

4. IMPLANTACION DEL ALGORITMO EN HARDWARE

Una implantación hardware no es tan flexible como lo es una realización software. Su costo es más elevado, y la ejecución de versiones actualizadas necesitan un esfuerzo superior. Sin embargo, hay razones que impulsan a implantar ciertos cálculos bajo configuraciones hardware, siendo las más importantes la economía de tiempo y la paralelización de ciertas partes del proceso.

No siempre es posible reemplazar la realización software. Unas veces es debido a la complejidad excesiva de la misma, con lo que la implantación hardware resultaría de una complejidad aún mayor.

Normalmente habrá que hacer simplificaciones ó variaciones en el algoritmo inicial para adaptarlo a las nuevas herramientas de cálculo que

proporcionan las modernas arquitecturas hardware. Téngase en cuenta que se trabaja a un nivel jerárquico más bajo del que sería si se trabajase en lenguaje ensamblador. Por ejemplo, una de estas simplificaciones consiste en adaptar el algoritmo para manejar valores con números enteros en lugar de números reales.

Hay que mencionar que están apareciendo en el mercado circuitos integrados que trabajan con números reales, aunque son todavía muy lentos para la presente aplicación. Además, si el algoritmo no sufre cambios significativos por su adaptación a números enteros, siempre el cálculo con estos últimos producirá menor carga computacional y por lo tanto, un costo menor.

El presente capítulo trata sobre las modificaciones realizadas al algoritmo inicial para hacer posible su implantación en una configuración hardware. Se entiende que el objetivo principal de esta adaptación consiste en deteriorar lo menos posible el algoritmo tratado en el capítulo anterior.

Si se pretende implantar el algoritmo en hardware, para elegir que puntos son realmente bordes, se debe tener la información tanto de la primera como de la segunda derivada para detectar con la primera y localizar con la segunda; ambas calculadas mediante mecanismos hardware.

La primera derivada sólo indica la presencia de un borde pero no precisa su localización, dato que sí se obtiene con la segunda derivada. Por lo tanto, si el objetivo es localizar exactamente la posición de los puntos-borde, se deberá llegar por algún camino al cálculo de la segunda derivada ó en su defecto a una aproximación de numérica de la misma.

Si se pretende aproximar la segunda derivada con el Laplaciano ∇^2 , una de las mayores dificultades que aparecen al realizar su cálculo, atendiendo al costo computacional de las operaciones de convolución, es que el tamaño mínimo del filtro que se debe aplicar es excesivamente grande. Si el proceso se piensa

implantar por medio de alguna configuración hardware, el problema es aún mayor (Chen, 1987).

La alternativa propuesta en esta tesis para afrontar este problema consiste en emplear la primera derivada del gaussiano como elemento generador de todos los gradientes; este método obliga a considerar los vectores direccionales con respecto a los cuales se está calculando tanto la primera como la segunda derivada.

4.1 Pipeline para una Línea de Vídeo

En el espacio unidimensional el Laplaciano $\nabla^2 F$ coincide con la segunda derivada direccional, en la dirección en la que se está trabajando, porque una de las componentes del Laplaciano será cero. Así, si la dirección es X, $\nabla^2 F = \partial^2 F / \partial x^2$; si no se indica lo contrario, la dirección por defecto será siempre la del eje X.

Como se puede observar en la Figura 4.1, la primera derivada no es isotrópica con respecto a su centro como lo es la segunda. Por lo tanto, se presentaría un primer problema al tener que realizar su cálculo en cada dirección del plano. Normalmente sólo es necesario hacerlo en dos direcciones ortogonales.

Por otro lado, el tamaño efectivo del filtro para la primera derivada es sensiblemente menor que el tamaño de la segunda derivada, lo cual produce una economía de cálculo apreciable.

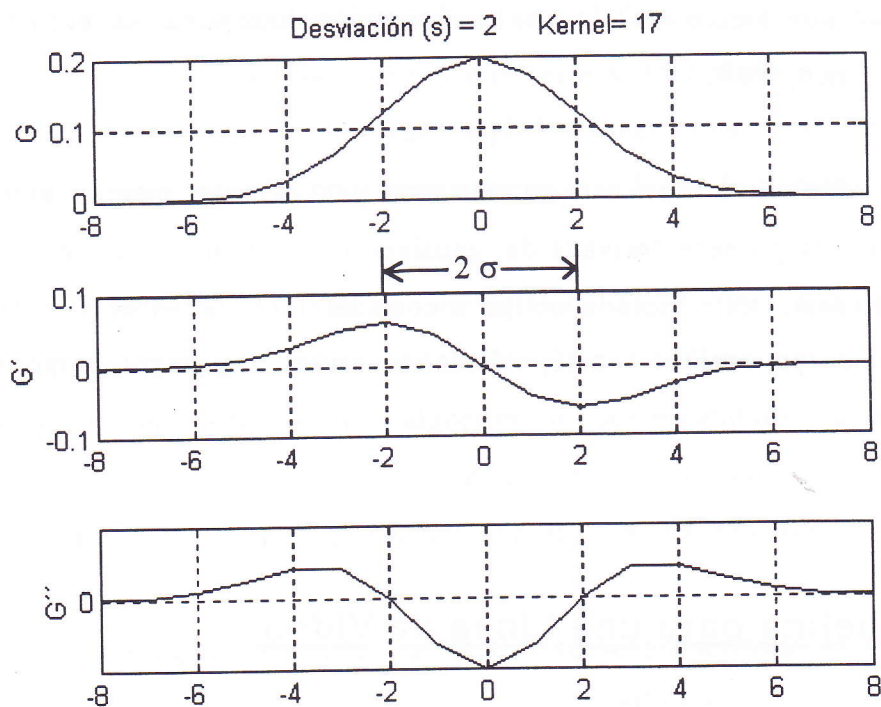


Figura 4.1 Gausiano $\sigma = 2$, y sus dos primeras derivadas en 1D.

Si se toman nuevamente las tres líneas modelo representadas en la Figura 3.1, y se les aplica el filtro $\nabla^2 G$, el resultado es el que ya hemos visto en el capítulo anterior y que se muestra en la Figura 4.2 para poder comparar con el resultado obtenido empleando dos veces la primera derivada.

En la Figura 4.3 se ha dibujado la respuesta al filtro ∇G , en la que se aprecia con claridad los sentidos en los cambios de intensidad. Véase en la Figura 4.4 el resultado obtenido al aplicar ∇G dos veces consecutivas a los tres modelos de líneas propuestos en la Figura 3.1. Puede observarse que para los tres casos estudiados, el resultado relativo obtenido en la Figura 4.4 es prácticamente el mismo que el obtenido con $\nabla^2 G$ en la Figura 4.2.

Se puede concluir que una alternativa válida para acortar el tiempo de procesamiento, al menos en 1D, es aplicar dos veces ∇G para generar $\nabla^2 F$. Si el procesamiento es realizado en hardware, es posible montar una estructura en

"pipeline" de tal manera que la segunda derivada comience a calcularse, antes de que se termine de calcular la primera completamente.

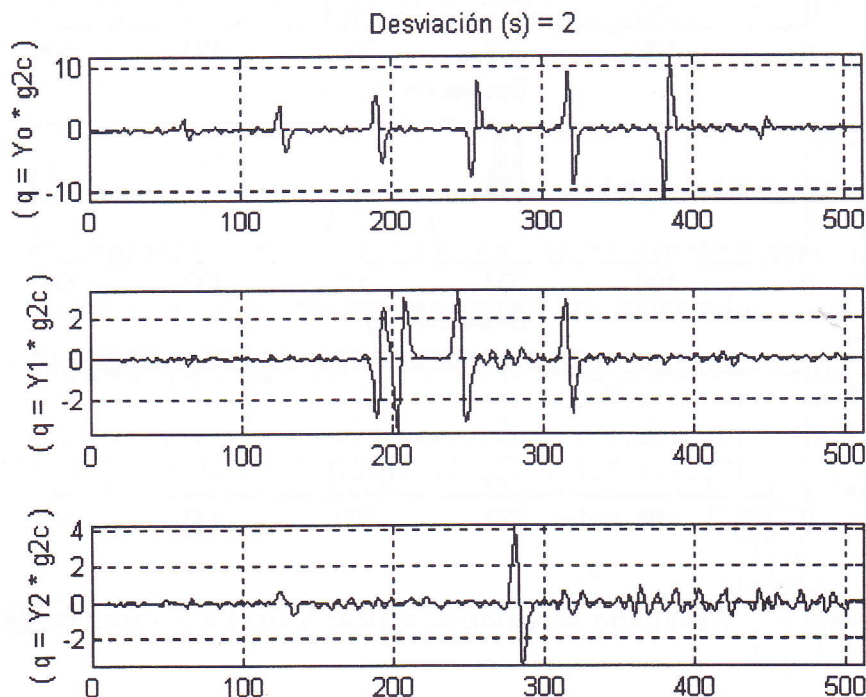


Figura 4.2 Segunda derivada de los tres modelos empleando ∇^2G .

Un ejemplo de "pipeline" típica en una dimensión es el mostrado en la Figura 4.5. Obsérvese que si el tamaño del filtro aplicado es de k elementos, la segunda derivada puede empezar a calcularse $2k-1$ elementos después de que el primer elemento de la primera derivada haya sido calculado.

Nótese además que existe un número de elementos en los extremos tanto de la primera como de la segunda derivada, denominados como P , que no contienen información significativa dado que el filtro desborda la línea original. Estos elementos, que equivalen en número a $2k-1$ elementos, pueden ser descartados para reducir el tiempo de cálculo en los procesos posteriores, ó de nivel jerárquico más bajo, al actual.

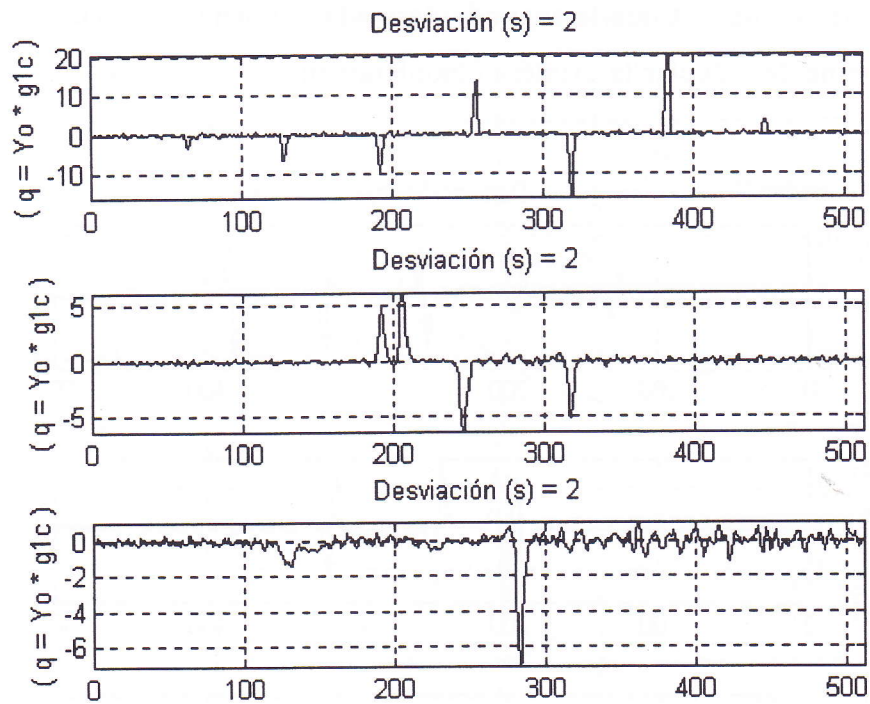


Figura 4.3 Resultado obtenido al aplicar $\nabla G(x)$ a los tres modelos.

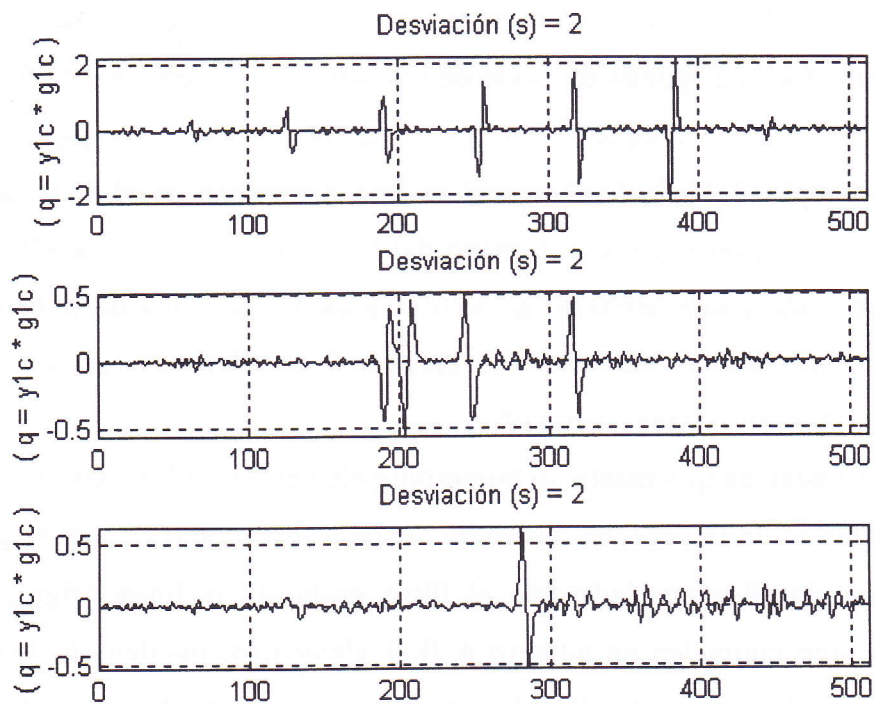


Figura 4.4 Resultado al aplicar $\nabla G(x)$ al los datos de la Figura 4.3

Este modelo se puede aplicar a cualquier operador en el espacio unidimensional. Los valores de los P pixels no significativos pueden ser reemplazados por el valor del pixel vecino significativo para rellenar sus posiciones con información.

El procedimiento esbozado en la Figura 4.5 puede ser empleado para procesar una línea de vídeo haciendo n igual al número de pixels que componen la línea. Asimismo, el procedimiento puede ser extendido a dos dimensiones para procesar la imagen de vídeo completa.

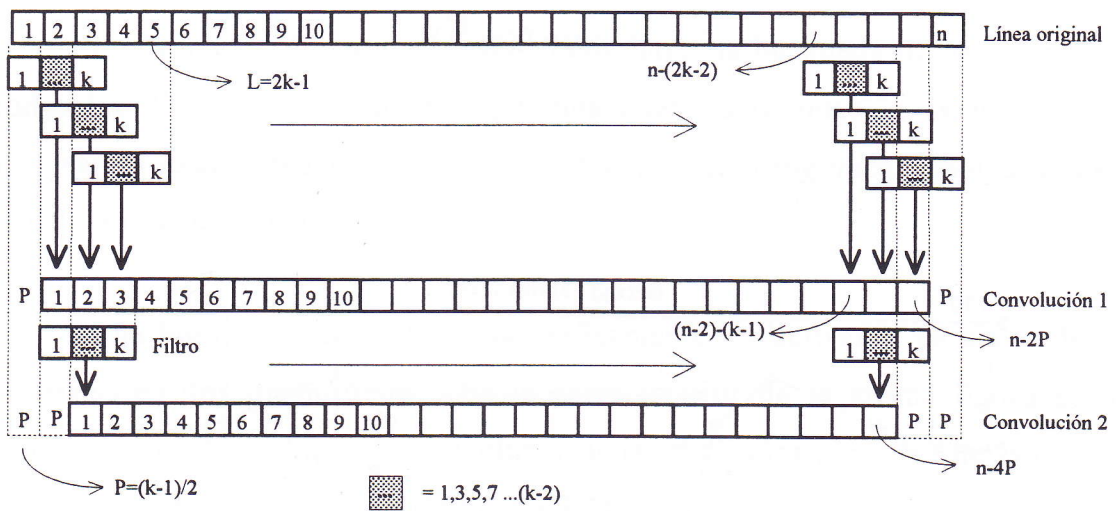


Figura 4.5 Estructura de procesamiento "pipeline" en una dimensión.

A continuación se estudian algunas formas de aplicar el algoritmo cuándo se trabaja en el espacio bidimensional y se analizan sus resultados.

4.2 Tamaño de G1 en Dos Dimensiones

El tamaño mínimo del operador primera derivada del gaussiano estará determinado por el ancho "w" de la porción central negativa del operador; la

relación se establece de la siguiente manera:

$$\nabla^2 G(x,y) = \left[\frac{x^2 + y^2}{\sigma^2} - 2 \right] \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}} = 0,$$

$$\frac{x^2 + y^2}{\sigma^2} - 2 = 0$$

$$x^2 + y^2 = 2\sigma^2,$$

$$x = 0 \Rightarrow y = \pm\sigma\sqrt{2} \quad ; \quad y = 0 \Rightarrow x = \pm\sigma\sqrt{2},$$

luego el ancho efectivo de la primera derivada será:

$$w = 2(\sigma\sqrt{2}).$$

(4.1)

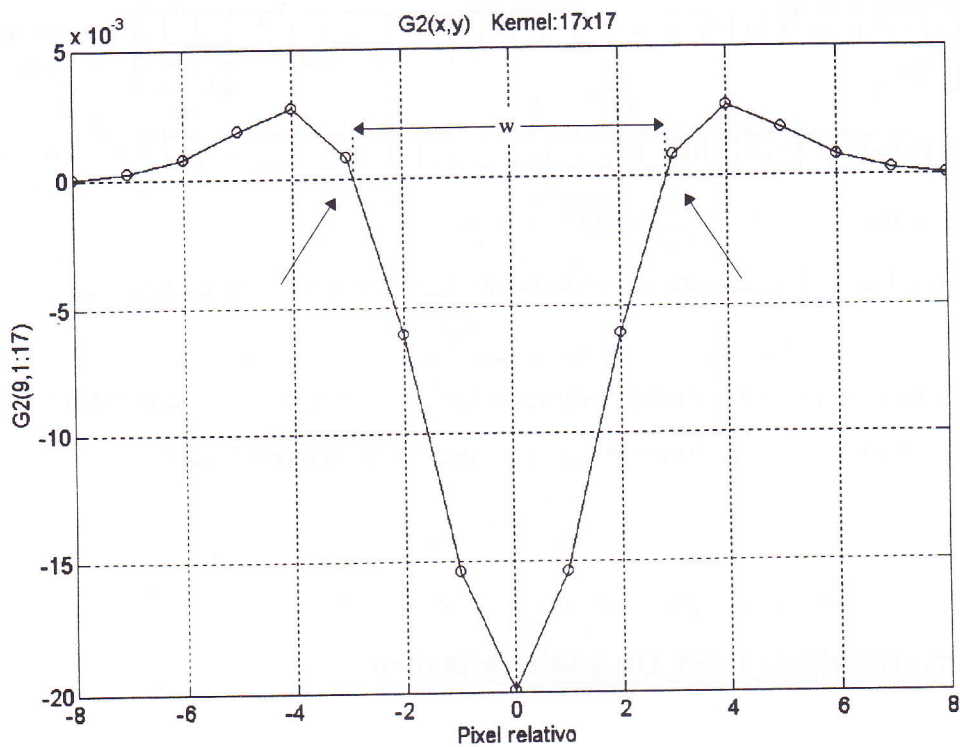


Figura 4.6 Corte de la segunda derivada en dos dimensiones, $\sigma=2$.

Nótese la diferencia en las gráficas de la segunda derivada dibujadas en la Figura 4.1 y en la Figura 4.6. En ambos casos se ha empleado un valor de desviación igual a dos. Sin embargo, mientras que en la Figura 4.1 la curva corta al eje horizontal en ± 2 , en la Figura 4.6 lo hace en $\pm w/2$, es decir en ± 2.8284 . Esto es debido a que las ecuaciones varían tanto para la primera como para la segunda derivada; sin embargo, el razonamiento es el mismo.

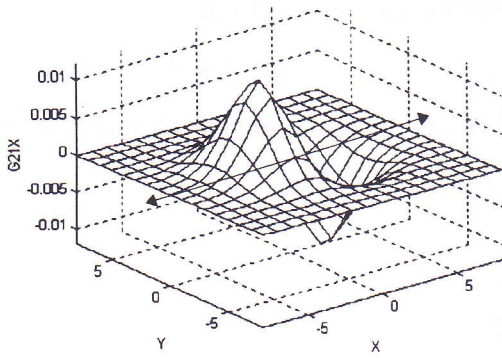
$$\sigma = \frac{\omega}{2\sqrt{2}} \quad (4.2)$$

En el espacio bidimensional se debería hacer la búsqueda de los bordes en todas las direcciones. Sin embargo, una buena aproximación es la obtenida empleando solamente las dos derivadas direccionales ortogonales empleadas para el cálculo del Laplaciano.

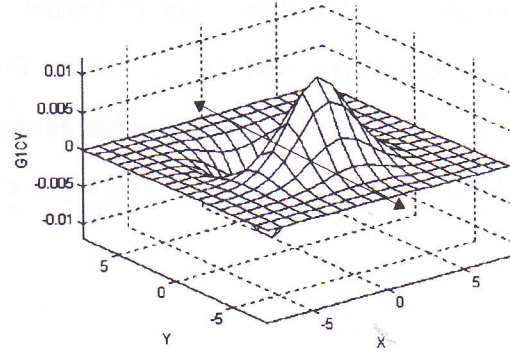
En la Figura 4.7 se representa gráficamente las derivadas parciales de un gaussiano en dos dimensiones. En la parte inferior de la misma figura se ha representado el resultado de convolucionar las derivadas parciales del gaussiano con la imagen representada en la Figura 3.14.

Una buena aproximación de la primera derivada en todas las direcciones se puede obtener calculando el máximo entre los valores absolutos de cada dirección ortogonal, (c) y (d). El resultado de aplicar lo anterior a la imagen de la bola (Figura 3.54) es mostrado en la Figura 4.8.

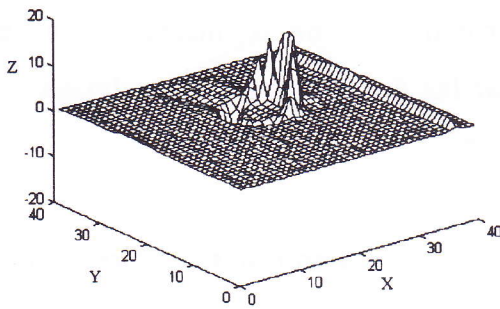
Se ha dibujado las líneas que normalmente permanecerían ocultas para destacar la forma interna de la derivada; de no hacer esto se vería solamente una especie de cono truncado. Obsérvese el buen resultado obtenido y mostrado en la Figura 4.8.



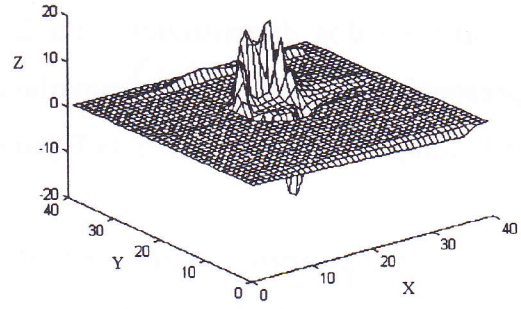
(a) $\nabla_x G(x,y) = \partial G(x,y) / \partial x$



(b) $\nabla_y G(x,y) = \partial G(x,y) / \partial y$



(c) $I * \nabla_x G(x,y)$



(d) $I * \nabla_y G(x,y)$

Figura 4.7 En (c) y (d), $F1_i$ obtenidas con los $G1_i$ dibujados en (a) y (b).

Maximos de F1X y F1Y - Desviación: 2

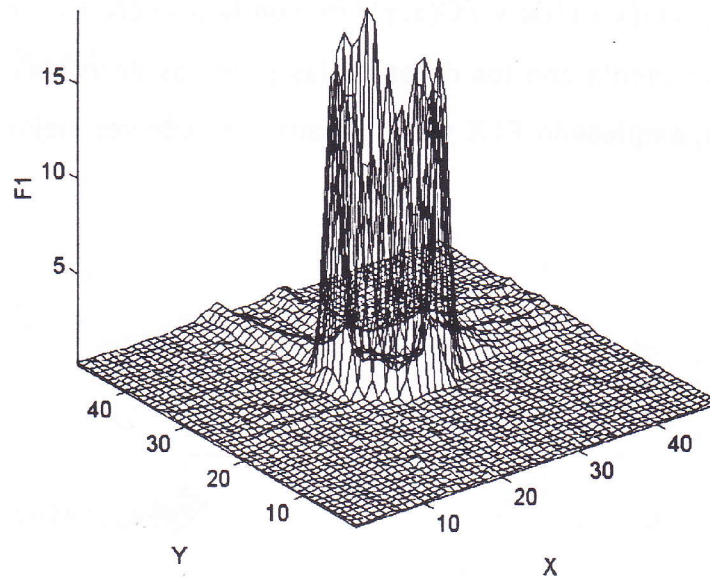


Figura 4.8 Primera Derivada de bola_m.

Se puede hacer algunas combinaciones con los operadores para calcular la segunda derivada. No todas ofrecen buen resultado.

4.3 Configuraciones Hardware

A continuación se muestran algunas *Configuraciones* y el resultado que se obtiene de cada una de ellas aplicándolas a la imagen de la bola presentada en la Figura 3.14.

4.3.1 Configuración A

Una primera configuración, llamada configuración tipo A, consiste en calcular la segunda derivada F2, con los máximos en valor absoluto de cada pixel de F2X y F2Y y conservando el signo del mayor. F2X y F2Y son obtenidas convolucionando $\partial G(x,y)/\partial x$ y $\partial G(x,y)/\partial y$ con la primera F1. Esta a su vez es obtenida componiéndola con los datos de las primeras derivadas parciales de la imagen, es decir, empleando F1X y F1Y. Esto se puede ver mejor en el siguiente diagrama:

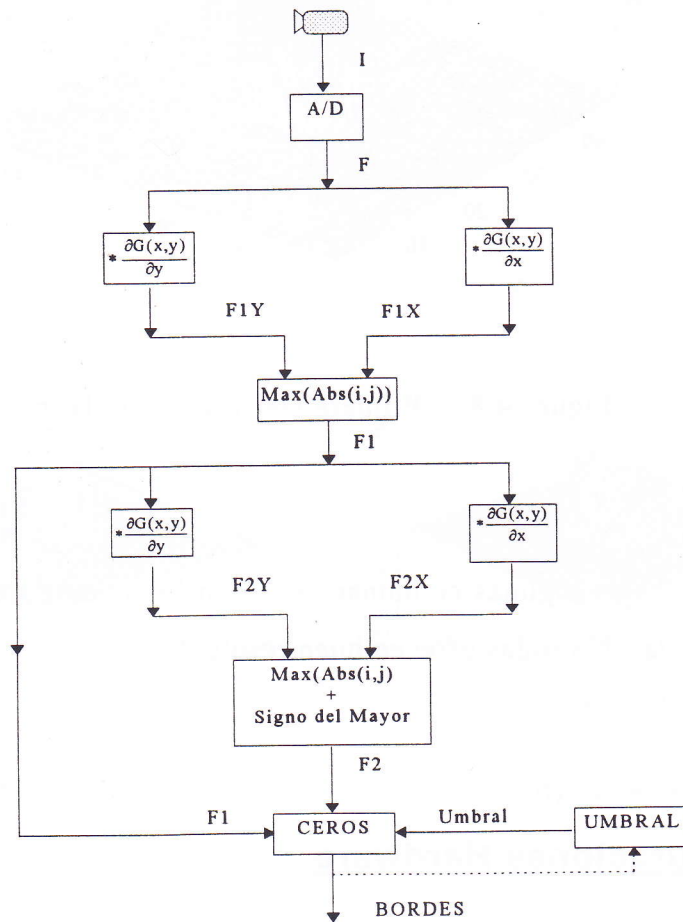


Figura 4.9 Configuración A

En la Figura 4.10 se presenta el resultado obtenido con la configuración A para desviaciones iguales a 0.8, 1.4, 2 y 3. El umbral empleado es del 2%. Se

ha escogido este valor para destacar el ruido que pueda aparecer al aplicar las diferentes arquitecturas. Los resultados mostrados a continuación son simulaciones programadas en el entorno Matlab. Los programas hacen exactamente lo que harían los bloques de procesamiento, por lo tanto la única diferencia es el carácter "offline" de los resultados.

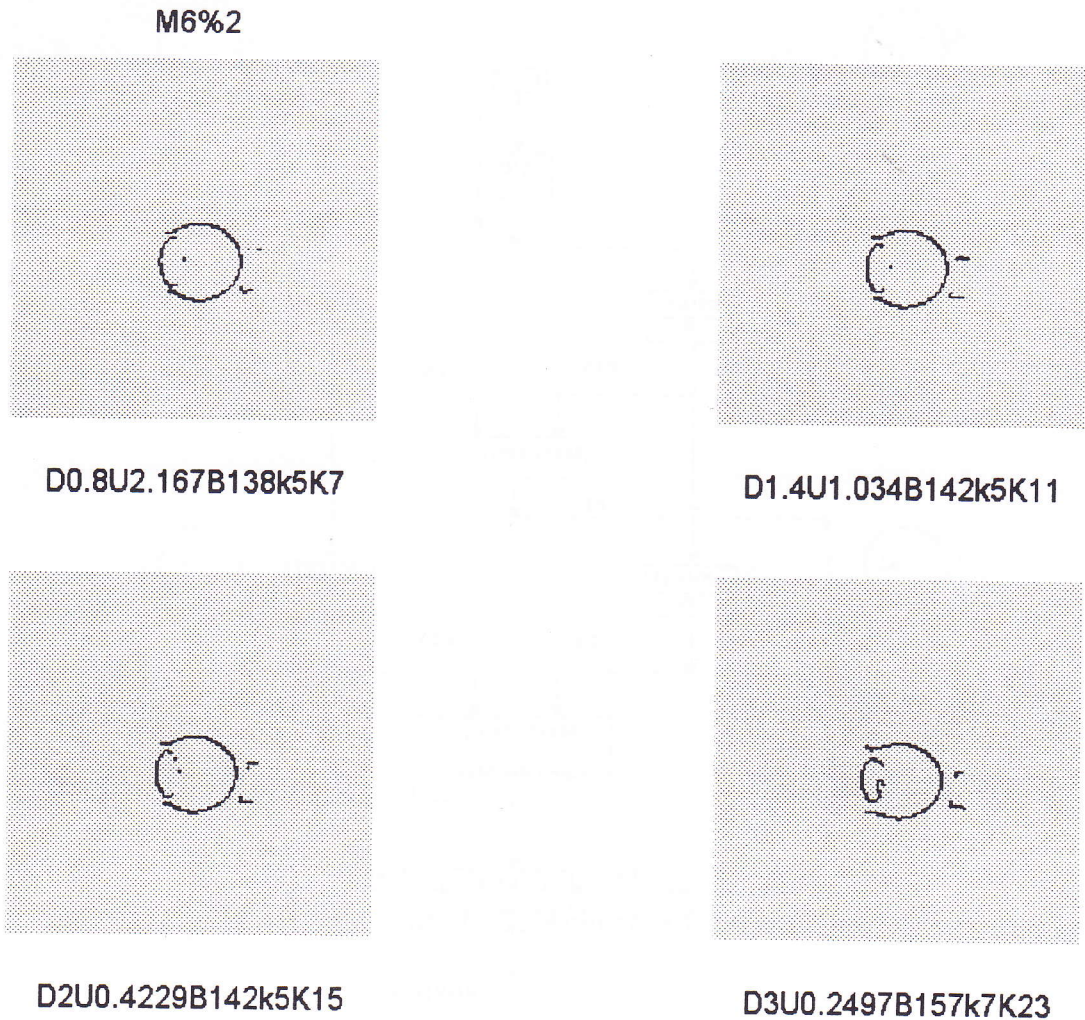


Figura 4.10 Resultado de aplicar la Configuración A.

Lo más destacable de la figura Figura 4.10 es la discontinuidad que se produce en torno a los límites del objeto. Asimismo se observa que ésta aumenta al aumentar la desviación. Con este resultado se eliminó la configuración tipo A.

4.3.2 Configuración B

Si en lugar de calcular las segundas derivadas parciales a partir de F1 se calculan a partir de F1X y F1Y, el resultado mejora significativamente. Por ello se presenta la configuración tipo B, cuyo diagrama de flujo es el siguiente,

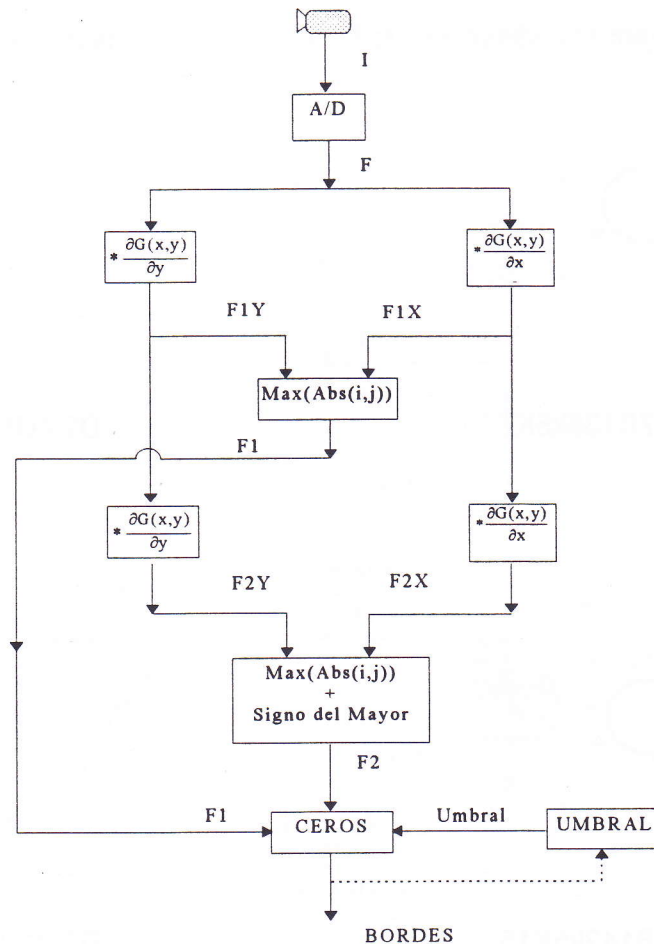


Figura 4.11 Configuración B.

Puede verse en el diagrama de la Figura 4.11 que no hace falta terminar de calcular F1 para empezar a calcular F2, dado que F2X se calcula directamente de F1X, y F2Y de F1Y. Sin embargo, es necesario reconstruir F1 para la detección de ceros que se realiza más adelante. La modificación hecha permite la

ejecución más rápida del proceso.

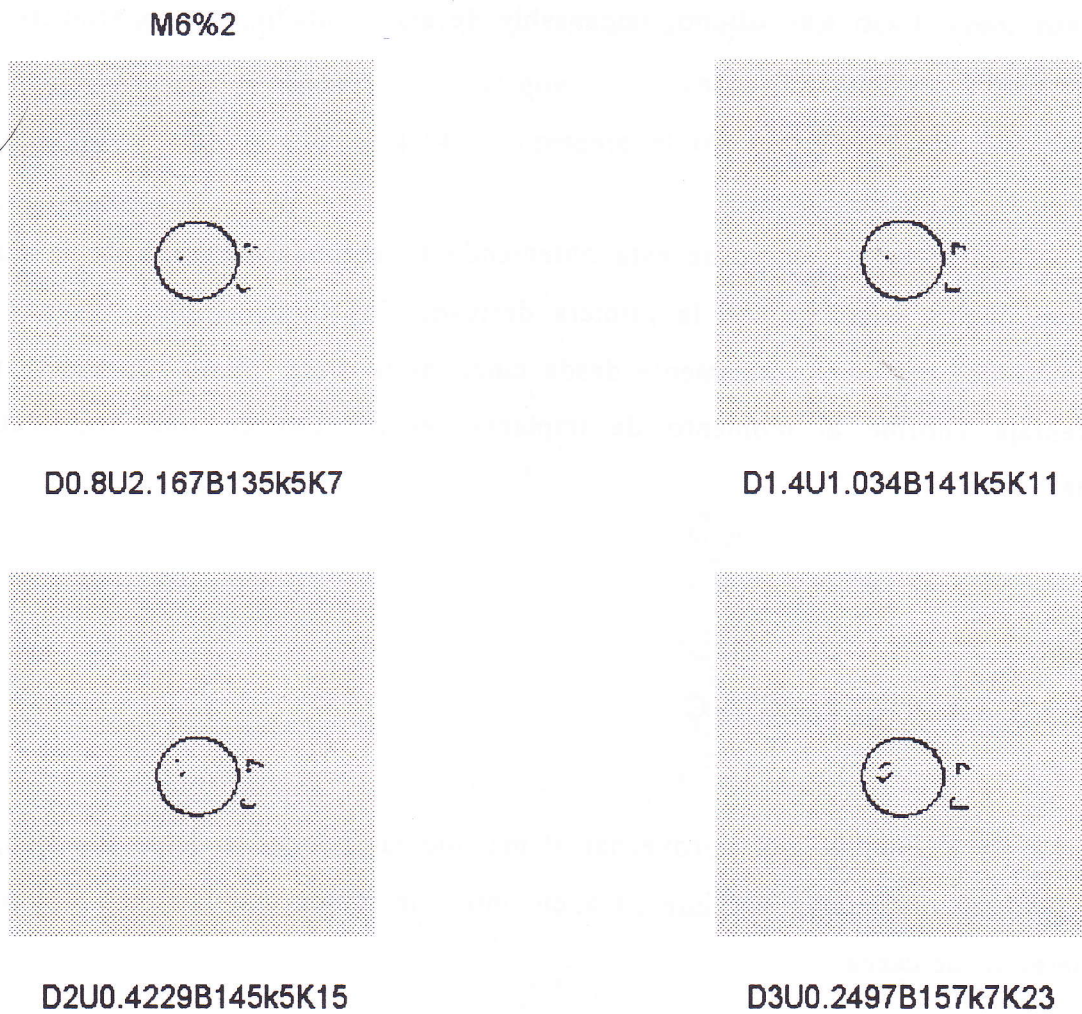


Figura 4.12 Resultado de Aplicar la Configuración B.

El resultado de aplicar la configuración B es muy superior al obtenido con la configuración A. Compara la Figura 4.10 y la Figura 4.12, puede verse que para la configuración B existe continuidad para los cuatro valores de desviación elegidos para este estudio. Los puntos bordes que aparecen a la derecha de la bola corresponden a bordes reales pertenecientes a la sombra de la misma.

Asimismo, téngase en cuenta que los valores escogidos son bastante

extremos puesto que los tamaños máximos necesarios para convolucionar directamente con sus respectivas segundas derivadas van desde siete hasta veintitrés. Caso este último, impensable de ser montado en una arquitectura hardware.

Sin embargo, como se está obteniendo la segunda derivada de la doble convolución sucesiva con la primera derivada, los tamaños para realizar las convoluciones varían solamente desde cinco hasta siete. Esto representa una ventaja enorme al momento de implantar el proceso en una arquitectura hardware.

4.3.3 Configuración C

En el empeño por aprovechar al máximo las bondades de la Pipeline, se introduce una nueva variante para encontrar la mejor disposición del bloque detector de ceros.

Existen dos posibilidades, la primera consiste en buscar los ceros de la manera como se ha venido haciendo hasta ahora, es decir, en las dos configuraciones precedentes. La segunda consiste en buscar los ceros de forma separada, es decir, buscando los pasos por cero en F2X y F2Y. Lógicamente la información para los detectores de ceros será F1X y F1Y en lugar de F1.

Al hacer la búsqueda de los ceros de forma separada, se ahorraría el cálculo que supone la composición de F2 y F1, dado que el proceso se llevaría de manera paralela y sólo convergería al momento de ir escribiendo los ceros encontrados en una matriz de resultados.

En el diagrama de flujo que se muestra en la Figura 4.13 se muestra la secuencia de cálculos anteriormente descrita, que se ha denominado Configuración C. El umbral, al igual que en los casos precedentes, puede ser realimentado. En la Figura 4.14 se presenta el resultado obtenido con esta configuración.

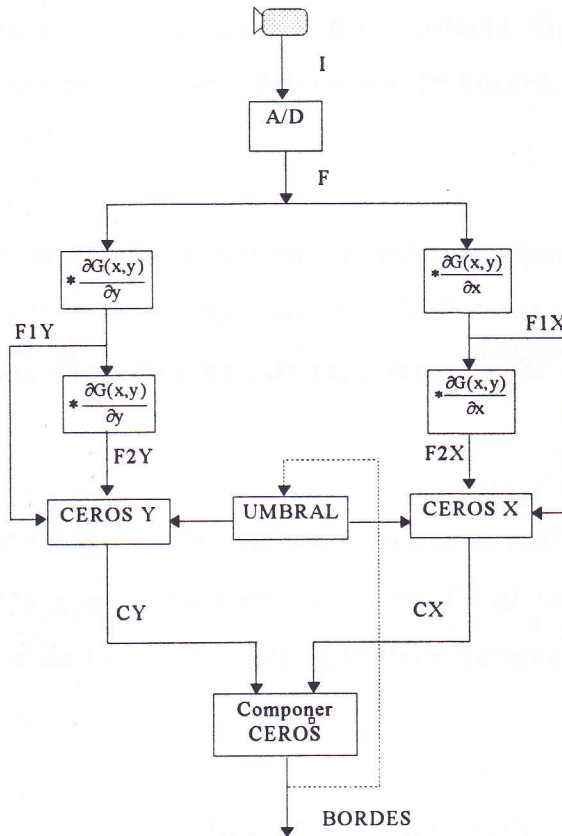


Figura 4.13 Configuración C.

Observando el resultado mostrado en la Figura 4.14 se puede ver cómo han aparecido una cantidad apreciable de bordes fantasma. Estos bordes fantasma están localizados formando líneas en dirección perpendicular a las diagonales, y tangentes al círculo que forman los bordes reales. Precisamente las direcciones diagonales son las direcciones que no han sido consideradas en el Laplaciano.

Si se aumenta el porcentaje del umbral la mayoría de las líneas desaparecen, pero se corre el riesgo de perder importantes bordes reales y obtener así discontinuidades en los límites de los objetos.

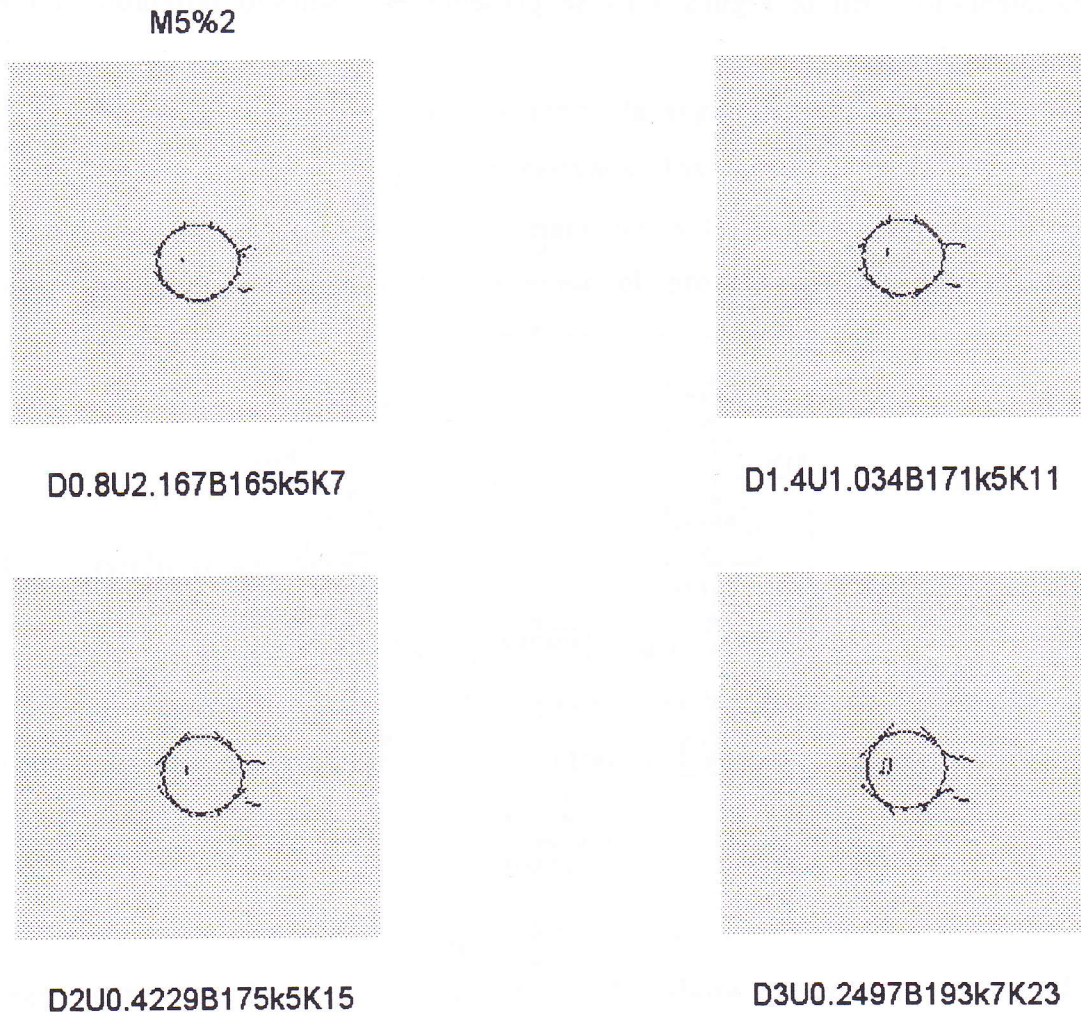


Figura 4.14 Resultado de aplicar la Configuración C

4.3.4 Configuración D

Se presenta asimismo una configuración adicional, D, cambiando el modo empleado en la aproximación de la primera derivada. El procedimiento consiste

en filtrar inicialmente la imagen digitalizada con un gaussiano G_0 ; el resultado FG , se convoluciona en paralelo con cuatro operadores direccionales diferentes. Estas direcciones se corresponden con los ángulos 0° , 90° , $+45^\circ$, y -45° grados medidos con respecto la parte positiva del eje X, y denominadas como X, Y, U y V respectivamente.

La motivación principal que impulsó a analizar esta configuración fue la existencia en el mercado de diferentes procesadores digitales de señal que, ofertados por sus fabricantes como detectores de bordes, realizan los cálculos antes descritos.

El dispositivo más representativo de esta configuración es el circuito integrado de la casa Plessey Semiconductors PDSP16401A. Este dispositivo se analiza en el siguiente capítulo y ha sido representado por líneas punteadas en la Figura 4.15.

Para encontrar el valor de la primera derivada se analiza el resultado de las cuatro convoluciones y se toma como valor de F_1 el valor máximo en valor absoluto. El cálculo de la segunda derivada debe hacerse derivando la primera así encontrada.

El inconveniente de esta configuración es que existe un primer operador G_0 que tiene un tamaño de kernel demasiado grande para aplicaciones en tiempo real. De cualquier manera se presenta esta configuración, cuyo diagrama de flujo puede verse en la Figura 4.15.

El resultado obtenido al aplicar la configuración D es el que se muestra en la Figura 4.16. Puede concluirse rápidamente que a desviaciones bajas aparecen bordes fantasma, que se convierten en discontinuidades al aumentar la desviación.

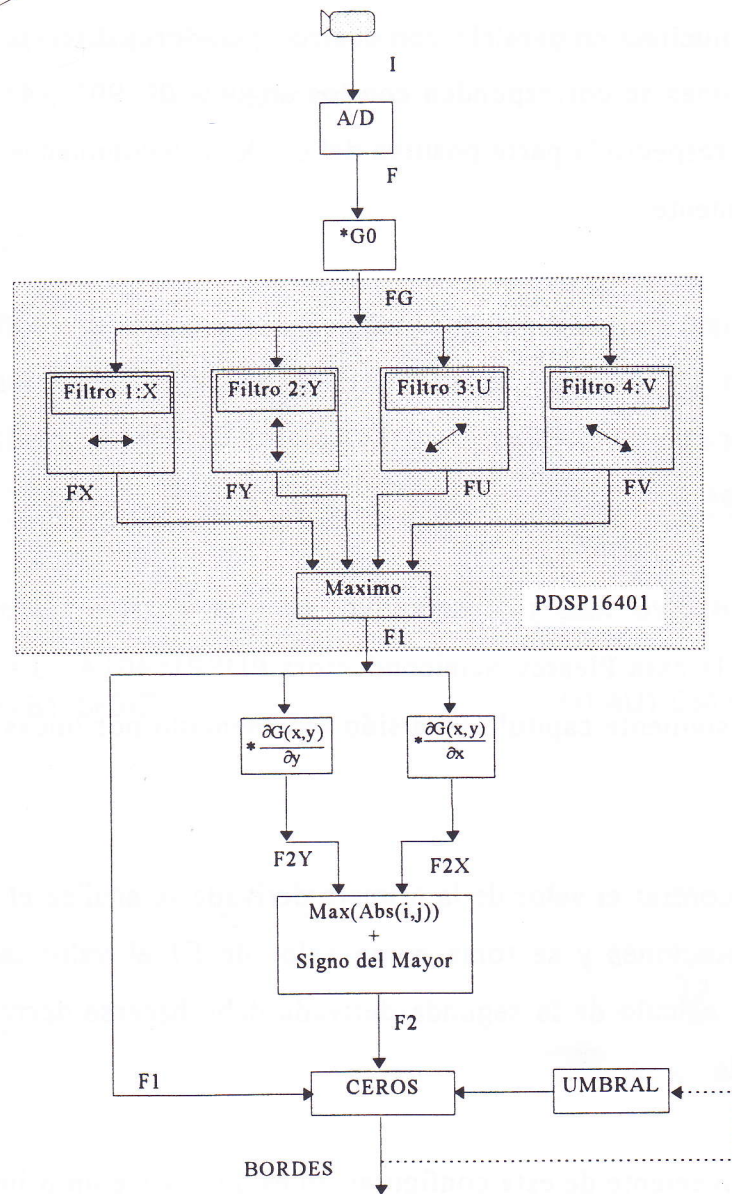


Figura 4.15 Configuración D.

Además, puede observarse, como es lógico, que el número de puntos bordes encontrados va disminuyendo al aumentar el valor de la desviación. Sin embargo, en el ejemplo presentado se observa que el número de puntos bordes es el mismo para una desviación de 1.4 como para una desviación de 2. Cuando la desviación se aumenta a 3, si se nota disminución en el número de puntos encontrados.

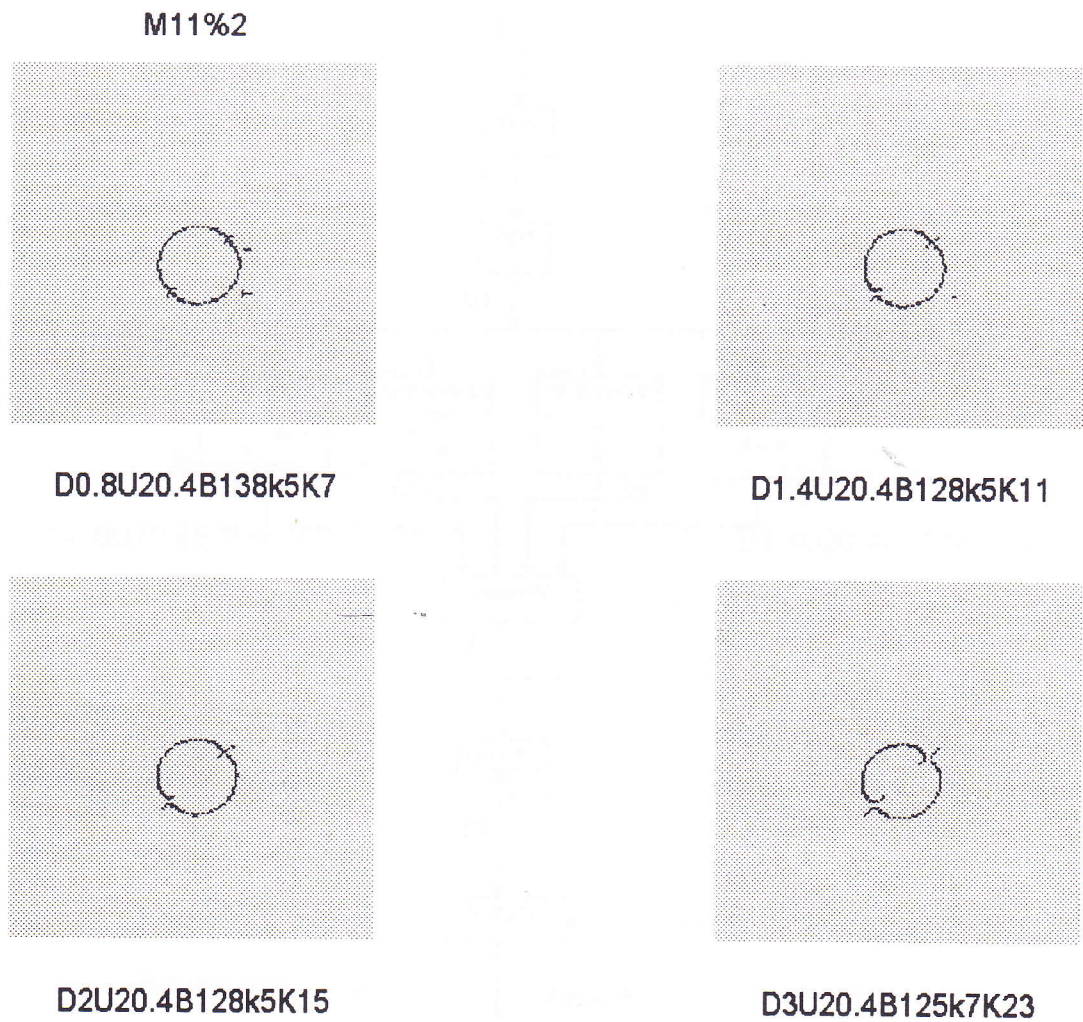


Figura 4.16 Resultado de la Configuración D.

4.3.5 Configuración E

Sin embargo, si la segunda derivada se obtiene directamente, es decir, convolucionando directamente la imagen digitalizada F con la derivada segunda del gaussiano, se obtiene una configuración distinta. A esta configuración se le ha denominado con la letra E y su diagrama de flujo se ha representado en la Figura 4.17.

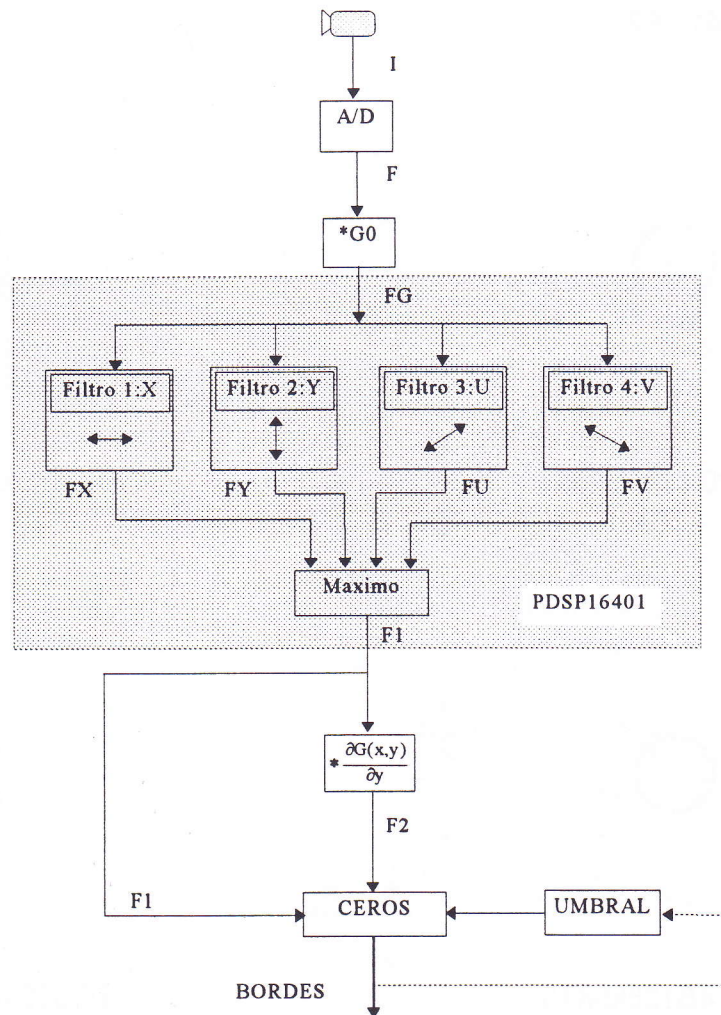


Figura 4.17 Configuración E.

El resultado obtenido al aplicar la configuración E es el que se muestra en la Figura 4.18. Las discontinuidades que aparecían en la Figura 4.16 han desaparecido, sin embargo, el tamaño del kernel ha aumentado de "k" a "K" por estar empleando directamente la segunda derivada del gaussiano para obtener la segunda derivada de la función. Por otro lado, puede observarse que a desviaciones bajas aparecen puntos bordes aislados. Éstos pueden ser eliminados al aumentar el porcentaje del umbral, pero siempre corriendo el riesgo de eliminar información real significativa. Este resultado era el esperado, dado que se está convolucionando con los tamaños máximos de los gaussianos, lo que aproxima la estimación al algoritmo inicial.

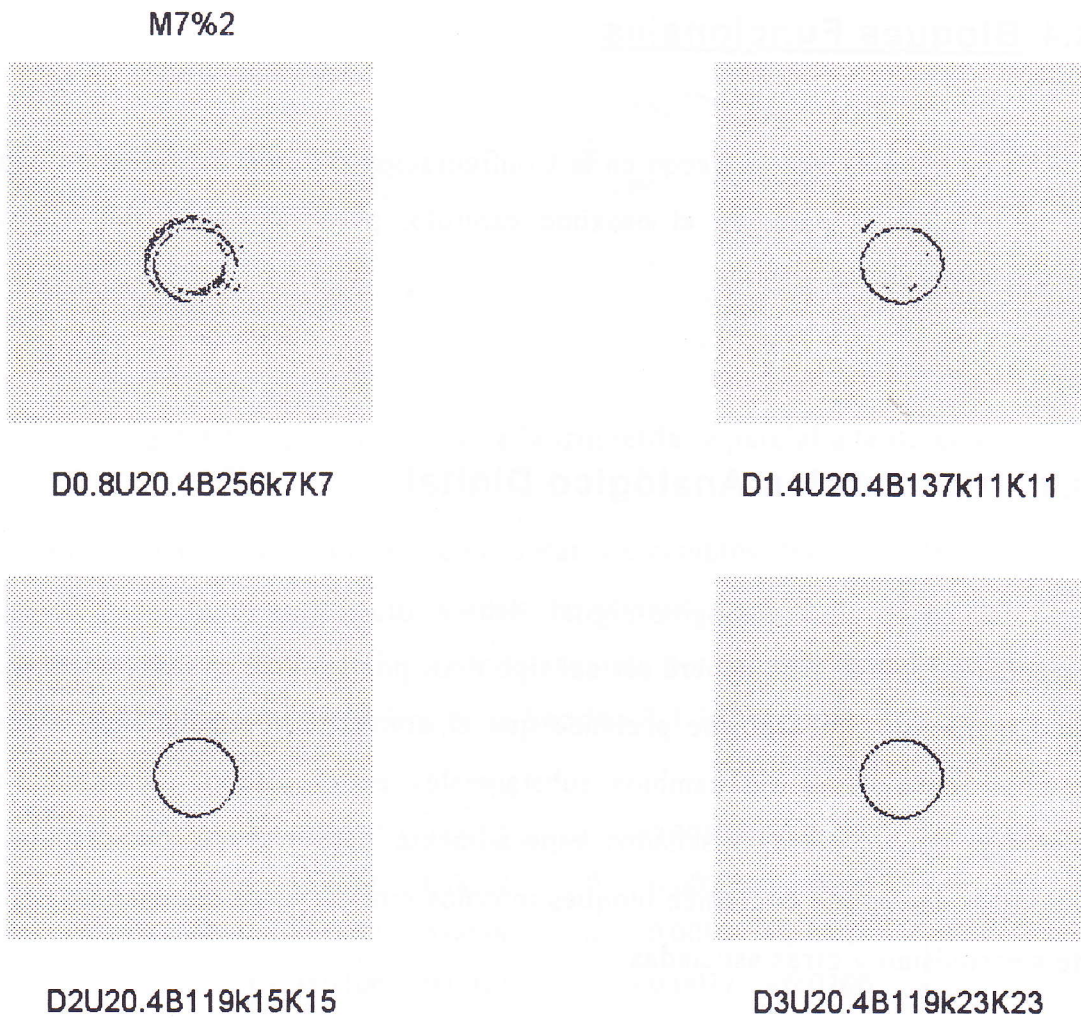


Figura 4.18 Resultado de la Configuración E.

Curiosamente, para un valor de desviación de 0.8 aparecen ciertas discontinuidades en el contorno de la bola sobre la parte inferior derecha. Estas discontinuidades se presentan también para otros valores bajos de desviación.

Una observación simple de los resultados obtenidos permite concluir que el comportamiento de la Configuración tipo B es el mejor. Por esta razón, en adelante se profundiza un poco más en esta configuración para lograr su implantación en una configuración hardware.

4.4 Bloques Funcionales

Los bloques que aparecen en la Configuración B serán definidos en cuanto a requerimientos para, en el próximo capítulo, pasar al diseño del hardware propiamente dicho.

4.4.1 Conversión Analógico Digital

La conversión analógico-digital deberá discretizar una señal de vídeo estándar. En este caso deberá ser del tipo PAL por ser este el sistema empleado en España; sin embargo, se pretende que el convertidor pueda, dado el caso, cambiar de sistema sin cambios substanciales en el diseño. Existen en el mercado convertidores diseñados especialmente para trabajar con señales de vídeo, es decir, que contienen bloques internos que permiten manejar las señales de sincronismo y otras asociadas.

4.4.2 Primera Derivada del Gausiano (PDG)

Se deben considerar en primer lugar dos factores que influyen en la conformación de los parámetros del filtro. El primero consiste en el truncamiento realizado al considerar como nuevo tamaño de filtro el valor w de la Ecuación (4.1). En la Figura 4.19 se pueden ver ambos filtros; destaca la diferencia del kernel en las dos representaciones, obsérvese que cuándo se emplea el filtro completo se necesita convolucionar con máscaras de 15×15 elementos, sin embargo, para el PDG llamado parcial, solamente se emplea una máscara de 5×5 .

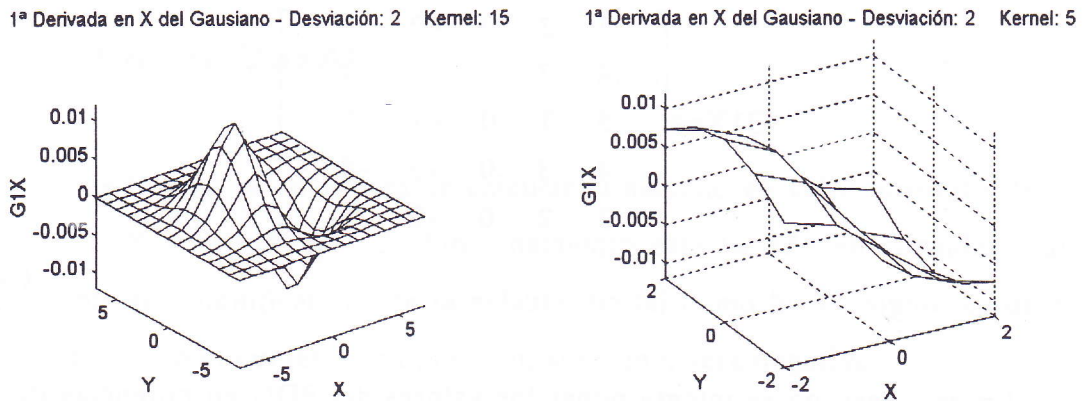


Figura 4.19 $\partial G/\partial x$ completo a la izquierda, y parcial a la derecha.

El segundo factor consiste considerar los cambios debidos a la estimación del PDG para números enteros; es decir, en cambiar los valores reales del PDG a valores enteros, tal como se hizo con la segunda derivada en el capítulo tres. El criterio es el mismo que se aplicó en la Sección 3.1.3.2.

$$G1Xr = \begin{bmatrix} 0.0073 & \overset{n}{\circlearrowleft} 0.0053 & 0 & -0.0053 & -0.0073 \\ 0.0106 & 0.0077 & 0 & -0.0077 & -0.0106 \\ 0.0121 & 0.0088 & 0 & -0.0088 & -0.0121 \\ 0.0106 & 0.0077 & 0 & -0.0077 & -0.0106 \\ 0.0073 & 0.0053 & 0 & -0.0053 & -0.0073 \end{bmatrix}$$

(4.3)

Se busca el valor mínimo n en la máscara compuesta de valores reales, y se calcula el factor que multiplicará a cada miembro de la máscara empleando la Ecuación (3.29). En (4.3) y (4.4) se muestran las matrices G1Xr y G1Xe que conforman el filtro en números reales y su aproximación a números enteros. El resultado obtenido es el que se muestra en la Figura 4.20. Como sucedía con el caso de la aproximación de la segunda derivada a números enteros, en la Figura 4.20 se observa que el PDG apenas ha sufrido alteraciones en su forma, además, en este caso no existe el problema del desplazamiento dado que la suma total será siempre cero.

$$G1Xe = \begin{bmatrix} 3 & 2 & 0 & -2 & -3 \\ 4 & 3 & 0 & -3 & -4 \\ 5 & 3 & 0 & -3 & -5 \\ 4 & 3 & 0 & -3 & -4 \\ 3 & 2 & 0 & -2 & -3 \end{bmatrix}$$

(4.4)

En este caso no se intenta poner los valores del PDG en potencias de dos porque éste puede ser calculado offline. Una vez calculados los valores enteros, se puede cargar con éstos el circuito integrado DSP que se encargará de calcular la convolución que corresponda.

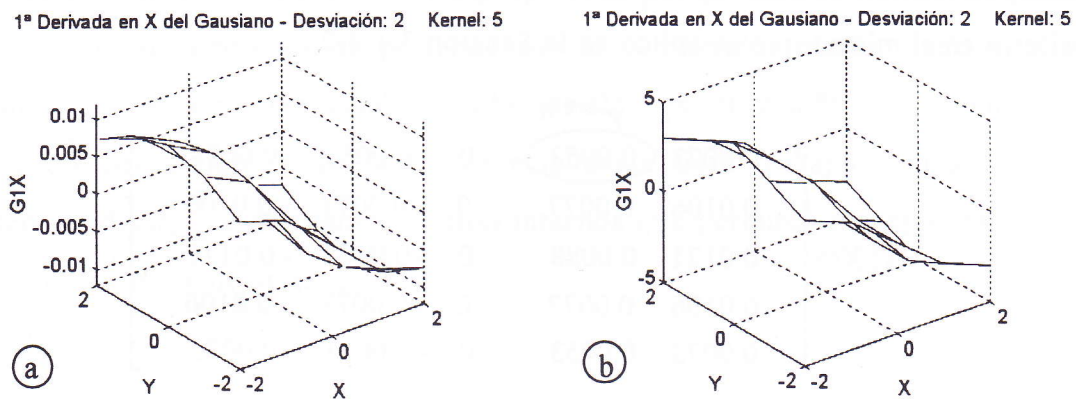


Figura 4.20 $\partial G/\partial x$ parcial con números reales y enteros.

En adelante cuándo se refiera a G1X ó PDG se hará referencia al filtro con kernel parcial y valores enteros tal como el representado en la Figura 4.20 y Ecuación (4.4). Por tanto el bloque funcional PDG estará formado por el hardware que calcule en tiempo real la convolución de la imagen que va llegando con el filtro PDG.

4.4.3 Bloque MaxAbs

Este bloque se encargará de calcular el máximo en valor absoluto de entre F1X y F1Y. El principio de funcionamiento está basado en considerar al bit MSB -bit más significativo- de la palabra digital como bit de signo, es decir, si este bit es cero el número será positivo, si es uno, será negativo.

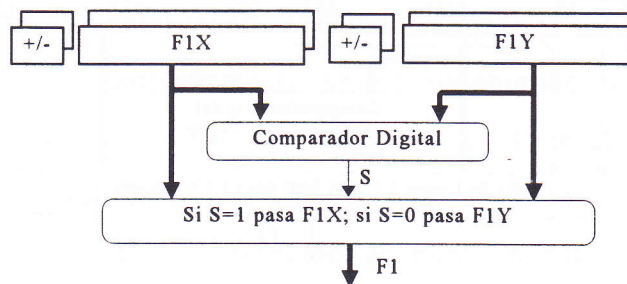


Figura 4.21 Diagrama funcional del bloque MaxAbs.

Si esto es cierto, para obtener el mayor número en valor absoluto de entre dos números, basta con comparar las representaciones de bits de los mismos sin considerar el bit MSB. La comparación se hace de manera lógica por medio de circuitos integrados que son en sí mismos sendos comparadores de ocho bits y que pueden ser conectados en cascada para palabras de mayor tamaño.

Normalmente estos comparadores, cuyo tiempo medio de respuesta rondan los cuatro nanosegundos, suelen tener tres salidas digitales: una para indicar cuándo los dos números comparados son iguales, una segunda para indicar cuándo es menor, y una última para significar que es mayor. De esta manera se puede controlar un Switch digital a manera de multiplexor para obtener el resultado deseado. El diagrama funcional del bloque se muestra en la Figura 4.21.

4.4.4 Bloque MaxAbsSign

Esta etapa es en esencia similar a la anterior con la diferencia que el Switch deja pasar al mayor de los dos números comparados y le añade su signo original. La comparación se hace solamente entre los valores absolutos de los dos números, pero el multiplexado se hace sobre el número y su signo, tal como se puede observar en la Figura 4.22.

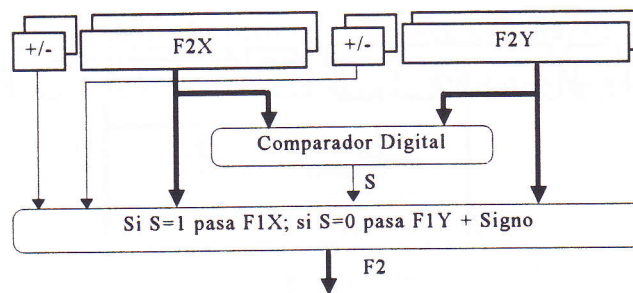


Figura 4.22 Diagrama funcional del bloque MaxAbs.

El signo no es más que el bit más significativo de las palabras digitales que ingresan al comparador y que no es conectado al mismo. En este caso se emplea el signo para hacer posible la posterior búsqueda de paso por cero de la segunda derivada de la imagen.

4.4.5 Bloque Ceros

Como se dijo en secciones anteriores la búsqueda de pasos por cero debe ser ejecutada de la forma más rápidamente posible. Por esta razón esta búsqueda se hace en dos direcciones ortogonales que coinciden con las empleadas con los gradientes.

A diferencia del algoritmo presentado en la Sección 3.1.2.3, la detección se hace con la primera derivada y la localización se sigue haciendo como en 3.1.2.3.

De esta manera, el criterio quedaría definido como sigue:

Sean l_a y l_b las distancias definidas en la Sección 3.1.2.3 y representadas en la Figura 3.27, se considera punto borde aquel punto que cumple las siguientes condiciones:

1. Detección:

Si $F1$ es mayor que Umbral, se ha detectado un punto borde. Para verificar esta condición basta con conectar a un comparador el Umbral elegido y la $F1$ para un tiempo t . Instantáneamente se irá obteniendo un bit que indicará cuándo $F1$ ha rebasado el Umbral impuesto. Será suficiente con atender a los cambios de este bit para saber si se ha presentado un punto borde. Esta operación se debe hacer en las dos direcciones elegidas que en este caso coinciden con los ejes X e Y.

2. Localización:

Considérese la Figura 3.27. Empleando el mismo tipo de comparador digital se puede comparar l_a y l_b . Si uno de los dos es cero, el punto-borde se encuentra localizado en aquél. Si no, como sucederá en casi todos los casos, el punto borde corresponderá al punto cuyo l_i sea menor. De esta manera se pueden localizar mejor los puntos bordes

$$C = \begin{bmatrix} \zeta & l_{a_y} & \zeta \\ l_{a_x} & l_b & \zeta \\ \zeta & \zeta & \zeta \end{bmatrix}$$

(4.5)

En la Figura 4.23 se presenta el diagrama funcional para este bloque. La distribución espacial es la que se indica en la matriz C de (4.5), que se llamará

en adelante matriz de búsqueda.

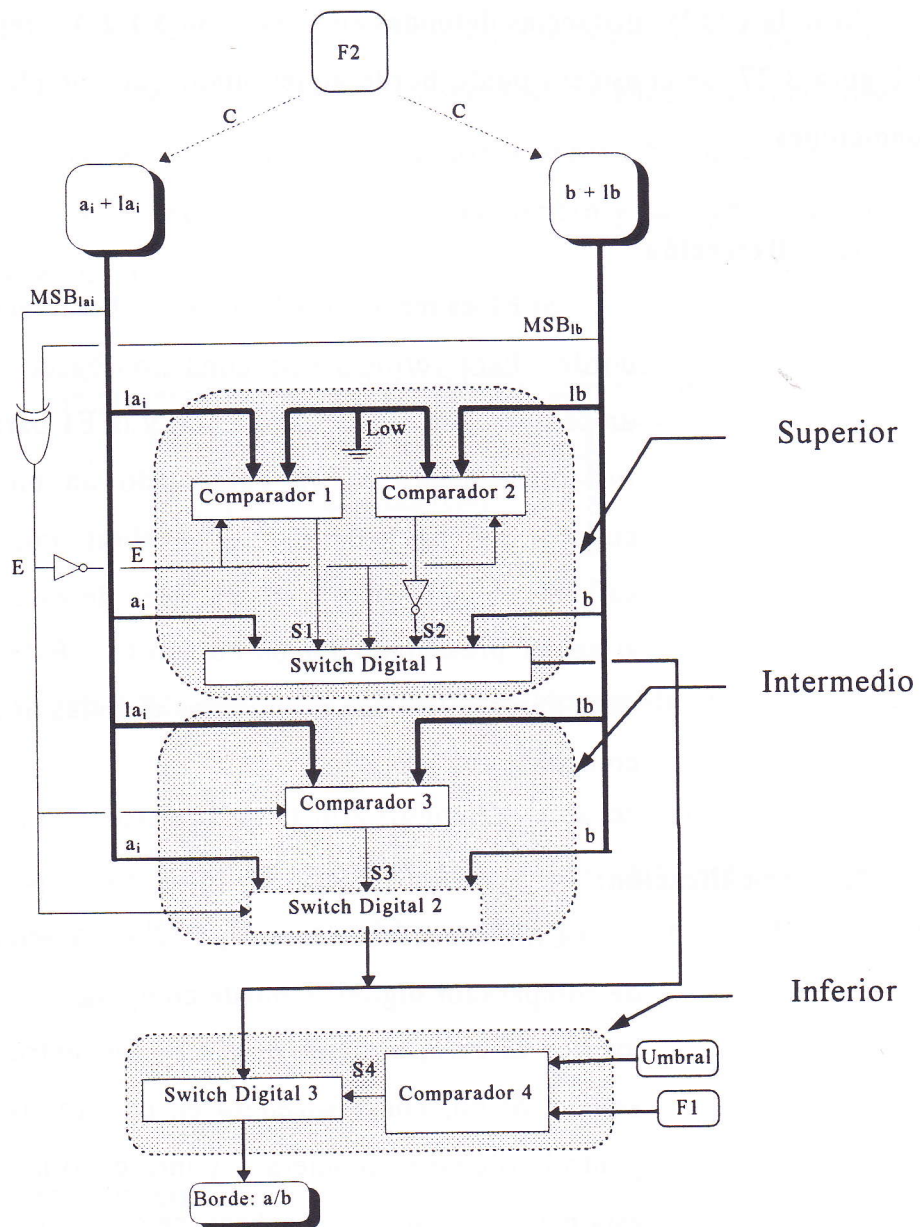


Figura 4.23 Diagrama funcional del bloque Ceros.

En la Figura 4.23 se han dibujado tres sub-bloques sombreados designados como superior, intermedio, e inferior. Básicamente se trata de determinar con la llave lógica EXOR si ha habido algún cambio de signo en el paso de $a \rightarrow b$. Si los MSB de las longitudes son diferentes, la EXOR dará un 1 lógico, si son iguales un 0 lógico. Si son iguales, lo que quiere decir que no ha habido cambio de

signo, puede que una de ellas haya tomado el valor exacto cero en intensidad. El sub-bloque superior se encarga de detectar si la a ó b son cero, en cuyo caso el sub-bloque intermedio está inhabilitado y a ó b es localizado como punto-borde si uno de sus a ó b son cero. Los comparadores uno y dos pueden ser comparadores normales ó simples llaves OR multipuerta; si la salida de ésta es cero, quiere decir que todos los bits de "1" son ceros, luego la palabra completa será el borde.

Si los MSB son diferentes E tomará el valor de uno lógico y habilitará el sub-bloque intermedio deshabilitando al mismo tiempo el superior. En este caso el Comparador 3 se encarga de ver cuál de los dos, a ó b , es menor. El punto borde será aquel a ó b cuyo a ó b sea el menor, tal como se vio en la Sección 3.1.2.3.

El sub-bloque inferior es el que verifica si la fuerza de cambio es suficiente para que el punto sea considerado como punto borde. Simplemente compara la primera derivada en "b" con el valor del umbral establecido. Este valor puede ser cambiado "on line" sin ningún problema y obtener así diferentes niveles de bordes.

Lo que tiene que ser calculado "off line" son los coeficientes de la primera derivada del gaussiano, en valores enteros, que sirven para generar todos los gradientes.

Por otra parte, tal como se muestra en la Figura 4.23, el proceso para a_i deberá duplicarse, una vez para la dirección X y otra para la dirección Y. El cálculo debe realizarse de forma paralela y en ambos casos servirá para determinar si existe en "b" un punto borde generado en la dirección X ó en la Y. Los valores de a_i , la_i , b , y lb son designados por la matriz de búsqueda C mencionada anteriormente en (4.5). Esta matriz se va desplazando a lo largo de F2 haciendo coincidir b con cada punto de la misma.

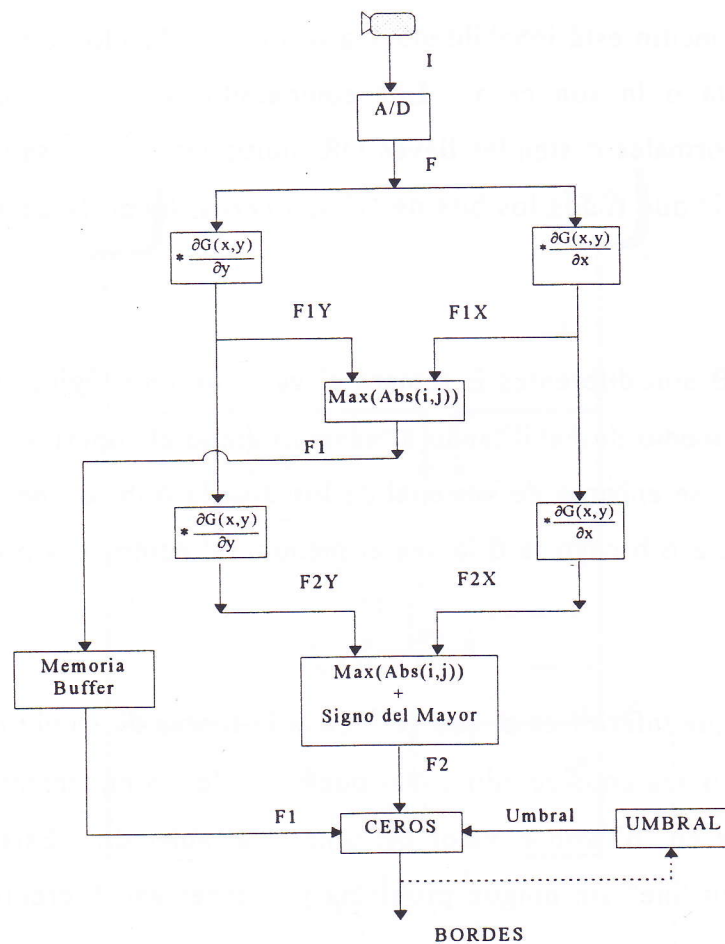


Figura 4.24 Configuración final.

En la Figura 4.24 se presenta la configuración B denominada ahora como configuración final porque el hardware se hará en base a esta configuración. Se ha incluido a la izquierda un bloque llamado Memoria Buffer para ir almacenando los valores de F1, dado que éstos son calculados más rápidamente que F2. En el próximo capítulo se desarrolla el diseño del hardware basándose en los bloques presentados en el presente.

A continuación se muestra el resultado obtenido al tratar la imagen con una secuencia como la mostrada en la configuración final. Compárese el resultado con el obtenido y mostrado al final del capítulo anterior (Figura 3.58).

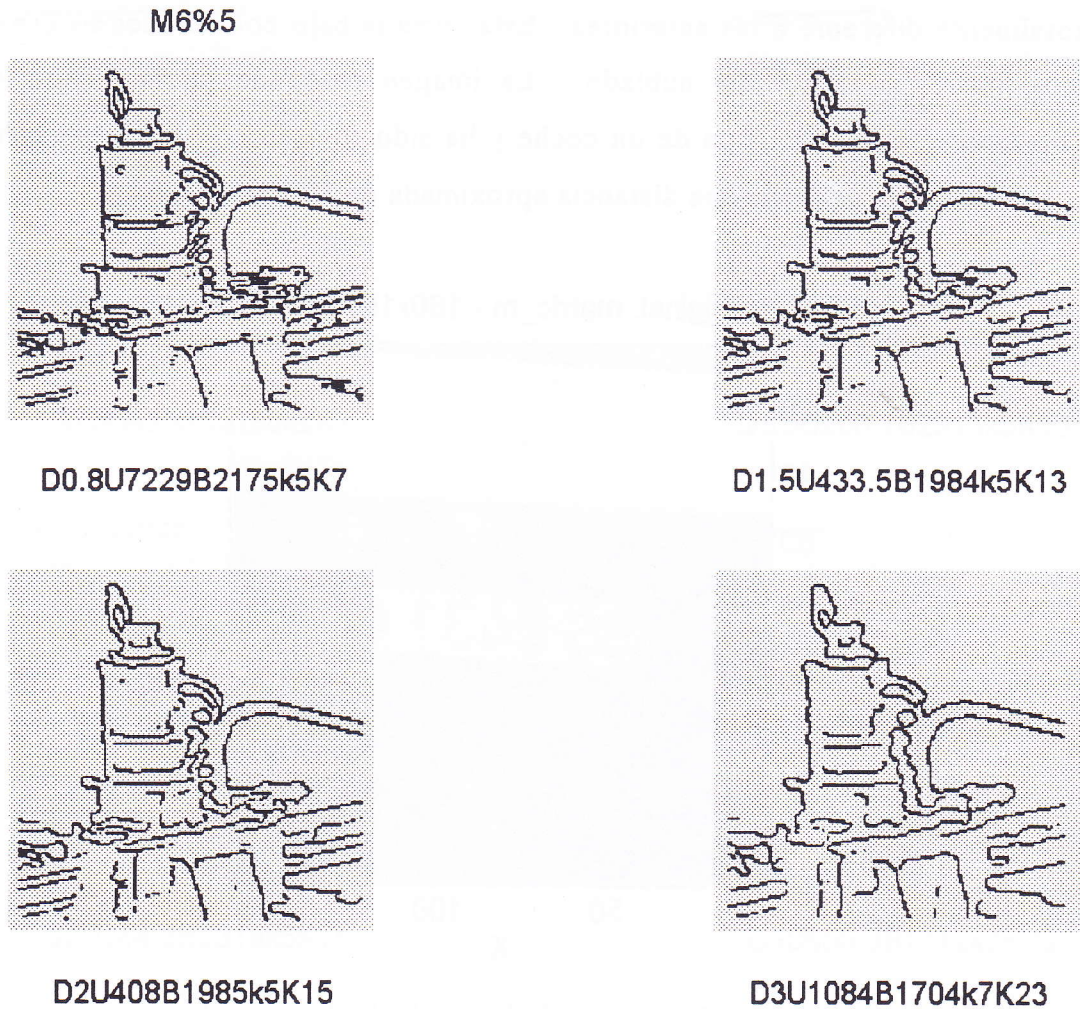


Figura 4.25 Resultado obtenido con la Configuración Final.

Se puede ver que los bordes en los extremos de la imagen no han sido truncados como lo fueron en la Figura 3.58. Este efecto era debido al excesivo tamaño de $G2(x,y)$ lo que obligaba a recortar el *efecto de desbordamiento* en la convolución con la imagen. Sin embargo, en la Figura 4.25 se puede ver que ese truncamiento es mínimo debido a la notable reducción en el tamaño del filtro $G1(x,y)$. Por ejemplo, para un valor de desviación igual a tres la $G2(x,y)$ necesita un tamaño de máscara de 23×23 puntos; sin embargo, para $G1(x,y)$ este valor se reduce a 7×7 puntos. Esto en términos de economía de cálculo en operaciones de convolución significa un ahorro notable.

A continuación se muestra el resultado obtenido con otra imagen totalmente diferente a las anteriores. Está tomada bajo condiciones externas al laboratorio, y en un día nublado. La imagen mostrada en la Figura 4.26 corresponde a la matrícula de un coche y ha sido tomada con la misma cámara que las anteriores desde una distancia aproximada de 50 mts.

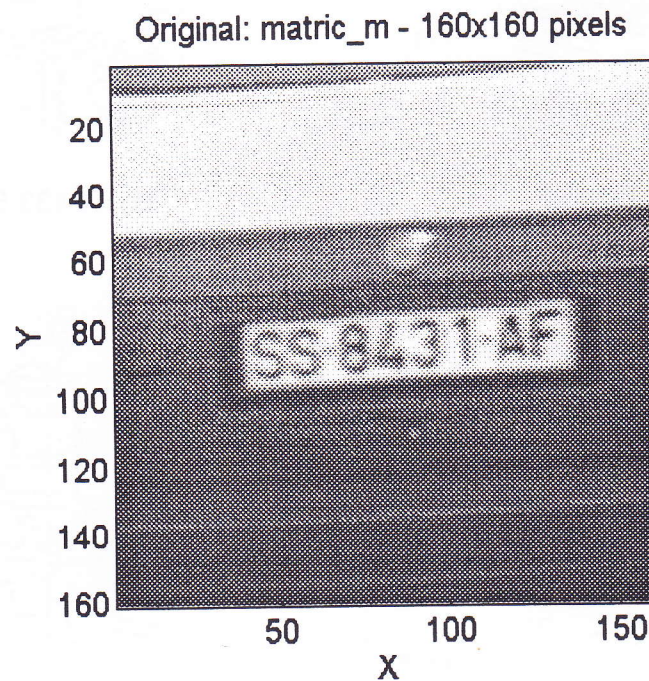
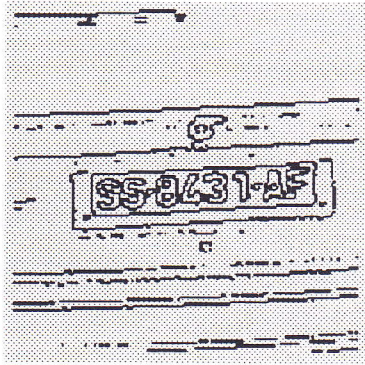


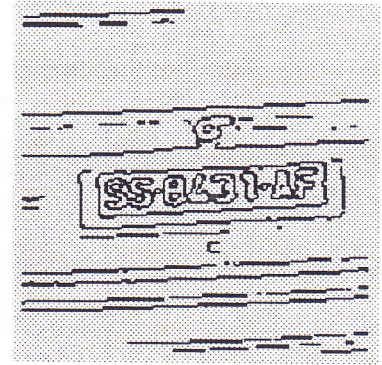
Figura 4.26 Imagen de la matrícula de un coche.

Se ha querido resaltar el efecto ocasionado al variar drásticamente el valor del porcentaje del Umbral de 3% a 10% (Figuras 4.27 y 4.28). Nótese la cantidad de detalles que desaparecen de un resultado a otro. Ambos ejemplos han sido obtenidos con la Configuración Final antes mencionada, es decir, obteniendo los gradientes con la primera derivada del gausiano, y ésta aproximada a números enteros. Asimismo, la segunda derivada es aproximada convolucionando dos veces con $G1(x,y)$.

M6%3



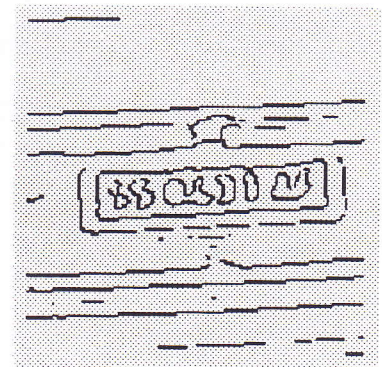
D0.8U4338B2250k5K7



D1.5U260.1B2313k5K13



D2U244.8B2319k5K15



D3U650.3B1712k7K23

Figura 4.27 Resultado obtenido empleando un porcentaje de 3%.

En la Figura 4.27 se puede observar que el número de puntos bordes encontrados disminuye a razón inversa con la desviación. Sin embargo, no ocurre lo mismo al pasar de una desviación de 1.5 a 2, donde los puntos encontrados también han aumentado de 2313 a 2319. Este resultado se repite para valores bajos de porcentaje. Si se hace la misma observación en la Figura 4.28, se notará que para un porcentaje de umbral del 10%, los puntos bordes si han disminuido a medida que la desviación ha aumentado. Obsérvese la limpieza de los bordes para una desviación de 0.8 y un porcentaje del 10% en la Figura 4.28.

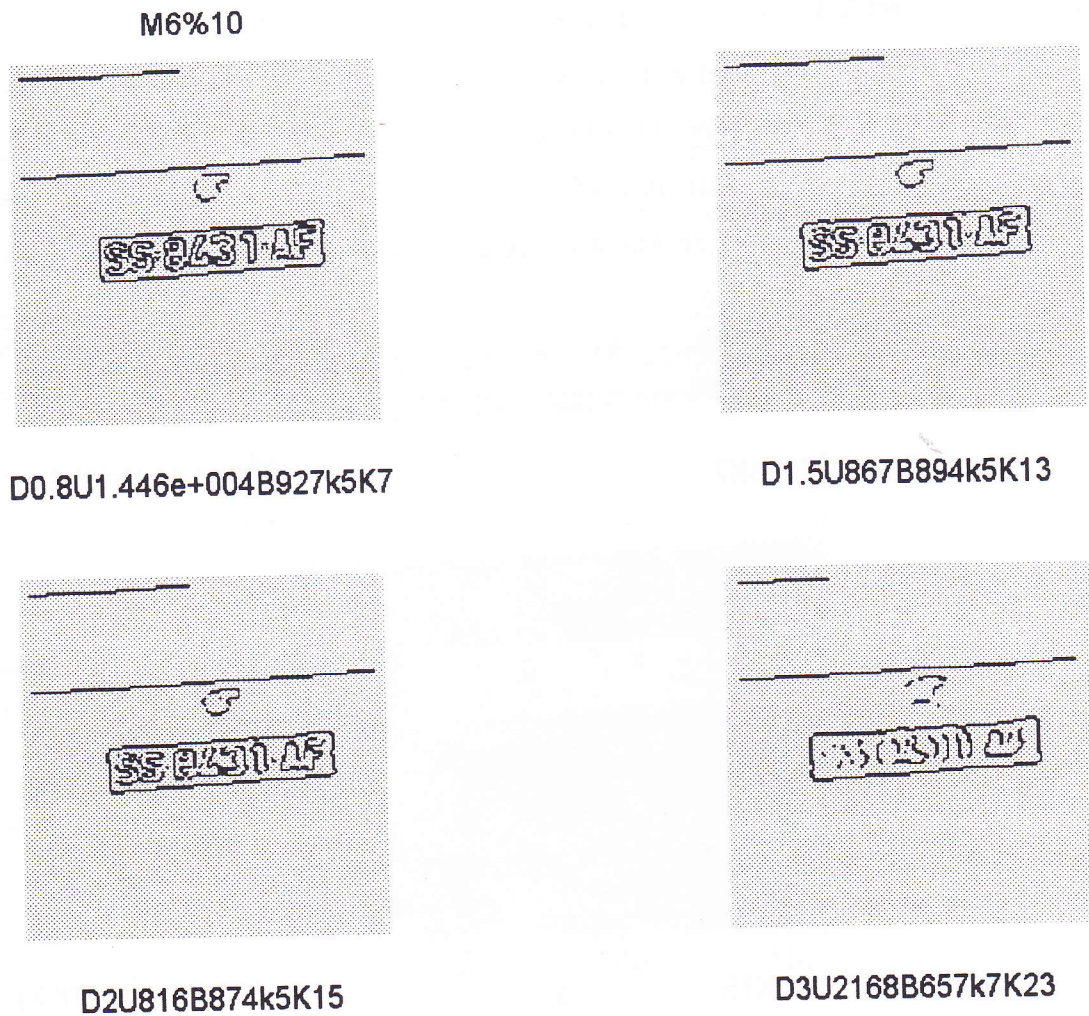


Figura 4.28 Resultado obtenido empleando un porcentaje de 10%.

4.5 Sumario del Cuarto Capítulo

En este capítulo se presenta una alternativa para solucionar el problema que supone el excesivo tamaño del filtro segunda derivada del gaussiano para valores de desviación relativamente grandes. Se propone emplear dos veces consecutivas el filtro primera derivada del gaussiano, truncándolo en $\pm\sqrt{2}\sigma$. Esta estimación, aunque no es equivalente a convolucionar directamente con la segunda derivada del gaussiano, se comporta de manera muy aceptable.

Se ha estudiado diversas configuraciones para implantar el algoritmo en hardware. Se ha encontrado que la denominada Configuración B es la que se comporta mejor.

Se ha establecido las funciones que deben cumplir cada bloque funcional de la configuración elegida. Cada bloque funcional se acerca, en su definición, a un nivel de procesamiento de señal cada vez más bajo, para terminar desembocando en las características que debe contener el hardware de proceso y su electrónica asociada.

En el siguiente capítulo que viene se presentará el diseño del hardware propiamente dicho atendiendo a la denominada Configuración Final, empleando los bloques descritos en el presente capítulo.

5. DISEÑO DEL HARDWARE

Para implantar la Configuración Final bajo una arquitectura hardware se hace necesario dividir la configuración elegida en bloques funcionales que en sí mismos formen una unidad de entrada y salida de datos para, dado el caso, poder cambiar la configuración simplemente cambiando la posición y número de éstos bloques. Este planteamiento dota al circuito resultante de cierta flexibilidad y permite, de alguna manera, aproximar el comportamiento de los bloques a funciones independientes dentro de un programa estructurado de software con la diferencia de que cada llamada a función se corresponde con la adición de un bloque nuevo similar a otro ya existente.

En adelante, al circuito resultante de la Configuración Final se le

denominará como circuito Bordes, bloque Bordes, ó simplemente Bordes, para diferenciar el diagrama funcional del circuito hardware. El circuito Bordes así concebido se ha subdividido en los ocho bloques funcionales que se definen en la Figura 5.1. Se ha empleado abreviaturas para no hacer muy larga la denominación de cada bloque. Las abreviaturas comienzan con la letra **B** para indicar que pertenecen a un bloque genérico de jerarquía superior llamado Bordes.

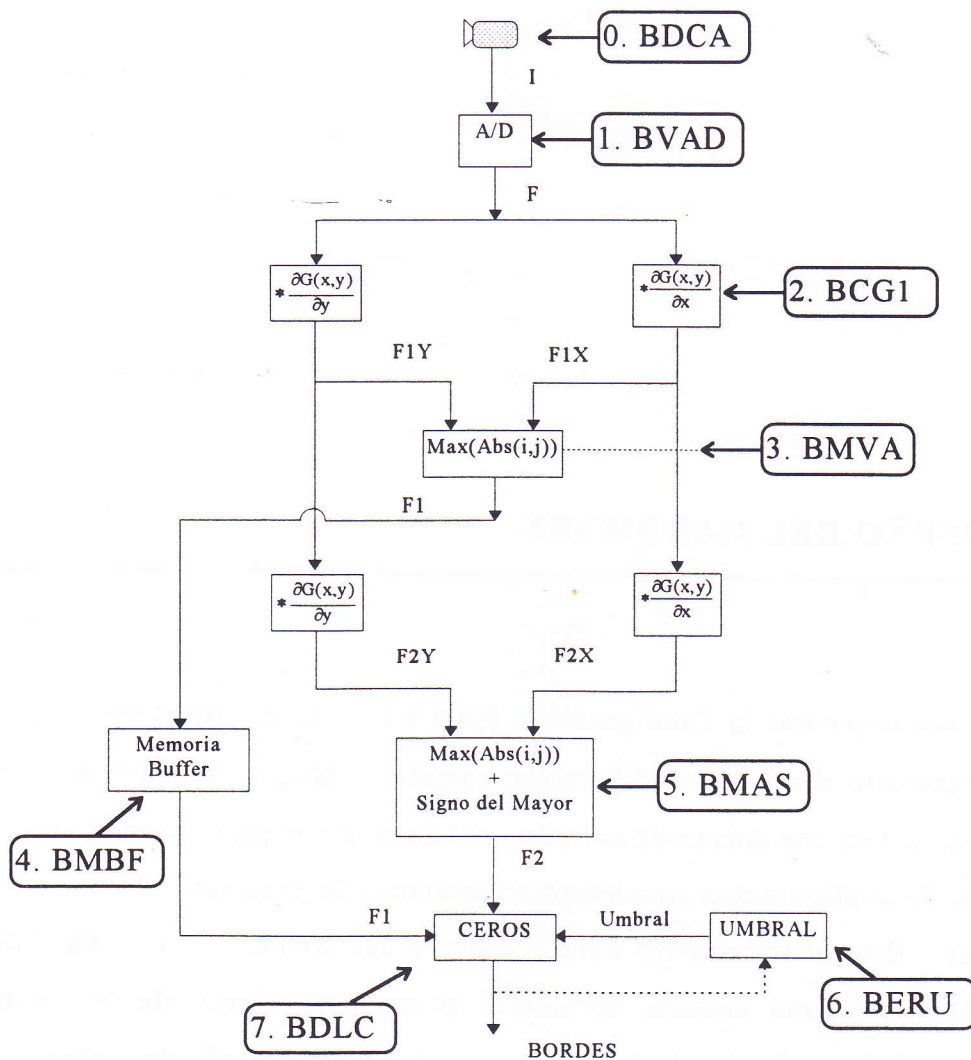


Figura 5.1 Asignación de Abreviaturas a los Bloques.

En la Figura 5.1 se han señalado los bloques funcionales que serán tratados de manera independiente. Todos los bloques son críticos, sin embargo, no se

puede ignorar que algunos son más importantes que otros. Entre los que requieren mayor atención se encuentran el BVAD, BMVA, BDALC y BMBF.

Tabla 5-1 Significado de las Abreviaturas empleadas en los Bloques.

Número	Abreviatura	Significado
0	B_DCA	<u>D</u> ispositivo de <u>C</u> arga <u>A</u> coplada.
1	B_VAD	Convertidor de <u>V</u> ídeo <u>A</u> nalógico- <u>D</u> igital.
2	B_CG1	<u>C</u> onvoluciona con <u>G</u> 1.
3	B_MVA	Encuentra el <u>M</u> ayor en <u>V</u> alor <u>A</u> bsoluto.
4	B_MBR	<u>M</u> emoria <u>B</u> uffer de <u>R</u> etardo.
5	B_MAS	<u>M</u> ayor en valor <u>A</u> bsoluto + <u>S</u> igno del mayor.
6	B_ERU	<u>E</u> stablece y <u>R</u> ealimenta el <u>U</u> mbraL.
7	B_DLC	<u>D</u> etecta y <u>L</u> ocaliza los pasos por <u>C</u> ero.

En adelante aparecerán términos, muy utilizados para designar dispositivos hardware, en su denominación original en lengua inglesa. Estos términos son comúnmente empleados en la lengua castellana, por los que la traducción de los mismos podría alterar su significado.

5.1 Dispositivo de Carga Acoplada (BDCA)

Este bloque consiste en una cámara estándar con un sensor de tipo CCD ("Chare Coupled Device"). Este bloque es el encargado de proporcionar la señal de vídeo que contiene la información que será tratada en los bloques posteriores. La cámara empleada es la XC-77CE de Sony que es una cámara monocroma de alta calidad con 753x581 elementos, un área sensible de 8.8x6.6 ó sensor de 2/3 de pulgada, y opera con el sistema internacional CCIR (50 campos).

Por tratarse del sensor ó transductor que capta la señal correspondiente a la escena y la trasforma en la información de la que parten todos los cálculos, se ha incluido un apéndice dedicado exclusivamente a este bloque. De este modo, el Apéndice B recoge un resumen del tipo de señal que proporciona la cámara antes mencionada e incluye una colección de gráficos obtenidos con un osciloscopio conectado directamente a la salida de la misma. Esto se ha hecho porque dentro de las normas *recomendadas* por el CCIR existen algunas variantes adoptadas por los fabricantes, e interesa conocer en detalle la señal con la que se ha de trabajar.

5.2 Convertidor de Vídeo Analógico Digital (BVAD)

Para tratar la señal de Vídeo Compuesto, descrita en el Apéndice B, es necesario convertir la información que viaja en ella en una señal de tipo digital. Para ello es necesario elegir un dispositivo electrónico apropiado, y tomar una decisión sobre qué parte de la señal será empleada para la aplicación. Es evidente que lo mejor sería emplear la totalidad de la información, pero, como se dijo anteriormente, existen condicionamientos de tiempo y costo computacional que obligan a elegir unas dimensiones de imagen apropiadas a la velocidad de procesamiento de la "pipeline" que comienza precisamente con este bloque.

Para el primer caso, es decir, para la elección del convertidor analógico digital, no existe prácticamente ningún problema. Existen en el mercado una variada oferta de componentes diseñados especialmente para convertir señales de vídeo compuesto a información digital. La diferencia que existe entre unos y otros consiste en la cantidad de funciones asociadas que incluyen dentro de la misma pastilla.

Es muy importante que el convertidor elegido incluya la mayor cantidad de funciones asociadas, o lo que es lo mismo, que necesite el mínimo de componentes externos para poder realizar una conversión correcta. Esto conlleva a elevar el precio del componente pero evita gran cantidad de problemas al momento de trabajar el trazado del circuito impreso que contendrá el componente. La mayoría de estos problemas son debidos a ruidos generados por otros componentes y captados por el convertidor como si fuera señal de información válida.

A lo largo del trabajo de investigación se han estudiado algunos convertidores habiéndose encontrado tres de prestaciones satisfactorias para esta aplicación. Dos de ellos de la Casa Brooktree y uno de la Casa Plessey.

Los dos primeros, el Bt208 y Bt218, son convertidores CMOS A/D de vídeo del tipo Flash de 8 bits y trabajan a 18 y 30 MSPS (Mega Sampling Per Second) respectivamente. Ambos cumplen los requerimientos de velocidad necesarios, son muy baratos comparados con los demás convertidores, pero necesitan demasiados componentes asociados.

El tercero, el SP94308, compatible con los sistemas estándares PAL y NTSC, es un convertidor de 8 bits que trabaja a 20 MHz que no requiere muestreador-retenedor, tiene un amplificador interno de reloj, y un latch ó candado interno en su salida. Las salidas digitales son compatibles con tecnología CMOS y TTL, tiene buffer driver de corriente interno, y está protegido para descargas de electricidad estática. Aunque su precio se eleva notablemente con respecto a los anteriores - casi cinco veces - se obtienen ventajas al diseñar porque se ahorra en otros componentes, se reduce el tamaño de la tarjeta y baja el ruido inducido. Por las ventajas expuestas se ha trabajado con el SP94308.

La entrada de reloj ó clock ha sido obtenida directamente de la cámara, la

cual proporciona una salida opcional de 14.1875 MHz, lo que significa que el periodo de muestreo será de 70.4845 ns. En la Figura 5.2 se presenta un diagrama de bloques del convertidor.

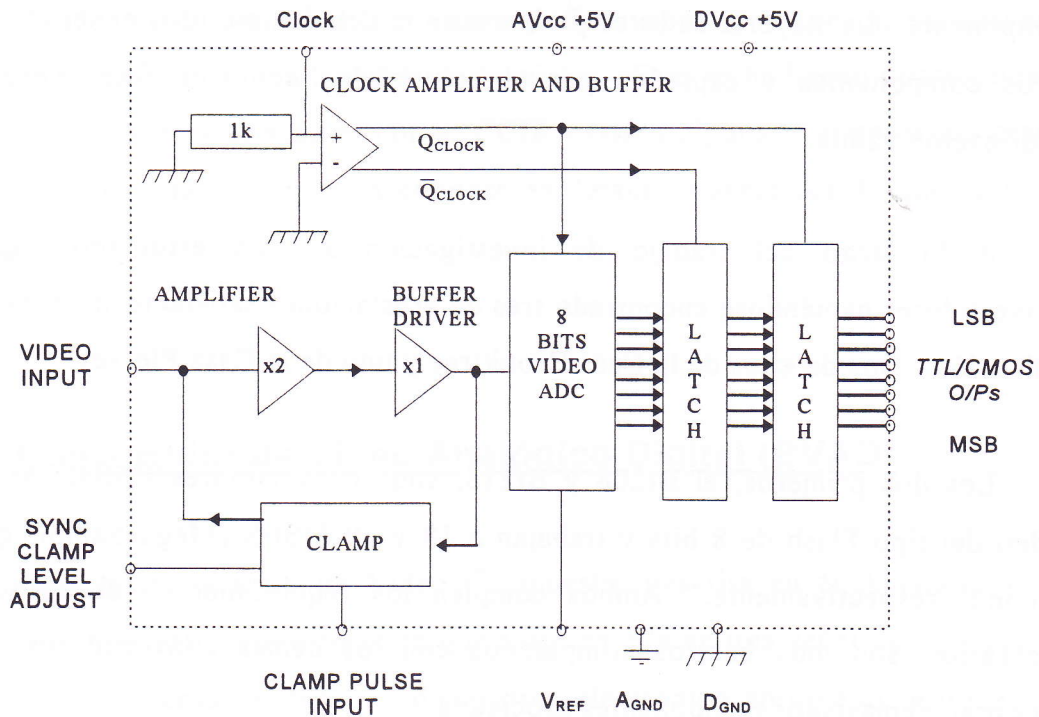


Figura 5.2 Diagrama de Bloques Interno.

Nótese que para funcionar sólo necesita a la entrada, la señal de vídeo compuesto, el clock, y la señal de sincronismo. La señal de sincronismo se puede obtener fácilmente con un "array" de transistores.

Como la velocidad de conversión, 20 MHz, es mayor que la velocidad de muestreo, 14.1875 MHz, no es necesario elegir la parte de la trama que será convertida. Es decir, conforme va ingresando la señal al convertidor, va saliendo la información digitalizada después de un tiempo $t_{conv}=2t_c+t_p$. La Figura 5.3 muestra los tiempos de conversión para este dispositivo.

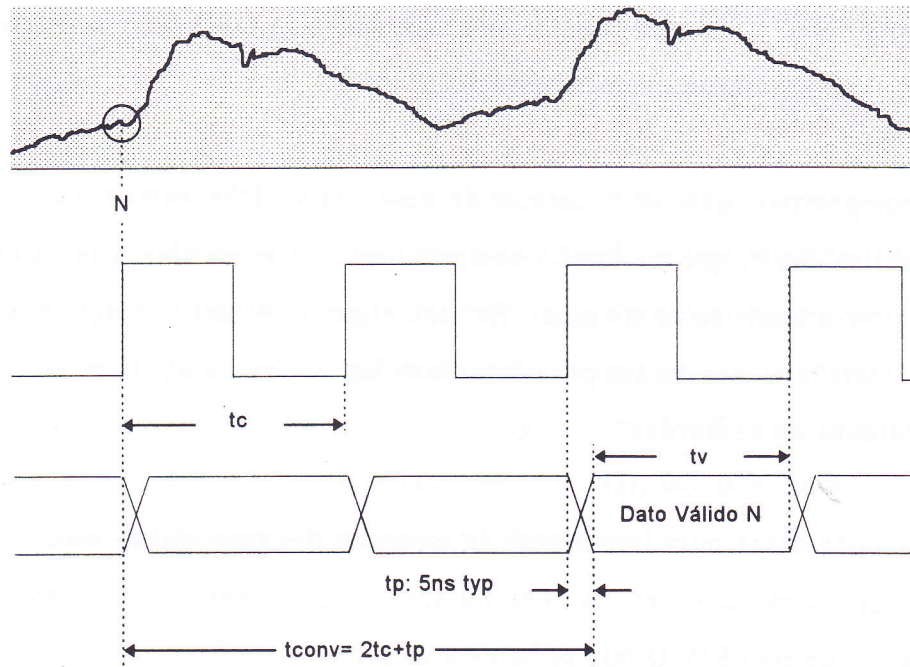


Figura 5.3 Diagrama de Tiempos del Convertidor.

El periodo de muestreo es 70.4845 ns. Por lo tanto, el tiempo de retardo que corresponde a t_{conv} será,

$$t_{conv} = 2t_c + t_p = 2(70.4845) + 5 \text{ ns} = 145.969 \text{ ns.}$$

(5.1)

Este tiempo de retardo se propaga a lo largo de cada conversión, de cada línea, y de cada imagen. De este modo este tiempo ó t_{conv} , ó delay1, será el primer retardo de la "pipeline".

5.3 Convolución con G1 (BCG1)

Este bloque se repite cuatro veces en la Configuración Final. La velocidad de proceso en BCG1 es crítica y determina el tamaño máximo de la fracción de la

imagen que será tratada, por tanto, el procesador digital de señal que se escoga influirá de manera directa en ésta característica del sistema.

El dispositivo que se encargue de esta etapa debe ser de características especiales, es decir, que su diseño se aproxime lo mas posible a la tarea para la que será implantado en el sistema. Existen algunos SIMD ("Single Instrucción Multiple Data") desarrollados para esta finalidad, inclusive algunos son ofertados como "detectores de bordes".

Por considerar muy importante la elección del dispositivo electrónico que calcule las convoluciones, a continuación se presenta un breve análisis comparativo de tres SIMD que se han estudiado en esta Tesis.

5.3.1 Procesador PDSP16401

Un claro ejemplo de SIMD es el Detector de Bordes en Dos Dimensiones Plessey PDSP16401A. Este dispositivo calcula una buena aproximación de la primera derivada derivando en cuatro direcciones diferentes y quedándose con el valor máximo. De esta manera obtiene la dirección de máximo cambio. Uno de los inconvenientes consiste en que las máscaras son fijas y de tamaño tres por tres.

En un principio se intentó implementar el detector de bordes con este dispositivo. sin embargo, sin ser deficiente, no se ajusta a característica de kernel variable imprescindible en el algoritmo antes discutido. No obstante si se hubiera empleado como aproximación de la primera derivada, hubiera sido necesario implementar un bloque previo que calcule el gaussiano de la imagen. Esta etapa previa debería haber tenido una longitud tal que no truncara de manera sensible el filtro. Esto sólo es posible agregando un bloque previo que

haga esto, y que además sea capaz de convolucionar tamaños de filtro grandes a gran velocidad.

El costo que implica un bloque adicional no justifica el supuesto ahorro en proceso que se pueda obtener. Por otro lado, al aplicar máscaras de tres por tres se puede introducir componentes en frecuencia que no existían inicialmente en la imagen. De cualquier manera, en el capítulo cuatro se muestra una simulación del resultado que se obtendría - ver la Configuración D y E- con este dispositivo (Figuras 4.16 y 4.17) . Puede apreciarse con claridad que para desviaciones grandes se introducen bordes fantasmas no deseados.

5.3.2 Procesador ZR33771-45

Este DSP de Zoram es un SIMD cuya descripción comercial es "Two-Dimensional Convolver". Este DSP puede ser configurado con diferentes valores previamente programados, ya sea con simetría horizontal, vertical, ó ambas. Su máxima frecuencia de salida es 45 MHz y soporta tamaños de máscara desde 2x2 hasta 7x7 simétricos y asimétricos. Admite dos bancos de coeficientes que se pueden intercambiar on line.

El ZR33771-45 tiene un acumulador de 24 bits y un bus de salida de 16 bits. El tamaño máximo de los coeficientes es de 9 bits, y el de datos es de 8 bits con la opción de signo. Además tiene un post-procesador que permite operaciones de desplazamiento, truncamiento y redondeo al dato de salida. Está construido con tecnología CMOS, y es TTL compatible

En la Tabla 5-2 se observa las características del DSP. Un ligero análisis permite concluir que el empleo de este dispositivo permite hacer convoluciones con máscaras de tamaño 5x5 a 15MHz, sin embargo, para tamaños de 7x7 la

velocidad de procesamiento se reduce a 11.25 MHz, lo que obligaría a procesar parte de la imagen digitalizada con la consiguiente pérdida de información.

Otro inconveniente que presenta este dispositivo es que -al menos en la información técnica no lo menciona- no permite su conexión en cascada, lo que obliga a incluir lógica asociada adicional para lograrlo.

Tabla 5-2 Tamaño de Máscara, Simetría y Velocidad de Muestreo.

MO[2:0]	Tamaño de Kernel	Simetría de Kernel	Velocidad en MHz
000	2x2	No	45.00
010	3x3	Si	45.00
001	3x3	No	15.00
110	4x4	Si	22.50
011	4x4	No	11.25
100	5x5	Si	15.00
101	6x6	Si	15.00
111	7x7	Si	11.25

Por último, necesita un Bloque de Memoria para almacenar los coeficientes y un secuenciador de reloj para desencadenar la carga de los mismos. Asimismo, las líneas de retardo que se deben almacenar para realizar la convolución deben ser almacenadas en un segundo bloque de memoria adicional fuera del dispositivo. A esto hay que sumar la necesaria lógica asociada y los multiplexores externos necesarios para alimentar las líneas.

Este dispositivo puede ser empleado para calcular las convoluciones. Sin embargo, hay que considerar sus limitaciones tanto de velocidad como de funciones integradas en el mismo chip.

5.3.3 Procesador PDSP16488

El PDSP16488 es un SIMD de aplicación específica para tratamiento de imágenes. Este dispositivo realiza una convolución en dos dimensiones de los pixels contenidos dentro de una ventana, con un conjunto de *coeficientes previamente almacenados*. Un array acumulador multiplicador permite las opciones listadas en la Tabla 5-3.

Tabla 5-3 Configuraciones de Un Solo Dispositivo.

Tamaño del Dato	Tamaño de la Ventana		Máxima Velocidad del Pixel	Líneas de retardo
	Ancho	Profundidad		
8	4	4	40 MHz	4x1024
8	8	4	20 MHz	4x1024
8	8	8	10 MHz	8x512
16	4	4	20 MHz	4x512
16	8	4	10 MHz	4x512

Para considerar los datos de esta tabla conviene recordar que la velocidad de muestreo del convertidor es de 14.1875 MHz. La zona tenuemente sombreada corresponde al tamaño de los datos que vienen del convertidor. dentro de esta zona se ha sombreada una más oscura para indicar la velocidad de 20MHz inmediatamente superior a la requerida. Sin embargo la otra característica, el tamaño de la ventana, no se ajusta a los requerimientos mínimos del sistema que se está diseñando. Nótese que el tamaño de la ventana no es simétrico.

El Procesador posee una RAM interna del 32k bits que puede ser configurada para proporcionar cuatro u ocho líneas de retardo. La longitud de

cada retardo puede ser programado según los requerimientos del usuario hasta un máximo de 1024 pixels/línea. Los retardos de línea se organizan en dos grupos, los cuales pueden ser internamente conectados en series ó pueden configurarse para aceptar entradas separadas de pixels. Esto le permite soportar operaciones con vídeo en modo entrelazado y no-entrelazado.

Los coeficientes de ocho bits son internamente almacenados y pueden ser descargados de un ordenador host ó de una EPROM. Para soportar una EPROM no es necesario incluir lógica adicional y un solo dispositivo puede soportar hasta 16 "convolvers".

El Procesador contiene un sumador de expansión y un retardador de red de trabajo que le permite ser conectado en cascada con otros procesadores del mismo tipo.

Una *precisión intermedia* de 32 bits es dispuesta para evitar cualquier pérdida de información por overflow, pero el resultado final no ocupará todos estos bits. Sin embargo, se ha previsto un multiplicador en la ruta de salida para permitir alinear el resultado del bit más significativo hasta un total de 32 bits.

Este DSP cubre las necesidades del sistema tal como se puede leer en la Tabla 5-4. Incluso se podrían implementar máscaras de 15x15 a 20MHz, pero esto requeriría un excesivo número de "convolvers", ocho en este caso. Si se tiene en cuenta que este bloque aparece cuatro veces, se necesitarían 32 DSPs para una máscara de estas dimensiones, lo cual resulta tremendamente excesivo. Sin embargo, para tamaños de máscara de 5x5 y 7x7 el número de DSPs necesarios se reduce a 2, con lo que el total se reduciría a 8. Este número de procesadores con ser grande. no es excesivo.

Tabla 5-4 Dispositivos Necesarios para Diferentes Ventanas

Velocidad Max. de Pixel (MHz)	Tamaño del pixel	Tamaño de la Ventana						
		3x3	5x5	7x7	9x9	11x11	15x15	23x23
10	8	1	1	1	4	4	4	9
10	16	1	2	2	-	-	-	-
20	8	1	2	2	6	6	8	-
20	16	1	4	4	-	-	-	-
40	8	1	4*	4*	-	-	-	-
40	16	2	-	-	-	-	-	-

* La velocidad máxima está limitada a 30 MHz por la expansión de las líneas de retardo.

De los tres dispositivos DSP vistos anteriormente el que más prestaciones ofrece es el PDSP14188. Además, su arquitectura permite, comparada con los otros, ampliar el tamaño del kernel si ese fuera el caso. Por esto, se concluye que este DSP es la mejor opción, por el momento, para desarrollar este bloque. Se prevé que procesadores más rápidos aparezcan en los próximos años, pero este dispositivo, con tamaño máximo de máscara de 15x15 bien puede soportar unos años.

Como se propone este procesador para calcular las convoluciones del sistema, en el Apéndice C se pasa a una descripción más detallada del mismo por ser este bloque el más importante.

Para poder procesar en tiempo real es necesario que el sistema trabaje a la máxima velocidad permitida. Aunque uno de los objetivos iniciales fue el de proyectar un sistema que sea capaz de procesar una imagen de 512x512 pixels, los trabajos han permitido obtener un sistema que podrá trabajar con tamaños de línea de hasta 1024x1024.

Como el reloj del sistema es de 14.1875MHz, y es el mismo que se emplea tanto para digitalizar la imagen como para procesarla en el DSP, se debe configurar éste a una frecuencia de trabajo que sea igual ó mayor que 14.1875Mz.

Si se consulta la Tabla 5-3, que corresponde a las posibles configuraciones para un solo dispositivo, se verá que la única posibilidad que tiene el DSP para procesar a una frecuencia de pixel mayor que la 14.1875Mz corresponde a una ventana máxima de 8x4, para un tamaño de dato de 8 bits. Esto significa que para poder configurar un tamaño de ventana máximo de 8x8 se necesitarán dos DSPs dispuestos en cascada y configurados cada uno con ventanas de 8x4, dando entre los dos los 8x8 necesarios (ver Figura 5.4).

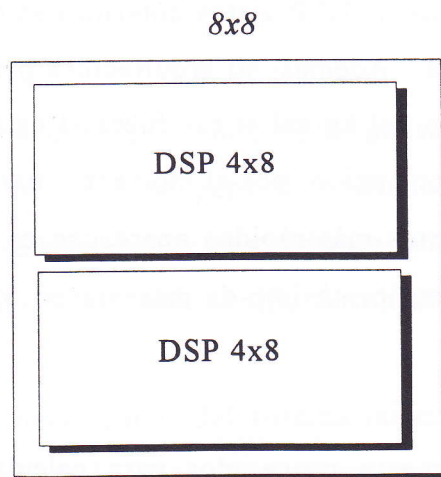


Figura 5.4 Ventana de 8x8 Configurada con 2 DSPs.

Por otro lado, obsérvese que la opción elegida, la zona más oscura en la Tabla 5-3, soporta 4 líneas de 1024 pixels de longitud almacenadas dentro del DSP. Esto simplifica notablemente el circuito y permite procesar líneas cuyo tamaño sea igual ó menor a 1024. Téngase en cuenta que la configuración elegida tiene 2 DSPs, luego la capacidad de almacenamiento de líneas se duplica haciendo un total de 8 líneas de 1024 pixels cada una. Este número de líneas es

precisamente el necesario para hacer convoluciones en tiempo real con ventanas de 8x8 coeficientes.

La Tabla 5-4 corrobora el razonamiento anterior al expresar que para ventanas simétricas de coeficientes de 5x5 y 7x7, el número necesario de dispositivos es de dos unidades. En la misma tabla se ha sombreado la opción elegida.

Como la configuración elegida trabaja a la frecuencia de 20MHz, y por tanto, tarda en calcular un nuevo dato 50ns, y la velocidad con que llegan los datos es de uno cada 70.4845 ns, no se generarán retardos significativos aparte del tiempo empleado en calcular el primer pixel, y el que corresponde a la espera de las N primeras líneas necesarias para iniciar la convolución.

Estos tiempos son ideales para procesar señales de vídeo no-entrelazado, no obstante, como se desea obtener la máxima resolución, y ésta se da para señales de vídeo compuesto en modo entrelazado, necesariamente se introducirán retardos debidos a que el primer campo debe ser almacenado, en lo que se llamará Retardo de Campo Externo, y esperar a que el segundo llegue e ir introduciendo líneas del campo impar y del par alternativamente al DSP.

Un campo aparece a la salida de la cámara cada 20ms, es decir a 50Hz, por lo tanto, el primer retardo introducido es el debido al almacenamiento en la memoria de Retardo de Campo Externo, y que son exactamente 20ms.

Por otra parte, las 8 primeras líneas del segundo campo tardan en llegar $8 \times 64 \mu\text{s}$, es decir 0.512ms, que sumados a los 20ms dan un total de 20.512ms.

Este tiempo de retardo que pudiera parecer excesivo, no lo es tanto porque es el tiempo que tardan en comenzar a salir los datos de la primera convolución. Sin embargo, este retardo se va absorbiendo a medida que empieza a llegar el

segundo campo, de tal manera que cuando ha llegado el último pixel de la última línea del segundo campo, quedan solamente por efectuarse kernel-1 operaciones de convolución, las que se efectuarán en $\text{kernel} \times 64 \mu\text{s}$.

Si se emplea un kernel de siete, el retardo introducido será $448 \mu\text{s}$. Esto quiere decir que $448 \mu\text{s}$ después de que haya llegado el último pixel de la imagen, saldrá el último pixel de otra imagen digitalizada correspondiente a la convolución de la primera con la máscara programada. A éste tiempo se llamará delay2.

7						7	0
	5					5	0
		3		3			0
			⊕				0
		3		3			0
	5					5	0
7						7	0
0	0	0	0	0	0	0	0

Figura 5.5 Distribución de coeficientes para ventanas de 7,5, y 3.

Obsérvese en la Figura 5.5 que para ventanas más pequeñas que 7×7 , es decir para 5×5 ó 3×3 , no es necesario reconfigurar el bloque, basta con cambiar los coeficientes, y rellenar con ceros los lugares de la ventana de 7×7 que no tienen coeficientes. De este modo, se configura para que siempre realice convoluciones de 7×7 .

Para la interconexión de los DSPs se ha empleado un circuito propuesto por el fabricante precisamente para una aplicación que coincide con las necesidades planteadas para este bloque funcional.

5.4 Bloque Mayor en Valor Absoluto (BMVA)

La tarea que tiene que desempeñar este bloque es la de comparar dos números y dar siempre a la salida el valor del mayor en valor absoluto. Como ya se comentó en la Sección 4.3.3, se trata de una simple pastilla integrada que por diseño desempeña la tarea antes mencionada. Para lograr el objetivo trazado simplemente hay que conectar a esta pastilla todos los bits de las palabras a comparar a excepción de los bits de signo. De esta manera la comparación y el resultado se hará siempre en valores absolutos.

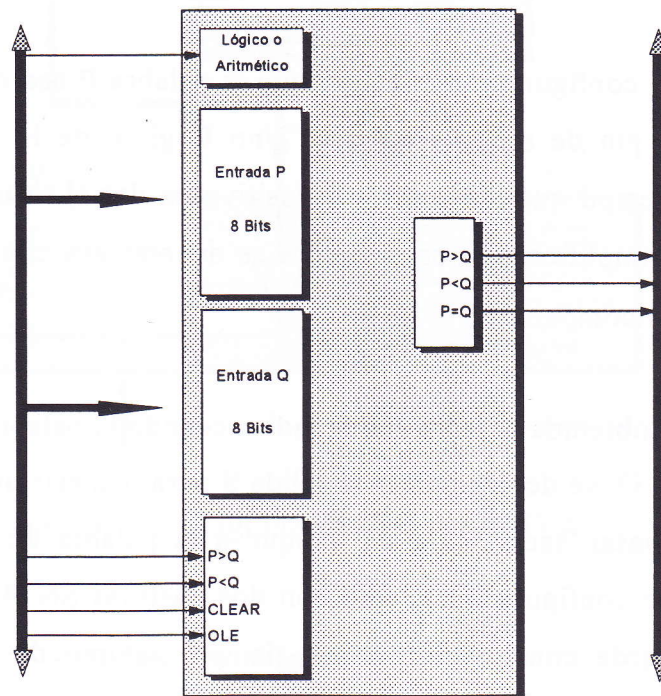


Figura 5.6 Diagrama de Bloques de las Conexiones del Comparador.

El dispositivo electrónico elegido es un circuito integrado de Texas Instruments denominado SN74AS866. Este dispositivo es capaz de realizar comparaciones aritméticas -considerando el signo- y lógicas a alta velocidad sobre palabras de 8 bits, ó sobre sus complementos de 2. A las palabras de

entrada se les denomina con las letras P y Q. De este modo, a la salida del comparador se tienen tres salidas que indican si $P > Q$, $P < Q$ ó $P = Q$. Es posible configurar el dispositivo para que una de las tres salidas se active, $P > Q$, $P < Q$ ó $P = Q$, cuando se verifica la condición de igualdad.

Este dispositivo debe ser configurado de la siguiente manera para que cumpla con los requerimientos del presente bloque:

- Realizar la comparación lógica. Se debe poner el pin L/A a 0 lógico.
- Que el pin de salida $P > A$ sea igual a 1 lógico cuando se cumpla la condición $P \geq Q$ en las palabras de entrada. Se debe conectar el pin de entrada $P > Q$ en 1 lógico y $P < Q$ a 0 lógico.

Con esta configuración, siempre que la palabra P sea mayor ó igual que la palabra Q, el pin de salida $P > Q$ será Uno Lógico, de lo contrario será Cero Lógico. El tiempo que tarda este dispositivo en dar el resultado deseado es de 13 ns. Para simplificar la nomenclatura se denominará con la letra S al pin de salida $P > Q$ del comparador.

Una vez obtenida la señal S que indica cuando la palabra P es mayor ó igual que la palabra Q, se debe emplear la salida S para manejar un conmutador lógico que permita pasar hacia fuera del bloque a la palabra de mayor valor. Este conmutador se configura fácilmente con dos pastillas SN74AS240, que realizan una operación de conmutación en un tiempo máximo de 9.5 ns, y una llave inversora NOT.

La unión del diagrama de bloques del circuito comparador mas el del circuito conmutador da como resultado el diagrama de bloques final para este Bloque del sistema procesador. La Figura 5.7 muestra el diagrama de bloques resultante.

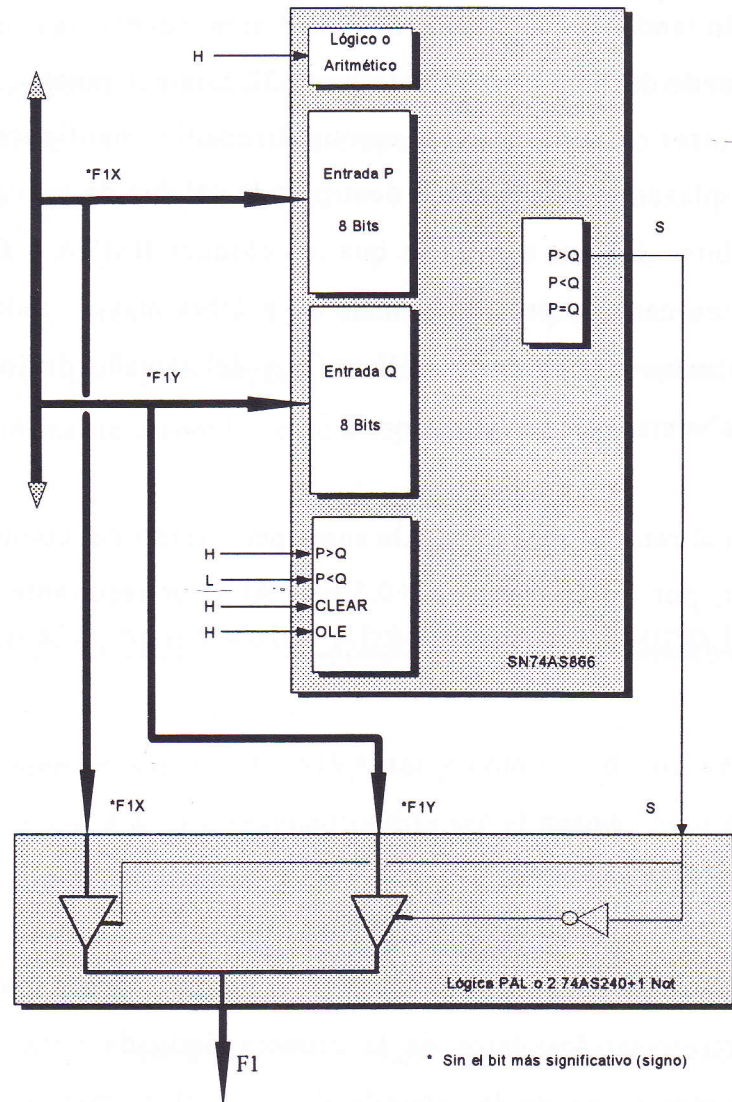


Figura 5.7 Diagrama del Bloque BMVA.

Compárese este y todos los diagramas del presente capítulo con los correspondientes a los mismos bloques del capítulo anterior. Se verá que el criterio seguido es el de considerar cada bloque como una caja negra que introduce un retardo a la pipeline y produce un resultado diferente a su entrada. Esta manera de trabajar permite, dado el caso, reemplazar la electrónica de una parte, ó de la totalidad de un determinado bloque, por otra tecnología más eficiente.

Por otra parte señalar que se están considerando palabras de 8 bits para las derivadas, y sin embargo la salida del DSP que se encarga de calcular la convolución, puede dar una precisión de hasta 32 bits por pixel, como se vio en secciones anteriores. Es más, cuenta con un dispositivo multiplicador que hace las veces de desplazador del resultado dentro de la palabra de salida dentro de un máximo de 32 bits. Lo que sucede es que los bloques BMVA y BMAS pueden ser conectados en cascada para un tamaño de palabra mayor, todo depende del bus de la plataforma que se emplee ("host"), y del tamaño de los coeficientes empleados en la ventana.

En cuanto al retardo será igual a la suma del retardo del comparador más el del conmutador, por lo tanto será $13+9.5$ ns. Al valor resultante, 22.5 ns se le ha denominado delay3.

5.5 Bloque Memoria Buffer (BMBF)

Este bloque es un banco de memoria RAM del tipo "first input first output" que sirve para retrasar los datos de la primera derivada para que no exista desfase entre éstos y los de la segunda derivada al momento de ingresar al bloque BDLC.

El número de desplazamientos entre F1 y F2 viene determinado por delay2. Esto se deduce al considerar que el último punto común de las señales es a la salida del bloque BCG1, y por lo tanto F1 empieza a estar disponible para el bloque detector de ceros $\text{delay3}=22.5$ ns.

Sin embargo, considérese los tiempos de F2. Para que comience a estar disponible para la comparación tarda delay2 segundos que proviene de la segunda convolución, más delay3 que corresponde al retardo del bloque BMAS.

Esto quiere decir que la diferencia de tiempos es solamente delay_2 , que equivale a $448\mu\text{s}$. Habrá que introducir una cantidad de posiciones de memoria que introduzcan un retardo equivalente más el tiempo interno de retardo necesario para que el bloque BDLC empiece a disponer de F1, siendo éste, considerando la lógica interna, igual a 2 veces delay_3 .

Por tanto el tiempo que debe retener este bloque a los valores de F1 es igual a $448.045\mu\text{s}$. para lograr este objetivo puede emplearse las memorias citadas anteriormente ó simplemente Desplazador de Registros "shift register".

5.6 Bloque Mayor en Valor Absoluto más Signo (BMAS)

Este bloque es similar al BMVA tal y como se puede ver en el capítulo anterior, Sección 4.4.4. En realidad podría ser el mismo, pero para mantener la independencia de los bloques se ha considerado éste como un bloque adicional.

Este bloque recibe la misma información que BMVA con la diferencia de que a éste se le suministra también el signo de cada palabra. En la Figura 4.21 se puede ver que el único cambio introducido respecto a la Figura 4.22 consiste en introducir al conmutador dos bits adicionales que corresponden a los signos de las palabras de entrada. Estos cambios se muestran en la Figura 5.8.

Físicamente esto se traduce en la inclusión de lógica adicional en la PAL que se encarga de la conmutación, ó en la adición de dos buffers adicionales, uno para cada signo. La parte correspondiente a la comparación permanece sin variación. Para clarificar la exposición se incluye a continuación el diagrama de bloques adicional del hardware para esta parte del sistema.

El retardo introducido por este bloque es denominado como delay_4 y

equivale al mismo valor que tiene delay3, es decir, 22.5 ns.

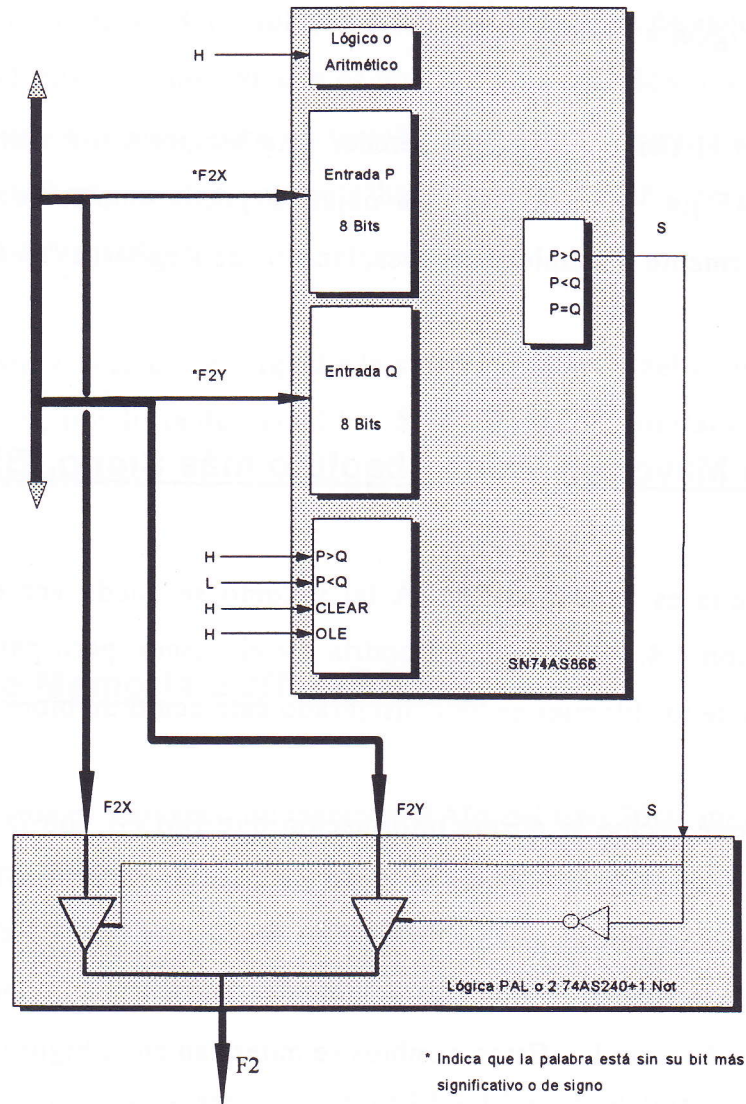


Figura 5.8 Diagrama del Bloque BMAS.

5.7 Bloque Ceros (BDLC)

Para detectar los pasos por cero se emplea la matriz de búsqueda definida en el capítulo anterior como:

$$C = \begin{bmatrix} \zeta & la_y & \zeta \\ la_x & lb & \zeta \\ \zeta & \zeta & \zeta \end{bmatrix} = \begin{bmatrix} \zeta & la_y \\ la_x & lb \end{bmatrix}$$

(5.2)

Esta matriz que fue definida en el capítulo anterior como una matriz de 3x3 elementos para indicar que el punto actual sobre el que se estaba buscando el paso por cero era el punto b con un valor de intensidad igual a lb, se redefine ahora a su verdadero tamaño. Este tamaño es de 2x2, donde los elementos que realmente interesan son las intensidades de los puntos b, a_x y a_y, que son lb, la_x y la_y respectivamente. Los valores lb, la_x y la_y, no son más que simples valores de la segunda derivada F2 que vienen del bloque anterior.

Físicamente este bloque está formado por comparadores y conmutadores del mismo tipo de los que se han descrito en este capítulo, por lo tanto se conoce el retardo que introduce cada uno, y se puede calcular el retardo global del bloque.

Como la búsqueda de ceros se hace en las dos direcciones ortogonales paralelas a los lados de la imagen es necesario duplicar este bloque, una vez para cada dirección. Esto no introduce un retardo adicional al que ya introduce un solo bloque porque el nuevo bloque trabaja en paralelo con el anterior. Es más, no se necesita tampoco un control lógico adicional porque ambos bloques intentan saber si existe un punto borde en el punto b de la matriz de búsqueda, y por lo tanto, las salidas de cada bloque en particular están unidas de tal manera que si uno ha detectado que b es un punto borde la salida será inmediatamente puesta a 1 lógico. El retardo introducido por este bloque equivale a la suma de tres veces el retardo delay2, por lo tanto será 67ns.

La Figura 5.9 muestra la ubicación de los comparadores y conmutadores, ó switch digitales, que se encargan de analizar si b es ó no un punto borde.

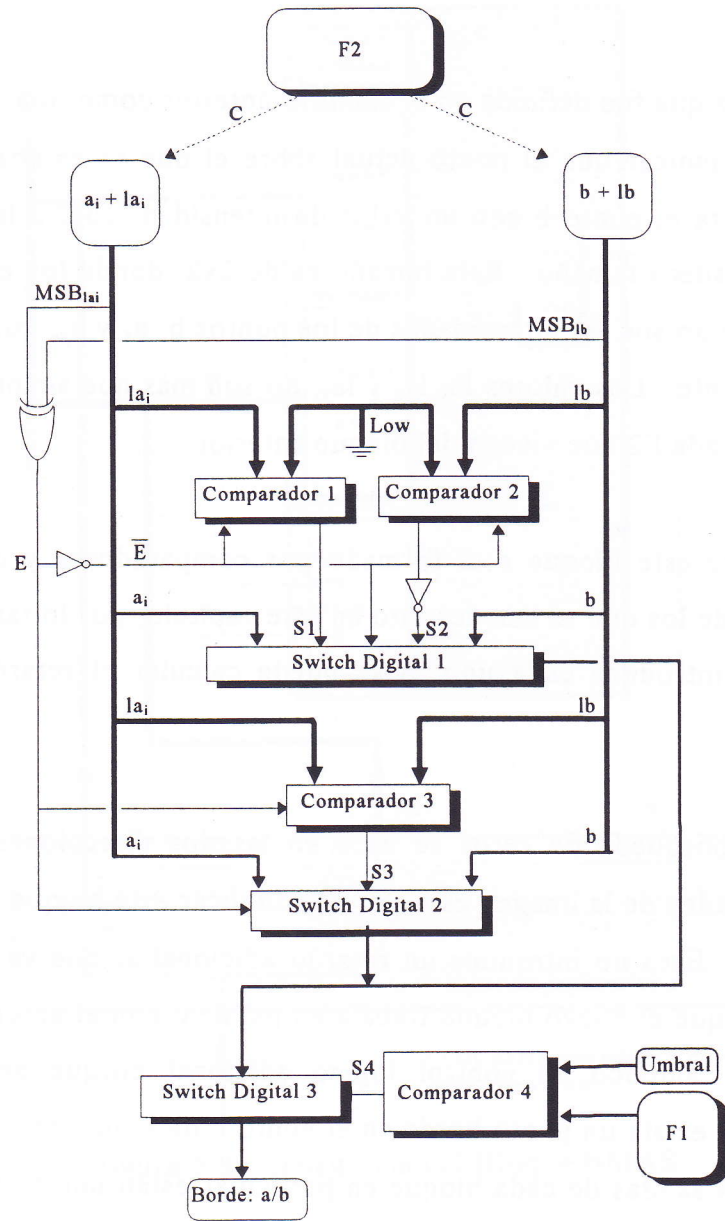


Figura 5.9 Diagrama del Bloque BDLC.

5.8 Retardo Acumulado

El retardo acumulado viene a ser el tiempo después del cual, terminada de ingresar la imagen completa, ha terminado de salir la imagen correspondiente al mapa de bordes en la salida del sistema general. Este tiempo es el que han introducido los bloques que se ha encargado de procesar la señal y no equivale a la suma de los tiempos parciales, ya que se han procesado en paralelo algunas tareas. Por lo tanto existen tiempos que están solapados. En la Figura 5.10 se muestra el Diagrama de retardos calculado con los retardos parciales introducidos por los componentes descritos en el presente capítulo.

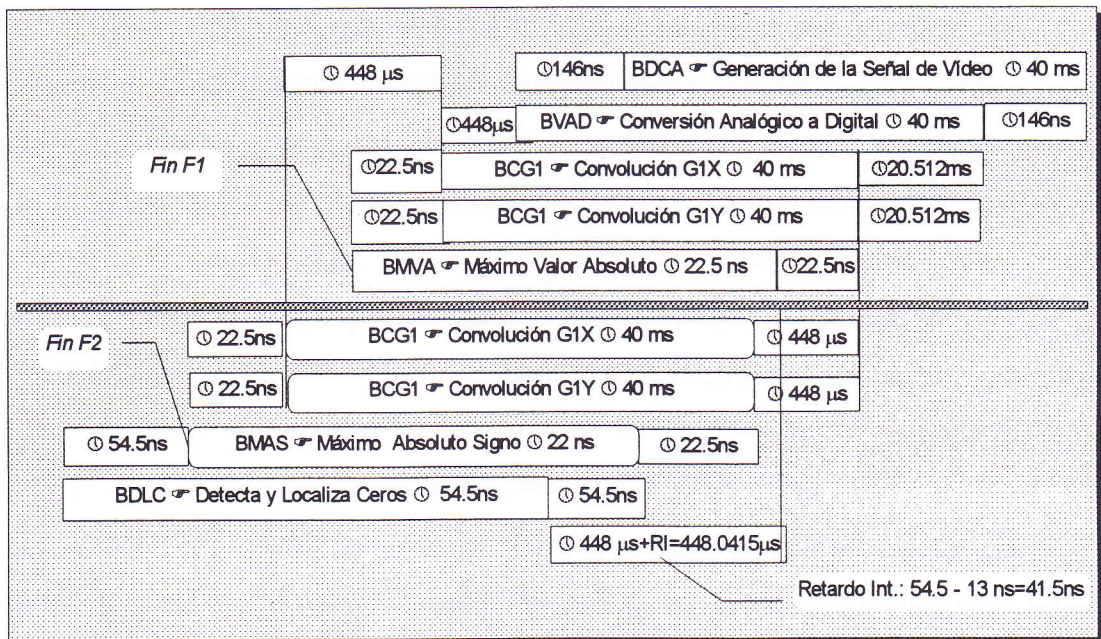


Figura 5.10 Diagrama de Retardos

Estos tiempos pueden variar si se emplean componentes fabricados con otras tecnologías. De cualquier manera, los cálculos se han realizado considerando las tecnologías más accesibles en el mercado, y señales digitales compatibles TTL. Si por alguna razón se desea introducir, para una parte ó la totalidad de la lógica del sistema, dispositivos lógicos programables, se deberá

recalcular los tiempos de la "pipeline", partiendo del diagrama presentado.

Se puede ver en el diagrama de retardos que el tiempo transcurrido desde que comienza a llegar la trama de la señal de vídeo hasta que comienza a salir la imagen de bordes es la suma de los tiempos de la parte derecha del diagrama, es decir:

$$t_i = 146 \times 10^{-9} + 20.51 \times 10^{-3} + 448 \times 10^{-6} + 22.5 \times 10^{-9} \text{ segundos.}$$

$$t_i = 20.96 \text{ ms.}$$

(5.3)

Este resultado era esperado porque el sistema debe esperar a que llegue el segundo campo para empezar a desarrollar los cálculos correspondientes al mapa de bordes, y esto ocurre 20 ms después de haber empezado a llegar el primer campo.

Como el segundo campo empieza a llegar a los 20 ms después de haber empezado a llegar el primero, el tiempo real de retraso en este punto es tan sólo 960 μ s, ó 0.96ms.

Si se analiza el mismo diagrama desde otro punto de vista, se verá que el tiempo transcurrido desde que termina de llegar la señal de vídeo hasta que termina de salir la imagen de bordes es equivalente a la suma:

$$t_s = 146 \times 10^{-9} + 448 \times 10^{-6} + 448 \times 10^{-6} + 22.5 \times 10^{-9} + 54.5 \times 10^{-9} \text{ segundos,}$$

$$t_s = 896.223 \times 10^{-6} = 0.896 \text{ ms.}$$

(5.4)

Puede verse que t_s es menor que 1ms. Si se observa detenidamente el diagrama de la Figura B-20 del Apéndice B, se verá que el tiempo nominal

transcurrido entre cada campo es igual a t_1 , de la tabla B-1 se obtiene que su valor es igual a 1.6ms.

Este resultado significa que la imagen correspondiente al mapa de bordes estará lista antes de que comience a llegar la siguiente imagen original. Es más, quedan:

$$t_p = 1.6 \text{ ms} - 0.896 \text{ ms} = 0.704 \text{ ms}, \quad (5.5)$$

para realizar cálculos de posprocesamiento, si ése fuera el caso. Sin embargo, el tiempo efectivo de que se dispone para ese eventual posprocesamiento es aún mayor, dado que, como en el caso del seguimiento de trayectorias, no es necesario que termine de salir la imagen de bordes para que ésta pueda empezar a ser tratada por algún algoritmo de "tracking", sino que se prolonga la arquitectura "pipeline" de tal manera que t_1 segundos después de haber empezado a llegar la imagen original, que tarda 40 ms en pasar íntegramente por el sistema, podrá empezarse ese posprocesamiento, es decir, 20.96 ms después.

Se habrá notado que estos cálculos consideran la trama completa de la imagen digitalizada, sin embargo, como sucede en la mayoría de los casos en los sistemas de visión tradicionales, si el tamaño de imagen es de 512x512 pixels, la disponibilidad de tiempo útil para posprocesamiento aumenta de manera notable. Esto demuestra la efectividad de la arquitectura de hardware planteada en la Configuración Final.

También se puede observar que el dispositivo que introduce mayores retardos al sistema es, como era de esperar, el encargado de calcular las operaciones de convolución. Si, como se espera, en el futuro se aplica la misma arquitectura a sistemas basados en dispositivos más rápidos, y se emplea un DSP que pueda calcular las convoluciones a mayor velocidad, el ahorro de tiempo

puede ser muy significativo. Sin embargo, como el elemento determinante es el DSP, normalmente bastará con cambiar el DSP para obtener un rendimiento superior. Téngase en cuenta que mientras el DSP introduce retardos del orden de los milisegundos, el resto de los bloques los hace del orden de los nanosegundos ó microsegundos en el peor de los casos.