorno donde se encuentra. Muchos son los autores que se han preocupado por este tema abordándolo desde diferentes frentes: medida difusa de los píxeles [1], cálculo de la entropía relativa [3], comparación de histogramas o bi-histogramas [4].

El problema de intentar hacer un sistema de seguimiento estándar para cualquier escenario, reside en la amplia diversidad de colores y formas distintas que se encuentran en el, los cuales pueden parecerse más o menos al objeto que pretendemos aislar. Además sin conocer la forma o color del objeto nos encontramos con un dilema, incluso después de haber encontrado el umbral que distinga entre ambos, no sabríamos decir con seguridad si el objeto es la parte más oscura o la más clara, por lo que a la hora de binarizar la imagen y extraer el objeto se tendrá que decidir por una u otra. Este problema se treta más adelante en el apartado de algoritmos de binarización.

Puesto que la utilización de algoritmos con contribuciones estadísticas están hasta el momento fuera del alcance con un solo procesador, proponemos otros dos algoritmos. El primero busca la simplicidad y rapidez de procesamiento. Consiste en procesar todos los píxeles de la imagen hasta hallar el valor medio predominante de esta. Es tan sencillo como eficaz, sin embargo está pensado para ser utilizado en ausencia del objeto que se pretende seguir, es decir, antes de que el objeto entre en el escenario. Por ello se propone a su vez otro algoritmo. El segundo pertenece a los basados en bi-histogramas. Fue propuesto por T.W. Ridler y S. Calvard [5]. El histograma es inicialmente segmentado en dos partes utilizando un umbral inicial de 2B-1 (donde B = numero de bits por píxel), siendo la mitad del limite superior. El valor medio (mf, 0) de los píxeles asociados al fondo de la imagen, y el valor medio (mb, 0) de los píxeles asociados al objeto son computados. Se haya un nuevo umbral como media de los dos valores medios. El proceso se repite, con el nuevo umbral encontrado, hasta que el umbral no cambia en las sucesivas iteraciones. La ecuación se describe :

$$?_k = (m_{f,k-1} + m_{b,k-1})/2 \text{ hasta } ?_k = ?_{k-1} \qquad (1)$$

Fig 1(a). Imagen de muestra.

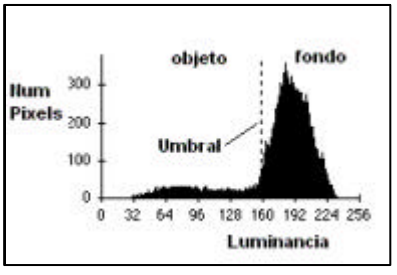


Fig 1(b). Histograma de Luminancia

Este algoritmo es mucho mas eficaz a la hora de separar objeto y fondo, sin embargo requiere mayor tiempo de procesamiento.

Para conseguir resultados óptimos, se recomienda utilizar una combinación de ambos. Primero utilizaríamos el umbral medio hallado al iniciar la exploración, mientras que más adelante corregiríamos las desviaciones respecto al umbral real, con la ayuda del algoritmo iterativo.

4.2 Algoritmos de binarización

El objetivo último de la segmentación es conseguir aislar el objeto para poder determinar, entre otras cosas, la posición dentro de la imagen o la cantidad de movimiento. Como se indica en el apartado anterior, una vez hemos hallado el umbral de corte para separa objeto y fondo, se debe poder distinguir si el objeto es la parte más oscura o por el contrario la más clara.

Para un escenario conocido, este problema se resuelve fácilmente. Si nos encontramos en un escenario con colores oscuros y el umbral queda emplazado por encima de estos, claramente se debe coger el objeto de aquellos píxeles por encima del umbral (Fig. 2(a)). Y si por el contrario nos encontramos un escenario con tonos claros, el objeto quedara encerrado en la parte inferior del histograma de luminancia (Fig. 2(b)).

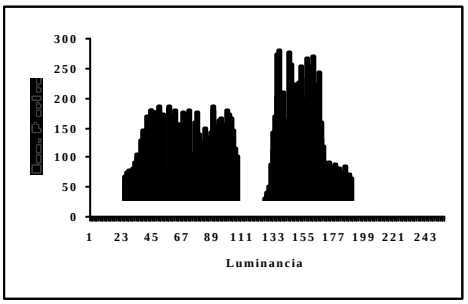


Fig 2(a). Distribución de Píxeles por Luminancia en Fondo poco luminoso.

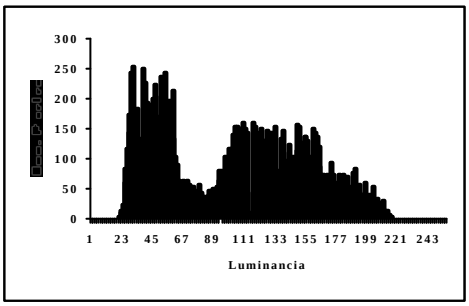


Fig 2(b). Distribución de Píxeles por Luminancia en fondo muy luminoso.

Si por el contrario lo que se conoce es el objeto, se podría utilizar un algoritmo más eficiente que consiste en marcar un rango de binarización para un objeto determinado. Es decir, dentro de los 256 posibles valores de la escala de grises, se reserva un rango para el objeto, y el resto seria el fondo de la imagen (Fig. 3).

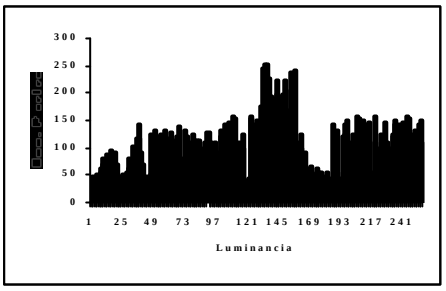


Fig 3. Objeto conocido.

La implementación de los algoritmos anteriormente descritos, se ha realizado en lenguaje C y precompilado (el prototipo) en Linux, y posteriormente compilado por el compilador GNU de MicroBlaze mb-gcc.

La complejidad total en líneas de código es de: 589 LDC. y ocupa un total de 1273185 Bytes de memoria.

5 Conclusiones

En este artículo se ha conseguido demostrar las ventajas que ofrece un procesador embebido como MicroBlaze en una FPGA. La minimización del coste y espacio son factores muy importantes a la hora de implementar un sistema de seguimiento de objetos. Otro factor tan importante como los anteriores es la facilidad que ofrecen las librerías de GNU a la hora de prototipar los algoritmos directamente en el PC, lo cual minimiza el tiempo de desarrollo drásticamente.

Las facilidades que ofrece un sistema de este tipo, son esperanzadoras a la hora de implementar algoritmos más complejos de procesado de imagen, e incluso la utilización de varios microprocesadores aumentaría sustancialmente el rendimiento.

6 Agradecimientos

Este trabajo ha sido financiado por los proyectos 07T/0052/2003-3 de la Consejería de Educación de la Comunidad de Madrid y 161000 de la Fundación General de la U.A.M.

7 Referencias

[1] **Image thresholding using measures of fuzziness**. Yumusak, N.;Temurtas, F.;Cerezci, O.;Pazar, S.; Industrial Electronics Society, 1998. IECON '98. Proceedings of the 24th Annual Conference of the IEEE, Volume: 3 , 31 Aug.-4 Sept. 1998.

[2] **Optimum Run Length Codes.** Meyr,H. Rosdolsky, H. Huang,T. Res. Div., Hasler Ltd., Berne, Switzerland. Communications, IEEE Transactions on [legacy, pre - 1988].

[3] **An efficient blood vessel detection algorithm for retinal images using local entropy thresholding.** Chanwimaluang, T.; Guoliang Fan; Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on , Volume: 5 , 25-28 May 2003.

[4] **Indexing via color histograms.** Swain, M.J. Ballard, D.H. Dept. of Comput. Sci., Rochester Univ., NY, USA; Computer Vision, 1990. Proceedings, Third International Conference on.

[5] **Picture Thresholding Using an Iterative Selection Method.** T.W. Ridler, S. Calvard. IEEE Trans. System, Man y Cibernetics, SMC-8 (1978).

[6] **Decodificador de video TVP5145 de TI.**
http://focus.ti.com/docs/prod/folders/print/tvp5145.html