UNIVERSIDAD DE SONORA

DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE INVESTIGACION EN FISICA
ING. EN TECNOLOGÍA ELECTRÓNICA

"IMPLEMENTACIÓN DE FILTROS DIGITALES PARA EL PROCESAMIENTO DE PATRONES DE ILUMINACIÓN"

T E S I S

PARA OBTENER EL TÍTULO DE:
INGENIERO EN TECNOLOGÍA ELECTRÓNICA

PRESENTA:
RAUL ALEJANDRO PALAFOX VALENCIA

DIRECTORES DE TESIS:
DR. LUIS GONZÁLEZ LÓPEZ
DR. JOSE RAFAEL BENITO NORIEGA LUNA

Hermosillo, Sonora, México Marzo del 2007

Universidad de Sonora

Repositorio Institucional UNISON

UNIVERSIDAD DE SONORA

DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE INVESTIGACION EN FISICA
ING. EN TECNOLOGÍA ELECTRÓNICA

"IMPLEMENTACIÓN DE FILTROS DIGITALES PARA EL PROCESAMIENTO DE PATRONES DE ILUMINACIÓN"

TESIS
PARA OBTENER EL TÍTULO DE INGENIERO EN TECNOLOGÍA ELECTRÓNICA

PRESENTA:
RAUL ALEJANDRO PALAFOX VALENCIA

DIRECTORES DE TESIS:
DR. LUIS GONZÁLEZ LÓPEZ
DR. JOSE RAFAEL BENITO NORIEGA LUNA

Hermosillo, Sonora, México Marzo del 2007

Este trabajo lo dedico

. . . A Dios por darme la oportunidad de vivir esta experiencia y llenarme de sabiduría en toda el ciclo de mi carrera.

. . . A Joaquín Alberto Corella Vega compa, amigo, hermano donde quiera que estés "Te estaremos esperando siempre".

. . . A mi Madre, Nana, Padre y mi Tata por su eterno apoyo en todos los ciclos de mi vida.

Agradecimientos.

A CONACYT por el apoyo brindado con la beca de tesis del proyecto 44071-A1 "Investigación y desarrollo de sistemas óptico-digitales para el proceso de información tridimensional"

A los maestros de la carrera de Ingeniería en Tecnología Electrónica de la Universidad de Sonora por estar siempre cuando se les necesitaba y apoyándonos en todos nuestros objetivos en especial al Dr. Luis Gonzales, Dr. Benito Noriega, Dr. Fernando Mendoza, Dr. Alicia Vera.

A mis amigos de la carrera por todos los momentos que compartimos en especial a Temis, Roberto y Chema.

A mi Familia por confiar en mí y ayudarme a salir adelante siempre en varios aspectos de mi vida.

A mis Bandas de Guerra sub campeones nacionales que para mí siempre serán las mejores gracias por darme la oportunidad de pasar todos esos momentos de mi vida inolvidables y sobre todo gracias por creer en mí prestarme la confianza en ser parte de ustedes mis niños SEC. 4 LINCES y los mas grandecitos LEGION SERI recuerden "la unión hace la fuerza".

A mis amigos de mi vida por el apoyo como siempre en especial a Paul Durazo, Jorge Mendoza, Martin Bermúdez, Adrian Pacheco y a todos los wasibiris.

A Selegna Melendrez por su apoyo incondicional creer que puedo lograr cualquier objetivo de mi vida gracias.

Tabla de contenido

LISTA DE FIGURAS

CAPÍTULO 1

1.- INTRODUCCIÓN

En los últimos años el procesamiento digital de imágenes ha sido ampliamente utilizado en diversas disciplinas tales como: Medicina, Biología, Física e Ingeniería.[2] Mediante el procesamiento digital de imágenes es posible manipular imágenes digitales en una computadora con el fin de obtener información objetiva de la escena captada por una cámara. Son dos las tareas fundamentales del Procesamiento Digital de Imágenes:

- Mejoramiento de una imagen digital con fines interpretativos
- Toma de decisiones de manera automática de acuerdo al contenido de la imagen digital

Como aplicaciones típicas se pueden mencionar: detección de presencia de objetos, inspección visual automática, medición de características geométricas y de color de objetos, clasificación de objetos, restauración de imágenes y mejoramiento de la calidad de las imágenes.

En este trabajo de tesis se aplica el procesamiento de imagenes para mejorar los patrones de luz en imágenes de tipo láser obtenidas en dos dimensiones. Estos patrones de luz láser son muy dispersivo por lo que estas imágenes requieren ser procesadas o filtradas para eliminar ese tipo de ruido de la imagen.

Estos filtros se aplican por medio de un FPGA ya que se requiere de su procesamiento en tiempo real.

En esta tesis se plantea el diseño de filtros digitales para patrones de iluminacion láser empleando FPGA mediante bloques de Simulink Xilinx los cuales realizan el método de convolución de mascara.

en la figura 1.1 se muestra un diagrama de bloques del sistema en general

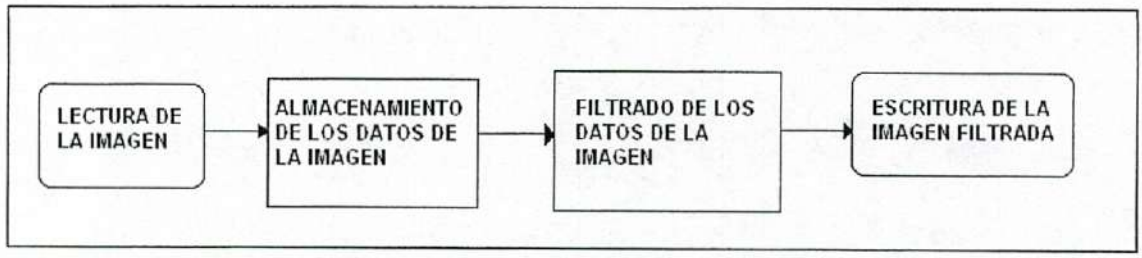


Figura 1.1 Diagrama de bloques de sistema en general

1.1 Organización del documento

El CAPÍTULO 2 de esta tesis trata sobre sistemas tanto de una dimensión como bidimensionales, también de la representación en dos dimensiones de una imagen, de la linealidad de los sistemas y la convolución, haciendo énfasis en la convolución espacial ya que es la que se va implementar.

En el CAPÍTULO 3 se habla del diseño de filtros digitales para procesar imágenes, se explican y se da un ejemplo tomando una referencia de cada uno de estos filtros entre los que podemos mencionar: sobel xy, sobel x, sobel y, detector de bordes, gaussian, smoothing, blurring y sharpean.

En el CAPÍTULO 4 se habla de la implementacion del sistema de filtrado, de la implementación con simulink, lectura de la imagen, almacenamiento de la información, filtrado de la información y escritura de la información ya filtrada.

CAPÍTULO 2

2.- FILTRADO DIGITAL

2.1 Sistemas.

Un sistema es una interconexión entre dispositivos que aceptan una o mas señales de entrada y funciona sobre ellos para producir una o mas señales de salida. En este caso los filtros son un ejemplo de sistema usado en aplicaciones de procesamiento de señales. Para ver esto más claro representamos al sistema como H y denotamos las señales "x" de entrada y "y" de salida. Para ilustrarlo podemos ver la figura 2.1[5].

$$y = H[x] \tag{1}$$

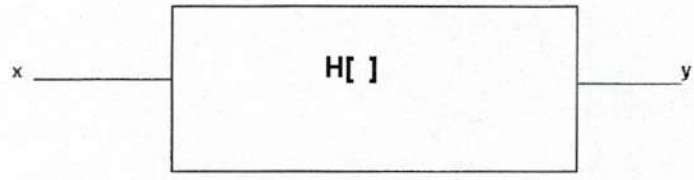


Figura 2.1 bloque de un sistema con entrada x y salida y.

2.1.1 Linealidad de un sistema.

Se dice que un sistema es lineal cuando su respuesta producida aplicación al mismo tiempo de dos funciones de diferentes entradas es la suma de las dos respuestas individuales, es decir para el sistema lineal la respuesta debida a varias entradas se calcula tratando una entrada a la ves y sumando los resultados. Este principio recibe el nombre de "superposición". [10].

2.1.2 Representación de una imagen espacial en dos dimensiones.

Las computadoras no pueden manejar imágenes continuas para ser procesadas, ellas manejan órdenes de números digitales. Entonces en el trabajo realizado se necesitan imágenes de dos dimensiones por lo tanto se requiere representar imágenes como campos de dos dimensiones de puntos. A cada punto o rejilla 2D de este campo se le llama píxel. Los píxeles son localizados en las rejillas del campo. La posición de cada píxel esta dada en notación por matrices[4].

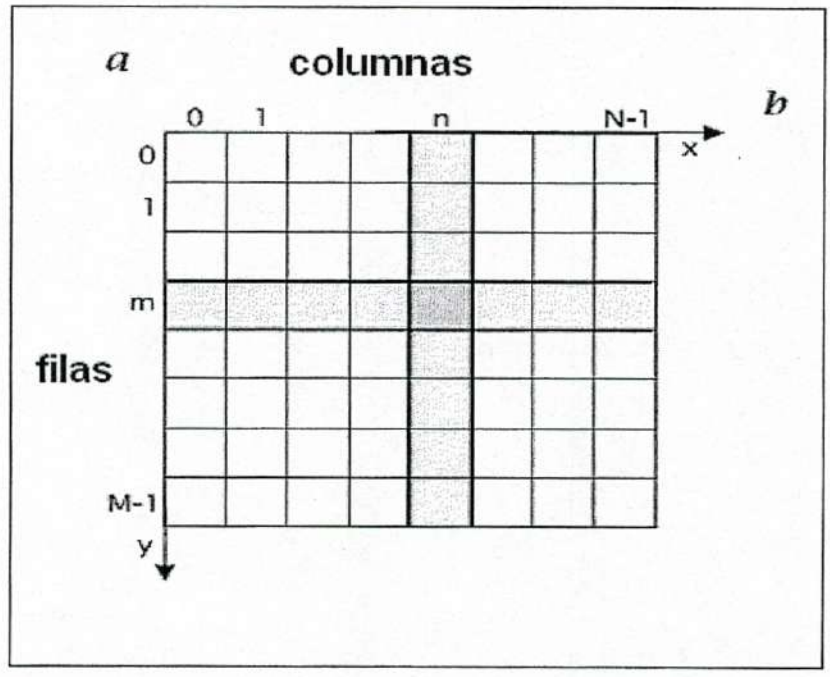


Figura 2.2 Representación de una imagen digital por campos de puntos discretos sobre una rejilla rectangular en 2 D.

En la figura 2.2 se puede ver que el primer indicador m se refiere a la posición de la fila y en n el de la posición de la columna. Una imagen digital contiene M * N píxeles y esta representada por una matriz de M * N, donde el indicador n que corresponde a las filas, corre desde 0 a N-1 y el indicador m desde 0 hasta M-1 dados los números de filas y números de columnas. Cada píxel no representa justamente un punto en la imagen pero es la célula elemental de cada rejilla dada. El valor de cada píxel representa el promedio de irradiancia en la célula correspondiente de una manera apropiada[4].

2.1.3 Sistemas en dos dimensiones.

Un sistema de dos dimensiones en la forma más general es un mapeo de algunas de las entradas de las funciones en dos dimensiones $F_1(x,y), F_2(x,y),...,F_N(x.y)$ a un sistema de salida de funciones de dos dimensiones $G_1(x,y), G_2(x,y),...,G_M(x.y)$, donde $-\infty$ es menor que x, y es menor que ∞, esto denota independencia, de las variables espaciales continuas de las funciones. Este mapeo puede ser representado por los operadores $0\{.\}$ para $m = 1,2,....,M$, qué relaciona la entrada con el sistema de la salida de funciones por el sistema de ecuaciones[1].

En casos específicos el mapeo puede ser de muchos a pocos, de pocos a muchos, o uno a uno. El mapeo de uno a uno se define como:

$$G(x,y) = O\{F(x,y)\} \tag{2}$$

2.2 Convolución Espacial

La convolución espacial como su nombre lo dice es uno de los tratamientos de imágenes más empleados y conocidos, ya que esta convolución discreta es utilizada para el suavizado de imágenes, el afilado de imágenes, detección de bordes etc. En este proceso se calcula un determinado punto en función de su valor y del valor de los puntos que le rodean, aplicando una simple operación matemática en función de la cual se obtendrá un valor resultante para el punto en cuestión.

La convolución espacial en general se puede representar de esta manera:

$$g(x,y) = f(x,y) \otimes a(x,y) \tag{3}$$

Donde $g(x,y)$ representa la imagen filtrada, $f(x,y)$ es la imagen de entrada y $a(x,y)$ es la función respuesta al impulso del filtro a aplicar o también llamada "mascara de convolución".

La mascara de convolución es la que va determinar los valores por los cuales se va ser la suma de píxeles en el vecindario del píxel fuente, los cuales determinan unos

coeficientes a aplicar sobre una área determinada. Dependiendo de la aplicación de filtro que se desee utilizar, el valor de los coeficientes de la mascara de convolución varía. La intensidad general de la imagen resultante se ve afectada por la suma de los pesos de la mascara de convolución.

En los filtros pasa bajos usados en el suavizado de imágenes, la suma de los coeficientes de la mascara de convolución es 1. Las mascaras usadas en la detección de bordes tienen coeficientes negativos y positivos y suman un total de 0.

La "ventana de convolución" es una ventana deslizante la cual se centra en cada píxel de una imagen de entrada y genera nuevos píxeles de salida. Para aplicar la máscara a esa zona se multiplican los valores de los puntos que rodean al píxel que estamos tratando por su correspondiente entrada o coeficiente en la máscara y luego se suman esos productos. El resultado es el nuevo valor para el píxel central, tal y como se puede ver en la siguiente figura. El proceso es repetitivo, incrementa el tiempo en función del tamaño de la imagen, pero tiene la ventaja de que es sencillo y eficaz.

Los nuevos píxeles son colocados en una nueva imagen, todo esto se puede visualizar mejor en la figura 2.3.

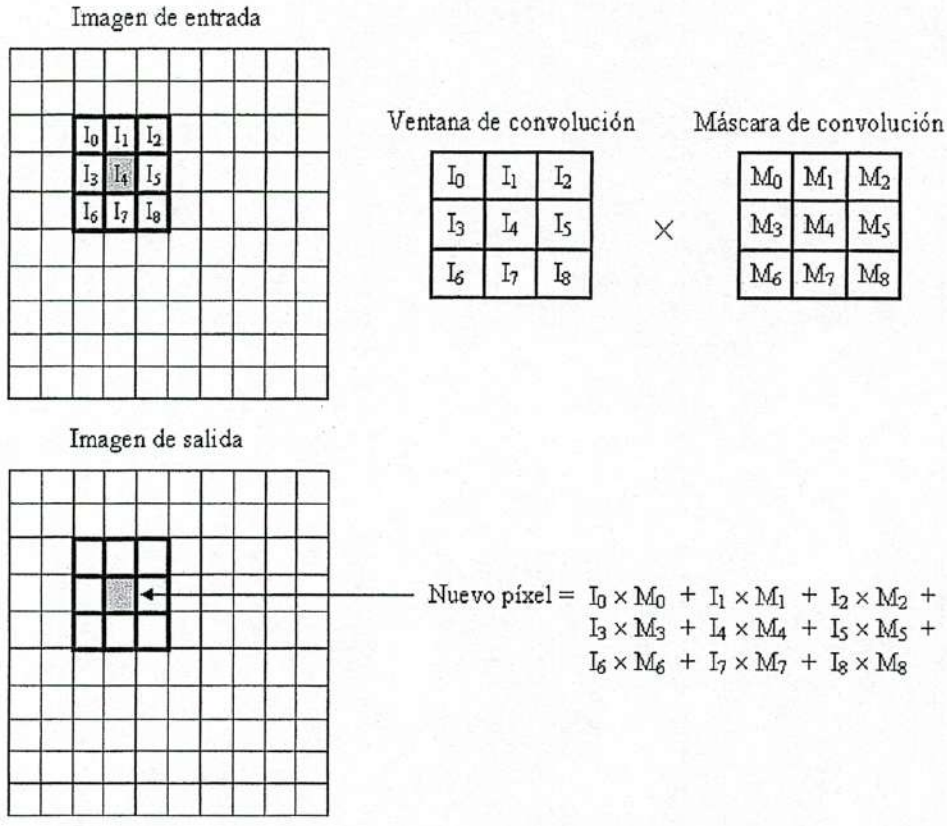


Figura 2.3 Representación de la función de la mascara de convolución y la ventana de convolución.

Existen una gran variedad de mascaras las cuales realizan diferentes acciones en una imagen. Algunas suavizan la imagen mientras otras resaltan bordes, también hay algunas que obscurecen la imagen. Las mascaras mas utilizadas son las de los filtros pasa bajos, pasa altos y detección de bordes. En la figura 2.4 presentaremos un ejemplo de una matriz donde al aplicarla no se genera ningún cambio.

0	0	0
0	1	0
0	0	0

Figura 2.4 matriz la cual no genera ningún cambio.

Cabe mencionar que el tamaño de la matriz cambia, en el ejemplo tenemos una de 3x3 pero para mejorar la definición del filtrado podemos utilizar matrices más grandes, claro esto lleva más trabajo. La primera duda que surge al aplicar la convolución espacial a una imagen es como manejar los bordes de la imagen, ya que cuando es empieza a centrar, se inicia en el punto (0,0) y la ventana deslizante sobresale del borde superior y lateral. Una de las soluciones es poner el valor de cero en las celdas vacías de la ventana de convolución, ha esto se le llama zero-padding, es fácil de implementar pero no funciona si los bordes de la imagen son muy importantes. Otra solución es empezar la convolución en la primera posición donde la ventana no sobresalga de la imagen. Si la máscara de convolución es de tamaño 3x3, se empezaría convolucionando con el píxel en (1,1) en vez del píxel en (0,0), este método también es sencillo de implementar. En la imagen de salida, los bordes convolucionados son copiados para crear una imagen con el mismo tamaño que la imagen de entrada. En la figura 2.3 se muestra como es el acomodo de la ventana de convolución para una de las soluciones presentadas.

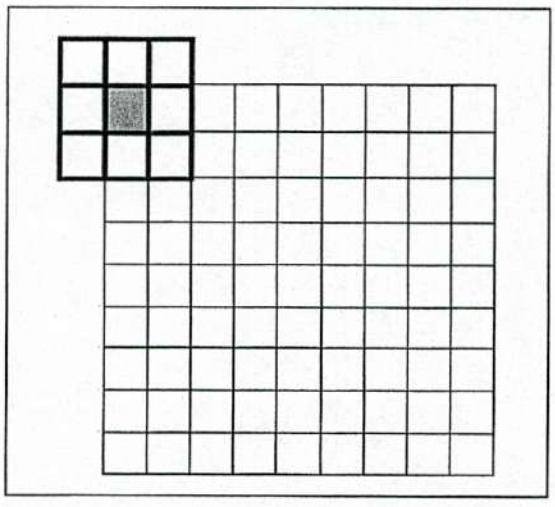


Figura 2.5 Representación del acomodo de la ventana de convolución.

El otro método es "envolver" la imagen, es decir, considerar como píxel contiguo al del borde izquierdo, el píxel del borde derecho y viceversa, así como con los del borde superior e inferior. Como se muestra en la figura 2.6.

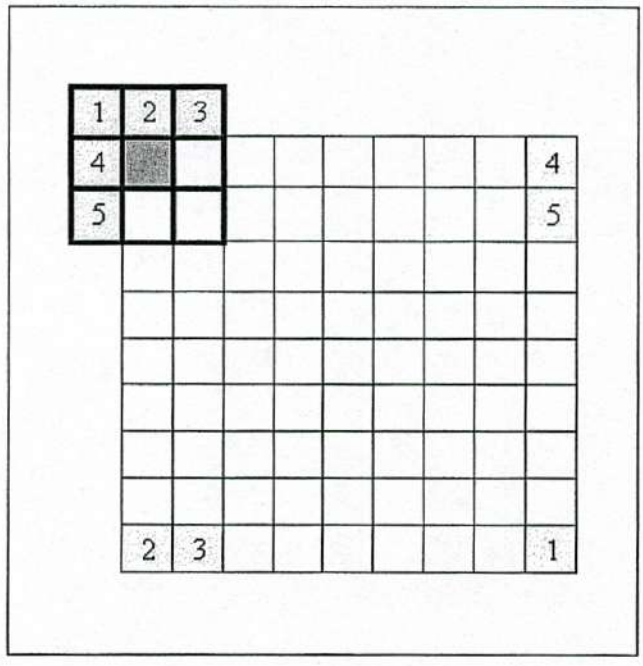


Figura 2.6 figura donde se muestra el método de envolver los bordes.

2.3 Filtros espaciales

Aquí en este tema vamos ha describir los diferentes tipos de filtros que se utilizarán.

El término espacial se refiere al hecho de que el filtro se aplica directamente a la imagen y no a una transformada de la misma, es decir, el nivel de gris de un píxel se obtiene directamente en función del valor de sus vecinos. El concepto de filtrado de una imagen está asociado a la representación de una imagen en el dominio de frecuencias. Si queremos filtrar una imagen de una posible contaminación de ruido aleatorio, lo que tendremos que hacer es diseñar un filtro que reduzca lo más posible la contribución de las altas frecuencias en la formación de la imagen. Los tipos de filtros que estudiaremos serán tres:

Filtros de paso bajo que eliminan frecuencias altas.

Filtros de paso alto que eliminan frecuencias bajas.

Filtros de paso de banda o medianos que eliminan determinados rangos de frecuencias.

El diseño de filtros espaciales se resume en calcular un conjuntos de pesos que definan una máscara cuya transformada de Fourier tenga el comportamiento de uno de los tres filtros básicos que vamos a estudiar. Se puede realizar la misma clasificación que se hizo para el filtrado en frecuencias con respecto al suavizado de una imagen o el realce de una imagen.

2.3.1 Filtros de suavizado

Estos filtros sirven para reducir el ruido de una imagen. Se clasifican en:

Filtros pasa bajas, para implementar este filtro los coeficientes de la máscara deben ser positivos, y además hemos de imponer que la suma de ellos sea igual a 1 para evitar que se salga del rango permitido de píxel.

Filtros mediana, uno de los principales inconvenientes que presenta el alisamiento a través de la operación de convolución es la paulatina desaparición de los saltos entre los niveles de gris de píxeles vecinos. Este tipo de filtros son adecuados para suprimir ruidos aleatorios y picos sin significado, ya que fuerzan a que un píxel sea lo más parecido posible a los píxeles de su entorno. El tamaño de la máscara fijará que clase de picos estamos considerando como no deseables dentro de la imagen.

2.3.2 Filtros de Realce

El principal objetivo de estos filtros es resaltar aquellas características de la imagen que por causa del mecanismo de captación o por error hayan quedado borrosos o sin distinguir en la imagen.

Filtros pasa altas, para estos filtros debemos tomar valores positivos en el centro de la función y valores negativos a su alrededor. Otra condición importante de estas máscaras es que la suma de sus valores sea cero. Esto es importante ya que garantiza que en una zona donde la imagen tenga variaciones pequeñas o sea plana el resultado del filtrado será cero o un valor pequeño. Sin embargo estos filtros pueden producir valores negativos que son causa de problemas por su difícil interpretación y representación como imágenes de niveles de gris, por lo que se han desarrollado otras formas de filtrado de paso alto exentos de estos problemas.

Filtros de enfatización, la idea es el realce de las altas frecuencias presentes en la imagen pero sin perder la contribución de las bajas frecuencias.

Filtros de derivadas, de forma paralela pero a la inversa de como actúa los filtros de alisamiento que están basados en integración, los filtros basados en la derivación de la imagen resaltan las altas frecuencias. El método más común de cálculo de la derivada en un punto de la imagen es el cálculo de su gradiente. Este tipo de filtros dan como salida una nueva imagen cuyos valores son, en cada píxel, el módulo del gradiente en dicho punto de la imagen de entrada. Las distintas aproximaciones dadas para la estimación de estos valores así como para el cálculo del módulo del gradiente han servido como base para la definición de distintos tipos de máscaras de realce.

Operadores de Robert con máscaras de 2 x 2.

Operadores de Prewitt con máscaras de 3 x 3.

Operadores de Sobel, donde las máscaras además de calcular el módulo del gradiente, producen un alisamiento en la imagen que es beneficioso.

CAPITULO 3

3. DISEÑO DE FILTROS DIGITALES PARA PROCESAR IMÁGENES

En este capitulo se evaluaran los diferentes filtros utilizados (sobel *x*, sobel *y*, sobel *x-y*, edge, bluring, smoothing, gaussian ,sharpen, identify), para ver el resultado de la aplicación de cada uno de estos filtros de la mascara de convolución de 5x5 tomamos una imagen que se muestra en la figura 3.1.

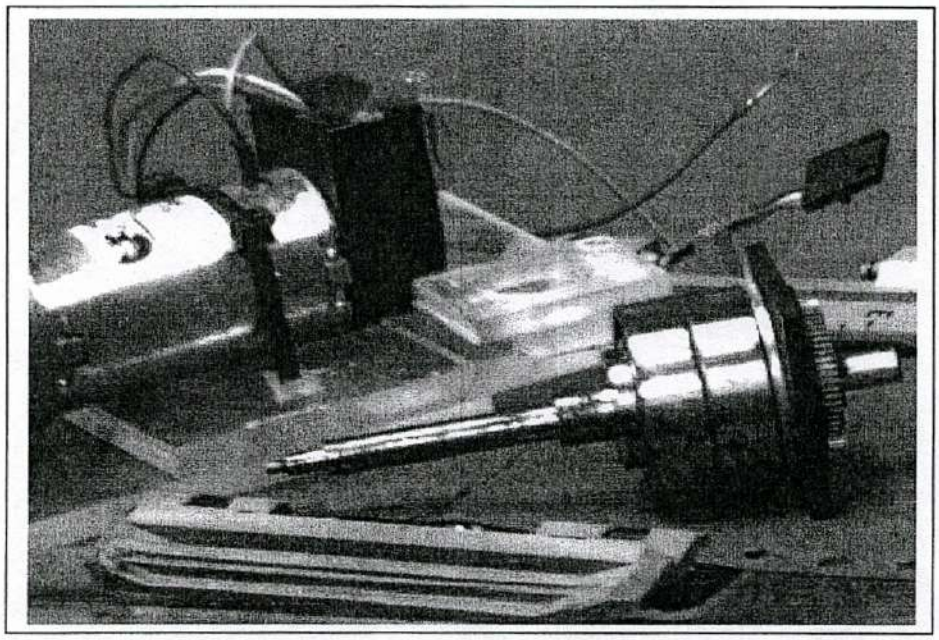


Figura 3.1 Imagen original

3.1 Filtro Sobel

El filtro sobel comprende de 3 filtros los cuales son el sobel *y*, el sobel *x* y por ultimo el sobel *xy*.

El operador Sobel es utilizado en procesamiento de imágenes, especialmente en algoritmos de detectores de bordes. El operador sobel aplicado sobre una imagen digital en escala de grises, calcula el gradiente de la intensidad de brillo de cada punto (píxel) dando la dirección del mayor incremento posible (de negro a blanco) también calcula el monto de cambio en esa dirección, es decir, devuelve un vector. El resultado muestra

que suavemente cambia una imagen en cada punto analizado, y a su vez que tanto un punto determinado representa un borde en la imagen y también la orientación a la que tiende ese borde.

Matemáticamente, el gradiente de una función de dos variables (para este caso la función de intensidad de la imagen) para cada punto es un vector bidimensional cuyos componentes están dados por las primeras derivadas de las direcciones verticales y horizontales. Para cada punto de la imagen, el gradiente del vector apunta en dirección del incremento máximo posible de intensidad, y la magnitud del gradiente del vector corresponde a la cantidad de cambio de intensidad en esa dirección. Lo anterior implica que el resultado de aplicar el operador sobel sobre región de una imagen con intensidad de brillo constante es un vector cero, y el resultado de aplicarlo en un punto sobre un borde es un vector que apunta cruzando el borde (perpendicular) en sentido de los puntos más oscuros hacia los más claros.

Matemáticamente, el operador utiliza dos mascaras de convolución de 3×3 para aplicar convolución a la imagen original para calcular aproximaciones a las derivadas, una mascara para los cambios horizontales y otro para las verticales. Si definimos A como la imagen original y G_x y G_y son las dos mascaras que representan para cada punto las aproximaciones horizontales y verticales de las derivadas de intensidad el resultado se calcula como:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \quad \text{Y} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{4}$$

En cada punto de la imagen, los resultados de las aproximaciones de los gradientes horizontal y vertical pueden ser combinados para obtener la magnitud del gradiente, mediante:

$$G = \sqrt{G_x^{\,2} + G_y^{\,2}} \tag{5}$$

Con esta información, podemos calcular también la dirección del gradiente:

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{6}$$

Donde, por ejemplo, Θ es 0 para bordes verticales con puntos más oscuros al lado izquierdo.

Debido a que la función de intensidad de una imagen digital sólo se conoce mediante puntos discretos, las derivadas de estas funciones no pueden ser definidas al menos que asumamos que existe una función continua que ha sido muestreada en los puntos de la imagen. Con algunas suposiciones adicionales, la derivada de la función continua de intensidad puede ser calculada como una función de la función de intensidad muestreada, la imagen digital. De lo anterior resulta que las derivadas en cualquier punto particular son funciones de los valores de intensidad virtualmente en todos los puntos de la imagen. Sin embargo, aproximaciones a estas funciones diferenciales pueden ser definidas con el nivel de precisión requerido.

El operador sobel representa una primera aproximación imprecisa del gradiente de la imagen, pero es de calidad suficiente para ser de uso práctico en muchas aplicaciones. Más precisamente, éste operador utiliza sólo valores de intensidad en una región de 3x3 al rededor de cada punto analizado para calcular el gradiente correspondiente, además de que utiliza sólo números enteros para los coeficientes que indican la aproximación del gradiente.

3.1.1 Detalles técnicos

Como una consecuencia de su definición, el operador sobel puede ser implementado mediante simples definiciones tanto en hardware como en software: sólo son utilizados ocho puntos de la imagen alrededor del punto a analizar para calcular el punto correspondiente de la imagen resultante, además sólo se requiere aritmética con números enteros para calcular una aproximación del vector gradiente. Además, los dos filtros discretos descritos arriba pueden ser separados:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{7}$$

y las dos derivadas G_x y G_y pueden ser calculadas con:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * A \quad \text{Y} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * A \tag{8}$$

En ciertas implementaciones, estos cálculos separados dan buena ventaja ya que implican menor operaciones aritméticas para cada punto.

Ya que el resultado del operador sobel es un mapeo de dos dimensiones del gradiente de cada punto, éste puede ser procesado y ser visto como una imagen, con las áreas de gradiente elevado (equivalentes a bordes) en negro y con los demás como blanco (el fondo de la imagen generada). Las siguientes figuras ilustran lo anterior, se muestra el cálculo del operador sobel sobre una imagen. Obsérvese las diferencias de gradiente (zonas negras) dadas al aplicar únicamente un gradiente en las figuras 3.3 y 3.4 donde se aplican los filtros sobel *x* y sobel *y* respectivamente. En la figura 3.2 se aplica el mapeo en las dos dimensiones tanto vertical como horizontal por lo tanto se aplica el filtro sobel *xy*.

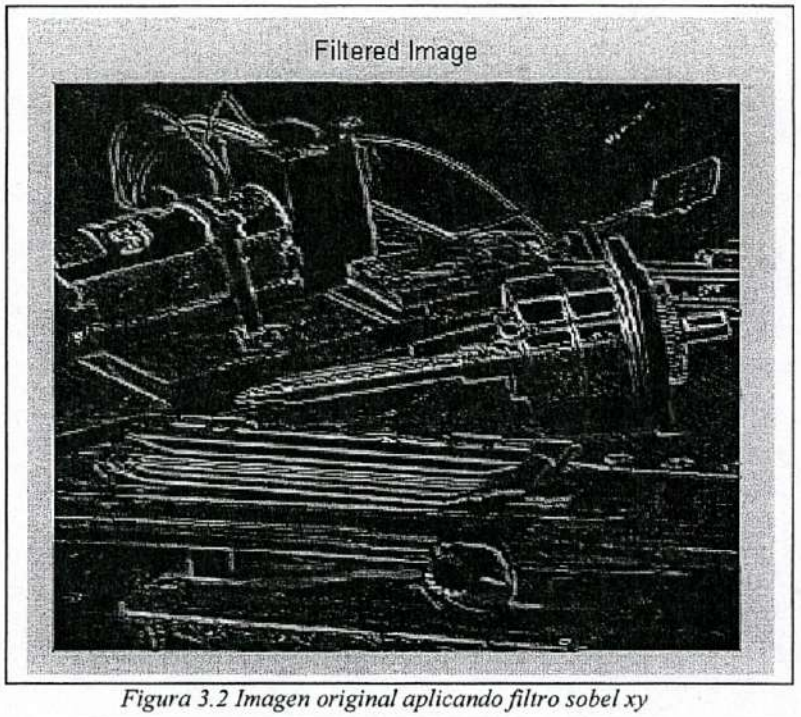


Figura 3.2 Imagen original aplicando filtro sobel xy

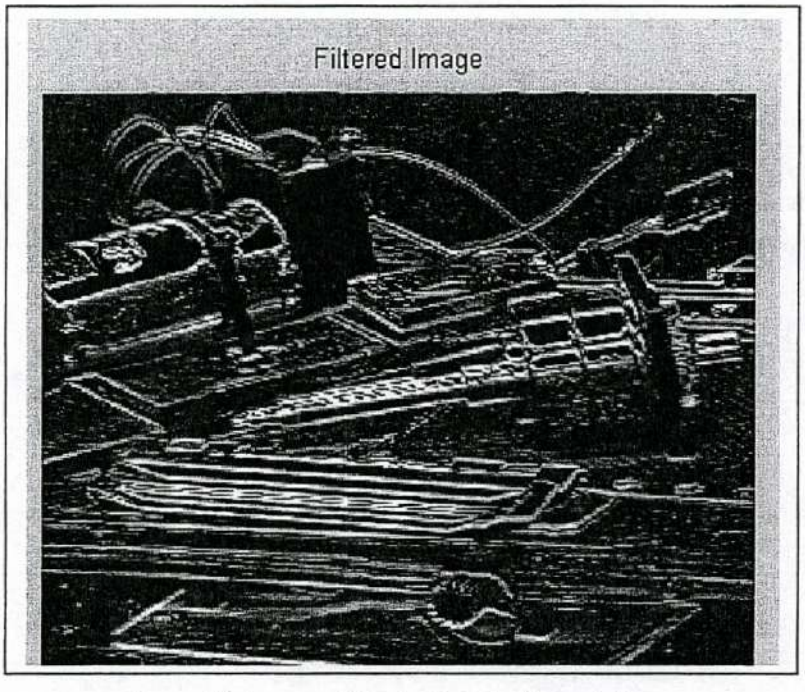


Figura 3.3 imagen original aplicando filtro sobel x

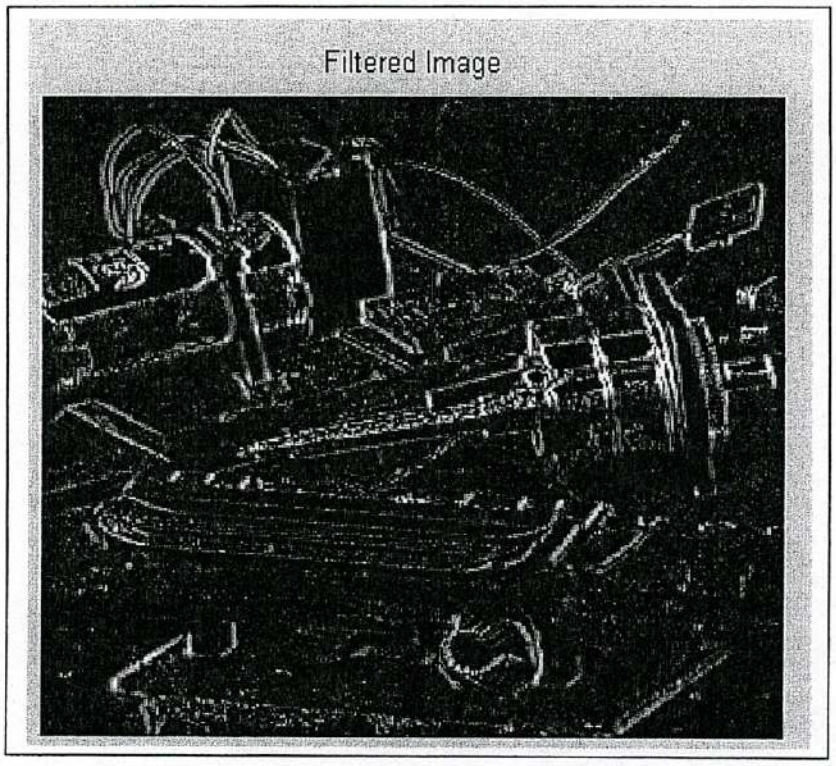


Figura 3.4 Imagen original aplicando filtro sobel y.

3.2 Filtro de Detección de bordes (edge).

Uno de los más importantes filtros es el de la detección de bordes. Importante porque de él se puede empezar a extraer importante información de la imagen, como pueden ser las formas de los objetos que la componen. Estos operadores son utilizados en aplicaciones para el reconocimiento de formas, aplicaciones industriales, militares, etc.

Dentro de las numerosas aplicaciones para la detección de bordes, los artistas digitales lo usan para crear imágenes con contornos deslumbrantes pues la salida de un detector de bordes puede ser agregada a una imagen original para realzar los bordes. La detección de bordes es a menudo el primer paso en la segmentación de imagen, que es un campo del análisis de la imagen, y se utiliza para agrupar los píxeles en regiones para determinar una composición de la imagen. La detección de bordes también es usada en el registro de imagen, el cual alinea dos imágenes que podrían ser adquiridas en momentos separados y de sensores diferentes.

Los bordes de una imagen contienen mucha de la información de la imagen. Los bordes cuentan donde están los objetos, su forma, su tamaño, y también sobre su textura. Los ejes o bordes se encuentran en zonas de una imagen donde el nivel de intensidad fluctúa bruscamente, cuanto más rápido se produce el cambio de intensidad, el eje o borde es más fuerte. Un buen proceso de detección de bordes facilita la elaboración de las fronteras de objetos con lo que, el proceso de reconocimiento de objetos se simplifica. Para poder detectar los bordes de los objetos, debemos de detectar aquellos puntos borde que los forman.

En general, los bordes de objetos en una imagen los podemos distinguir por los cambios más o menos bruscos de valor entre dos o más píxeles adyacentes. Podemos realizar una clasificación general de los bordes según sea su dirección en:

- Bordes verticales, cuando píxeles conectados verticalmente tienen valores diferentes respecto de los anteriores o posteriores.
- Bordes horizontales, cuando tenemos píxeles conectados horizontalmente, y estos tienen distintos valores respecto de los anteriores o posteriores.
- Bordes oblicuos, cuando tenemos una combinación de las componentes horizontales y verticales.

La diferencia entre los valores de los píxeles nos indica lo acentuado del borde, de forma que a mayores diferencias tenemos bordes más marcados y a menores tenemos unos bordes suavizados. Los filtros utilizados para la detección de bordes son filtros diferenciales, que se basan en la derivación o diferenciación. Dado que el promediado de los píxeles de una región tiende a difuminar o suavizar los detalles y bordes de la imagen, y esta operación es análoga a la integración, es de esperar que la diferenciación tenga el efecto contrario, el de aumentar la nitidez de la imagen, resaltando los bordes como lo podemos ver en la figura 3.5 donde se aplica el filtro edge.

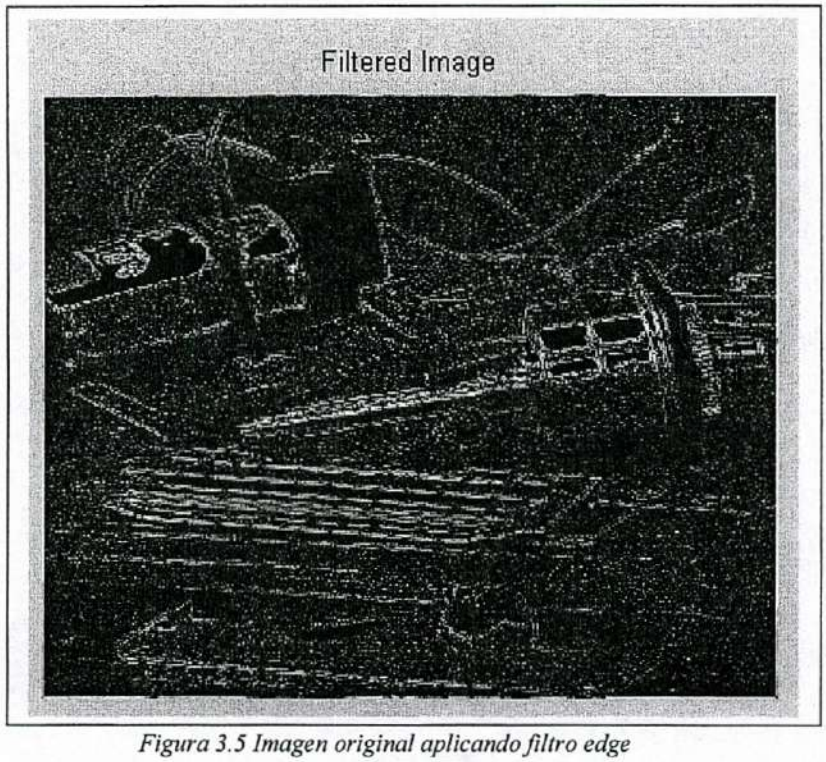


Figura 3.5 Imagen original aplicando filtro edge

3.3 Filtro Gaussian

El operador del filtro gausiano es un operación de convolución 2-D que es usado por desenfoque de imágenes (blur) y removedor de detalles y ruidos.

La función gausiana en una dimensión es:

$$g(x) = e^{-\frac{x^2}{2\sigma^2}} \tag{9}$$

con media cero y desviación típica σ.

Para procesamiento digital de imágenes se emplea la función gausiana bidimensional:

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}} \tag{10}$$

Estos filtros lineales con los pesos escogidos de acuerdo a una función gausiana son muy utilizados, así también son buenos para eliminar el ruido gausiano muy utilizados en la detección de bordes.

Propiedades

- Simetría rotacional. El mismo efecto en todas las direcciones.
- Un único lóbulo (pico).
- El peso de los píxeles va disminuyendo con la distancia al centro.
- Cuanto más alejado está un píxel, menos significativo es este.
- Conserva las bajas frecuencias y tiende a eliminar las altas.
- El grado del filtrado es controlado y manejado por σ.

La idea del filtro gausiano es para usar la distribución de dos dimensiones como una función de 'punto difundido', y este es logrado por convolución. Desde la imagen es almacenada como una colección de píxeles discretos que nosotros necesitamos para producir una aproximación discreta para una función Gausiana como en la figura 3.6 antes de que podamos funcionar la convolución. La siguiente figura muestra un valor-entero conveniente de la mascara de convolución que aproxima una gaussian con una media σ de 1.0.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figura 3.6 Aproximación discreta para función gausiana con media $\sigma = 1$.

Una ves que la mascara de convolución calculado, entonces el gaussian puede ser realizado usando métodos de convolución. La convolución puede ser factor para realizarse rápidamente desde la ecuación para la segunda dimensión isotropica Gausiana muestra que es separable sobre los componentes x y y. Así la convolución de

segunda dimensión puede ser realizada con la primera convolución con una primera dimensión Gausiana en la dirección y. (El Gaussian es de hecho el único operador totalmente circular simétrico que puede ser descompuesto de tal manera) la figura muestra en la primera dimensión de componente x de la mascara de convolución que al ser utilizado produce la mascara llena que se muestra en la figura 3.7 (después de escalar por 273, de redondear y de truncar una fila de píxeles alrededor del límite porque tienen sobre todo el valor 0. Esto reduce la matriz 7x7 al 5x5 demostrado arriba) el componente y es exactamente el mismo pero es orientado verticalmente.

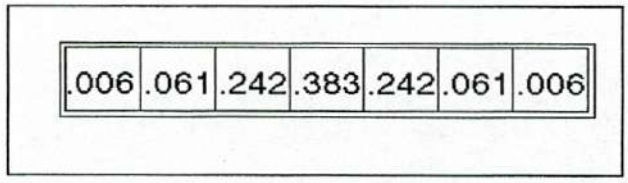


Figura 3.7 La primera dimensión de componente x de la mascara de convolución.

Otra manera de computar un Gausian con una desviación estándar grande esta convolucionar una imagen varias veces con un pequeño Gausian. Mientras que es computacional mente compleja, esta puede tener aplicabilidad si el procesamiento se realiza usando un hardware pipeline.

El filtro Gausiano no solo es útil en aplicaciones de ingeniería. Este también atrae la atención de la biología computacional porque se ha atribuido con una cierta cantidad de plausibilidad biológica por ejemplo algunas células en los caminos visuales del cerebro tienen a menudo una respuesta aproximadamente Gaussiano.

A continuación vemos en la figura 3.8 los resultados sobre la imagen original aplicando un filtro gaussian.

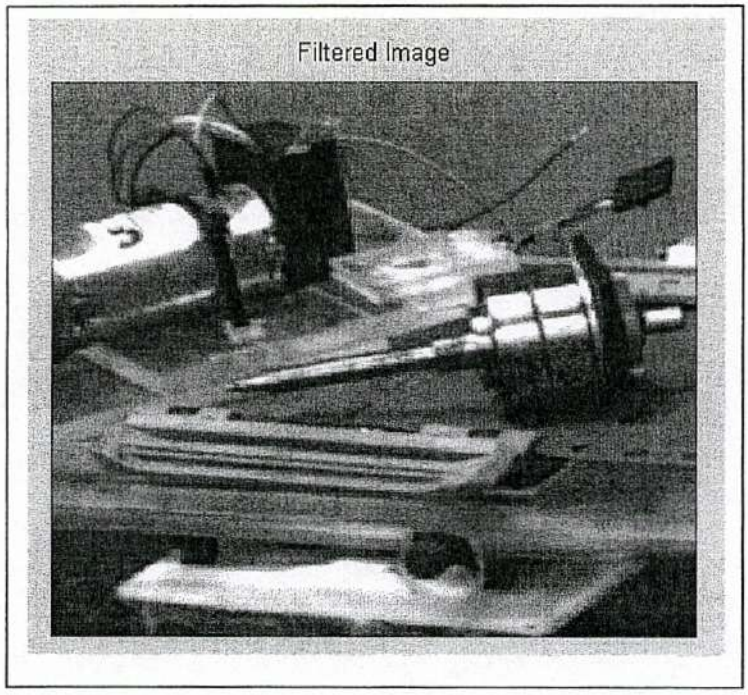


Figura 3.8 Imagen original aplicando filtro gaussian.

3.4 Filtro Smoothing o Suavizado

El suavizado de imágenes se utiliza para dos acciones las cuales son: dar a una imagen un difuminado o efecto especial y para la eliminación de ruido.

El suavizado o filtrado espacial paso bajo borra los detalles más finos de una imagen, es decir, atenúa las altas frecuencias, mientras se mantienen las bajas y medias frecuencias. Tiene un buen número de aplicaciones: algunas veces se emplea para simular una cámara desenfocada, o para quitar la vista a un fondo; mientras los fotógrafos usan un filtro de cámara para conseguir ese efecto, los artistas informatizados emplean filtros digitales.

El suavizado se alcanza mediante la convolución, y es fácil ver en la máscara de convolución que el suavizado es simplemente el promedio del vecindario. Promediar tiende a eliminar los valores extremos de un grupo, así los píxeles extremadamente claros u oscuros pueden hacerse más grises dependiendo de los vecinos del píxel. Cuanto más grande es la máscara, mayor es el efecto de suavizado y mayor el tiempo de cómputo requerido.[2]

3.4.1 Filtros Pasa Bajas

Las máscaras de los filtros pasos bajo deben tener todos sus coeficientes positivos y la suma de ellos debe ser igual a uno. Las máscaras de tamaño 3x3 más utilizadas se pueden ver en la figura 3.9.

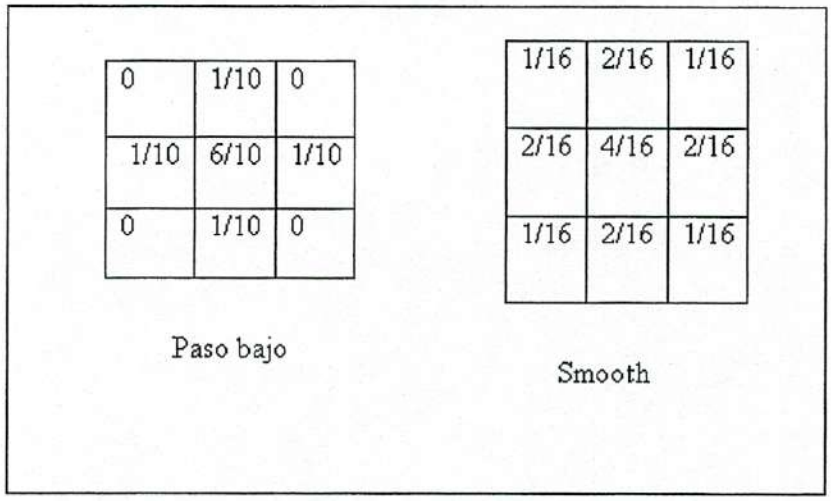


Figura 3.9 Mascara pasa bajas y smooth.

Las dos máscaras tienen el mismo efecto sobre la imagen, pero con la llamada Smooth, al tener los coeficientes un mayor valor, el efecto de difuminado es mucho más fuerte. En el caso de la máscara Smooth debemos darnos cuenta que la suma de los valores es 16 y no 1, por lo que los valores obtenidos con ella para los puntos de la imagen, se nos pueden salir del rango válido, por lo que el resultado debe ser normalizado dividiéndolo por la suma de la máscara, en este caso 16. En el caso de la máscara Paso Bajo la suma total es de 10, valor por el cual dividimos para normalizar el resultado.

El filtrado con una máscara paso bajo, como la Smooth, produce un efecto de difuminado de los bordes y emborronamiento de la imagen filtrada con respecto a la imagen original. Esta pérdida de los detalles pertenecientes a las altas frecuencias es lo que caracteriza a todos los filtros paso bajo o de suavizado.

Las máscaras vistas anteriormente no son las únicas que producen un efecto paso bajo, pero si son las más extendidas. Cada uno puede diseñar su propia máscara, sólo hay que

seguir las pautas dadas anteriormente. En la siguiente figura 3.10 mostramos otras máscaras que son ejemplos propios:

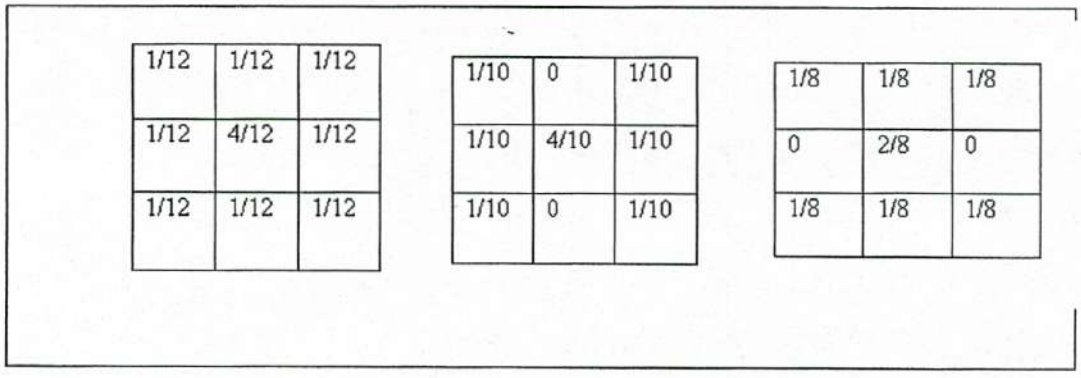


Figura 3.10 Ejemplos de mascaras pasa bajas y smooth

3.4.2 Filtro de Media

De entre la multitud de máscaras de filtro paso bajo destaca especialmente la máscara de media, que es la que efectúa el promedio de los valores del entorno. El filtro espacial de media reemplaza el valor de un píxel por la media de los valores del punto y sus vecinos. Su efecto es el difuminado o suavizado de la imagen y se aplica junto con el de mediana para eliminar ruidos. Este filtro lo implementamos con la siguiente máscara que se visualiza en la figura 3.11.

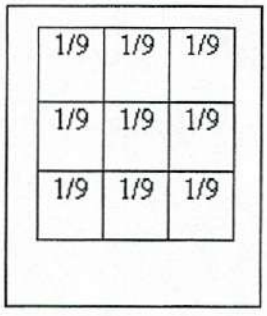


Figura 3.11 Mascara de filtro media 3x3

Normalmente el tamaño de la máscara se toma en función de la cantidad de suavizado que queramos aplicar en cada momento. La visualización del resultado es el único medio de saber si hemos elegido el tamaño adecuado.

Se puede observar que el efecto final del filtro de la media es un suavizado de la imagen por reducción o redistribución del valor de los píxeles. Este filtro tiene el resultado opuesto a los de detección de bordes, donde el objetivo de los filtros es acentuar las diferencias, por esta razón el filtro de la media es un filtro paso bajo. También hay que notar que este filtro no modifica la imagen en las zonas donde el valor de los píxeles es el mismo como se muestra en la figura 3.12 donde aplicamos en filtro smoothing.

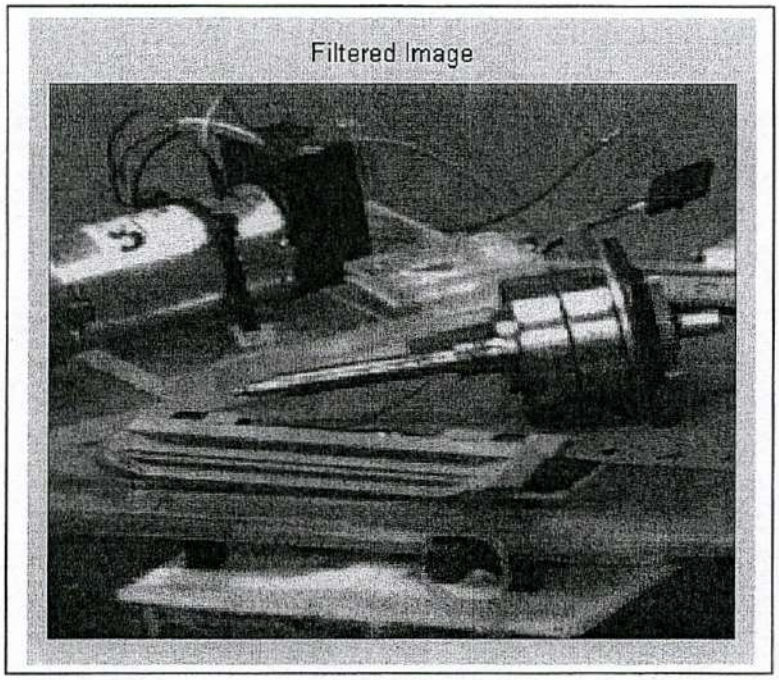


Figura 3.12 Imagen original aplicando filtro smooting

3.5 Filtro Blurring

El blur es un efecto ampliamente usado en software de gráficos como Adobe Photoshop, el GIMP y Saint.NET. Este es típicamente usado para reducir el ruido de la imagen y reducir los niveles de detalles. El efecto visual de esta técnica de blurring es un blur smoothing así como si miraras una imagen a través de una pantalla translúcida claramente diferente del efecto del bokeh producido por un lente fuera de foco o la sombra de un objeto bajo la iluminación usual. El gaussian smoothing es también usado como un pre-procesamiento en las etapas de los algoritmos de visión de la computadora para realzar la estructura de una imagen en diferentes escalas.

MECANICAMENTE

El gaussian blur es un tipo de filtro de imagen- emborronamiento que usa una distribución normal para hacer cálculos para aplicar la transformación a cada píxel en la imagen. La ecuación de la distribución gaussiana en dimensión N es

$$G(r) = \frac{1}{\sqrt{2\pi\sigma^2}^N} e^{-r^2/(2\sigma^2)} \tag{11}$$

o específicamente en dos dimensiones

$$G(u,v) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/(2\sigma^2)} \tag{12}$$

Donde r es radio blur $\left(r^2 = u^2 + v^2\right)$, y σ es la desviación estándar de la distribución gaussiana. Cuando aplicas en dos dimensiones, esta formula produce una superficie cuyo contorno son círculos concéntricos con una distribución Gaussiana desde el punto del centro. Los píxeles donde la distribución es no cero son usados para construir una matriz de convolución, el cual es aplicado para la imagen original. El valor de cada píxel es colocado para un promedio cargado de la vecindad de los píxeles. El valor del píxel original recibe el peso más pesado. Esto da como resultado un blur que conserva linderos y bordes mejor que otro, mas uniforme que filtros de blurring.

En teoría, la distribución en cada punto sobre la imagen será no cero, quiere decir que la imagen entera necesitaría ser incluida en los cálculos para cada píxel. En la práctica, cuando computamos una aproximación discreta de una función Gaussiana, los píxeles de afuera de aproximadamente 3σ son bastantes pequeños para ser considerados efectivamente cero. Entonces, los píxeles de afuera del rango pueden ser ignorados. Típicamente, un programa de imágenes necesita solo calcular una matriz con dimensiones $(6\sigma+1)\times(6\sigma+1)$ para asegurar todos los píxeles pertinentes son tomados en cuenta.

Además de ser circularmente simétrico, el Gaussian Blur puede ser aplicado a una imagen de dos dimensiones como dos cálculos de una sola dimensión independientes, y tan es llamado como separable linealmente. Eso es el efecto de aplicar una matriz de dos dimensiones puede también ser logrado aplicando una serie de simple dimensión de matrices Gaussianas en la dirección horizontal, entonces repitiendo el proceso en la dirección vertical. En términos computacionales, esta es una propiedad útil, desde la calculación puede ser realizado en tiempo $O(n\times M\times N)+O(m\times M\times N)$, a diferencia de $O(m\times n\times M\times N)$ por una mascara de convolución no separable, donde M, N son las dimensiones de la imagen filtrada y m, n son las dimensiones de el filtro.

Aplicando un múltiplo sucesivo blur para una imagen tiene el mismo efecto que el aplicar un sencillo, mas grande blur, quién es aquel cuyo radio es la raíz cuadrada de la suma de los cuadrados del radio del blur que donde fue actualmente aplicado. Por ejemplo,

Aplicando sucesivamente gaussian blur con radio de 6 y 8 da el mismo resultado como aplicar una simple blur de radio 10, desde $\sqrt{6\times 6+8\times 8}=10$. Por esta relación, el tiempo de procesamiento no puede ser salvado simulando un blur de Gaussian con blur sucesivos, más pequeños, el tiempo requerido será al menos tan grande como realizar el blur sencillo.

Blurring es comúnmente usado cuando reducimos el tamaño de una imagen. Cuando muestreamos abajo una imagen, este es común para aplicar un filtro pasa bajas para la imagen antes de retomar muestras. Esto es para asegurar que la información falsa de alta frecuencia no aparezca en la imagen del bajo muestreo. Los blur tienen buenas propiedades, por ejemplo no tener ningún borde agudo, y así no introducen a timbrar en la imagen filtrada como lo podemos ver en la figura 3.13 donde se aplica el filtro blur.

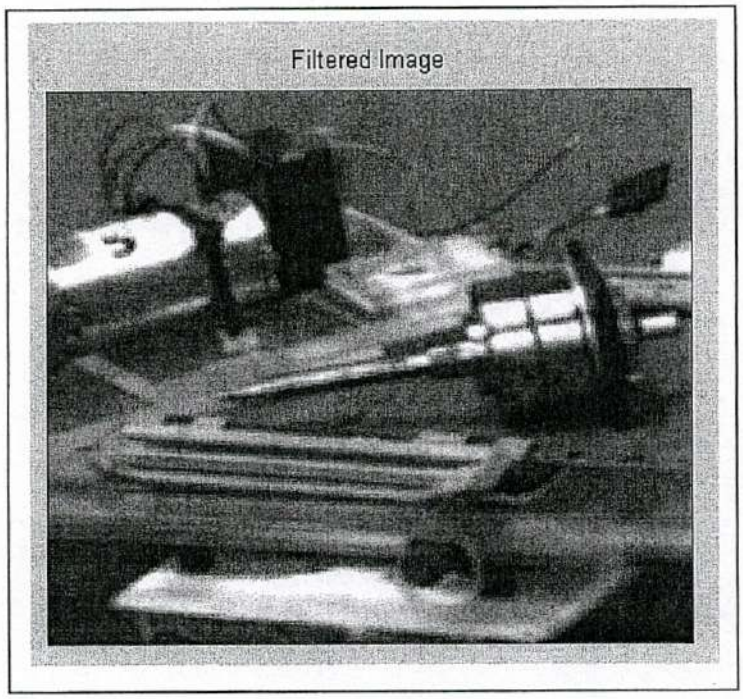


Figura 3.13 Imagen original aplicando filtro Blur.

3.6 Filtro Sharpean

Para entender los filtros sharpen primero debemos definir lo que es el realce en una imagen y los filtros pasa altas.

3.6.1 Realce

El objetivo principal del realce es el de destacar los detalles finos de una imagen o intensificar detalles que han sido difuminados, bien sea por error o bien por efecto natural del método de adquisición de la imagen. El realce de una imagen aumentará su contraste. Las utilidades del realce de las imágenes son variadas e incluyen aplicaciones que van desde la impresión electrónica y las imágenes médicas hasta las inspecciones industriales e incluso la detección autónoma de objetivos en las armas inteligentes.

Para la implementación del realce se utilizan, lo mismo que para el suavizado, técnicas basadas en la aplicación de filtros. El relace de una imagen se basa en el filtro paso alto. Un filtro paso alto eliminará los componentes bajos de frecuencia (como el medio de la imagen) y mostrará sólo los detalles altos.[2]

Por lo general, la máscara de convolución usada comúnmente en el realce, es decir, la máscara de filtro paso alto, tiene un coeficiente positivo en su centro y coeficientes sobre todo negativos alrededor del borde externo.

3.6.2 Filtro pasa alto

Para implementar un filtro paso alto, es decir permitir pasar las componentes de altas frecuencias y diluir las de baja frecuencia, es necesario que el filtro posea coeficientes negativos en la periferia y positivos en el centro. Así, cuando la máscara se encuentra sobre una zona uniforme, la salida proporcionada por la máscara será 0 o próxima a dicho valor. Normalmente, este tipo de filtro elimina también el término de frecuencia 0 con lo que la imagen resultante deberá tener valores de intensidad negativos. Como sólo estamos considerando niveles positivos de gris, los resultados del filtrado paso alto necesariamente implican alguna forma de desplazamiento o cambio de escala para que al final los niveles de gris queden dentro del rango. Las máscaras pasa altas de tamaño 3x3 más utilizadas se muestran en la figura 3.14.

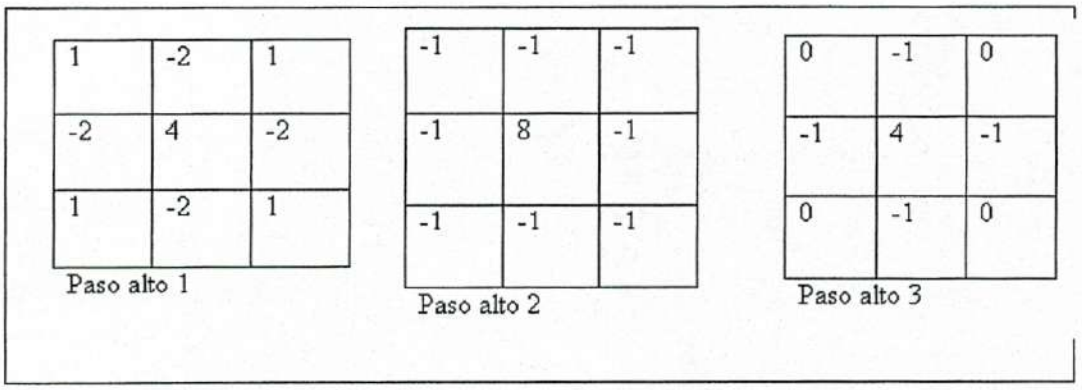


Figura 3.14 Mascaras paso alto mas utilizadas.

En estas máscaras, llamadas propiamente paso alto, hay que destacar que la suma de los coeficientes es cero. Así cuando la máscara está sobre una zona de la imagen de poco contraste o pequeña variación del nivel de gris, la salida de la máscara es cero o muy pequeña. Esto se traduce en que zonas uniformes con distintos niveles de gris son pasadas a un mismo nivel (el cero), con lo que perdemos información de la imagen. Además, eliminar los términos de baja frecuencia produce una disminución de la media

de los niveles de gris, reduciendo significativamente el contraste global de la imagen. Como en el caso de los filtros pasa bajas, las máscaras vistas anteriormente no son las únicas que producen este efecto, pero sí son las más extendidas. De igual forma se puede diseñar máscaras propias. Las siguientes máscaras que podemos ver en la figura 3.15, son ejemplos de lo mismo.

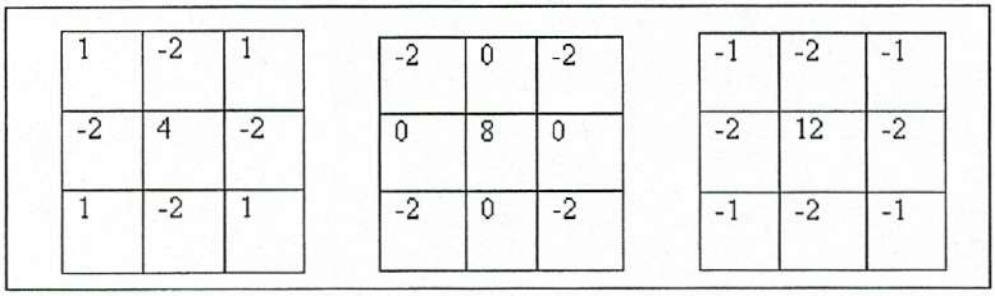


Figura 3.15 Ejemplos de mascaras pasa altos.

3.6.3 Filtros Sharpean.

Ya visto y explicado, al aplicar los filtros paso alto vistos en la parte anterior se disminuye considerablemente el numero de grises presentes en la imagen original, con lo que se reduce el contraste global de la imagen. Para realizar el realce propiamente dicho de la imagen empleamos los **filtros Sharpen** que constituyen una variación de las máscaras de realce ya vistas. Las máscaras Sharpen más comunes se muestran en la figura 3.16.

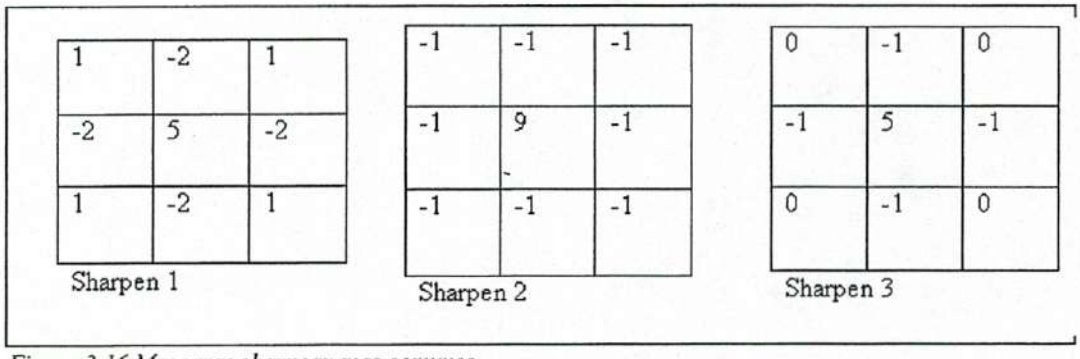


Figura 3.16 Mascaras sharpean mas comunes.

Como podemos observar, en este caso aumentamos en uno el valor centras de las máscaras paso alto para obtener la máscara Sharpen correspondiente. En la siguiente figura podemos apreciar como el filtrado paso alto disminuye considerablemente el número de grises presentes en la imagen original, reduciéndose el contraste global de la imagen. También podemos observar la mejora en nitidez y detalles que experimenta la imagen que ha sido realzada mediante el filtrado con las máscaras Sharpen. Se puede ver cómo esta máscara tiene un efecto de afilado de los bordes y mejora de los detalles borrosos sin que cambie la apariencia global de la imagen, como sí hace el filtrado con las máscaras paso alto todo esto se muestra en la figura 3.17 donde se aplica el filtro Sharpean.

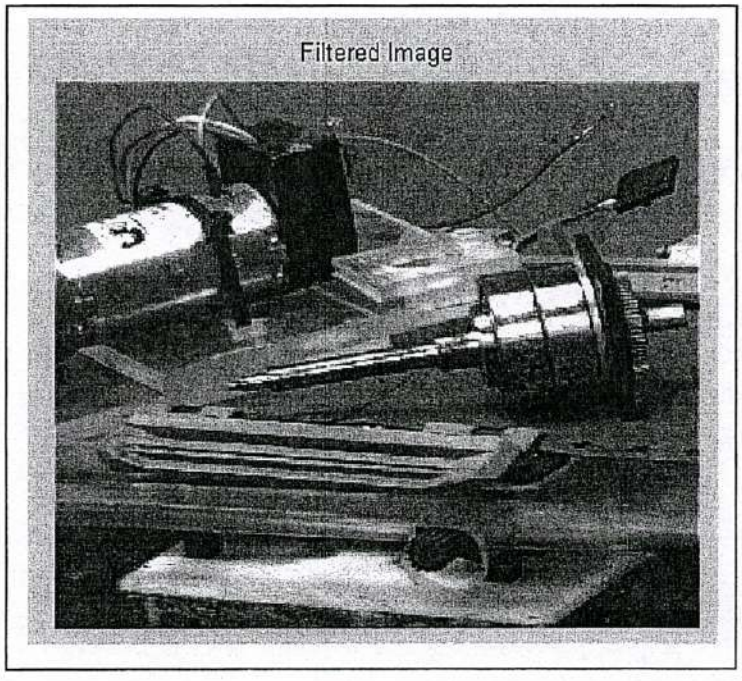


Figura 3.17 Imagen original aplicando filtro sharpean.

CAPITULO 4

4. IMPLEMENTACION DEL SISTEMA DE FILTRADO.

Como recordamos para la implementación del sistema de filtrado nos vamos ha basar en un diagrama de bloques en general que se describe en la figura 4.1

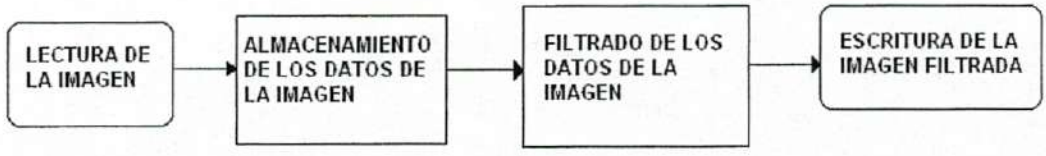


Figura 4.1 diagrama de bloques de sistema en general

4.1 Planteamiento del problema

El objetivo de esta tesis es desarrollar un sistema para procesar una imagen láser en escala de grises. La imagen original es muy "difusa" debido al ruido que se produce por las altas frecuencias (característica de los patrones láser). La imagen procesada es empleada para reconstruir objetos tridimensionales cuyo algoritmo requiere encontrar los puntos máximos del patrón láser. Como el patrón de origen es "difuso" no se puede reconocer de manera conveniente el punto máximo por lo tanto hay qua aplicar un filtro que nos permita que la imagen no presente manchas extras fuera del patrón de iluminación.

4.2 Implementación con Simulink

Para buscar el filtro adecuado a la imagen aplicaremos un sistema de bloques de SIMULINK en las secciones siguientes se explica el procedimiento.

Simulink.

Simulink es una plataforma para la simulación de multidominio y el Diseño basado en modelos de sistemas dinámicos. Provee un ambiente gráfico interactivo y un set modificable de bibliotecas de bloques que le dejan exactamente diseñar, simular,

implementar, y hacer pruebas de control, procesamiento de señales, comunicaciones, y otros sistemas variantes en el tiempo.

Los productos adicionados extienden el ambiente de Simulink con las herramientas para el modelado específico y diseño de tareas y para la generación de código, implementación de algoritmo, prueba, y verificación. Simulink es integrado con MATLAB, dando acceso inmediato a un rango extensivo de herramientas para la visualización de algoritmos de desarrollo, de datos, análisis de datos y el acceso, y la computación numérica.

4.2.1 Lectura de la imagen para filtrar.

En este sistema de bloques de Simulink lo que se busca es filtrar una imagen la cual tenemos que introducirla dentro de dichos bloques, para que la imagen pueda ser entendible por los bloques de simulink la tenemos que adecuar, ya que la imagen se puede manejar en diferentes formas ya sea en matriz o vector, en la figura 4.2 se presenta el programa con el cual se maneja en el sistema la imagen. En este programa primero se genera una matriz de datos apartir de una imagen dada ya que se tiene la matriz la convierte en vector de datos para que así pueda ser entendibles por los bloques.

```
% sysgenConv5x5_imageData
------crea una matrix leyendo la imagen dada
[sysgenConv5x5_imageData, map] = imread('mano.bmp');
lineSize = size(sysgenConv5x5_imageData,1);
NPixels = size(sysgenConv5x5_imageData,1) * size(sysgenConv5x5_imageData,2);
grayScaleImage = 0;
for I = 1:lineSize,
  %for J = 1:lineSize,
  for J = 1:512,
    pixel = double(sysgenConv5x5_imageData(I,J));
    mapValue = map(pixel + 1);
    grayPixel = mapValue * 255;
    grayScaleImage(I,J) = uint8(floor(grayPixel));
  end
end
% turn the array into a vector---------- convierte la matrix en un vector
grayScaleSignal = reshape(grayScaleImage,1,NPixels);

% insert a column of 'time values' in front -- the from workspace
% block expects time followed by data on every row of the input----inserta una columna de %
'valores de tiempo' enfrente --- desde el bloque de workspace espera el tiempo seguido por %
dato de cada raiz de la entrada
grayScaleSignal = [ double(0:NPixels-1)'  double(grayScaleSignal)'];
```

Figura 4.2 Programa de lectura de imagen.

El bloque que nos servirá para capturar o leer la imagen se llama simin el cual se encuentra en la biblioteca de Sources (fuentes). En la figura 4.3 a) se ilustra el bloque de lectura de imagen sin el programa incluido y el la figura 4.3 b) se ilustra el bloque de lectura con el programa incluido.

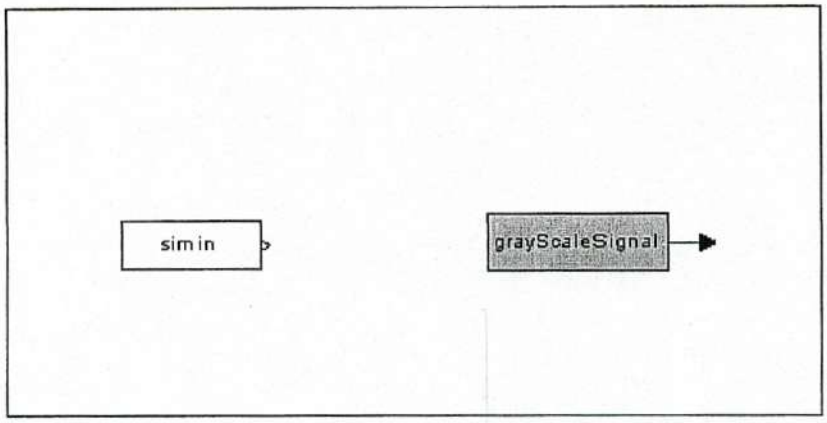


Figura 4.3 a) Bloque de lectura de imagen sin el programa b) bloque de lectura con el programa incluido.

A continuación se describe brevemente las propiedades de dicho bloque y cual es su función dentro de lo que queremos nosotros. El bloque lee los datos del espacio de trabajo (workspace) MATLAB. Los datos de bloque son parámetros especificados en el espacio de trabajo por una expresión en MATLAB que evalúa a una matriz (arreglo 2D) una estructura contiene un arreglo de valores de señal y pasos de tiempo. El bloque desde el espacio de trabajo interpola el parámetro de los datos determina la salida del bloque en el intervalo del tiempo para el cual se proveen los datos del espacio de trabajo. Si seleccionas la opción de interpolación de datos, el bloque utiliza la interpolación Lagrangian linear para computar los valores de los datos para los pasos del tiempo que ocurren entre los pasos del tiempo para los cuales el espacio de trabajo provee datos. En la figura 4.4 se ilustra la caja de parámetros del bloque de lectura.

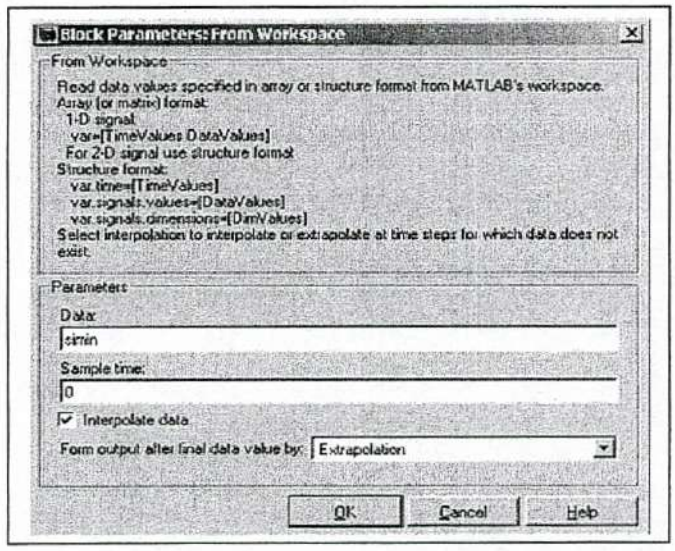


Figura 4.4 Caja de parámetros del bloque de lectura de imagen.

Una expresión que evalúa a un arreglo o una estructura que contiene un arreglo de tiempos de simulación y valores correspondientes de la señal. Por ejemplo supongamos que el espacio de trabajo contiene un vector de la columna de tiempos nombrados T y un vector de los valores correspondientes de la señal nombrados U. En este caso tendríamos que poner en este campo [T,U]. Si el arreglo o la estructura requerida del la señal contra tiempo existe ya en el espacio de trabajo, tenemos que introducir el nombre de la estructura o la matriz en este campo, este es el caso de nosotros el arreglo que tenemos lleva el nombre de grayScaleSignal por lo tanto en el campo de datos se escribe el nombre ya mencionado para que de ahí tome el arreglo desde el espacio de trabajo. Ya que se tiene la imagen en escala de grises en forma de vector sigue el siguiente bloque

Gateway In

El bloque gateway in se ilustra en la figura 4.5.

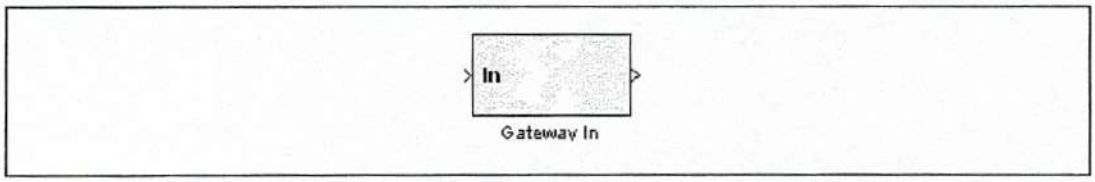


Figura 4.5 Bloque gateway in.

Este bloque se encuentra en las siguientes bibliotecas de Xilinx: elementos básicos, tipos de datos e índice. El bloque de Xilinx Gateway In se encarga de que los datos proporcionados por el usuario de Simulink puedan ser entendibles para Xilinx. Estos bloques convierten el número entero de Simulink, doble y los tipos de datos del punto fijo en el tipo de punto fijo del generador del sistema. Cada bloque define un puerto a un nivel superior de la entrada en el diseño de HDL generado por el System Generator. Este bloque por lo tanto se encarga de acomodar los datos que proveen el primer bloque en 8 bits para que pueda ser leído por los siguientes.

El siguiente bloque es un registro este nos sirve para ir pasando los datos en forma sincronizada para el buen manejo de los mismos. El bloque lo podemos encontrar en las siguientes bibliotecas de Xilinx Blockset: elementos básicos, control lógico, memoria e índice. El bloque de registro Xilinx modela un flip flop D, teniendo latencia de un periodo de muestreo. En la figura 4.6 nos muestra el bloque de registro

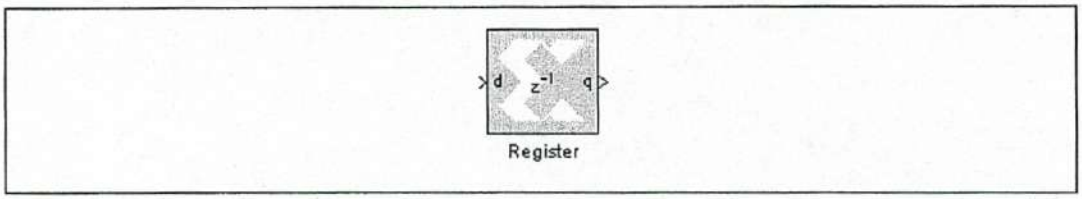


Figura 4.6 Bloque de registro

Interfase del bloque

El bloque tiene un Puerto de entrada y un Puerto opcional de reset. El valor inicial de salida es especificado por el usuario en los parámetros de bloques en la caja de dialogo. Los datos presentados en la entrada aparecerán en la salida después de un período de la muestra. En el reset, el registro asume el valor inicial especificado en la caja de diálogo de los parámetros. La figura 4.7 nos ilustra la caja de los parámetros del bloque de registro.

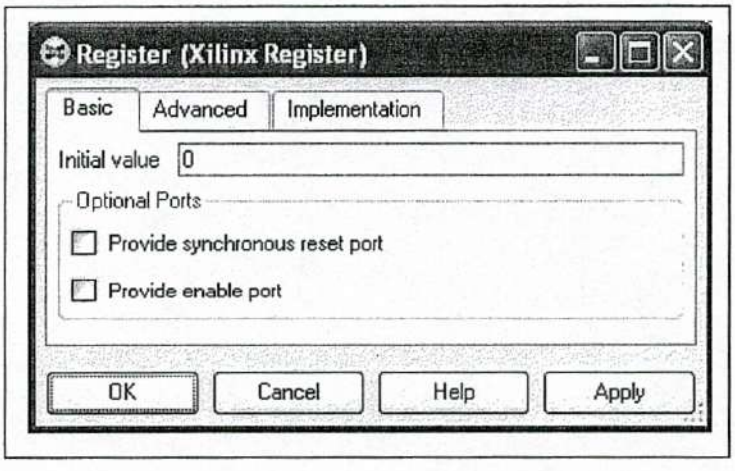


Figura 4.7 Parámetros del bloque de registro.

4.2.2 Almacenamiento de la información de la imagen.

El Bloque que nos servirá para acomodar y alinear los datos antes de ser filtrados es el Virtex2 5 Line Buffer (Imaging). Este nos sirve para acomodar los datos que entren coordinados para después introducirlos en la mascara. En la figura 4.8 se ilustra el bloque de almacenamiento Virtex2 5 Line Buffer.

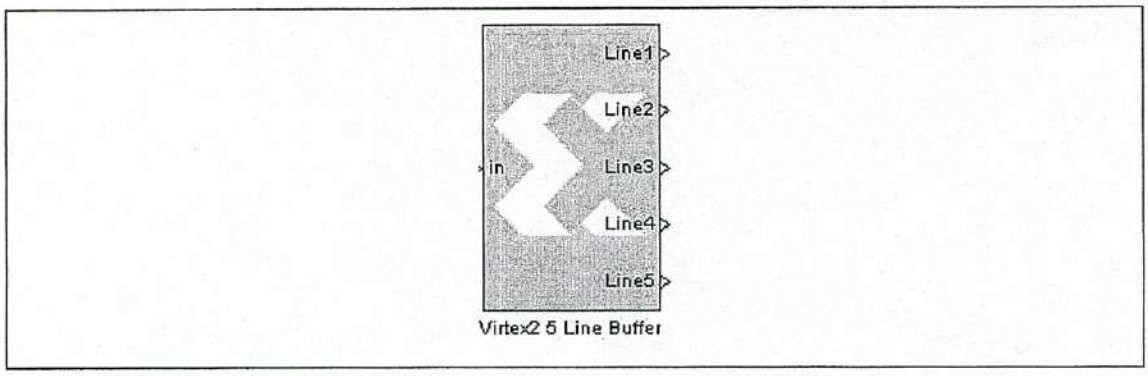


Figura 4.8 Bloque de almacenamiento Virtex2 5 Line Buffer (Imaging)

La línea de entrada del bloque Virtex2 5 Line Buffer de Xilinx se refiere a un bloque de almacenadores internos que protegen a una corriente secuencial de píxeles para construir 5 líneas de salida. Cada línea es retrasada por N muestras, donde N es la longitud de la línea. La línea 1 es retrasada 4*N muestras cada una de las líneas siguientes se retrasa por menos muestras N y la línea 5 es una copia de la entrada. El Xilinx Virtex2 Line Buffer es un bloque proporcionado por Xilinx el cual se localiza en Imaging library de Xilinx Reference Blockset. En la figura 4.9 se ilustra la caja de parámetros para el bloque del almacenador Virtex2 5 Line Buffer.

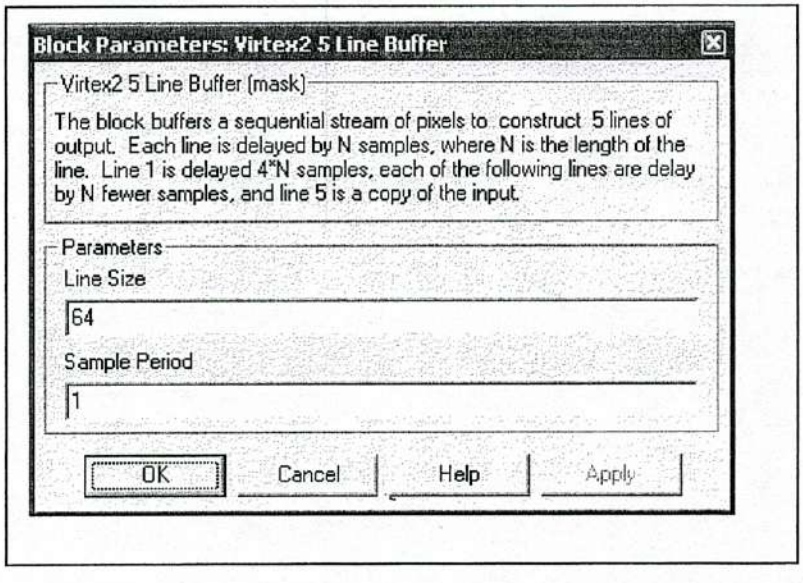


Figura 4.9: Virtex2 5 Line Buffer parámetros del bloque caja de dialogo

Los parámetros especificados en este bloque de referencia son:

Tamaño de la línea: numero de muestras en cada línea a ser retrasados.

Periodo de muestreo: rango de muestreo en el cual el bloque funcionara.

4.2.3 Aplicación de la mascara a la imagen.

Para la aplicación del filtro utilizamos el bloque 5x5 Filter (Imaging) en este bloque entran los datos de la imagen que fue leída al principio y aquí se aplica la convolución son mascaras de 5x5 en lo siguiente describimos el bloque y su función y parámetros. En la figura 4.10 se ilustra el bloque 5x5 Filter (Imaging).

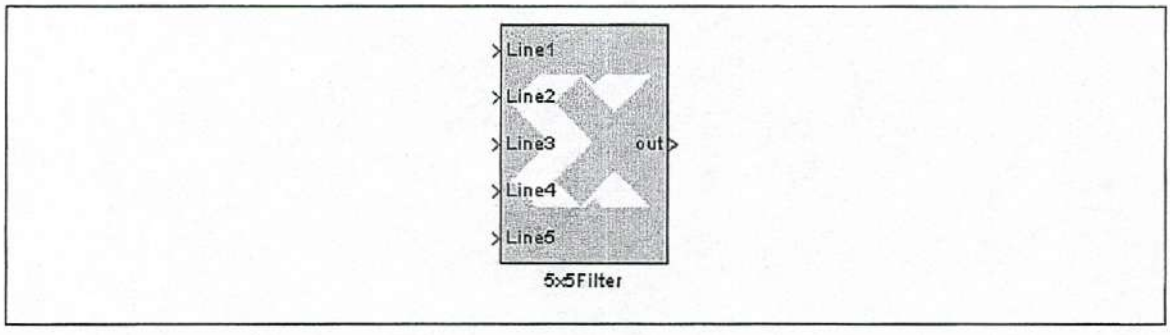


Figura 4.10 Bloque 5x5 Filter (Imaging)

El bloque Xilinx 5x5 Filter es implementado usando un filtro FIR MAC de 5 etapas. Nueve diferentes filtros 2-D se han proporcionado para filtrar imágenes en escala de grises. El filtro puede ser seleccionado cambiando el parámetro de la máscara en el bloque del filtro 5x5 los coeficientes del filtro 2-D son almacenados en un bloque de RAM, y el modelo no hace ninguna optimización específica para estos coeficientes. Puedes sustituir tus propios coeficientes y tu factor de posicionamiento modificando la máscara del bloque del filtro 5x5, bajo la etiqueta de inicialización. Los coeficientes usados se demuestran abajo para los 9 filtros. La salida del filtro es multiplicada por el factor de posicionamiento nombrado. En las figuras 4.11, 4.12 y 4.13 se muestran las diferentes mascaras 5x5 utilizadas para hacer el filtrado dentro del bloque filtro 5x5 estos datos son almacenados dentro de una memoria interna.

```
edge = [ 0  0  0  0 0; ...
0 -1 -1 -1 0; ...
0 -1 -1 -1 0; ...
0  0  0  0 0];
edgeDiv = 1;
```

```
sobelX = [ 0  0  0  0 0; ...
0 -1  0  1 0; ...
0 -2  0  2 0; ...
0 -1  0  1 0; ...
0  0  0  0 0];
sobelXDiv = 1;
```

```
sobelY = [ 0  0  0  0 0; ...
0  1  2  1 0; ...
0  0  0  0 0; ...
0 -1 -2 -1 0; ...
0  0  0  0 0];
sobelYDiv = 1
```

Figura 4.11 Mascaras de convolución de edge, sobel x y sobel y.

```
sobelXY = [ 0  0  0  0 0; ...
0  0 -1 -1 0; ...
0  1  0 -1 0; ...
0  1  1  0 0; ...
0  0  0  0 0];
sobelXYDiv = 1;
```

```
blur = [ 1  1  1  1 1; ...
1  0  0  0 1; ...
1  0  0  0 1; ...
1  0  0  0 1; ...
1  1  1  1 1];
blurDiv = 1/16;
```

```
smooth = [ 1  1  1  1 1; ...
1  5  5  5 1; ...
1  5 44  5 1; ...
1  5  5  5 1; ...
1  1  1  1 1];
smoothDiv = 1/100;
```

Figura 4.12 Mascaras de convolución de sobel xy, blur y smooth.

```
sharpen = [ 0  0  0  0 0; ...
0 -2 -2 -2 0; ...
0 -2 32 -2 0; ...
0 -2 -2 -2 0; ...
0  0  0  0 0];
sharpenDiv = 1/16;
```

```
gaussian = [1 1 2 1 1; ...
1 2 4 2 1; ...
2 4 8 4 2; ...
1 2 4 2 1; ...
1 1 2 1 1];
gaussianDiv = 1/52;
```

```
identity = [ 0  0  0  0 0; ..
0  0  0  0 0; ...
0  0  1  0 0; ...
0  0  0  0 0; ...
0  0  0  0 0];
identityDiv = 1;
```

Figura 4.13 Mascaras de convolución de sharpen, gaussian e identify.

En la figura 4.14 se ilustra la caja de parámetros del bloque de filtro 5x5.

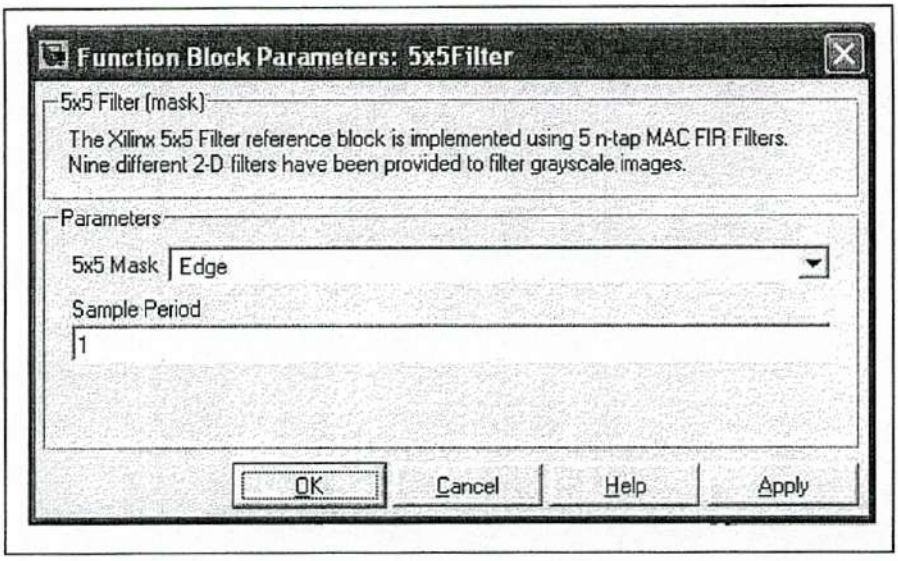


Figura 4.14 Bloque de parámetros de filtro 5x5

Los Parámetros especificados por el bloque son:

Mascara 5x5: los coeficientes para un Edge, Sobel X, Sobel Y, Sobel X-Y, Blur, Smooth, Sharpen, Gaussian, o Identity filter pueden ser seleccionado.

Periodo de muestreo: el periodo de muestreo al cual es requerido que funcione la señal de entrada.

4.2.4 Escritura de la imagen filtrada.

Los siguientes bloques que vamos a utilizar son los finales del sistema tenemos otro registro que ordena los datos para pasar al gateway out que el ordena los datos para así pasar al bloque mas importante de esta parte final que es el simout este bloque escribe un arreglo para poder determinar el formato de salida de la imagen ya filtrada.

Gateway Out

Este bloque se encuentra en las siguientes bibliotecas Xilinx Blockset: elementos básicos, tipos de datos, e índice. En la figura 4.15 se ilustra el bloque gateway out

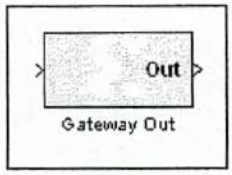


Figura 4.15 Bloque gateway out

Este bloque convierte el tipo de datos del punto fijo 8 bits del generador del sistema en el doble de Simulink. En el siguiente bloque encontramos el bloque simout este escribe datos desde el espacio del trabajo. Lo puedes encontrar en la biblioteca Sink. En la figura 4.16 a) se muestra el bloque de escritura de imagen y en la figura 4.16 b) se muestra el bloque de escritura de imagen con la imagen ya incluida.

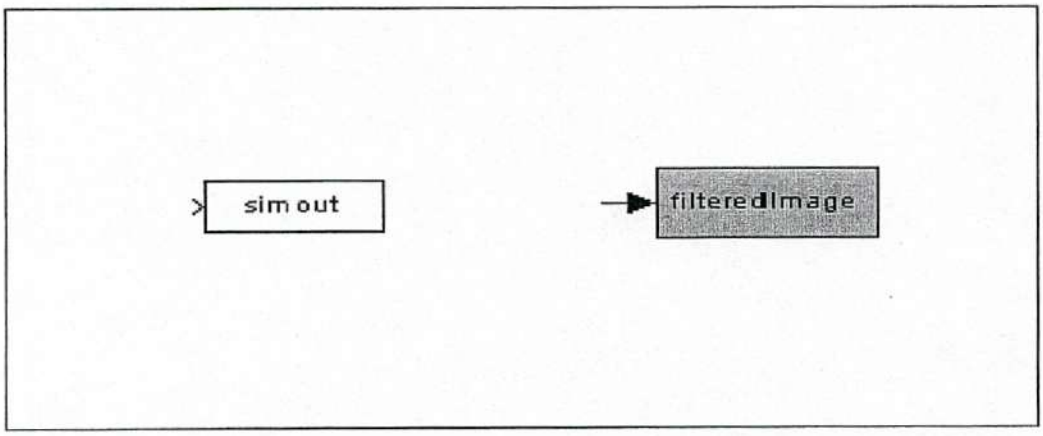


Figura 4.16 a) Bloque de escritura de imagen b) bloque de escritura de imagen ya con la imagen escrita

El bloque escribe su salida para un arreglo o estructura que tiene el nombre especificado por el parámetro conocido variable del bloque. El nombre que es especificado es filteredImage y este es el arreglo que escribe. En la figura 4.17 se ilustra el programa de escritura de imagen el cual al existir ciertas variables genera la matriz de datos para así tener la imagen filtrada.

```
if (exist('filteredImage','var') & exist('lineSize','var') & exist('NPixels','var'))
   filteredImageSize=size(filteredImage);
   designLatency = 20+2*lineSize;
   if ((~isempty(filteredImage)) & (filteredImageSize(1) >= (designLatency+NPixels-1)))

      % Reshape Simulink Output into a 2-D Image
      %rawImage = uint8(floor(reshape(filteredImage(designLatency:designLatency+NPixels-
1), lineSize, lineSize)));
      rawImage = uint8(floor(reshape(filteredImage(designLatency:designLatency+NPixels-
1), lineSize, 512)));

      % Plot Original and Filtered Images
      h = figure;
      clf;
      colormap(gray(256));

      set(h,'Name',' Filtering Results');
      subplot(1,2,1);
      image(grayScaleImage), ...
         axis equal, axis square, axis off, title 'Original Image';


      subplot(1,2,2);
      image(rawImage), axis equal, axis square, axis off;
      filterTitle = 'Filtered Image';
      title(filterTitle)
      colormap(gray(256));
   end
end
```

Figura 4.17 Programa de escritura de imagen.

4.2.5 SISTEMA DE FILTRADO DE BLOQUES DE SIMULINK

El sistema de bloques cada uno prediseñado por Xilinx este realiza el proceso de filtrado de imágenes el cual esta compuesto por la lectura de la imagen del programa la conversión de datos para ser entendibles por los bloques de Xilinx de doble ha punto fijo 8 bits después un almacenador de datos que esta conectado directamente al bloque de filtrado de las imágenes otra conversión de datos ahora de punto fijo 8 bits a doble para ser entendible por Simulink y por ultimo el bloque de escritura de imagen. En la figura 4.19 se muestra el sistema de filtrado de bloques de Simulink.

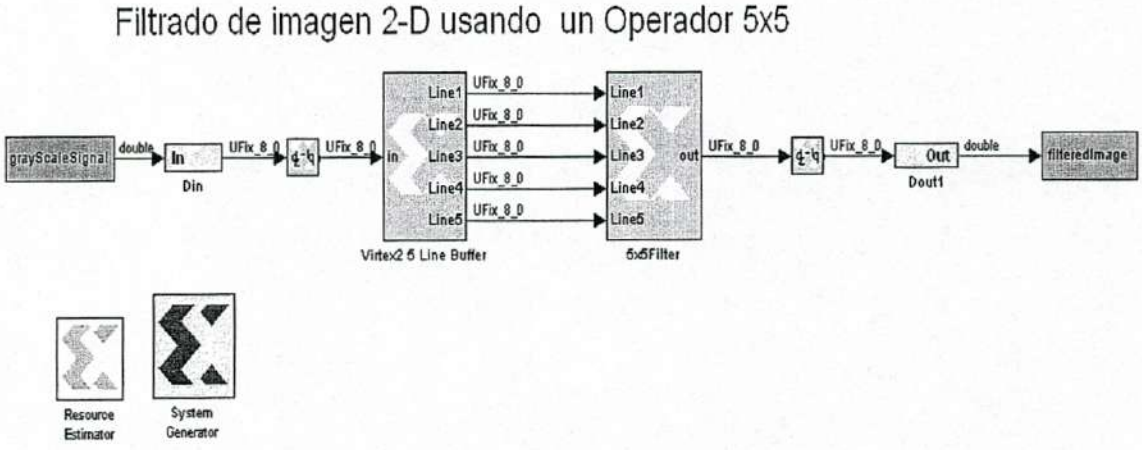


Figura 4.18 Sistema de filtrado en bloques de simulink completo

Si este sistema de filtrado se desea implementar en el dispositivo FPGA Virtex II Pro de Xilinx ocupará 309 celdas, 5 multiplicadores dedicados, y 5 bloques de ram que opera a 213MHz. El rendimiento de procesamiento del diseño es MHz / 5 = 42.6 millones pixels/ second. Para una imagen de 64x64, este es 42.6x10^6/(64x64) = 10,400 frames/sec. Para una imagen de 256x256 el rendimiento del procesamiento es de 650 frames /sec, y para una imagen de 512x512 es de 162 frames/sec.

CAPITULO 5

5. RESULTADOS

Como se planteó al principio del capítulo anterior nuestro objetivo es desarrollar un sistema para procesar la imagen de un patrón de iluminación láser en escala de grises que presenta ruido de alta frecuencia indeseable. El requerimiento de nuestro problema consiste en aplicar un filtro que permita que la imagen no presente manchas extras fuera del patrón de iluminación que puedan confundir al algoritmo de detección de máximos para su posterior procesamiento en la reconstrucción de objetos tridimensionales. Nuestra propuesta consiste en elegir un filtro pasabajas adecuado.

Después de implementar una serie de filtros pasabajas a la imagen con el patrón difuso, encontramos que por sus características el filtro Gaussiano, que es un filtro promediador, presenta el mejor desempeño.

En la figura 5.1 a) se muestra la imagen láser difusa y en la figura 5.1 b) se muestra la imagen resultante después de aplicar el filtro pasabajas Gaussiano. Obsérvese como el patrón resultante ha sido suavizado.

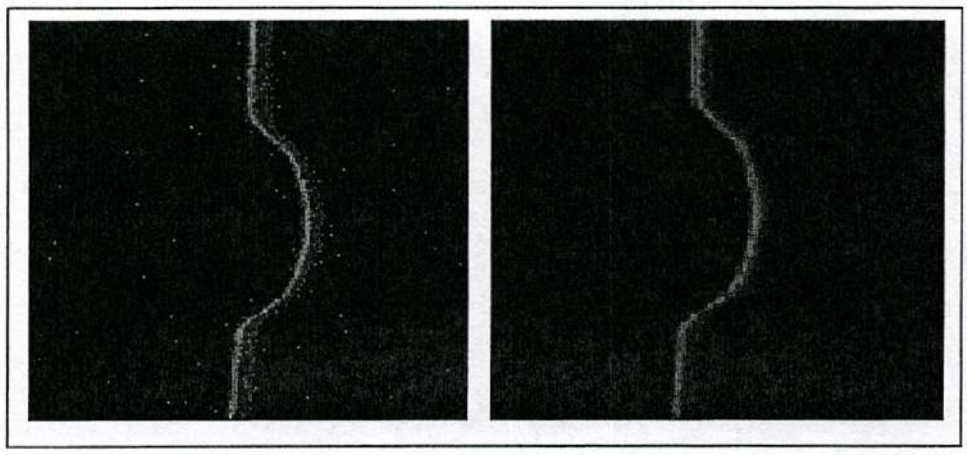


Figura 5.1 a) imagen láser difusa b) imagen láser filtrada(Gaussian).

Para mostrar en detalle el efecto de nuestro sistema de filtrado, en las siguientes figuras 5.2 a y b se presenta un acercamiento del patrón de iluminación original y el procesado,

respectivamente. Como podemos observar las motas de luz producidas por la fuente láser han sido atenuadas. De esta manera, el algoritmo de detección de máximos puede ser aplicado adecuadamente.

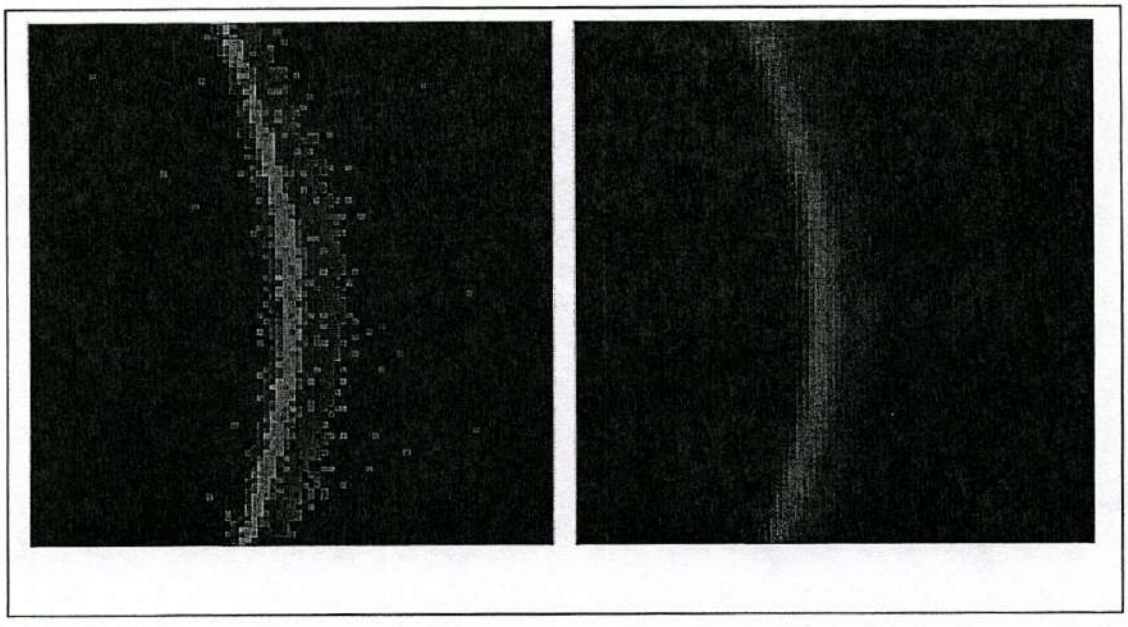
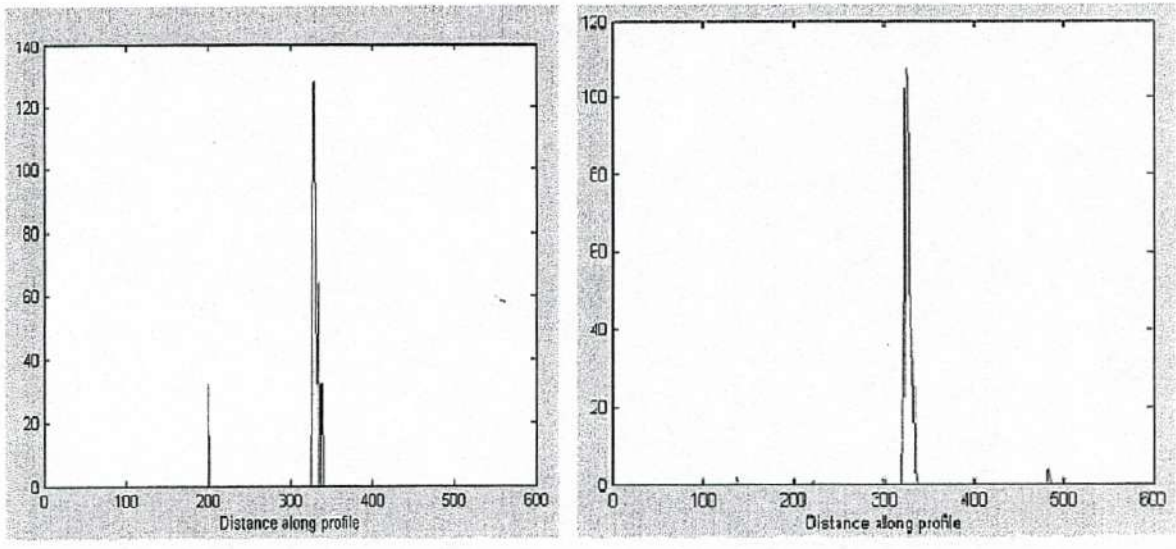


Figura 5.2 a) imagen láser con zoom difusa b) imagen láser con zoom filtrada (Gaussian).

En la figura 5.3 a y b podemos apreciar mediante una grafica del barrido transversal del patrón de iluminación original.



5.3 a)Grafica de la imagen antes de aplicarle el filtro b) Grafica de la imagen al aplicarle el filtro.

CAPITULO 6

6. CONCLUSIONES

Se ha presentado un diseño de sistema de bloques en el programa ya mencionado Simulink de MatLab junto con los beneficios que este mismo trae. En esta tesis se presento un sistema de bloques el cual se ha diseñado para hacer el filtrado de algunas imágenes láser en escala de grises, el cual pueda ser eficiente y flexible ademas este sistema nos puede proporcionar los códigos de los filtros para ser implementados en dispositivos programables en este caso de Xilinx (FPGA, DSP).

Aunque el sistema fue desarrollado para procesar un patrón de iluminación láser. Su aplicación se puede extender al procesado de imágenes en general. El proceso de filtrado de imágenes del sistema esta compuesto por la lectura de la imagen del programa la conversión de datos para ser entendibles por los bloques de Xilinx de doble ha punto fijo después un almacenador de datos que esta conectado directamente al bloque de filtrado de las imágenes otra conversión de datos ahora de punto fijo a doble para ser entendible por Simulink y por ultimo el bloque de escritura de imagen. Los filtros que pueden ser implementados por el sistema son algunos de detección de bordes como los sobel x, sobel y, sobel xy y el edge también los filtros pasa bajas como el smoothing, el blur y el gaussian y por ultimo un filtro pasa altas como lo es el filtro sharpean.

Este filtro ocupa 309 celdas, 5 multiplicadores dedicados, y 5 bloques de ram de las partes de una tarjeta Xilinx xc2v250-6 y opera a 213MHz. El rendimiento de procesamiento del diseño es MHz / 5 = 42.6 millones pixels/ second. Para una imagen de 64x64, este es 42.6x10^6/(64x64) = 10,400 frames/sec. Para una imagen de 256x256 el rendimiento del procesamiento seria 650 frames /sec, y para una imagen de 512x512 seria 162 frames/sec.

BIBLIOGRAFÍA

[1] William K. Pratt, *Digital Image Processing*, Third edition, Adisson Wiley, 2001, 735 pág.

[2] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, second edition, Pretince Hall, 2001, 779 pág.

[3] Rulph Chassaing, Digital Signal Processing and Applications with the C6713 and C6416 DSK, Wiley, 2005, 518 pág.

[4] Bernd Jahne, Digital Image Processing, Springer, 2002, 585 pág.

[5] C. Britton Rorabaugh, Digital Filter Desingner's Handbook, TAB Books Division of McGraw-Hill, Inc., 332 PÁG.

[6] B. A. Shenoi, INTRODUCCION TO DIGITAL SIGNAL PROCESSING AND FILTER DESIGN, WILEY-INTERSCIENCE, 2006, 423 pág.

[7] JAE S. LIM, Two Dimensional Signal and Image Processing, Prentice Hall, 694 pág.

[8] "Max" Maxfield, *The Design Warrior's Guide to FPGAs Divices, Tools and Flows*, Newnes, Elsevier, 2004, 542pág.

[9] Steve Winder, Analog and Digital Filter Desing, Second Edition, Newnes, 2002, 450 pág.

[10] Katsuhiko Ogata, *INGENIERIA DE CONTROL MODERNA,* Prentice Hall, Pearson, tercera edicion, 1998, 997 pág.

[11] Anil K. Jain, Fundamentals of Digital Image Processing, Prentice Hall, *1989, 565 pág.*

[12] *Les Thed, Practical Analog and Digital FILTER DESIGN, Artech House, Inc., 2004, 267 pag.*

[13] K. S. Thyararajan, *Digital Image Processing with Application to Digital Cinema, Focal Press, 2006*, 378 pag.

[14] *Jan Teuber,Digital Image Processing, Prentice Hall*, 1993, 246 pag.

[15] A. Erhardt, Ferron, Theory and Applications of Digital Image Processing, *University of Applied Sciences Offenburg, 2000, 52 pag.*

A. APENDICE

A.1 Descripción de la tarjeta XUP Virtex-II Pro

A.1.1 Descripción General

El sistema de desarrollo XUP Virtex-II Pro proporciona una plataforma de hardware avanzada que consiste en un alto desarrollo de plataforma FPGA Virtex-II Pro rodeado por una colección de componentes periféricos que pueden ser usados para crear un sistema complejo y para demostrar la capacidad de la plataforma FPGA Virtex-II Pro.

A.1.2 Diagrama de bloques

En la Figura A.1 se muestra un diagrama de bloques del sistema de desarrollo XUP Virtex-II Pro.

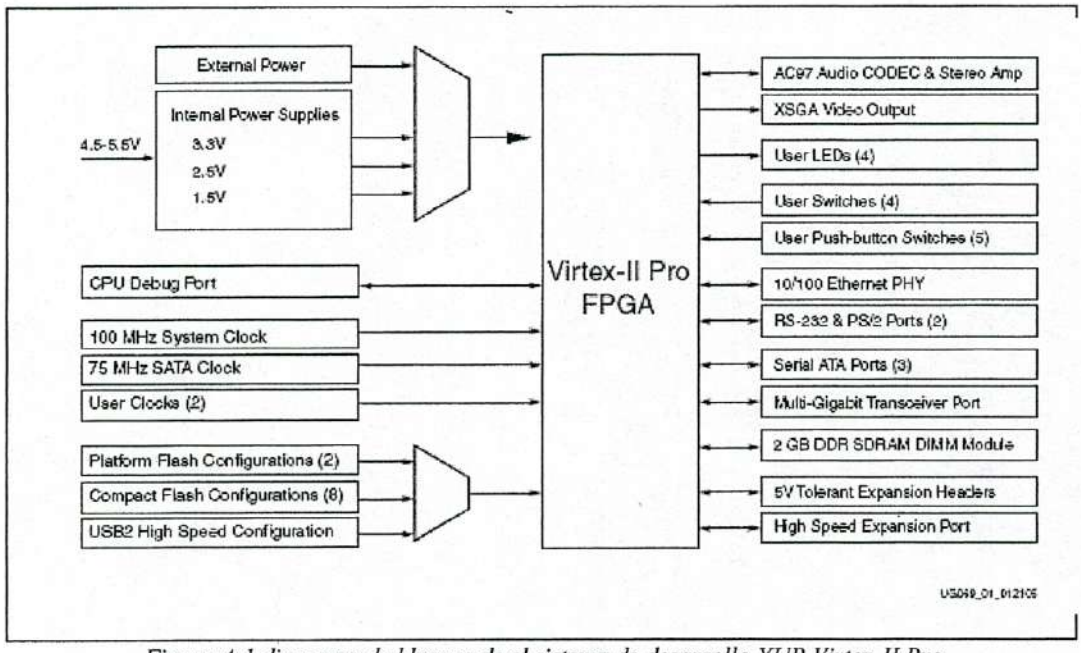


Figura A.1 diagrama de bloques de el sistema de desarrollo XUP Virtex-II Pro

En la figura A.2 se muestra una fotografía real de la tarjeta de desarrollo XUP Virtex-II Pro

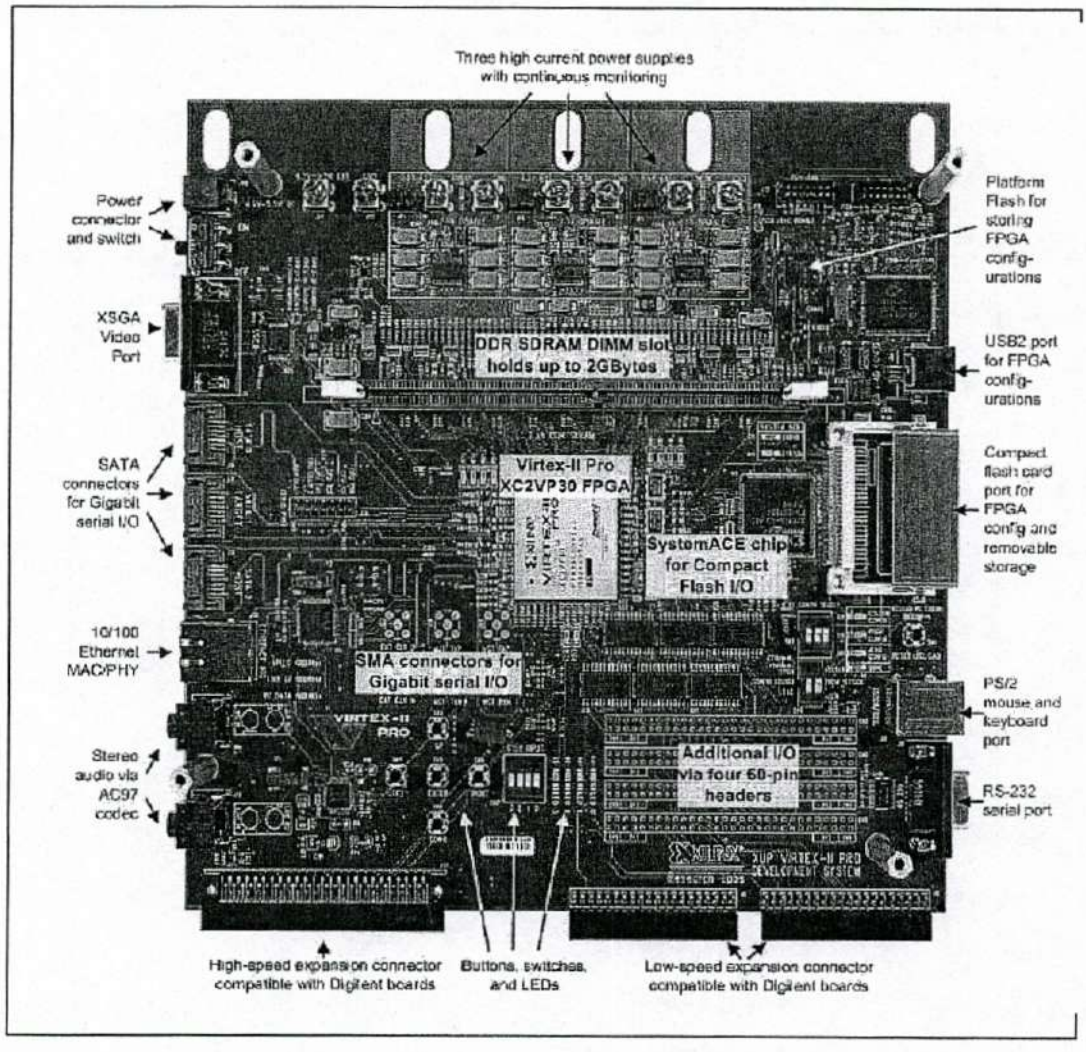


Figura A.2 fotografía real de la tarjeta de desarrollo XUP Virtex-II Pro

A.2 Características de XC2VP30

Características	XC2VP30
Celdas	13969
Tamaño del arreglo	80 x 46
RAM distribuida	428 Kb

Bloques multiplicadores	136
Bloques de RAM	2448 Kb
DCMs	8
PowerPC RISC Cores	2
Multi-Gigabit Tranceivers	8

A.2.1 Fuentes de poder y configuración FPGA

El sistema de desarrollo XUP Virtex-II Pro es accionado desde una fuente de poder regulada de 5V. Las fuentes de poder en la tarjeta conmutan a 3.3V, 2.5V, y 1.5V para el FPGA, y los componentes periféricos y los reguladores lineales accionan el MGTs.
La tarjeta tiene proporcionada para la medida de corriente actual para todas las fuentes de alimentación digitales de FPGA, así como el uso de la energía externa si la capacidad de las fuentes de alimentación a bordo de la conmutación se excede. El sistema de desarrollo XUP Virtex-II Pro proporciona algunos métodos de configuración del FPGA Virtex-II Pro. Los datos de configuración pueden ser originados desde la plataforma interna flash PROM (dos potenciales configuraciones), y la configuración externa desde la plataforma embebida del cable USB o la interfase del puerto paralelo.

A.2.2 Transreceptores multi-gigabit

Cuatro de los ocho transreceptores Multi-Gigabits (MGTs) que están presentes en el Virtex-II Pro FPGA se traen hacia fuera y pueden ser utilizados por el usuario.
Tres de los canales bidireccionales de MGT se terminan en el accesorio serial de la tecnología avanzada (SATA) y los cuatro canales terminan en los conectores miniatura secundarios (SMA). Los transreceptores MGT son equipados con una fuente de reloj de 75MHz que es independiente del sistema de reloj para soportar la comunicación estándar SATA.Una fuente de reloj MGT adicional esta disponible por el conector secundario SMA. Dos de los puertos con conectores SATA son configurados cono

puertos Host y los tres puertos SATA son configurados como un puerto objetivo para tener en cuenta el establecimiento de una red simple del tablero-a-tablero.

A.2.3 Sistema RAM

El sistema de desarrollo XUP Virtex-II Pro tiene proporcionado para la instalación de un estándar proveído por el usuario JEDEC dual en línea de 184 pines de doble rango de datos con sincronía dinámica del modulo de la memoria RAM.
La tarjeta soporta almacenadotes y módulos de memorias desalamacenadoras con una capacidad de 2GB o menos en cualquier organización de 64 bits o 72 bits. La organización de 72 bits debe ser usada si se detecta un error ECC y la corrección es requerida.

A.2.4 Regulador del sistema ACE Flash compacto

El sistema del regulador avanzado del la configuración ambiente (sistema ACE) controla y maneja datos de la configuración del FPGA. El control proporciona una interfase inteligente entre un canal del FPGA y varias fuentes que soportan la configuración. El regulador tiene algunos puertos: el puerto compact flash, el puerto de configuración del JTAG, el puerto para el microprocesador (MPU) y el puerto de prueba JTAG. El sistema de desarrollo XUP Virtex–II pro soporta un simple regulador de sistema ACE. El puerto de configuración del JTAG se conecta con el FPGA y a conectores de expansión. El puerto de prueba de JTAG se conecta al puerto jefe de JTAG y al USB2 e interfase CPLD, y los puertos MPU directamente para el FPGA.

A.2.5 Interfase veloz de Ethernet

El sistema de desarrollo XUP Virtex-II Pro proporciona una interfase veloz de Ethernet IEEE transreceptor que soporta ambas aplicaciones 100BASE-TX y 10BASE-T. Este soporta la operación completa duplex a 10 Mb/s y 100 Mb/s, con auto negociación y detección paralela. El PHY proporciona una interfase media independiente (MII) por el accesorio para el regulador del acceso media (MAC) implementado en el FPGA cada tarjeta es equipada con un numero de serial de silicio que lo identifica únicamente cada tarjeta con 48 bits números de serie. Este número de serie es recuperado usando el protocolo "1-Wire". Este número de serial puede ser usado como el sistema de dirección

MAC. El sistema de desarrollo XUP Virtex-II Pro proporciona tres puertos seriales: uno simple RS-232 y dos puertos PS/2. El puerto es configurado como un DCE con hardware handshake usando un estándar de conector serial DB-9. Este conector es típicamente usado para comunicaciones con una computadora Host usando un cable serial estándar de 9 pin para el puerto COM. Los dos puertos PS/2 pueden ser usados para añadir un teclado o un mouse para el sistema de desarrollo Virtex-II Pro. Todos los puertos de seriales son equipados con circuitos cambiadores de nivel porque los FPGAs Virtex-II Pro no pueden interconectar directamente a los niveles voltaicos requeridos por RS-232 o PS/2.

A.2.6 Uso de LEDs, Switches, y Push Buttons

Un total de Cuatro LEDs son proporcionados para propósitos definidos por el usuario. Cuando el FPGA entra en una lógica de 0, el LED correspondiente se enciende. Un DIP switched 4 posiciones y 5 push buttons son proporcionados para entradas de usuario. Si el DIP switched esta en alto, o cerrado o encendido, o el push button es presionado una lógica 0 es vista por la FPGA, de otra manera una lógica 1 es indicada.

A.2.7 Conectores de expansión

Un total de 80 pins Virtex-II Pro de entrada y salida son traídos fuera para cuatro cabeceras de 60 pins proporcionadas por el usuario y dos conectores de ángulo recto de 40 pin para que el usuario defina su uso. Las cabeceras de 60 pin son diseñados para aceptar los conectores de cable de cinta, con cada segunda señal una tierra para la señal integral. Algunas de estas señales son compartidas con las montadas enfrente de los conectores de ángulo recto.

A.2.8 Salida XSGA

El sistema de desarrollo XUP Virtex-II Pro incluye un video DAC y 15 pin de alta densidad el conector D-sub para soportar la salida XSGA. El video DAC puede operar con un reloj de píxel de arriba de los 180MHz. Este permite una salida compatible VESA de 1280 x 1024 a 75 Hz restaurada y una máxima resolución de 1600 x 1200 a 70 Hz restaurada.

A.2.9 Audio CODEC AC97

Una audio CODEC y un amplificador de potencia stereo son incluidos sobre el sistema de desarrollo XUP Virtex-II Pro para proporcionar alta calidad en trayectoria de audio y proporcionar todo de la funcionalidad análoga en un sistema de audio en la PC. Estas características son un completo duplex stereo ADC y DAC, con un mezclador analógico, combinar las entradas de las líneas de nivel, entrada de micrófono y datos PCM.

A.2.10 Puerto de depurado

El FPGA es equipado con una cpu interfase de eliminador de errores y una cabecera de 16 pin. Este conector puede ser usado en conjunción con las herramientas de los terceros, el Xilinx Parallel Cable IV o el cable USB de la plataforma Xilinx para eliminar errores de software como funciona en cualquier base del procesador de PowerPC 405. ChipScope Pro™ puede también ser usado para desarrollar un depurador en tiempo real y verificador de diseño del FPGA. ChipScope Pro inserta una lógica analizadora, bus analizador, y entrada-salida virtual discreta en los códigos del software en el diseño del FPGA. Estos códigos permiten el diseño para ver todas las señales internas y nodos junto con el FPGA incluyendo el bus local procesador (PLB) o el bus periférico sobre el chip (OPB) soportando los códigos del PowerPC 405. Las señales son capturadas y traídas fuera por la plataforma embebida del cable USB.

A.2.11 Interface de programación de USB 2

El sistema de desarrollo XUP Virtex-II Pro incluye un microcontrolador embebido USB 2.0 capaz de tener comunicaciones con cualquier HOST USB a alta velocidad (480 Mb/s) o una máxima velocidad (12 Mb/s). Esta interfase es usada para programar o configurar Virtex-II Pro FPGA en el modo Boundary-Scan (IEEE 1149.1/IEEE 1532). Las velocidades del reloj de la tarjeta son seleccionables de 750 Khz. para 24 MHz. El microcontrolador USB 2 tiene fijación para una computadora de escritorio o una laptop con un cable USB A-B de alta velocidad.