Arquitectura hardware para el seguimiento de objetos utilizando procesamiento piramidal

Marco Aurelio Nuño Maganda, Miguel Arias Estrada

Instituto Nacional de Astrofísica, Óptica y Electrónica,
Luis Enrique Erro # 1, Santa María Tonanzintla, Puebla, México.
nmaganda@cseg.inaoep.mx , ariasm@inaoep.mx

Resumen. En este trabajo se presenta una arquitectura hardware para el seguimiento de objetos. La ventaja de esta arquitectura con respecto a otras es el uso de pirámides para hacer el seguimiento. La arquitectura se divide en dos grandes bloques: la arquitectura que obtiene la pirámide de la imagen y la arquitectura que efectúa el proceso de correlación de la imagen con un patrón dado, utilizando operaciones basadas en pirámides. Se presentan resultados para ambos bloques de la arquitectura y se bosqueja el funcionamiento de ambos bloques interactuando uno con el otro.

1 Introducción

La visión por computadora es una de las principales ramas de la Inteligencia Artificial que se explora actualmente[1]. Las aplicaciones de visión obtienen un conjunto de imágenes de una determinada fuente, las procesan y arrojan resultados en forma de imágenes o de datos que después son analizados dependiendo de la aplicación.

A lo largo del tiempo se han utilizado varios dispositivos para el procesamiento digital de imágenes. Uno de ellos es la PC, la cual comenzó a ser utilizada como elemento de procesamiento cuando se comenzaron a producir en serie y su costo se redujo. Una de las desventajas de la PC es la forma secuencial de realizar el procesamiento. Se ha contemplado el uso de supercomputadoras o estaciones de trabajo dedicadas, pero éstas tienen un costo muy elevado con respecto a las PCs. Otra de las alternativas es el uso de plataformas de desarrollo con FPGAs (Field Progammable Gate Arrays) como elemento central de procesamiento. Los FPGAs son dispositivos digitales programables que a diferencia de los microprocesadores, no se programa en ellos un conjunto de instrucciones, sino la descripción de una arquitectura en función de componentes básicos [2]. Posteriormente, a través de las herramientas proporcionadas por los fabricantes, se implementa esa arquitectura mapeando la descripción de los componentes básicos en los recursos disponibles en el FPGA. La potencia de cómputo de los FPGAs se mide en el número de bloques lógicos o en el número de compuertas equivalentes contenidas en él. El desarrollo de FPGAs con mayor cantidad de recursos cada vez nos permite la implementación de sistemas complejos de visión en tiempo real. En la actualidad existen dispositivos FPGA con capacidad equivalente a millones de compuertas digitales.

A. Gelbukh, M. Hernández Cruz (Eds.) Avances en la ciencia de la computación en México, CORE-2003, pp. 198–208, 2003.

Por otro lado, el seguimiento de objetos tiene múltiples aplicaciones en diversos sectores, como son la industria, en robótica y en aplicaciones militares. Se han explorado diversas técnicas de seguimiento de objetos, algunas muy robustas y otras dedicadas a determinadas características. Se han intentado varios tipos de seguimiento, como pueden ser por extracción de esquinas, por extracción de bordes, por modelado 3D de objetos, por segmentación, por umbralización, por fusión de sensores y por correlación. Todos los tipos tienen sus ventajas y desventajas y también son dependientes de la aplicación del seguimiento en sí.

Una de las representaciones de imágenes utilizada para el procesamiento de video es la pirámide multiresolución, que separa la imagen original de una resolución dada en un conjunto de imágenes de diferentes resoluciones espaciales [3].

2 Principios

La potencia de la pirámide consiste en que las operaciones basadas en ella son mucho más rápidas que las mismas operaciones sobre la imagen original [3]. Esta simplificación se debe a que se trabaja con resoluciones más burdas, en donde hay menos píxeles que deben ser procesados. Cada nivel es ¼ más pequeño que el nivel precedente, resultando en una reducción en el procesamiento de un factor de 16, 64, 256 y así sucesivamente.

Muchos algoritmos que trabajan a nivel piramidal son llamados algoritmos burdos-a-finos (coarse-to-fine algorithms). Estos algoritmos procesan la imagen en una resolución muy burda (muy baja) y obtienen un primer resultado del procesamiento, el cual es impreciso, pero da una noción del resultado que se desea obtener. Los resultados se refinan repitiendo el procesamiento a resoluciones mayores, utilizando como referencia los resultados obtenidos en resoluciones más bajas.

2.1 Pirámide Gaussiana

Para generar una pirámide gaussiana, se parte de la suposición de que la imagen está representada por el arreglo G_0 de F Filas por C columnas. Cada valor representa la intensidad de luz del punto correspondiente de la imagen. Esta imagen es el nivel 0 de la pirámide gaussiana. El nivel 1 de la pirámide contiene una imagen G_1, la cual es reducida a partir de una versión de la imagen G_0 a la que se le aplicó un filtro pasa-bajas [4]. Cada valor dentro del nivel 1 es calculado como un promedio ponderado de los valores en el nivel 0 dentro de una ventana de 5x5. Cada valor del nivel 2, representado por G_2, es obtenido a partir de los valores del nivel 1 aplicando la misma ventana.

El proceso de promediación ponderada de nivel a nivel está implementado por la función *Reduce:*

$$G_k = \text{Reduce}\,(G_k\text{-}1) \tag{1}$$

Lo cual significa que para los niveles $0 \le l \le N$ y nodos i, j, $0 \le i \le F$, $0 \le j \le C$

$$G_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) G_{l-1}(2i+m, 2j+n). \tag{2}$$

donde N se refiere al número de niveles de la pirámide mientras que F_i y C_i son dimensiones del i-ésimo nivel. La densidad de los nodos es reducida por la mitad una dimensión o por cuatro en dos dimensiones de nivel a nivel El patrón de pesos debe cumplir con las siguientes propiedades:

a) Por lo general, se utilizan máscaras de tamaño 5 x 5, aunque no hay restricción con respecto al tamaño de la máscara.

b) Debe ser separable

$$w(m,n) = \hat{w}(m)\hat{w}(n)$$

c) Los componentes del patrón deben estar normalizados

$$\sum_{m=-2}^{2} \hat{w}(m) = 1$$

d) Además, estos componentes deben ser simétricos

$$\hat{w}(i) = \hat{w}(-1) \quad para\ i = 0,1,2$$

En la Fig. 1 se muestra el resultado de obtener la pirámide gausiana de una imagen. En la tabla 1 se muestra el número de filas y columnas que tendrá cada una de imágenes de la pirámide, partiendo de una imagen con 480 filas por 640 columnas.

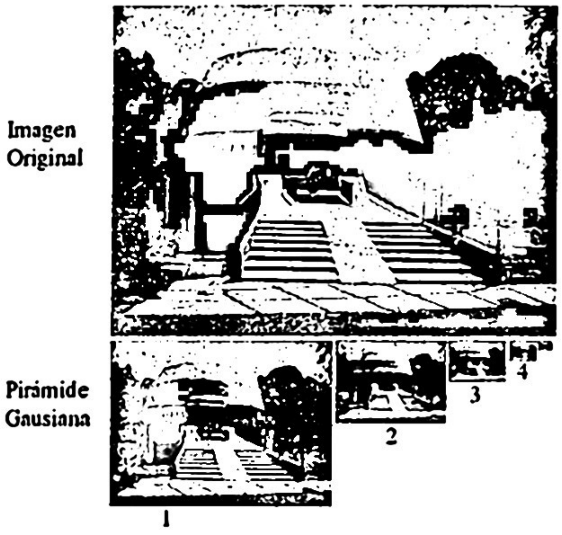

Fig. 1. Pirámide Gaussiana de la figura "INAOE"

Tabla 1. Número de filas y columnas de la pirámide de la imagen "INAOE"

Nivel	Filas	Columnas
0	480	640
1	240	320
2	120	160
3	60	80

2.2 Medidas de correlación

Se han explorado diversas medidas de similitud basadas en correlación. Las diferentes medidas exploradas durante el desarrollo de este trabajo se mencionan a continuación:

1) Suma de Diferencias Absolutas (SDA) [5]

$$SDA=\sum_{i,j}\left|I_1(x+i,y+j)-I_2(x+i+d,y+j)\right| \tag{6}$$

2) Suma de Diferencias al Cuadrado (SDC) [5]

$$SDC=\sum_{ij}\left(I_1(x+i,y+j)-I_2(x+i+d,y+j)\right)^2 \tag{7}$$

3) Correlación Cruzada Normalizada (CCN) [6]

$$CCN=\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\frac{fr(i,j)fp(i,j)}{\sqrt{fr(i,j)^2}\sqrt{fp(i,j)^2}} \tag{8}$$

3 Sistema propuesto

Se propone una arquitectura hardware basada en un FPGA de seguimiento de objetos. Esta arquitectura se divide en dos grandes bloques, la arquitectura que obtiene la pirámide gaussiana de una imagen y la arquitectura para la correlación de objetos basándose en los resultados arrojados por la primera.

3.1 Arquitectura para la obtención de pirámides

Se diseñaron dos variantes de la arquitectura. Estas variantes dependen del número de píxeles almacenados por dirección de memoria. Se supone el uso de imágenes en niveles de gris, donde un píxel ocupa 8 bits (1 byte). Entonces, si tenemos una memoria de 16 bits, se pueden codificar 2 píxeles por localidad. En el caso de memorias de 32 bits, se pueden codificar 4 píxeles por localidad.

Los módulos básicos que integran la arquitectura son:

1. Memoria RAM. En esta memoria se almacena la imagen a procesar, así como las imágenes que son arrojadas como resultado del procesamiento. La arquitectura fue diseñada para ser flexible con cualquiera de estas dos restricciones:

   a) Se dispone sólo de una memoria principal, en donde además de almacenarse la imagen a procesar, también se almacenan los resultados del procesamiento.

   b) Se dispone de más de una memoria. En una de éstas se almacena la imagen a procesar y en otras se almacenarán los resultados del procesamiento.

Para esta arquitectura se tomó en cuenta la primera restricción. Para pasar a la segunda restricción, debe hacerse un cambio mínimo en el código sin necesidad alterar el funcionamiento de los otros módulos que integran la arquitectura.

2. Generador de Direcciones. Este módulo se encarga de proporcionar a la arquitectura la dirección RAM en donde se leerá el dato o la dirección RAM en donde se almacenará el resultado. El Generador de Direcciones recibe datos de los Generadores de Control, los cuales se encargan de determinar cuándo se tiene listo en banco de registros el contenido de una localidad de memoria de resultados. En caso de que no se necesite hacer escritura a memoria, el Generador de Direcciones genera la dirección de lectura del siguiente dato a procesar.

3. Procesadores de Convolución. Estos procesadores se agrupan en niveles de procesadores. Cada nivel puede tener 2 ó 3 procesadores, dependiendo del tamaño de palabra de las memorias (2 procesadores para memorias de 32 bits y 3 procesadores para memorias de 16 bits). La principal característica de los Procesadores de Convolución consiste en que entre más niveles de procesadores se declaren, arquitectura tendrá un mejor desempeño. El número mínimo de niveles que deben declararse es uno, mientras que el máximo dependerá de los recursos disponibles en el FPGA en donde se va a sintetizar el diseño.

4. Banco de Registros. En este Banco de Registros se almacenan los resultados parciales de cada uno de los Módulos de Convolución. El Generador de Direcciones se encarga de leer el contenido de los registros del Banco cuando alguno de módulos Generadores de Control le indica que es tiempo de escribir en la memoria de datos el resultado de las operaciones realizadas por los Módulos de Convolución. En otro caso, los resultados de los Módulos de Convolución seguirán siendo almacenados en el Banco de Registros.

En la Fig. 2 se muestran los componentes de la arquitectura piramidal, así como señales de control que entran a los bloques. Estas señales de control son generadas el generador de direcciones, el cual recibe señales de control de los generadores control y decide en qué momento hacer una operación de escritura y en qué momento hacer una operación de lectura a la memoria de datos.

3.2 Arquitectura de correlación

Los componentes principales de la arquitectura de correlación son:

1. Memoria de Imagen. Como se mencionó en la arquitectura piramidal, la imagen original se almacena en una memoria y el resultado de la pirámide puede estar almacenado en la misma memoria o en una memoria diferente. La Memoria Imagen es la región de memoria en donde se encuentra el nivel de la pirámide gausiana que será utilizado para hacer la correlación con el patrón a buscar.

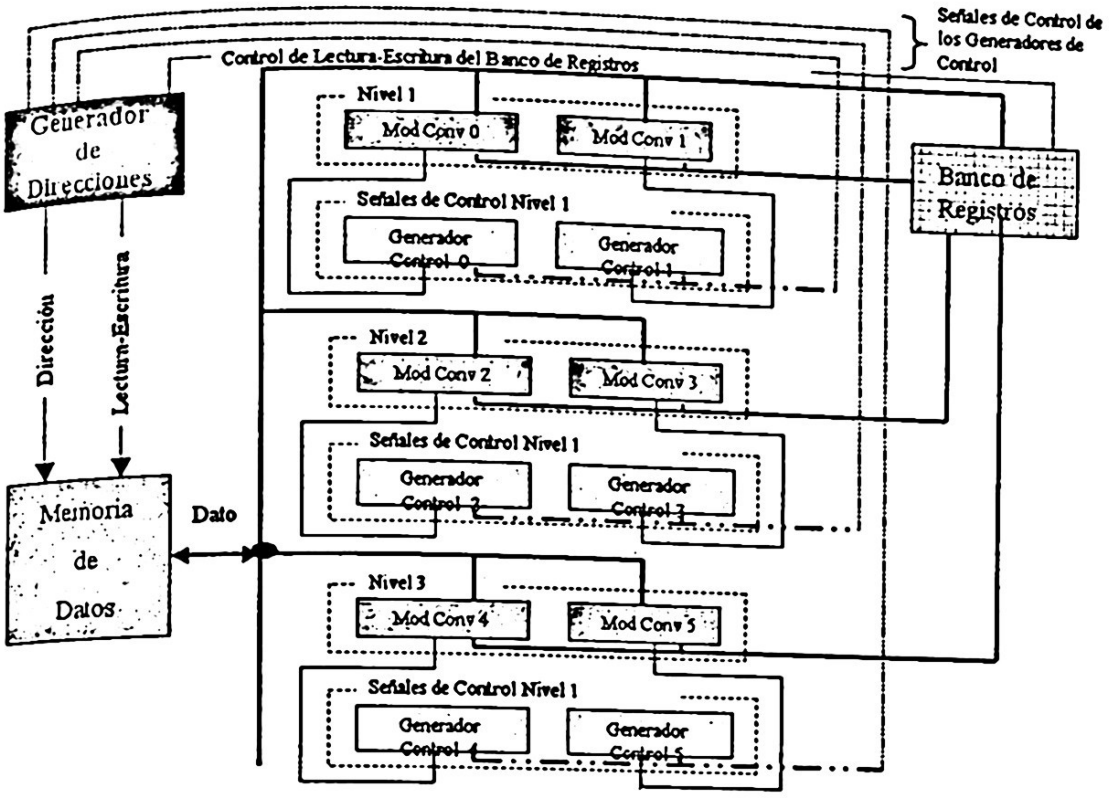

Fig. 2. Bloques Principales de la Arquitectura Piramidal

2. Memoria de Patrón. En el proceso de seguimiento de objetos, se debe buscar la similitud entre un patrón y una imagen. Entonces, la memoria patrón contendrá este patrón que se desea correlacionar. Si se cuenta con una sola memoria para realizar todo el procesamiento, esta memoria será una región diferente de la memoria principal en donde está almacenado el patrón.
3. Control de Correlación. Este se encarga de enviar las señales de control a las memorias de Datos y de Patrón, a los registros de Imagen y de Patrón y a los registros de Comparación Local y Global.
4. Registros de Imagen. Hay dos registros que se encargan de almacenar temporalmente dos palabras de memoria con la finalidad de obtener la correlación de 4 ventanas al mismo tiempo, ya que se desea realizar el menor número de lecturas repetidas como sea posible.
5. Registro de Patrón. Hay un registro que se encarga de leer de la imagen de patrón una palabra que se correlacionará con las 4 posibles ventanas de correlación.
6. Bloques de correlación (fcorr). Estos bloques se encargan de hacer el proceso de correlación de los valores de los registros de imagen y de patrón. Con esto, además de hacer un sistema altamente modular, podemos programar cualquier función de correlación deseada.
7. Comparador Local (ComparadorL). Una vez que se han terminado de procesar las cuatro ventanas de correlación, el control de correlación envía una señal a este comparador para obtener el mejor coeficiente de correlación de los cuatro que se

están procesando actualmente. Con esto, se tiene una idea de cuál de las ventanas es la que tiene mayor similitud con el patrón buscado.

8. Comparador Global (ComparadorG). Se encarga de comparar el contenido del Registro de Mejor Correlación con la salida del Comparador Local. En caso que el Mejor Local sea "mejor" (dependiendo de la medida de correlación) se actualizará el Registro de Mejor Correlación.
9. Registro de Mejor Correlación (MejorGlobal). En este registro se va almacenando el valor de mejor correlación de toda la imagen. Este registro servirá para saber cuánto cambió el grado de correlación con respecto al cuadro anterior.
10. Registros de Coordenadas. En estos registros se almacenan los valores donde va encontrando la mejor correlación. Al final del barrido de la imagen, estos registros contienen la fila y la columna de la mejor correlación, la cual sirve para desplegar el recuadro donde se encuentra tentativamente el objeto o para que en la siguiente iteración del proceso, se tomen esos resultados para acotar la búsqueda.

En la Fig. 3 se muestra la conexión entre todos los bloques que integran arquitectura de correlación, así como las principales señales de control y de datos.

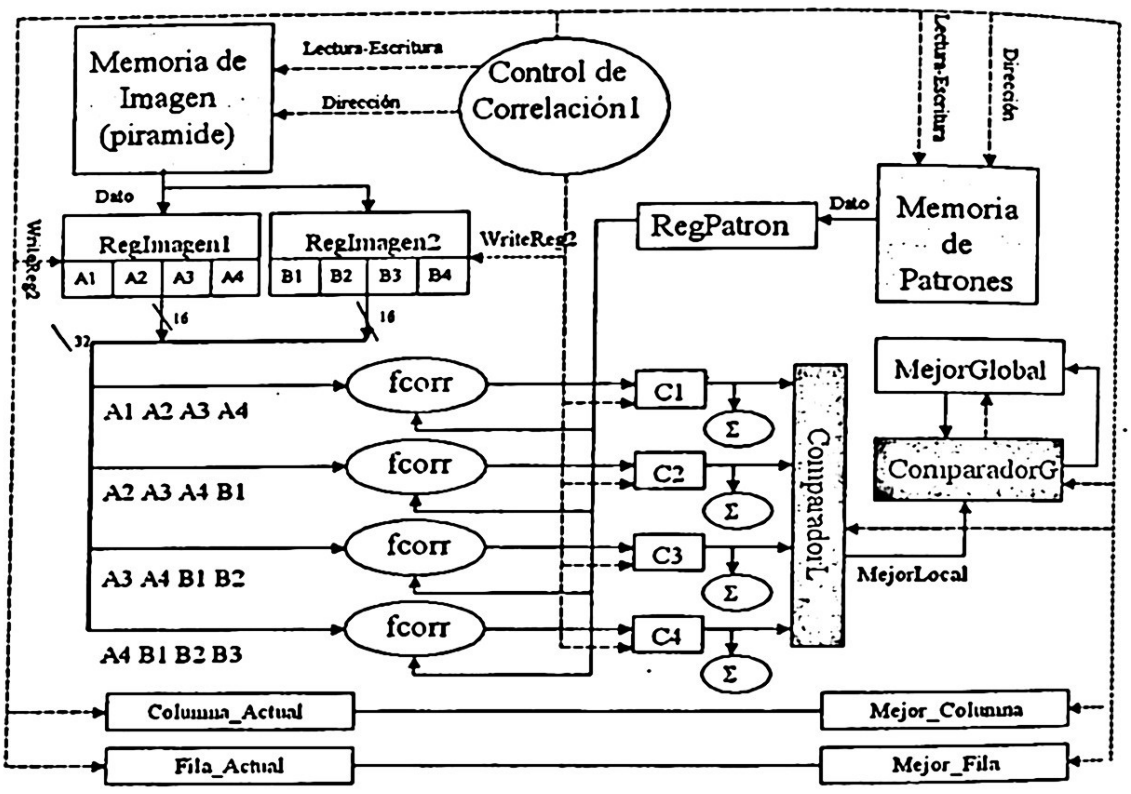

Fig. 3. Arquitectura de Correlación

3.3 Integración de las arquitecturas anteriores

En esta sección se describe brevemente la integración de las arquitecturas de pirámides y de correlación para dar paso a la arquitectura de seguimiento.

La arquitectura piramidal se encarga de obtener la pirámide de la imagen actual, y posteriormente pasa las direcciones donde se localizan las imágenes a la arquitectura

de correlación. La arquitectura de correlación se encarga de decidir si es el primer cuadro a procesar. En caso de ser el primer cuadro, toma las coordenadas de inicio de patrón (fila y columna) y hace una copia de la región correspondiente al patrón en una región de memoria diferente de donde se encuentra almacenada la pirámide. Con este proceso se obtiene la pirámide del patrón, que es una copia de los respectivos píxeles de las imágenes que componen la pirámide correspondientes al patrón a seguir. En caso de no ser el primer cuadro, entonces se hace la búsqueda del patrón almacenado en el cuadro anterior, iniciando por el nivel más alto de la pirámide. La búsqueda se divide en dos partes: la búsqueda hecha en el nivel más burdo (o de menor resolución), en donde la correlación del patrón buscado se lleva a cabo con toda la imagen. En las búsquelas hechas en niveles menos burdos, la búsqueda se realiza acotando el espacio de búsqueda de acuerdo con los resultados obtenidos en el nivel inferior. En la Fig. 4 se muestra de manera gráfica este proceso. Una vez que se ha llegado al nivel cero de la pirámide, se actualiza el patrón en todos los niveles, se envía una señal de término de proceso y se procesa la siguiente imagen.

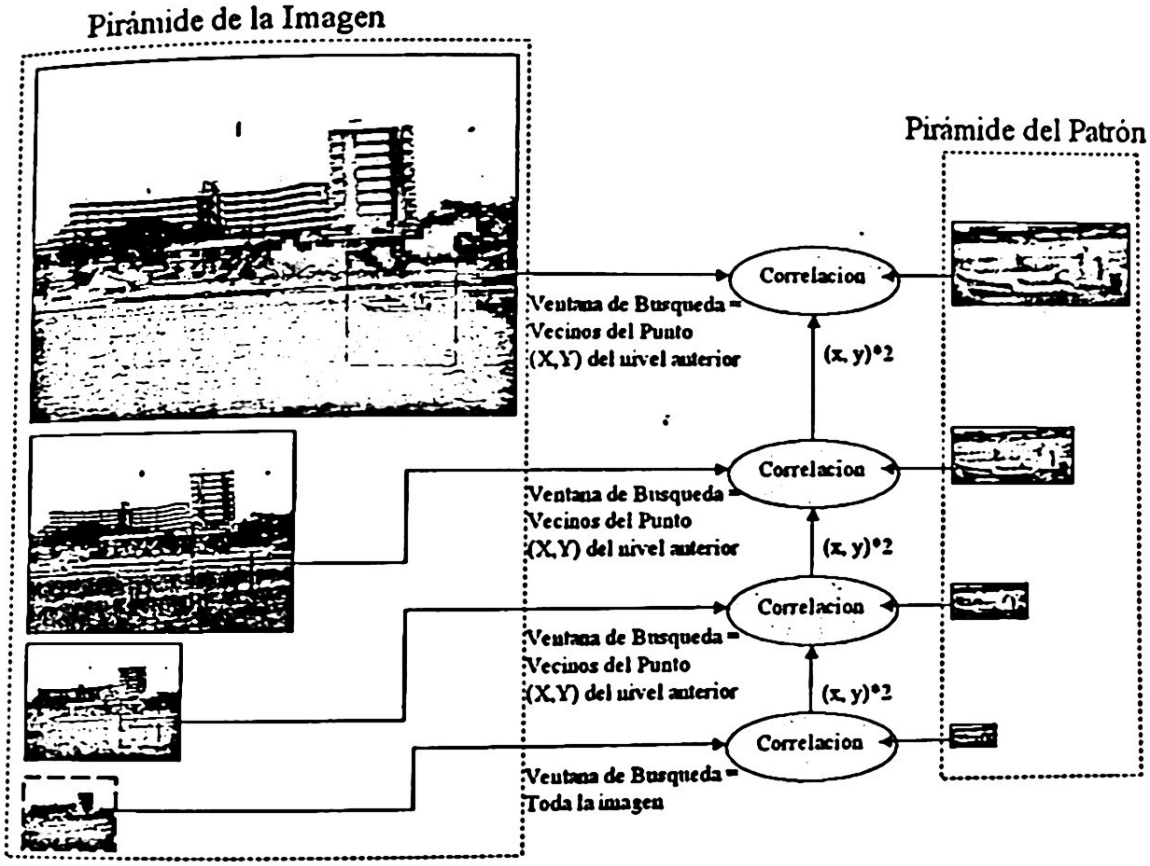

Fig. 4. Integración de las Arquitecturas Piramidal y de Correlación

4 Implementación y resultados

Los resultados presentados para la arquitectura de seguimiento fueron obtenidos por medio de simulaciones hardware. Handel-C es una herramienta innovadora que propone la reducción del tiempo de desarrollo de arquitecturas hardware, utilizando un lenguaje basado en el estándar ANSI-C. Las arquitecturas piramidal y de correlación

fueron codificadas en este lenguaje. Para la obtención de los resultados presentados, las imágenes utilizadas se convirtieron en archivos de texto, que se abrieron en Handel-C para pasarse a la arquitectura y posteriormente obtener archivos de texto donde se almacenaron los resultados del procesamiento. Una vez que se obtuvieron los resultados, se realizó una herramienta de desplegado de las secuencias (en MATLAB) que utilizará los datos arrojados por la arquitectura para desplegar la posición tentativa del objeto.

En las Figuras 5 y 6 se muestran los resultados para cuatro imágenes de dos diferentes secuencias. En la primera secuencia se observa que el seguimiento del objeto es bueno, en la segunda secuencia, se utiliza una secuencia un poco más difícil debido a que se introduce una serie de movimientos rotacionales que dificultan el seguimiento para algunos sistemas, pero que en el caso de esta arquitectura, los efectos de rotación sobre el seguimiento final son mínimos.

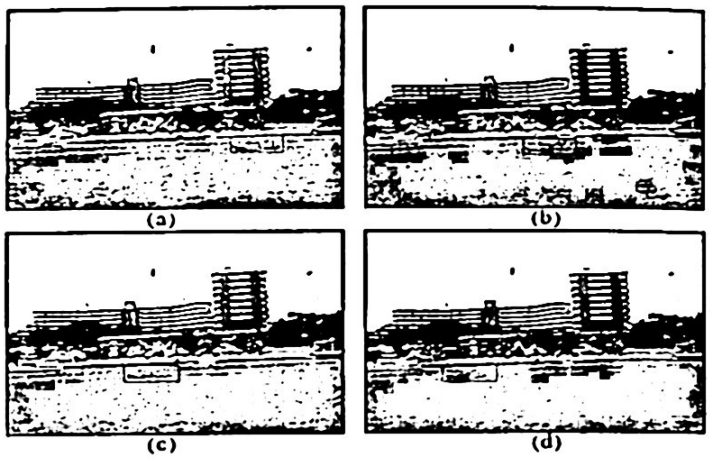

Fig. 5. Resultados del Seguimiento para la Secuencia "Marina"

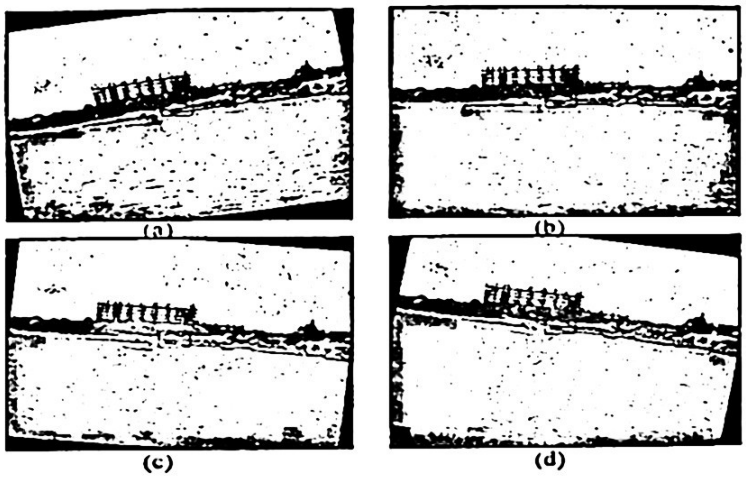

Fig. 6. Resultado del Seguimiento para la Secuencia "Casita Rotada"

En la Fig. 7 se muestra el resultado de la implementación de la arquitectura piramidal. Para su implementación se utilizó la tarjeta RC1000 de Celoxica, la cual se comunica con la PC por medio del bus PCI. La interfaz con la PC se programó en Visual C++.

Fig. 7. Arquitectura Piramidal en Funcionamiento

Se efectuaron varias pruebas a la arquitectura antes de su implementación. Una de ellas consistió en obtener el número de ciclos de reloj utilizados para procesar la imagen completa, y de esta forma obtener el tiempo de procesamiento de una imagen (suponiendo una frecuencia de reloj de 20 Mhz), que en este caso fue de 480 líneas por 512 píxeles. Los resultados obtenidos se resumen en la tabla 2.

Tabla 2. Rendimiento de la arquitectura en ciclos de reloj y en tiempo de procesamiento.

Número de Niveles de Procesadores en Paralelo	Rendimiento en Ciclos	Tiempo de Procesamiento (ms)	Imágenes por Segundo
1	262, 080	13 ms	76
2	181, 440	9.1 ms	109
3	154, 560	7.7 ms	129
4	141, 010	7.1 ms	141

Posteriormente se hizo la implementación, la cual fue realizada en la tarjeta RC1000 de Celoxica, que tiene como elemento central de procesamiento un FPGA Virtex (fabricado por Xilinx) con capacidad de 2 millones de compuertas equivalentes. Los recursos utilizados por la arquitectura para el procesamiento de pirámides se muestran a continuación:

Tabla 3. Recursos utilizados por la arquitectura del FPGA Virtex

Recurso del FPGA	Total Utilizado	Porcentaje Utilizado
Slices	6, 583 de 19, 200	34 %
4 Input LUT	8, 795 de 38, 400	22 %

Conclusiones

En este trabajo se presentó una arquitectura para el seguimiento de objetos que tiene como características principales la flexibilidad con respecto al grado de paralelismo que se desea implementar. El grado de paralelismo implementado dependerá de los

recursos disponibles en el FPGA utilizado para su implementación, de la velocidad que se desee alcanzar y de la aplicación que utilizará el seguimiento de objetos.

También se puede concluir que Handel-C es un lenguaje que acorta el tiempo de diseño e implementación de arquitecturas hardware complejas. Esto permite hacer un mayor número de pruebas y verificaciones antes de llegar a un diseño final que puede ser utilizado en aplicaciones donde se requiera menor espacio o menor consumo de potencia. Además, se pueden diseñar aplicaciones que tengan un alto rendimiento con la finalidad de alcanzar el desempeño en tiempo real.

6 Trabajo Futuro

Se ha validado la arquitectura de seguimiento a nivel simulación. Como trabajo futuro se hará la implementación definitiva en la Tarjeta RC1000 de Celoxica, además de hacer experimentos con otras aplicaciones de la pirámide gaussiana, como son la generación de mosaicos de imágenes, la estabilización de imágenes, entre otras. También queda como trabajo futuro optimizar el código para obtener un rendimiento superior, con la finalidad de implementar otros algoritmos que utilicen el seguimiento de objetos.

Referencias

1. Pajares, G., De La Cruz, J.: Visión por Computador. Editorial. Ra-ma., México, (1995)
2. Brown, S., Rose, Jonathan.: Architecture of FPGAs and CPLDs: A Tutorial., IEEE Design and Test of Computer, Vol. 13, No. 2, pp. 42-57, (1996)
3. Sarnoff Corporation.: Pyramid-Based Video Processing., http://www.sarnoff.com/government_professional/vision_technology/core_technology/pyramid_video_processing.asp, (2002)
4. Burt, P.J., Adelson E .H.: The Laplacian pyramid as a compact image code. IEEE Transactions on Communications. COM-31, pp 532-540, (1983)
5. Xicoténcatl P, Juan Manuel.: Arquitectura Hardware de Visión Estéreo en Tiempo Real. INAOE, México. (2000)
6. Peregrina A, Cesar Darío.: Seguimiento de Objetos por medio de Visión Activa. INAOE, Mexico. (2002)