FILTROS ESPACIALES

La característica esencial de las operaciones de píxel tratadas en el capitulo anterior fue que el nuevo valor del píxel calculado finalmente depende única y exclusivamente del valor del píxel original y es ubicado en la misma posición. Aunque es posible realizar muchos efectos sobre imágenes utilizando operaciones de píxel existen condiciones bajo las cuales no es posible utilizarlas para poder generar determinados efectos tal y como se tiene en el caso de suavizado (véase figura 5.1) o detección de bordes de imágenes. Filtros pueden ser considerados como operaciones en donde el nuevo píxel calculado depende no únicamente del píxel original, sino de otros píxeles que están en una determinada vecindad en relación a el. Una observación importante es que este tipo de operaciones permiten encontrar el valor de un nuevo píxel a partir de un conjunto de píxeles provenientes de la imagen original si que esto signifique que exista un cambio en la geometría de la imagen resultante de tal forma que la relación entre las imágenes original y resultado sigue siendo al igual que en las operaciones de píxel 1 a 1.

(a) (b)

Figura 5.1 Si se utiliza una operación de píxel no puede realizarse efectos tales como el suavizado. (a) Imagen original y (b) imagen suavizada.

5.1 ¿QUE ES UN FILTRO?

En una imagen se observan lugares o píxeles donde localmente existe un cambio abrupto del nivel de intensidad ya sea que este aumente o disminuya significativamente. De manera contraria se presentan también lugares o píxeles en la imagen en donde la intensidad de la imagen permanece constante. Una primera idea de suavizado de una imagen es por ello que cada píxel simplemente sea remplazado por el promedio de sus vecinos.

Por lo que para calcular el valor del píxel de la imagen suavizada $I'(x,y)$ se emplea el píxel de la imagen original $I(x,y) = p_0$ más sus 8 píxeles vecinos $p_1, p_2, \ldots, p_8$ de la imagen original I y se calcula el promedio aritmético de esos nueve valores:

$$I'(x,y) \leftarrow \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9} \tag{5.1}$$

En coordenadas relativas ala imagen lo anterior podría ser expresado como:

$$I'(x,y) \leftarrow \frac{1}{9} \cdot \begin{bmatrix} I(x-1,y-1) & + & I(x,y-1) & + & I(x+1,y-1) & + \\ I(x-1,y) & + & I(x,y) & + & I(x+1,y) & + \\ I(x-1,y+1) & + & I(x,y+1) & + & I(x+1,y+1) & \end{bmatrix} \tag{5.2}$$

Lo cual podría ser descrito en forma compacta:

$$I'(x,y) \leftarrow \frac{1}{9} \cdot \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(x+i, y+j) \tag{5.3}$$

Este promedio local calculado muestra todos los elementos típicos presentes en un filtro. Realmente este filtro es el ejemplo de uno de los tipos de filtros mas utilizados, los llamados *filtros lineales.*

De lo anteriormente visto debe de quedar clara la diferencia entre las operaciones de píxel y las de filtro, sobre todo por que el resultado de las operaciones de filtro no solo depende de un solo píxel de la imagen original sino de un conjunto de ellos. Las coordenadas actuales del píxel en la imagen original (x,y) definen normalmente la región $R(x,y)$ de vecinos considerada. La figura 5.2 muestra las coordenadas del píxel actual y su relación con la región de vecinos considerada.

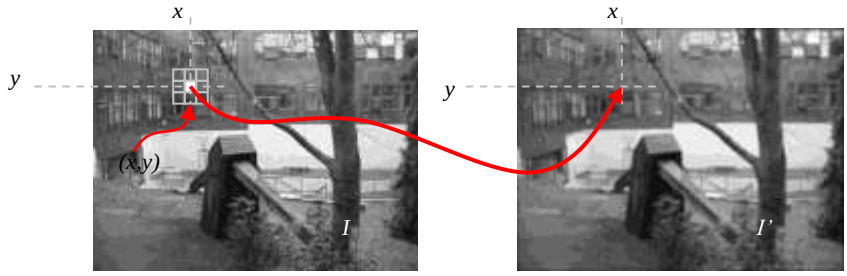


Figura 5.2 Principio de las operaciones de filtro. Cada nuevo píxel $I'(x,y)$ se calcula a partir de una determinada región $R(x,y)$ de vecinos cuyas coordenadas centrales se relacionan con las coordenadas del píxel a calcular.

El tamaño de la región del filtro $R(x,y)$ es un parámetro importante de un filtro, ya que determina cuantos y cuales vecinos del píxel de la imagen original serán considerados para calcular el nuevo píxel. En el ejemplo tratado anteriormente se utilizo un filtro de suavizado con una región 3x3, el cual se encuentra centrado sobre la coordenada (x,y). Con filtros mas grandes, 5x5, 7x7 o inclusive 31x31 píxeles se obtienen mayores índices de suavizado.

La forma de la región del filtro puede ser cualquiera, sin embargo la forma cuadrada es la mas utilizada por que además de facilidad de cálculo (ya que no es necesario calcular la rejilla de píxeles por alguna función tal y como sería necesario si se tratase de una circular) permite considerar píxeles vecinos en todas las direcciones, lo cual es una propiedad deseable de un filtro. También se podría además considerar de una manera diferente los valores de intensidad tomados en cuenta por el filtro, es decir puede asignarse un peso diferente a cada píxel que participe en la región dependiendo de su relación con el píxel central de tal forma que píxeles mas cercanos al píxel central se les confiera mayor importancia mientras que los mas alejados menor.

Considerando lo anterior podría haber diferentes alternativas para poder generar un filtro sin embargo se necesita un método sistemático que nos permita especificar, producir y utilizar lo. También puede observarse cómo fue tratado arriba la existencia de filtros que tienen características lineales y no lineales, que en forma implícita les impone ciertas propiedades a las operaciones que de ellos se desprenden. La diferencia entre ambos tipos de procesamiento es la forma en que el píxel dentro de la región $R(x,y)$ de procesamiento se encuentra conectado en la operación, pudiendo ser ésta lineal o no lineal. En lo que sigue del capítulo se tratara ambos tipos de filtros y se mostrarán algunos ejemplos prácticos.

Como fue visto las operaciones que serán tratadas en este capitulo se refieren a la suma de las multiplicaciones de cada píxel por los valores de una matriz de

coeficientes en una vecindad definida por la propia matriz. Esta matriz de coeficientes es conocida en la comunidad de visión por computadora como *filtro, máscara, kernel* o *ventana*.

5.2 FILTROS LINEALES ESPACIALES

Los filtros lineales son así designados por que los valores de intensidad de los píxeles dentro de la región de procesamiento se combinan linealmente para generar el píxel resultado. Un ejemplo de este tipo lo es el tratado en la ecuación 5.3, en donde nueve píxeles que forman parte de una región 3x3 se combinan linealmente en una suma multiplicada por el factor (1/9). Con el mismo mecanismo puede definirse un múltiple número de filtros, con diferentes comportamientos sobre una imagen, con el solo hecho de cambiar los pesos con los que los valores de los píxeles, que se encuentran dentro de la región de procesamiento, se combinan linealmente.

5.2.1 La matriz del filtro

Los filtros lineales quedan completamente especificados al definir el tamaño y la forma de la región de procesamiento, así como los correspondientes pesos o coeficientes que determinan la manera en que linealmente los valores de los píxeles de la imagen original se combinan. Los valores de estos coeficientes o pesos son definidos en la matriz de filtro $H(i, j)$ o simplemente mascarilla. El tamaño de la matriz $H(i, j)$ determina el tamaño de la región de procesamiento $R(x, y)$ del filtro y sus valores definen los pesos con los cuales los correspondientes valores de intensidad de la imagen son multiplicados para obtener su combinación lineal. Considerando lo anterior el filtro para suavizar una imagen tratado en la ecuación 5.3 puede ser definido por la siguiente matriz de filtro:

$$H(i,j) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.4}$$

De donde puede verse que cada uno de los nueve píxeles que se encuentran definidos en la mascarilla contribuye con un noveno del valor final del píxel resultado.

Fundamentalmente la matriz del filtro $H(i, j)$ al igual que una imagen es una función real discreta de 2 dimensiones, es decir $H : \square \times \square \rightarrow \square$. De esta manera la matriz del filtro posee su propio sistema de coordenadas, donde el origen es el centro de la misma, por consiguiente tendremos coordenadas positivas o negativas (véase figura 5.3). Importante es hacer notar que los coeficientes de la matriz del filtro fuera de sus coordenadas definidas son cero.

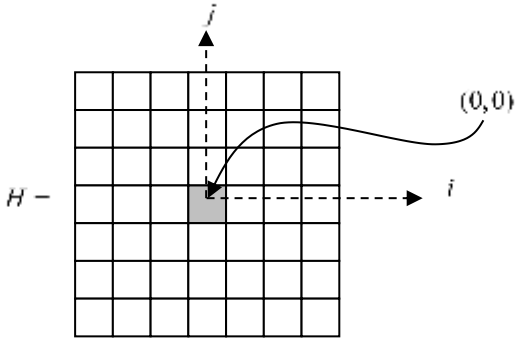


Fig. 5.3. Matriz del filtro o rejilla y su correspondiente sistema de coordenadas.

5.2.2 Operación de los filtros

El efecto de un filtro sobre una imagen queda completamente definido a través de los coeficientes de la matriz $H(i, j)$. La forma en como se realiza la operación del filtro sobre la imagen, como muestra la figura 5.4 es la siguiente:

Para cada píxel (x, y) de la imagen se llevan a cabo los siguientes pasos:

1. La matriz del filtro $H(i, j)$ es posicionada sobre la imagen original I en el píxel $I(x, y)$, de tal forma que su centro de coordenadas $H(0,0)$ coincida con el.

2. Los píxeles de la imagen que se ubiquen dentro de la región $R(x, y)$ definida por la matriz del filtro $H(i, j)$ son multiplicados por los coeficientes correspondientes a cada posición de la matriz del filtro y el resultado parcial de cada una de estas multiplicaciones es sumado.

3. El resultado del anterior procesamiento es colocado en la posición $I'(x, y)$ de la imagen resultado.

En otras palabras todos los píxeles de la imagen resultado $I'(x, y)$ son calculados utilizando la siguiente ecuación:

$$I'(x,y) = \sum_{(i,j)\in R(x,y)} I(x+i, y+j) \cdot H(i,j) \tag{5.5}$$

Para un filtro típico con una matriz de coeficientes de tamaño 3x3 se tendría concretamente la siguiente ecuación:

$$I'(x,y) = \sum_{j=-1}^{1}\sum_{i=-1}^{1} I(x+i, y+j)\cdot H(i,j) \tag{5.6}$$

Para todas las coordenadas de la imagen (x, y). Algo importante que debe ser considerado en este punto es que la anterior ecuación no es aplicable literalmente para todos los píxeles de la imagen, ya que en los bordes de la imagen existirán puntos en los cuales al centrar la matriz de coeficientes del filtro estos no tengan correspondencia y no puede establecerse en forma clara un resultado. Este problema será de nueva cuenta tratado con sus soluciones en la sección 5.5.2.

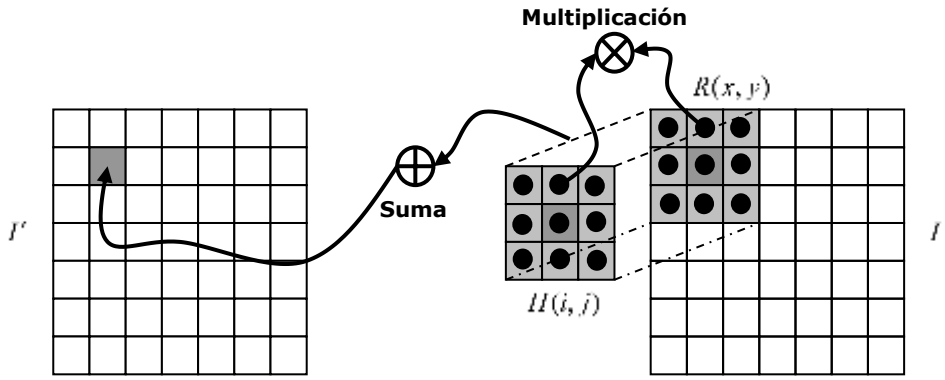


Figura 5.4 Filtros lineales. El origen de la matriz de coeficientes del filtro es colocado en el píxel (x, y) de la imagen original I. Los coeficientes del filtro $H(i, j)$ son multiplicados por los correspondientes píxeles de la imagen obteniéndose así 9 diferentes productos. Todos los productos son sumados y el resultado es colocado en la posición (x, y) de la imagen resultado I'.

5.3 CÁLCULO DE LAS OPERACIONES DE FILTRO EN MATLAB®

Después de que es ya conocida la forma de realizar las operaciones de filtro y ante la advertencia de observar sus efectos en los extremos de la imagen se mostrará la forma en que mediante programación en MatLAB es posible implementar este tipo de operaciones.

La aplicación de los filtros espaciales a las imágenes se lleva a cabo por la aplicación de una convolución entre el píxel de interés (x, y), sus vecinos y los coeficientes de la matriz de coeficientes del filtro. El mecanismo de procesamiento de filtrado espacial mostrado en la figura 5.4. La parte crucial de la programación de estas operaciones recae en la idea de la relación de las coordenadas del filtro con las de la imagen, ya que el proceso consiste en mover el centro del filtro de izquierda a derecha y de arriba a abajo realizando el procesamiento correspondiente en cada píxel de la imagen. La figura 5.5 muestra una ilustración de este proceso. Para un filtro de un tamaño *mxn* se tiene que $m = 2a+1$ y $n = 2b+1$, donde a y b son enteros no negativos. Lo anterior significa que los filtros son diseñados siempre en dimensiones impares, con la dimensión más pequeña de 3x3. Aunque esto no es un requerimiento, al trabajar con matrices de coeficientes impares facilita el procesamiento al existir un único centro y realizar el procesamiento en forma simétrica.

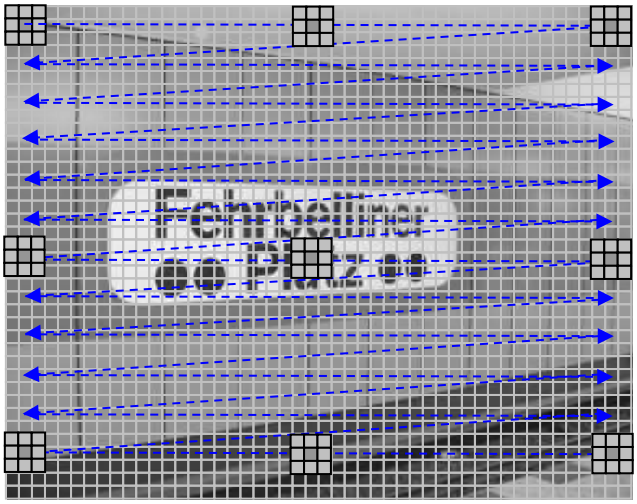


Figura 5.5 Aplicación del filtro espacial sobre una imagen.

El valor del nuevo píxel calculado en la posición (x, y) generado a partir del filtro depende de los valores de intensidad de la imagen que corresponden con los coeficientes del filtro en la región definida $R(x, y)$. Considerando que se tienen los coeficientes del filtro y los valores de intensidad de la imagen mostrados en la figura 5.6, el nuevo valor del píxel será:

$$I'(x,y) = I(x-1,y-1)\cdot H(-1,-1) + I(x-1,y)\cdot H(-1,0) + I(x-1,y+1)\cdot H(-1,1) + \quad (5.7)$$

$$I(x,y-1)\cdot H(0,-1) + I(x,y)\cdot H(0,0) + I(x,y+1)\cdot H(0,1) +$$
$$I(x+1,y-1)\cdot H(1,-1) + I(x+1,y)\cdot H(1,0) + I(x+1,y+1)\cdot H(1,1)$$

Un aspecto importante de implementación que debe ser tomado en cuenta es el efecto que tienen los bordes de la imagen en el procesamiento efectuado por el filtro ya que si se recorre la matriz de coeficientes del filtro centrada en todos los píxeles de la imagen habrá píxeles de la matriz de coeficientes H a los cuales no les corresponda un píxel de la imagen I. Mas adelante será tratado este problema y la forma en que puede ser resuelto. Una representación de este problema es mostrada en la figura 5.7.

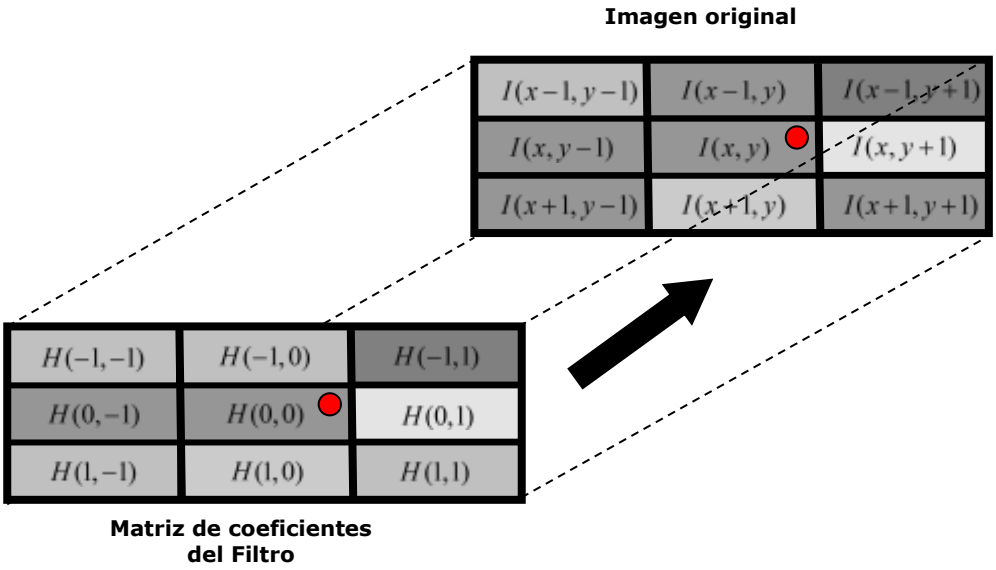


Figura 5.6 Proceso de filtrado espacial.

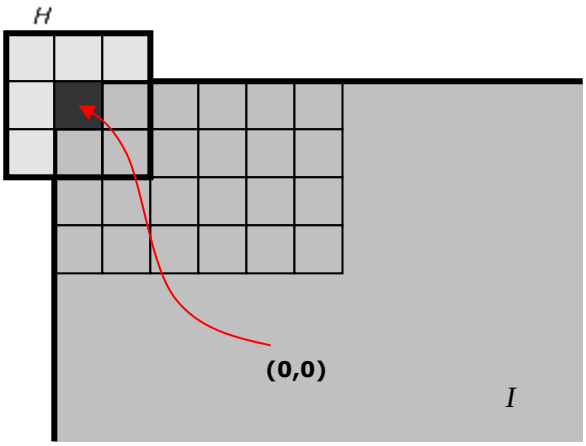


Figura 5.7 Problema de los bordes o fronteras de la imagen en el procesamiento de la convolución efectuada por el filtro. En el píxel (0,0) de la imagen si se centra el filtro habrá coeficientes de H sobre los cuales no existe correspondencia de píxeles en la imagen.

Una solución sencilla es simplemente considerar en el procesamiento un tamaño menor de la imagen, es decir si consideramos una imagen de *M*x*N* en lugar de recorrer el centro del filtro sobre la imagen de 1 a *N* y de 1 a *M* el recorrido se hará quitando los bordes de la imagen siendo ahora de 2 a *N*-1 y de 2 a *M*-1. De esta manera para cada coeficiente del filtro existirá un píxel correspondiente de la imagen. Para el procesamiento completo de la operación. Sin embargo como el borde de la imagen no será procesado por el filtro, se tienen dos alternativas: la primera es generar una imagen resultado más pequeña en comparación a la original de *M*-2x*N*-2, la otra opción es como paso inicial copiar la imagen original en la imagen resultado y al momento de hacer el procesamiento por medio del filtro, que se modifiquen todos los píxeles exceptuando los de las fronteras de la imagen, esto es que se recorra el filtro de los píxeles de 2 a *N*-1 y de 2 a *M*-1, con ello los píxeles que no fueron alterados permanecerán con la información original, permaneciendo además la imagen resultado del mismo tamaño que la original.

El programa 5.1 muestra el código en MatLAB para implementar el sencillo filtro 3x3 que suaviza la imagen mediante el cálculo del promedio de los píxeles vecinos tratado en la ecuación 5.4.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Implementación del filtro 3x3 que suaviza
% la imagen utilizando el promedio de los vecinos
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 19/12/07
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Se obtienen los valores de las dimensiones de la Imagen
[m n]=size(Im);
%Se convierte la imagen a double para evitar problemas
%en la conversión del Tipo de dato
Im=double(Im);
%Se copia la imagen original en la resultado para evitar el
%problema del procesamiento de la frontera de la imagen
ImR=Im;
%Se aplican la operación del filtro (véase ecuación 5.4)
%Se recorre toda la imagen con el filtro a excepción de la
%frontera
for r=2:m-1
    for c=2:n-1
        ImR(r,c)= 1/9*(Im(r-1,c-1)+Im(r-1,c)+Im(r-1,c+1)...
            +Im(r,c-1)+Im(r,c)+Im(r,c+1) ...
                +Im(r+1,c-1)+Im(r+1,c)+Im(r+1,c+1));
     end
end
% Se convierte la imagen a entero para poder desplegarla
ImR=uint8(ImR);
```

Programa 5.1. Implementación del filtro 3x3 suavizador en MatLAB.

5.4 TIPOS DE FILTROS LINEALES

La función de un filtro lineal se encuentra exclusivamente especificada por el valor de su matriz de coeficientes y debido a que la matriz de coeficientes puede consistir de diferentes valores podría existir un número infinito de filtros lineales. De lo anterior resultan las preguntas ¿Para qué tipo de efecto puede utilizarse un filtro? y ¿Para un determinado efecto cuáles son los filtros mas idóneos? En la practica existen 2 clases de filtros lineales: filtros de suavizado y filtros de diferencia.

5.4.1 Filtros de suavizado

Los filtros tratados en las secciones precedentes tienen como consecuencia algún tipo de suavizado sobre la imagen original. Todo filtro lineal el cual su matriz de coeficientes consiste solo de coeficientes positivos realiza de alguna manera un efecto de suavizado sobre una imagen, ya puede comprobarse que el resultado del

mismo es una versión escalada del promedio de la región $R(x, y)$ cubierta por el filtro.

El Filtro "Box"

El filtro "Box" es el más sencillo y antiguo de todos los filtros que realizan suavizado sobre una imagen. La figura 5.8 muestra (a) la función tridimensional, (b) bidimensional y (c) mascarilla implementada por este tipo de filtro. El filtro "Box" debido a los bordes tan agudos que implementa y su comportamiento frecuencial resulta un filtro de suavizado, salvo por su sencillez, no muy recomendable. El filtro que como puede verse en la figura 5.8(a) parece caja (de ahí su nombre), afecta a cada píxel de la imagen de igual manera (ya que todos sus coeficientes tienen el mismo valor).

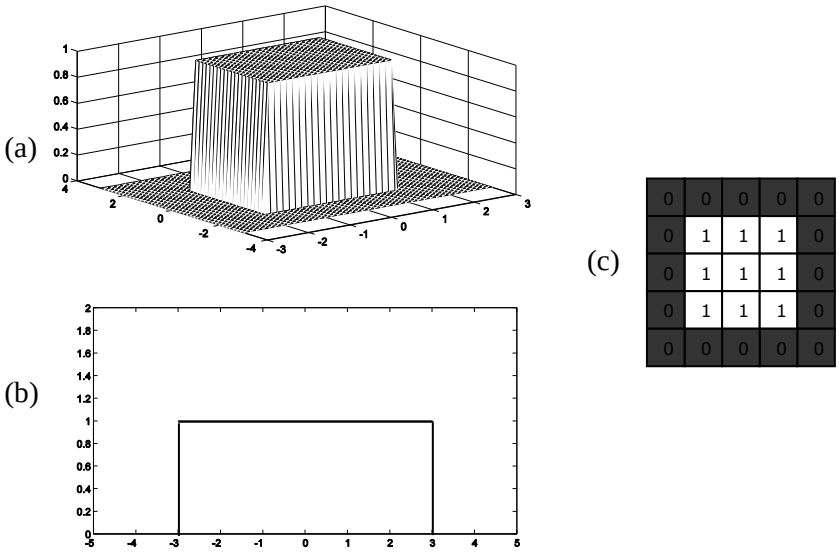


Figura 5.8 El filtro para suavizado "Box". (a) Función tridimensional, (b) bidimensional y (c) mascarilla que implementa el filtro.

Intuitivamente parecería en un filtro menos conveniente afectar a todos los píxel de la imagen de igual manera y mejor darle un mayor peso al punto central (que es el píxel principal) del filtro y considerar a los píxeles vecinos en una proporción menor. Una característica deseable de los filtros de suavizado lo es también el hecho de que sean isotrópicos, esto es que el efecto que se tengan sobre la imagen sea invariante a la rotación, lo cual no es el caso para el filtro "box" debido a su forma cuadrática.

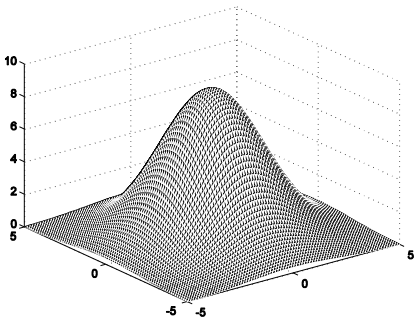
El filtro Gaussiano

El filtro Gaussiano corresponde a una función de Gauss bidimensional y discreta tal como:

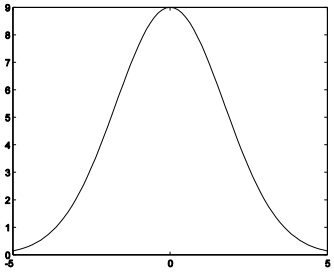
$$G_\sigma(r) = e^{-\frac{r^2}{2\sigma^2}} \qquad \text{o} \qquad G_\sigma(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.8}$$

Donde la desviación estándar σ representa el radio de cobertura de la función de Gauss tal y como se muestra en la figura 5.9.

(a)



(b)



(c)

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

Figura 5.9 El filtro Gauss para suavizado. (a) Función tridimensional, (b) bidimensional y (c) mascarilla que implementa el filtro.

El elemento central del filtro representa el peso máximo que participa en la combinación lineal de la operación, mientras que los valores de los demás coeficientes tienen menor influencia conforme estos se alejan del centro del filtro.

Una propiedad importante de este filtro es que es isotrópico[1]. Un problema que presenta el filtro de suavizado "Box" es que la atenuación que lleva a cabo en los píxeles de la imagen como producto de su operación, hace que también los bordes (regiones de la imagen donde existe un cambio de intensidad brusco), sean diluidos. El filtro de Gauss es menos dañino en este sentido permitiendo suavizar las regiones

[1] Invariante a la rotación

en donde los valores de intensidad son homogéneos sin diluir los bordes de la imagen.

5.4.2 Filtros de Diferencia

Si un filtro contiene como parte de sus coeficientes números negativos, su operación puede interpretarse como la diferencia de dos diferentes sumas: La suma de todas las combinaciones lineales de los coeficientes positivos del filtro menos la suma de todas las combinaciones lineales debidas a los coeficientes negativos, dentro de la región definida por el filtro $R(x,y)$. Si lo anterior lo expresamos matemáticamente quedaría especificado como:

$$I'(x,y) = \sum_{(i,j)\in R^+} I(x+i, y+j)\cdot |H(i,j)| - \sum_{(i,j)\in R^-} I(x+i, y+j)\cdot |H(i,j)| \qquad (5.9)$$

Con R^+ se refiere la parte del filtro con coeficientes positivos $H(i,j) > 0$ y a R^- ala parte del filtro con coeficientes negativos $H(i,j) < 0$. Como ejemplo se muestra el filtro 5x5 de Laplace de la figura 5.10 el cual realiza la diferencia entre el punto central (único coeficiente positivo con valor 16) y la suma negativa de 12 coeficientes con valores de -1 y -2. Los coeficientes restantes valen cero y no son considerados en el procesamiento. Este filtro implementa la función conocida como sombrero mexicano ("mexican hat"), teniendo la expresión:

$$M_\sigma(r) = \frac{1}{\sqrt{2\pi}\sigma^3}\left(1 - \frac{r^2}{\sigma^2}\right)e^{-\frac{r^2}{2\sigma^2}} \quad \text{ó} \qquad (5.10)$$

$$M_\sigma(x,y) = \frac{1}{\sqrt{2\pi}\sigma^3}\left(1 - \frac{x^2+y^2}{\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Mientras que con los filtros con coeficientes positivos el efecto que se obtiene es el de suavizado, en los filtros de diferencia (que tienen coeficientes positivos y negativos) el efecto es el contrario donde las diferencias entre niveles de intensidad de píxel son realzadas. Por consiguiente este tipo de filtros son utilizados normalmente para el realce de bordes, para su detección.

5.5 CARACTERÍSTICAS FORMALES DE LOS FILTROS LINEALES

Hasta ahora se ha tratado el concepto de filtro de una forma sencilla de tal forma que el lector pueda darse una idea rápida de para que son útiles y como poderlos construir. Aunque esto puede ser suficiente desde el punto de vista práctico, existen situaciones en donde el entendimiento de las propiedades matemáticas propias de las operaciones en las que se basan los filtros flexibiliza el diseño y análisis de los filtros.

La operación en la que se basa el accionar de los filtros es la convolución la cual ha sido tratada someramente a lo largo de este capitulo. La siguiente sección trata la convolución sus principales características.

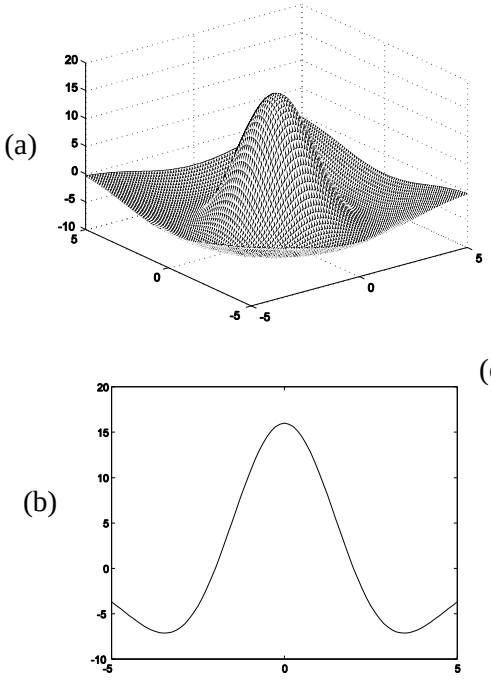


Figura 5.10 El filtro de Laplace (o de sombrero mexicano). (a) Función tridimensional, (b) bidimensional y (c) mascarilla que implementa el filtro.

5.5.1 Convolución lineal y Correlación

La operación de un filtro lineal, como fue vista en las secciones anteriores no fue un descubrimiento del área del procesamiento digital de imágenes, sino que es desde hace tiempo conocida por las matemáticas. La operación es llamada la convolución lineal, dicha operación permite conectar dos funciones continuas o discretas en una

sola. Para dos funciones discretas bidimensionales I y H la convolución lineal es definida como:

$$I'(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=\infty}^{\infty} I(x-i, y-j) \cdot H(i,j) \tag{5.11}$$

O en forma compacta:

$$I' = I * H \tag{5.12}$$

Las expresión de la ecuación 5.11 es muy parecida a la definida en 5.6, con la excepción de los limites de las sumatorias para i, j y el signo cambiado en las coordenadas $I(x-i, y-j)$. La primera diferencia se explica por el hecho de que el filtro $H(i, j)$ define una región $R(x, y)$ de influencia que contiene un conjunto finito de coeficientes, fuera de esta región los coeficientes son considerados cero o no relevantes, por lo que el límite de la sumatoria podría ser extendido sin alterar esto el resultado del cálculo.

En lo que respecta a las coordenadas si se vuelve a reformular la ecuación 5.11, de tal forma que:

$$\begin{aligned} I(x,y) &= \sum_{(i,j)\in R} I(x-i, y-j) \cdot H(i,j) \\ &= \sum_{(i,j)\in R} I(x+i, y+j) \cdot H(-i,-j) \end{aligned} \tag{5.13}$$

Dicha expresión vuelve a parecerse a la definida en la ecuación 5.5, cuando se definía la operación de la matriz de coeficientes del filtro, solo que la matriz de coeficientes se encuentra invertida en el sentido vertical y horizontal $H(-i,-j)$. Una inversión de este tipo puede ser interpretada como una rotación de 180° de la matriz de coeficientes $H(i, j)$ del filtro. La figura 5.11 muestra como se modela la inversión de las coordenadas mediante la rotación de la matriz de coeficientes del filtro $H(i, j)$.

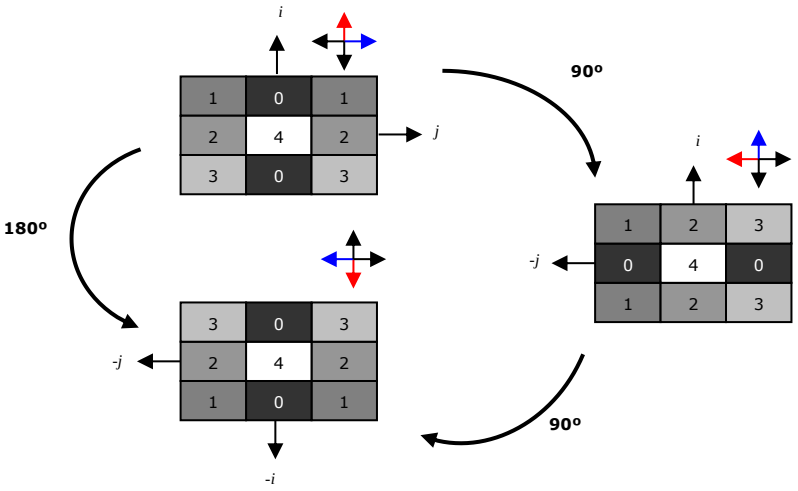


Figura 5.11. Modelaje de la inversión de coordenadas $H(-i,-j)$ *por la rotación de 180° de la matriz de coeficientes del filtro.*

Un concepto muy cercano a la convolución es la correlación. **Correlación** es el proceso de pasar la mascara o filtro $H(i,j)$ a través de la imagen tal y como fue definida la operación del filtro por la ecuación 5.5. Por lo que podemos decir que la convolución es igual a la correlación únicamente con la diferencia de que la matriz de coeficientes del filtro es rotada 180° (invertidos sus ejes coordenados) antes de realizar la combinación lineal de coeficientes y valores de píxel.

La operación matemática que se encuentra detrás de cada operación de filtro lineal es la convolución ($*$) y su resultado es exclusivamente dependiente del valor de los coeficientes de la matriz de filtro $H(i,j)$. La figura 5.12 muestra le proceso de convolución en imágenes.

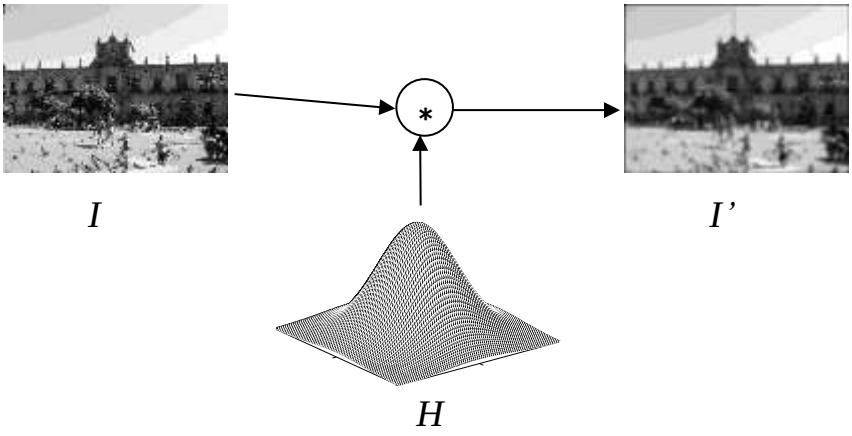


Figura 5.12. Convolución con el filtro gaussiano. A la imagen original I le es aplicada la convolución con la matriz de coeficientes $H(i, j)$ produciendo el resultado I'.

5.5.2 Propiedades de la convolución lineal

La importancia de la convolución se debe a que se basa en una serie de sencillas características matemáticas y a sus múltiples aplicaciones y formas en las que aparece modelando procesos y fenómenos. Como se verá mas adelante en este libro existe una estrecha relación entre la convolución, el análisis de Fourier y sus correspondientes métodos en la frecuencia.

Conmutatividad

La convolución es conmutativa, lo que significa que:

$$I * H = H * I \tag{5.14}$$

Representa el mismo resultado, si la imagen y el filtro intercambian posiciones de orden en la operación, por lo que no existe diferencia, si ala imagen I con la matriz de coeficientes $H(i, j)$ se convoluciona o al revés. Ambas funciones pueden intercambiar posiciones sin que esto signifique un cambio en los resultados de la operación.

Linealidad

Esto en imágenes implica algunas propiedades importantes. Si una imagen se escala con una constante a y después se a ese producto se convoluciona por la matriz de coeficientes H, se obtendría el mismo resultado que si primero se convoluciona la imagen y el resultado de esta operación es escalado por a. Esto es:

$$(a \cdot I) * H = a \cdot (I * H) \tag{5.15}$$

Además si dos imágenes son sumadas píxel a píxel y después al resultado se le aplica la convolución por H, daría el mismo resultado que si se convolucionarán ambas imágenes parcialmente por H y el resultado de cada convolución se sumara. Esto es:

$$(I_1 + I_2) * H = (I_1 * H) + (I_2 * H) \tag{5.16}$$

Podría en este contexto sorprender que la adición de una constante b a una imagen que después se convoluciona por la ventana H de un filtro no genera el mismo resultado que si primero se convoluciona la imagen y después se le suma la constante b. Esto es:

$$(b + I) * H \neq b + I * H \tag{5.17}$$

La linealidad de las operaciones de filtro por la utilización de la convolución es un concepto teórico importante, sin embargo en la práctica las operaciones de filtro se ven envueltas en procesos que limitan esta linealidad, estos podrían ser los errores de redondeo o bien la restricción del rango de las imágenes a los cuales se les restringe para visualización (“Clamping”).

Asociatividad

La convolución es un operador asociativo, lo que significa que el conjunto de operaciones que se realicen uno a uno en una secuencia pudieran cambian de orden de agrupamiento sin afectar al resultado original. Esto es:

$$A * (B * C) = (A * B) * C \tag{5.18}$$

La capacidad de asociación de los filtros es importante ya que pudiera resumirse la secuencia de la aplicación de un conjunto de filtros a uno solo, lo que normalmente es llevado a cabo para el diseño de matrices de coeficientes. El caso contrario también es válido, donde un filtro puede ser dividido en operaciones mas sencillas desempeñadas por filtros mas pequeños.

5.5.3 Separabilidad de los filtros

Una inmediata consecuencia de la ecuación 5.18 es la posibilidad de describir un filtro por la convolución de 2 o más filtros tentativamente más sencillos y pequeños

que el original. De tal forma que la convolución desempeñada por un filtro “grande” sobre una imagen podría se descompuesta por una secuencia de convoluciones realizadas por filtros pequeños, tal que:

$$I * H = I * (H_1 * H_2 * H_3 \ldots H_n) \tag{5.19}$$

La ventaja que se persigue con esta separación es la de aumentar la velocidad de las operaciones ya que aunque se aumenta el número de filtros, entre todos ellos al ser mas pequeños y sencillos realizan menos operaciones que el filtro original.

Separación x- y

Algo frecuente e importante en la utilización de filtros es la separación de un filtro bidimensional H en 2 de una sola dimensión H_x y H_y que operan en el sentido horizontal y vertical sobre una imagen. Si se supone que se tiene 2 filtros unidimensionales H_x y H_x que operan en cada una de las direcciones, así que:

$$H_x = \begin{bmatrix} 1 & 1 & \underline{1} & 1 & 1 \end{bmatrix} \qquad \text{o bien} \qquad H_y = \begin{bmatrix} 1 \\ \underline{1} \\ 1 \end{bmatrix} \tag{5.20}$$

Si se desea utilizar a ambos filtros uno con otro sobre la imagen I, podría ambos filtros fusionarse tal y como fue visto anteriormente de tal forma que:

$$I \leftarrow (I * H_x) * H_y = I * \underbrace{H_x * H_y}_{H_{xy}} \tag{5.21}$$

Por lo que seria exactamente lo mismo aplicar sobre la imagen el filtro H_{xy} que es producto de la convolución de ambos filtros direccionales H_x y H_x, donde:

$$H_{xy} = H_x * H_y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \underline{1} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{5.22}$$

Por lo que pudo ser visto el filtro para suavizado “Box” H_{xy} puede dividirse en 2 filtros de una sola dimensión aplicados en los dos diferentes sentidos sobre la imagen. La convolución desempeñada por el filtro completo H_{xy} necesita de 3x5=15 operaciones por píxel en la imagen original. En el caso de los dos filtros H_x y H_x se tendrían 3 +5= 8 operaciones por píxel de la imagen, lo cual a la vista es mucho menor.

En general el número de operaciones requeridas para un procesamiento crece dependiendo del tamaño del filtro con el cual se convoluciona la imagen, por lo que separar los filtros en las diferentes direcciones puede como fue visto acelerar el proceso de cálculo de algunas operaciones sobre imágenes.

Separación del filtro Gaussiano

Como fue visto anteriormente un filtro bidimensional puede ser dividido en dos filtros en las direcciones x-y, entonces puede generalizarse que si se tiene una función en dos dimensiones, es posible de igual manera dividirla en dos funciones, en donde cada función tratara con cada dimensión en particular. Es decir, matemáticamente:

$$H_{x,y}(i,j) = H_x(i) \cdot H_y(j) \tag{5.23}$$

Un caso prominente es el de la función de Gauss $G_\sigma(x,y)$ la cual puede ser dividida como el producto de dos funciones de una sola dimensión, se tiene por lo tanto que:

$$G_\sigma(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} \cdot e^{-\frac{y^2}{2\sigma^2}} = g_\sigma(x) \cdot g_\sigma(y) \tag{5.24}$$

De lo anterior es claro que un filtro Gaussiano $H^{G,\sigma}$ bidimensional puede dividirse en un par de filtros Gaussianos $H_x^{G,\sigma}$ y $H_x^{G,\sigma}$ unidimensionales en la forma en que puede ser realizado que:

$$I \leftarrow I * H^{G,\sigma} = I * H_x^{G,\sigma} * H_y^{G,\sigma} \tag{5.25}$$

La función Gaussiana decae lentamente por lo que un filtro para evitar errores debido al redondeo de coeficientes no puede implementar valores de σ<2.5. Para un filtro que incorpore una desviación estándar de $\sigma = 10$ se necesita un filtro de dimensiones 51x51. Si se separara el filtro en dos como fue visto el filtro implementaría la operación 50 veces más rápido que en la utilización del filtro original.

5.5.4 Respuesta al impulso de un Filtro

Existe también un elemento neutro en la operación de convolución el cual naturalmente también es una función, dicho elemento es llamado el impulso o la función Delta Dirac $\delta()$, con ella se establece que:

$$I * \delta = I \tag{5.26}$$

Lo que significa que esta función al ser convolucionada con una imagen el resultado es la imagen sin ningún tipo de cambio. La función Delta Dirac $\delta()$ en el caso discreto bidimensional es definida como:

$$\delta(i,j) = \begin{cases} 1 & \text{para } i = j = 0 \\ 0 & \text{si no} \end{cases} \tag{5.27}$$

La función Delta Dirac considerada como una imagen es visualizada como un solo punto brillante en el origen de coordenadas rodeado de un área de puntos negros infinita. La figura 5.13 muestra la función Delta Dirac en dos dimensiones y la forma en como esta es concebida como imagen.

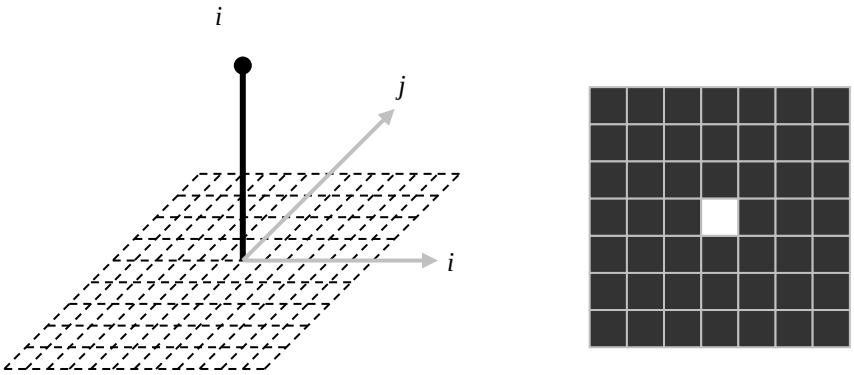


Figura 5.13 La función Delta Dirac en 2 dimensiones.

Si la función Delta Dirac es utilizada como filtro y aplicamos la convolución de esta función a una imagen, tal y como se definió en la ecuación 5.26 el resultado es de nueva cuenta la imagen original (véase figura 5.14).

El caso contrario es de igual manera interesante, si se toma a la función Detla Dirac como entrada (como si esta fuera la imagen) y se convolución con un filtro H el resultado seria la matriz de coeficientes del filtro (véase figura 5.15), esto es:

$$H * \delta = H \tag{5.28}$$

La aplicación de esta propiedad hace sentido, si las propiedades y características del filtro no se conocen y se desean medir. Bajo la condición de que el filtro a analizar se trata de un filtro lineal si aplicamos la función impulso se obtendría la información del filtro. La respuesta de un filtro a la función impulso es denominada Respuesta al Impulso del filtro y es de gran valor en el análisis y síntesis de sistemas en ingeniería.

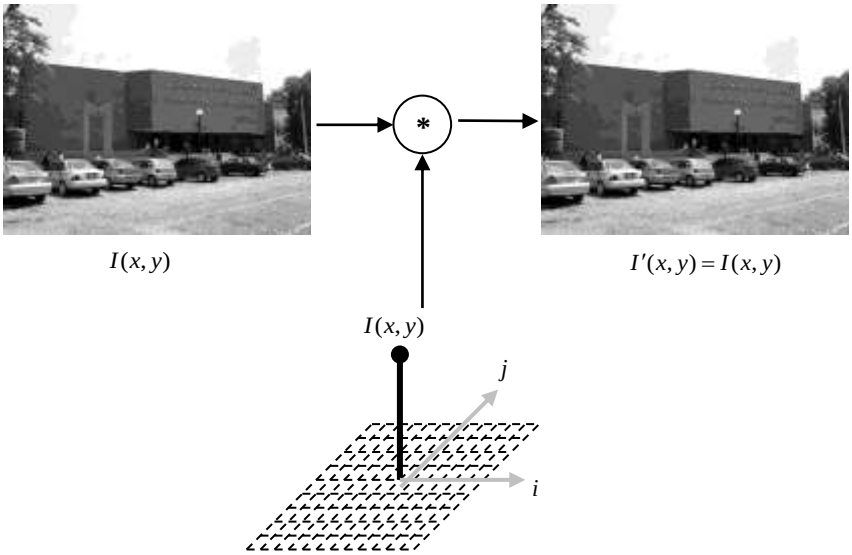


Figura 5.14 El resultado de la función Delta Dirac δ aplicada a una imagen I.

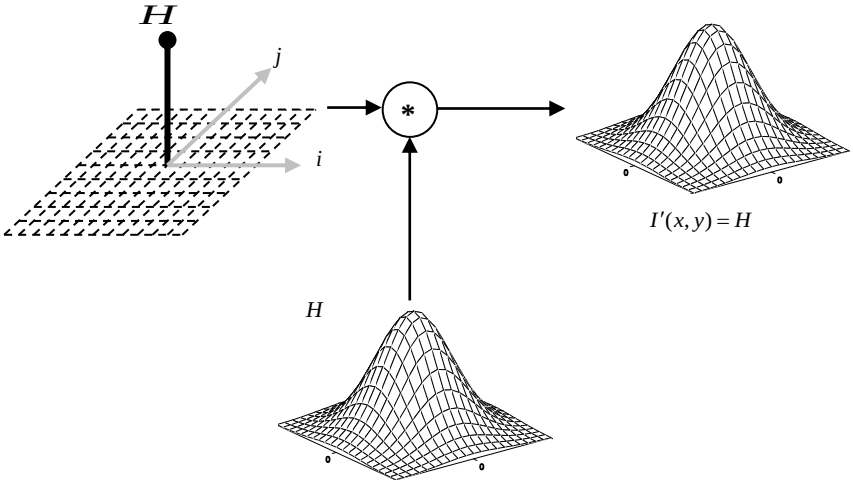


Figura 5.15 El resultado de convolucionar la función impulso δ y un filtro H.

5.6 AÑADIR RUIDO A IMÁGENES CON MATLAB®

Para probar la efectividad de los filtros sobre imágenes es necesario medir sus efectos a partir de imágenes que contengan artefactos o estructuras (ruido) que representen parámetros viables para ello. Como es lógico suponer es difícil obtener imágenes naturalmente con esas características, es decir no es sencillo obtener imágenes fabricadas desde el momento de la toma con cámara digital o escáner. Una manera más sencilla de tener imágenes con estas características es simplemente

fabricarlas artificialmente, lo que significa tomar la imagen por algún medio y añadirle los artefactos por programación.

Utilizando MatLAB es posible mediante sus elementos de programación inducir ruido en imágenes. El tipo de ruido que puede modelarse y añadirse puede ser tan complejo como sea necesario, en el caso de esta sección se muestra la forma de hacerlo para el caso de ruido de sal y pimienta. El ruido de sal y pimienta simula el hecho de añadir de forma aleatoria píxeles blancos (sal) y negros (pimienta) a una imagen, este tipo de ruido será utilizado para probar el efecto de filtros diseñados en las siguientes secciones.

Debido a la naturaleza estocástica de este tipo de ruido será necesario utilizar alguna función en MatLAB que permita de forma aleatoria dar un valor entre 1 y los valores máximos que describan el tamaño de la imagen y asignarlo al par que define la posición (x, y) del píxel. Una vez elegido el píxel se le asigna un valor que puede ser 255 (en el caso de la sal) y 0 (en el caso de la pimienta). La función en MatLAB que devuelve un número seudo-aleatorio uniformemente distribuido entre 0 y 1 es `rand`, cuyo formato puede ser descrito como:

```
y = rand
```

En donde `y` asume el valor devuelto por la función `rand`. Como el valor que se devuelve es uniformemente distribuido entre 0 y 1, para que se adapte al intervalo de interés solo será necesario multiplicarlo por el límite máximo del intervalo.

Un aspecto importante es que si solamente se distorsiona la imagen colocando solamente píxeles en la imagen con valores de 0 y 255 al momento de hacer un análisis visual o simplemente su despliegue se vuelve un tanto complicado, por esta razón se vuelve una mejor opción añadir estructuras[2]. La estructura con la que puede contaminar a la imagen para generar el ruido puede ser realmente cualquiera, aunque generalmente se elige por facilidad la rectangular. Una observación importante es que la estructura debe de ser en todo caso de menor dimensión a la del filtro utilizado, de lo contrario el filtro no tendrá la capacidad de eliminar la distorsión. En este caso se elige una estructura de 3x2, ya que en la mayoría de los filtros propuestos para explicar los efectos de ellos sobre imágenes son de dimensión 3x3. La figura 5.16 muestra la estructura añadida como ruido a la imagen considerando como posición relativa la del píxel superior de la izquierda.

[2] Grupos de píxeles, que siguen un determinado patrón regular, un rectángulo.

(x, y)	$(x+1, y+1)$
$(x+1, y)$	$(x+1, y+1)$
$(x+2, y)$	$(x+2, y+1)$

Figura 5.16 Estructura añadida como ruido, con el objetivo de facilitar su visualización en lugar de la distorsión de un píxel individual.

El programa 5.2 muestra el script de MatLAB que permite añadir 2000 puntos de ruido, donde 1000 corresponden a estructuras de valores de intensidad 255 (sal) y 1000 corresponden a estructuras de valores de intensidad de 0 (pimienta).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                              %
%Script de MatLAB para añadir ruido Sal        %
%y Pimienta                                    %
%                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                              %
% Erik Cuevas, Daniel Zaldivar, Marco Pérez    %
% Versión: 23/12/07                            %
%                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Se hace una copia en Ir1 de la imagen a la cual se le
%quiere añadir el ruido
Ir1=Ir;
[re co]=size(Ir1);
%Se calculan 1000 puntos de ruido con valor 255
for v=1:1000
%se calculan las posiciones de cada punto
%para x se escala el valor para el intervalo máximo re
x=round(rand*re);
%para y se escala el valor para el intervalo máximo co
y=round(rand*co);
%Como MatLAB no indexa a partir de 0 se protege el
programa
%para empezar en 1
if x==0
    x=1;
end
if y==0
    y=1;
end
```

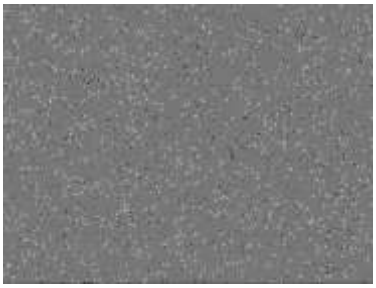
```
 %Se protege el programa para que si la posición
calculada
 %representa la frontera de la imagen se recalcule para
 %que pueda insertarse una estructura
 if x==600
     x=598;
 end
 if y==800
     y=798;
 end
 %se inserta la estructura con valores
 %de intensidad de 255 (sal)
 Ir1(x,y)=255;
 Ir1(x,y+1)=255;
 Ir1(x+1,y)=255;
 Ir1(x+1,y+1)=255;
 Ir1(x+2,y)=255;
 Ir1(x+2,y+1)=255;
 end
 %Se calculan 1000 puntos de ruido con valor 0
 %y se repite todo lo anteriormente dicho
 for v=1:1000
 x=round(rand*re);
 y=round(rand*co);
 if x==0
     x=1;
 end
 if y==0
     y=1;
 end
 if x==600
     x=598;
 end
 if y==800
     y=798;
 end

 Ir1(x,y)=0;
 Ir1(x,y+1)=0;
 Ir1(x+1,y)=0;
 Ir1(x+1,y+1)=0;
 Ir1(x+2,y)=0;
 Ir1(x+2,y+1)=0;
 end
```

Programa 5.2 Programa en MatLAB para añadir ruido de Sal y Pimienta a una imagen.

Una consideración importante, como se ve en el programa 5.2 que hay que tomar en cuenta, es la necesidad de proteger la posición aleatoria donde se colocara la estructura, ya que si esta es elegida en la frontera de la imagen, debido a que la estructura es colocada hacia abajo tomando como punto relativo el valor aleatorio escogido por la función `rand`, la estructura no se posicionara dentro de la imagen. Para ello en el programa se controla que si este es el caso el valor de posición de la estructura se establece 2 unidades hacia arriba de tal forma que la estructura pueda posicionarse. La figura 5.17 muestra imágenes que resultaron de la ejecución del programa 5.2.

(a)



(b)

(c)

Figura 5.17 Proceso de distorsión de una imagen con ruido de sal y pimienta efectuada mediante el programa 5.2. (a) Imagen original, (b) ruido de sal y pimienta añadido (se coloco un fondo gris para apreciar ambas estructuras 0 y 255) y (c) la imagen con el ruido.

Si bien se quiere ahorrar tiempo podría entones utilizarse la función implementada en el toolboox de procesamiento de imágenes que permite añadir a una imagen diferentes tipos de ruido. En esta sección solo se explicara la forma de hacerlo para el tipo de ruido específico de sal y pimienta.

El formato de la función es:

```
g=imnoise(f, 'salt & pepper', 0.2);
```

Donde `f` es la imagen a la cual se desea contaminar con el ruido de sal y pimienta y `0.2` es el porcentaje de píxeles (visto de otra manera el número de píxeles que serán de sal y pimienta) de la imagen que se desea contaminar, donde 0.2 por ejemplo advertirá que un 20% de los píxeles de la imagen serán contaminados. `g` por su parte recibe la imagen contaminada por el ruido configurado.

5.7 FILTROS NO LINEALES ESPACIALES

La utilización de filtros lineales para suavizar y quitar perturbaciones de imágenes tiene una gran desventaja, estructuras presentes en la imagen, tales como puntos, bordes y líneas son de igual manera degradados (véase figura 5.18). Este efecto no puede ser evitado mediante el empleo de alguna combinación de coeficientes de filtros lineales por lo que si desea es suavizar una imagen sin que esto signifique una reducción en la calidad de las estructuras presentes en la imagen, la utilización de los filtros lineales esta restringido. En la presente sección se presenta y explica un tipo especial de filtros que permiten resolver este problema cuando menos de una mejor manera que los filtros lineales. Evidentemente los filtros no lineales para poder llevar a cabo este tipo de efectos se basan en operaciones cuyas características y propiedades no son lineales.

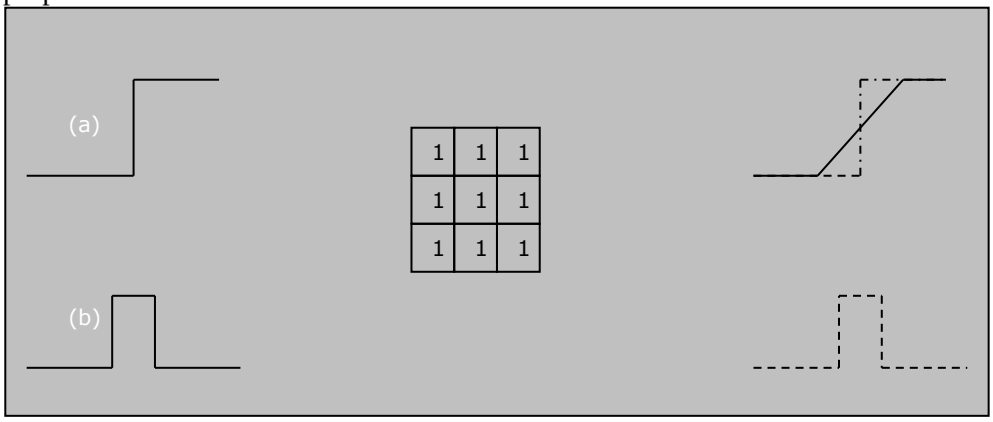


Figura 5.18 El suavizado lineal degrada estructuras importantes en la imagen, tales como bordes en escalón (a) o líneas (b).

5.7.1 Filtros Máximos y Mínimos

Los filtros no lineales calculan el resultado en una determinada posición (x, y) al igual que los filtros anteriores mediante el empleo de una determinada región relativa a la imagen original R. Los filtros no lineales más sencillos son los filtros máximos y mínimos. Los filtros máximos y mínimos son definidos correspondientemente como:

$$I'(x,y) = \min\left\{I(x+i, y+j) \middle| (i,j) \in R\right\} = \min(R(x,y)) \qquad (5.29)$$
$$I'(x,y) = \max\left\{I(x+i, y+j) \middle| (i,j) \in R\right\} = \max(R(x,y))$$

Donde $R(x, y)$ representa la región relativa a la posición (x, y) definida por el filtro (la mayoría de las veces es un rectángulo de 3x3). La figura 5.19 muestra el efecto del filtro mínimo sobre diferentes estructuras de la imagen. Como puede verse en la figura el escalón de la imagen aparece por acción del filtro mínimo desplazada hacia la derecha una cantidad igual al ancho de la región $R(x, y)$ definida por el filtro, mientras que una línea es eliminada por la acción del filtro, siempre y cuando la anchura de la línea es menor que a la definida por la del filtro.

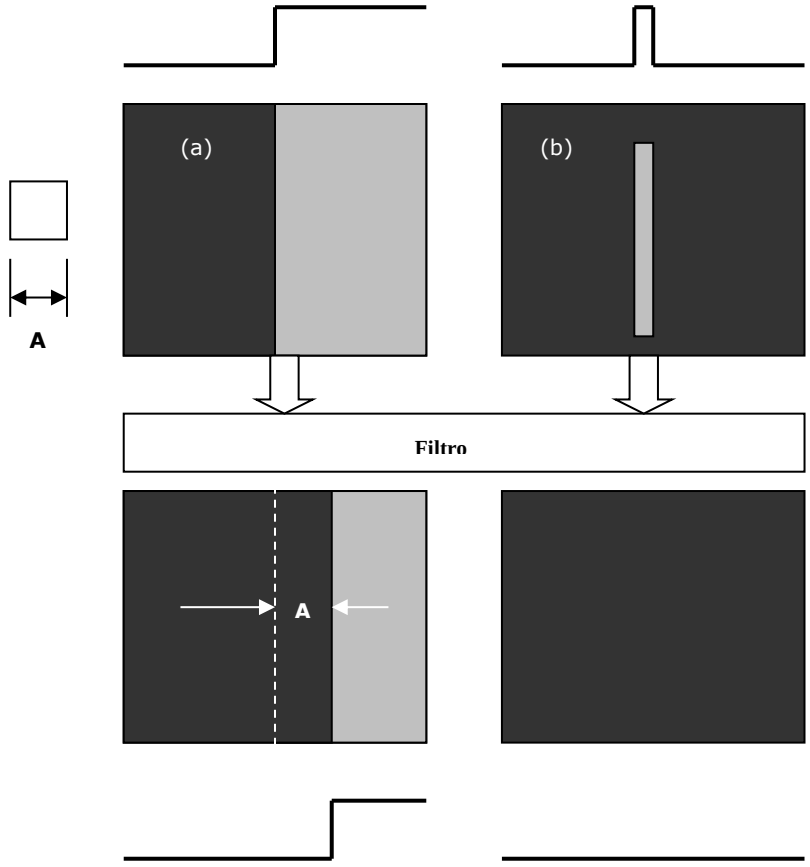


Figura 5.19 Efecto del filtro mínimo sobre diferentes formas locales en una imagen. La imagen original esta arriba y el resultado obtenido por operación del filtro esta abajo. El valor de A expresa en ancho del filtro utilizado que al mismo tiempo define la región R abarcada por él. El escalón representado en la imagen (a) por efecto de la aplicación del filtro es desplazado hacia la derecha en A unidades. La línea contenida en la imagen (b), si su anchura es menor a la de A es desaparecida por el efecto de la operación del filtro.

La figura 5.20 muestra la aplicación de los filtros mínimo y máximo sobre una imagen de intensidad, la cual fue artificialmente contaminada con el ruido conocido como sal y pimienta. El ruido de sal y pimienta simula el hecho de añadir de forma aleatoria píxeles blancos (sal) y negros (pimienta) a una imagen. El filtro mínimo elimina los puntos blancos de la imagen, debido a que en su operación encuentra el mínimo de una región de influencia $R(x, y)$ definida por el filtro, el cual centrado en el píxel contaminado blanco, lo sustituirá por el valor mínimo encontrado entre

sus vecinos. De igual manera los píxeles que tienen el mínimo valor permisible por la imagen (negro o pimienta) son agrandados en relación al tamaño **A** del filtro.

El filtro máximo presenta naturalmente el efecto contrario. Esto es, píxeles que represente distorsiones pertenecientes a valores de intensidad cero (pimienta) o pequeños serán eliminados, mientras que estructuras blancas serán aumentadas.

La figura 5.20 muestra el efecto de la aplicación de los filtros máximos y mínimos sobre una imagen.

(a)

(b)

(c)

Figura 5.20 Filtros mínimo y máximo. (a) La imagen original a la cual se le añadió artificialmente ruido de sal y pimienta (véase sección 5.6), (b) imagen resultado de aplicar el 3x3 filtro mínimo y (c) imagen resultado de aplicar el filtro 3x3 máximo.

5.7.2 El filtro de la Mediana

Evidentemente es imposible construir un filtro que pueda quitar todas las imperfecciones o ruidos de la imagen y al mismo tiempo mantener intactas estructuras de la imagen consideradas como importantes, ya que ningún filtro pude

diferenciar cuales estructuras son importantes para el observador y cuales no. El filtro de la mediana es un buen compromiso en este dilema.

El efecto de la atenuación de los bordes en una imagen generado por el efecto de suavizado de los filtros lineales generalmente es indeseado. El filtro de la mediana permite eliminar artefactos y estructuras no deseadas en la imagen sin afectar significativamente a los bordes. El filtro de la mediana pertenece a una clase especial de filtros los estadísticos, los cuales además de ser no lineales basan su accionar en alguna operación del tipo estadístico.

En Estadística la **mediana** es el valor de la variable que deja el mismo número de datos antes y después que él. De acuerdo con esta definición el conjunto de datos menores o iguales que la mediana representarán el 50% de los datos, y los que sean mayores que la mediana representarán el otro 50% del total de datos de la muestra.

Considerando que $x_1, x_2 \ldots, x_n$ son los datos de una muestra ordenada en orden creciente, la mediana quedaría definida como:

$$M_e = x_{\frac{n+1}{2}} \tag{5.30}$$

Si *n* es impar será M_e la observación central de los valores, una vez que estos han sido ordenados en orden creciente o decreciente. Si *n* es par será el promedio aritmético de las dos observaciones centrales, esto es:

$$M_e = \frac{x_{\frac{n}{2}} + x_{\frac{n+1}{2}}}{2} \tag{5.31}$$

Con lo anteriormente visto se puede decir que el filtro de la mediana sustituye cada píxel de la imagen por la mediana de los valores de intensidad dentro de la región de influencia $R(x, y)$ definida por el filtro, esto expresado formalmente quedaría como:

$$I'(x, y) = M_e(R(x, y)) \tag{5.32}$$

Para el cálculo de la mediana de los datos que conforman la región de interés $R(x, y)$, solo es necesario realizar dos pasos. Primero acomodar los valores de intensidad de la imagen que corresponden región de influencia definida por el filtro en forma de vector, después reacomodarlos en orden creciente, si existen valores repetidos también serán repetidos en el nuevo arreglo. La figura 5.21 demuestra el cálculo de la mediana en un filtro con una región de tamaño 3x3.

Debido a que un filtro normalmente se define como una matriz cuyo conjunto de datos es impar, el valor de la mediana corresponde siempre al valor central del vector ordenado en forma creciente correspondiente a la región de interés $R(x, y)$.

El filtro de la mediana puede decirse que no produce o calcula el nuevo valor con el que será sustituido el píxel de la imagen, sino que este es un valor existente en la región de interés del filtro producto del reacomodo de datos. La figura 5.22 ilustra el efecto de un filtro de la mediana sobre estructuras bidimensiolanes. De los resultados de esta figura puede ser observado como estructuras mas pequeñas de la mitad de la estructura desaparecen, mientras que estructuras iguales o mas que la mitad permanecen prácticamente sin cambio. La figura 5.23 muestra finalmente mediante la utilización de imágenes el efecto del filtro de la mediana para eliminar distorsiones como los de la sal y pimienta y lo compara con un filtro suavizador.

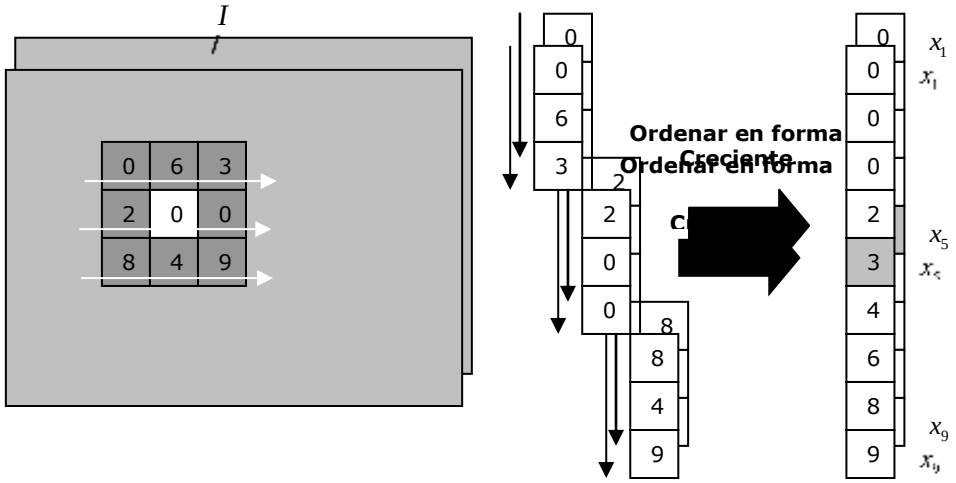


Figura 5.21 Calculo de la mediana considerando un filtro de tamaño 3x3.

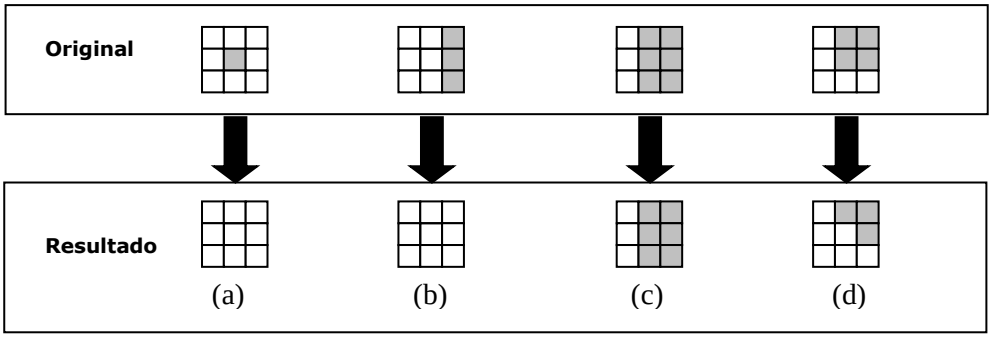


Figura 5.22 Efecto del filtro de la mediana sobre estructuras, puede verse como en (a) y (b) cuando las estructuras son mas pequeñas que la mitad del filtro las elimina, mientras que cuando son igual o mayores que la mitad como en (c) y (d) permanecen prácticamente sin cambio.

(a)

(b) (c)

Figura 5.23 Resultados y comparación del efecto del filtro de la mediana. (a) Imagen original contaminada por el ruido de sal y pimienta (véase sección 5.6). (b) Efecto del filtro suavizador "Box" sobre la imagen (a). Puede verse como el filtro es incapaz de eliminar los artefactos que distorsionan la imagen, el único efecto posible resulta en una atenuación de los mismos. (c) Efecto del filtro de la mediana. Puede observarse como este filtro elimina completamente el ruido presente en la imagen.

5.7.3 El filtro de la mediana con ventana de multiplicidad

El filtro de la mediana basa su cálculo en una medida de rango en un conjunto ordenado de datos, por lo que un dato que excepcionalmente tenga un valor muy pequeño o muy grande en comparación a los demás datos que son cubiertos por la región de interés $R(x, y)$ del filtro no alterara significativamente el resultado, su

único efecto sería desplazar el resultado en algún sentido. Por la anterior razón el filtro es muy bien situado para la eliminación de ruido del tipo sal y pimienta.

Con el filtro de la mediana todos los píxeles considerados en la región de interés del filtro tienen por decirlo de alguna manera la misma importancia en la determinación del cálculo del valor resultado. Como se ha mencionado en varias partes de este capítulo una situación deseable en un filtro sería darle un mayor peso al coeficiente central del filtro que corresponde al píxel de la imagen a sustituir.

El filtro de la mediana con ventana de multiplicidad puede ser considerado como una variante del filtro de la mediana, que a diferencia de éste posee una ventana que además de fijar la relación de los vecinos del píxel central a considerar le confieren una importancia diferente a cada uno de ellos dependiendo del correspondiente coeficiente de la matriz del filtro H . De esta manera al igual que los filtros lineales, los coeficientes de la matriz H escalan sus píxeles correspondientes de la imagen. Sin embargo en la operación efectuada por este filtro no se involucra una multiplicación del coeficiente del filtro por su correspondiente píxel, sino multiplicidad, que en este contexto significara que el coeficiente del filtro expresa la cantidad de veces que se presenta el valor de intensidad del píxel de la imagen en los datos sobre los cuáles se calcula la mediana.

Considerando lo anterior la matriz de coeficientes indicaría en su suma total el número de datos que son considerados para el cálculo de la mediana por lo que se tiene que el tamaño de datos T a considerar para el cálculo de la mediana para una ventana de 3x3 considerando la matriz:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{5.33}$$

Es:

$$\begin{aligned} T &= h_1 + h_2 + h_3 + h_4 + h_5 + h_6 + h_7 + h_8 + h_9 \\ T &= \sum_{i=1}^{9} h_i \end{aligned} \tag{5.34}$$

Como ejemplo se considera si se tiene el filtro H definido como:

$$H = \begin{bmatrix} 1 & 3 & 2 \\ 2 & \underline{4} & 1 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.35}$$

El tamaño de datos T sería:

$$\begin{aligned} T &= \sum_{i=1}^{9} h_i \\ &= 17 \end{aligned} \tag{5.36}$$

Una vez conocido la cantidad de datos que deben considerarse se procede a aplicar el mismo método que utilizo en la sección 5.6.2 para encontrar la mediana del conjunto de datos. Como fue también tratado en esa sección si el conjunto de datos definido por T es impar la mediana es el dato central de la colección ordenada en orden creciente, pero si es par entonces tendrá que calcularse el promedio de los datos centrales de la colección, tal y como es mostrado en la ecuación 5.31.

Considerando este método, los coeficientes de la matriz H deben ser positivos y si el coeficiente en una determinada posición es cero, se advertirá con ello que no se considerará ese píxel de la imagen para la determinación de la mediana. Un ejemplo de esto es el filtro de Cruz de la mediana definido en la ecuación 5.37, que como su nombre lo dice considera solo la cruz de los datos de la matriz de coeficientes.

$$H = \begin{bmatrix} 0 & 1 & 0 \\ 1 & \underline{1} & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.37}$$

La figura 5.24 ilustra el proceso de operación de este filtro.

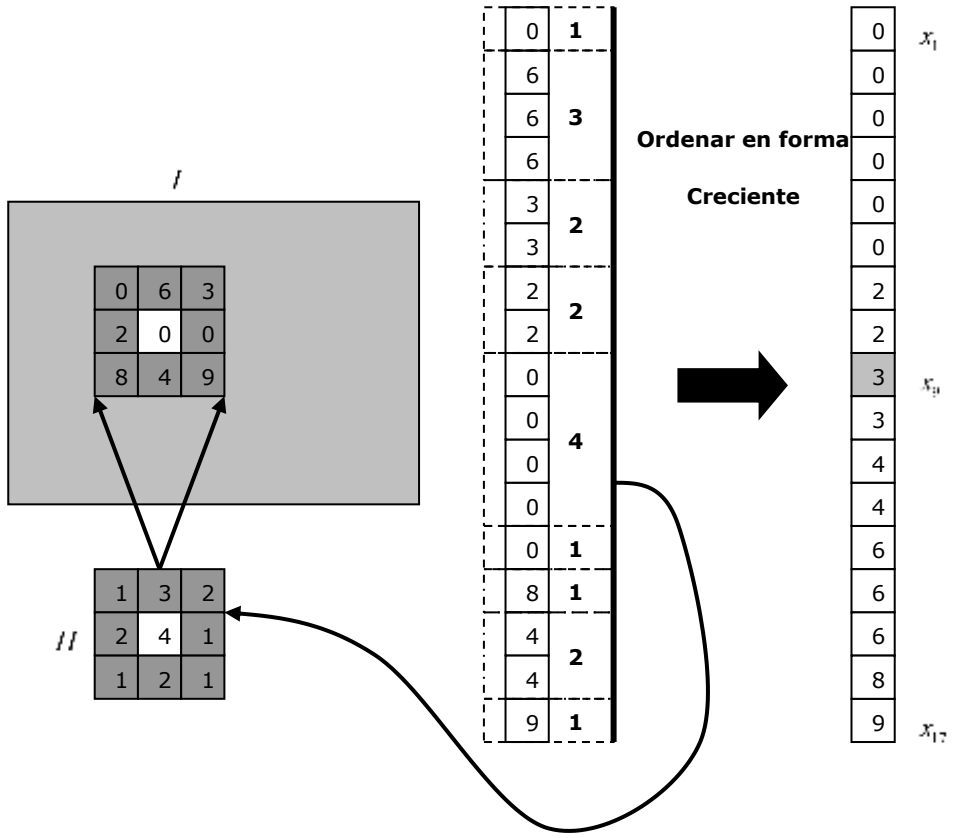


Figura 5.24 Cálculo desarrollado por el filtro de la mediando con una ventana de multiplicidad de 3x3.

5.7.4 Otros filtros no lineales

El filtro de la mediana y su variante de ventana de multiplicidad son únicamente algunos ejemplos de filtros no lineales, los cuales se utilizan frecuentemente y además resultan fáciles de explicar. Debido a que por el nombre de no lineales agrupamos a todos aquellos filtros que no tienen la propiedad de linealidad existen una gran cantidad de ellos que cumplen con la característica de no linealidad, tales como los filtros detectores de esquinas y los morfológicos que serán tratados en capítulos posteriores. Una diferencia importante de los filtros no lineales es de que la base matemática y teoría existente para este tipo de filtros no es la misma ni se encuentra tan desarrollada como en el caso de los filtros lineales los cuales al ser basados por la convolución heredan todas sus propiedades.

5.8 FILTROS ESPACIALES LINEALES EN MATLAB®

Para explicar las opciones que implementa MatLAB en sus funciones para el filtrado espacial de imágenes analizaremos la forma en la que se realizan las operaciones de correlación y convolución a detalle, mismas que ya fueron tratadas en secciones anteriores y veremos las posibles opciones que se tienen, con el objetivo de entender la flexibilidad con la que se pueden operar las funciones implementadas en MatLAB.

5.8.1 Tamaño de la correlación y convolución

Para ejemplificar la forma en la que se realizan las operaciones se considera por sencillez el caso unidimensional por lo que se supone se tiene una función f y un kernel k como lo muestra la figura 5.25(a). Para efectuar la operación de correlación de la función y el kernel movemos ambas secuencias de tal forma que su punto de referencia coincida tal y como lo muestra la figura 5.25(b). De 5.25(b) puede ser observado que existen coeficientes que no tienen un coeficiente correspondiente con el cual interactuar. Para solucionar esto lo mas sencillo es añadir ceros al inicio y al final de la función f (véase figura 5.25(c)) de tal forma que durante el desplazamiento del kernel durante la operación de correlación se garantice un coeficiente correspondiente entre ambas secuencias.

La operación de correlación entonces se efectúa, desplazando el kernel hacia la derecha, donde los valores de la secuencia de correlación en cada paso se obtienen mediante la suma de los valores multiplicandos de los correspondientes coeficientes de ambas secuencias. Como muestra la figura 5.25(c) el resultado del primer elemento de la secuencia de correlación es cero. El último valor de la secuencia de correlación lo representa el último desplazamiento geométrico efectuado por el kernel sobre la función f, de tal forma que el último valor del kernel se corresponde con el ultimo valor de la función original de f (sin añadir ceros) tal y como lo muestra la figura 5.25(d).

Si desarrollamos la operación tal y como fue descrita se obtendrá como resultado la secuencia de correlación mostrada en 5.25(e). Debe de ser notado que si se realiza el proceso inverso, es decir se mantiene fijo el kernel y el desplazamiento lo realiza la función f, el resultado no sería igual, por lo que el orden si importa.

Con el objetivo de fijar las relaciones entre las operaciones que aquí se describen y las opciones manejadas por las funciones de MatLAB, se dirá que si se realiza la operación de correlación tal y como fue descrita anteriormente (mediante el desplazamiento del kernel sobre la función considerando la adición de ceros para que exista correspondencia entre coeficientes) el resultado de la correlación será **`'full'`**.

Si se establece en lugar del proceso anterior considerar que el punto de referencia se encuentra en el coeficiente central del kernel *k*, entonces el resultado será diferente e igual al tamaño de la función *f*, esto se debe a que habrá menos desplazamientos los que se lleven a cabo entre el kernel y la función. A este tipo de resultado le llamaremos **`'same'`**. La figura 5.26 muestra una representación del resultado de la correlación con esta variante.

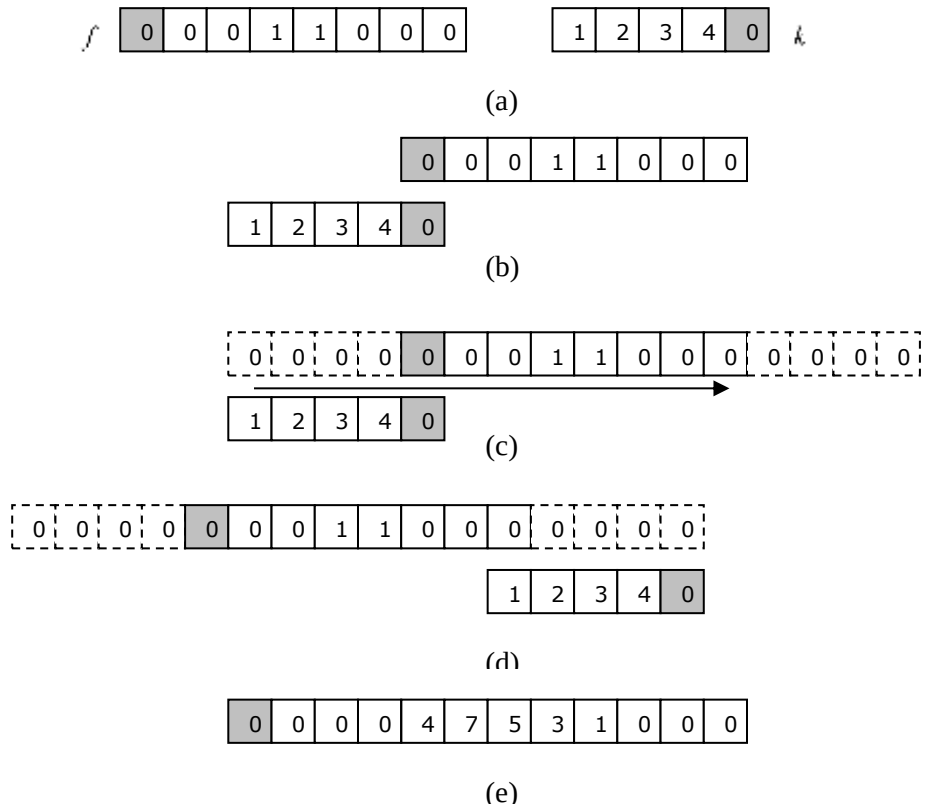


Figura 5.25. Operación de correlación unidimensional efectuada entre una función f y el kernel k. (a) Función y Kernel, (b) Alineación de ambas secuencias en su punto de referencia para efectuar correlación, (c) adición de ceros a la función para asegurar correspondencia de coeficientes e ilustración del desplazamiento del kernel como producto de la operación de correlación, (d) ultimo producto viable entre ambas secuencias y (e) resultado 'full' de la correlación.

Como comparación se establecen también en la figura 5.26 los mismos pasos con los que se desarrolló la operación de correlación 'full'. Algo importante de hacer notar es que el resultado de esta correlación es del mismo tamaño que la secuencia de la función *f*.

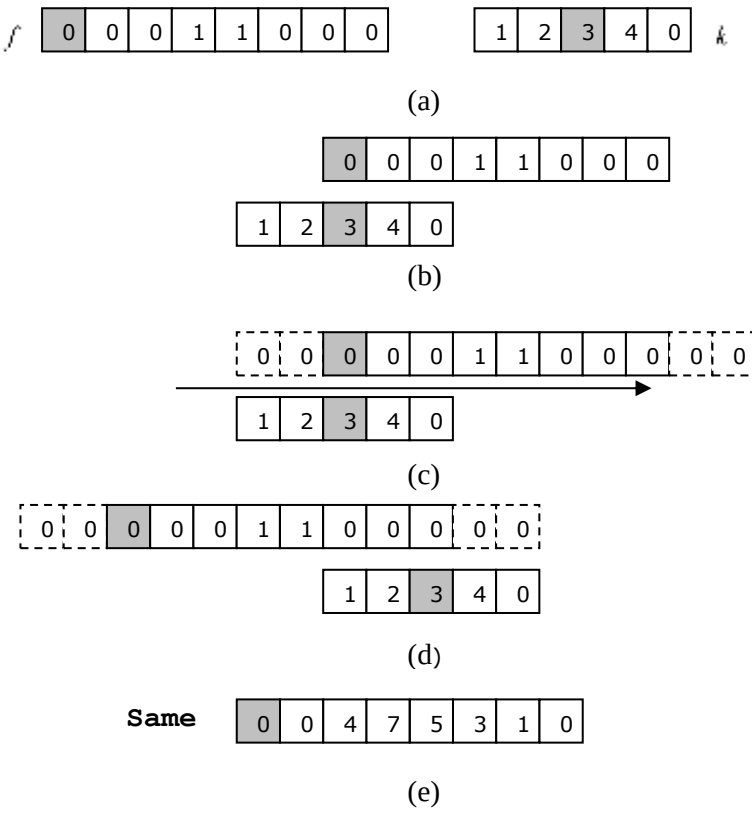


Figura 5.26. Operación de correlación unidimensional efectuada entre una función f y el kernel k considerando que el punto de referencia es el coeficiente central del kernel. (a) Función y Kernel, (b) Alineación de ambas secuencias en su punto de referencia para efectuar correlación, (c) adición de ceros a la función para asegurar correspondencia de coeficientes e ilustración del desplazamiento del kernel como producto de la operación de correlación, (d) ultimo producto viable entre ambas secuencias y (e) resultado 'same' de la correlación.

Para efectuar la convolución, como ya fue explicado, se realiza una rotación de 180° sobre el kernel *k*, y puede entonces realizarse el mismo proceso que el que se desempeñó con la correlación. Al igual que con la correlación es aquí también aplicable el caso de los productos `'full'` y `'same'` resultantes de la operación dependiendo de si se realiza el producto completo de la convolución considerando como punto de referencia algún coeficiente inicial del kernel (`'full'`) o bien el coeficiente central (`'same'`). La figura 5.27 muestra el proceso de cálculo sobre las mismas secuencias presentadas en el caso de la correlación.

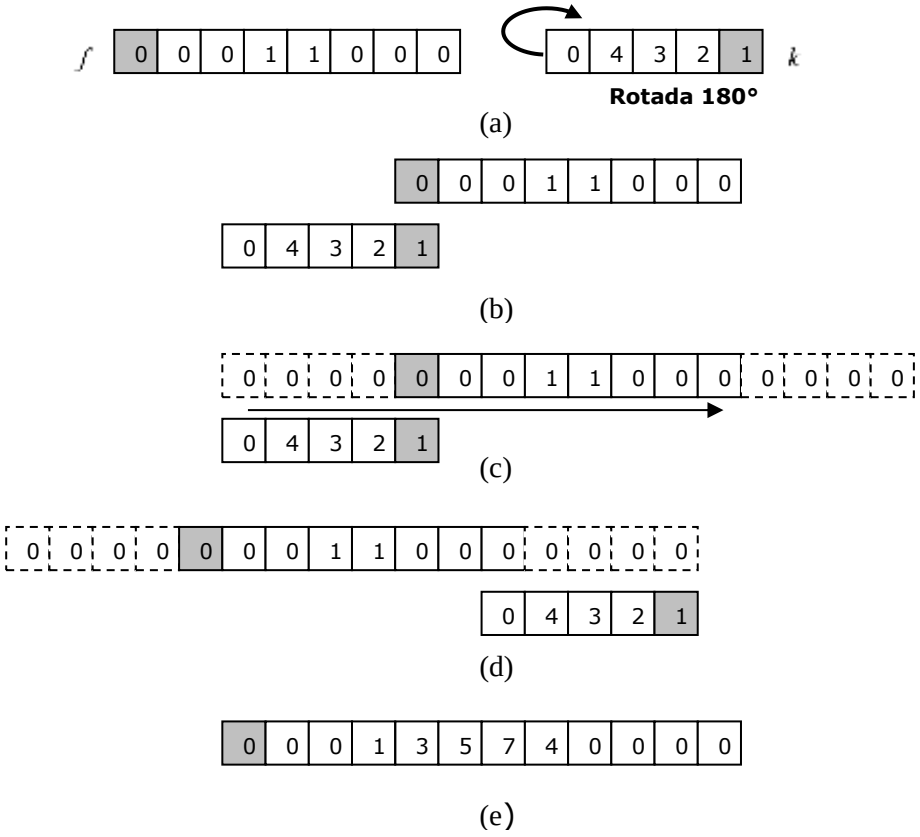


Figura 5.27 Operación de convolución unidimensional efectuada entre una función f *y el kernel rotado* k *180°. (a) Función y Kernel, (b) Alineación de ambas secuencias en su punto de referencia para efectuar correlación, (c) adición de ceros a la función para asegurar correspondencia de coeficientes e ilustración del desplazamiento del kernel como producto de la operación de correlación, (d) ultimo producto viable entre ambas secuencias y (e) resultado* `'full'` *de la correlación.*

De la figura es evidente que al rotar el kernel *k* 180° el punto de referencia se invierte. Los demás pasos son exactamente iguales a los mostrados para el caso de la correlación. Otra observación importante resulta del resultado de la convolución en su versión `'full'`, el cual es exactamente el mismo que el de la correlación para el mismo caso con la excepción de que el orden de la secuencia es exactamente el contrario, de ello se advierte la cercana relación existente entre ambas operaciones. De igual forma puede calcularse la convolución en su variante `'same'` con la única consideración de que habrá que rotarse el kernel, mismo que al ser invertido cambiara el orden de la secuencia mas no el punto de referencia el cual al ser

simétrico en relación a las dos puntas de la secuencia permanecerá en la misma posición. Lo anterior es ilustrado en la figura 5.28.

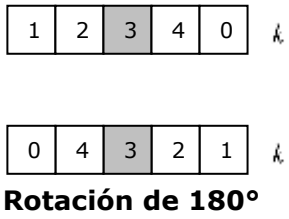
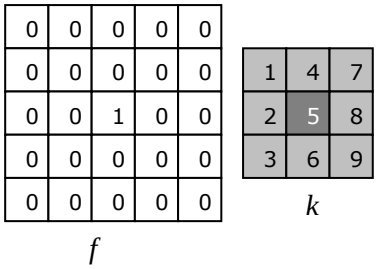
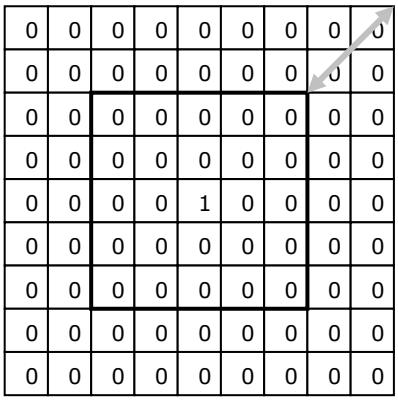


Figura 5.28 Rotación del kernel, para la realización de la convolución.

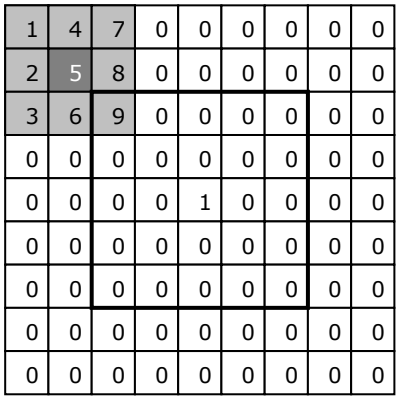
Los mismos conceptos aquí tratados pueden ser ampliados al caso de imágenes. La figura 5.29 ilustra las operaciones de correlación y convolución con sus variantes `'full' y 'same'`.


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
f
1 4 7
2 5 8
3 6 9
k

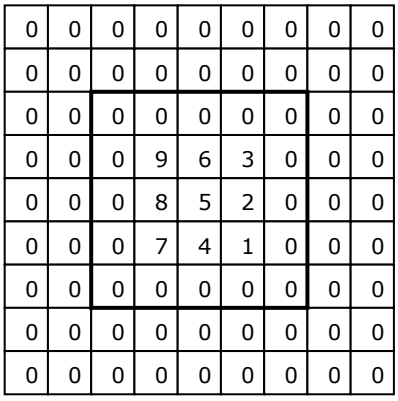
(a)


0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 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 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

(b)


1 4 7 0 0 0 0 0 0
2 5 8 0 0 0 0 0 0
3 6 9 0 0 0 0 0 0
0 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 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

(c)


0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 9 6 3 0 0 0
0 0 0 8 5 2 0 0 0
0 0 0 7 4 1 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

(d)

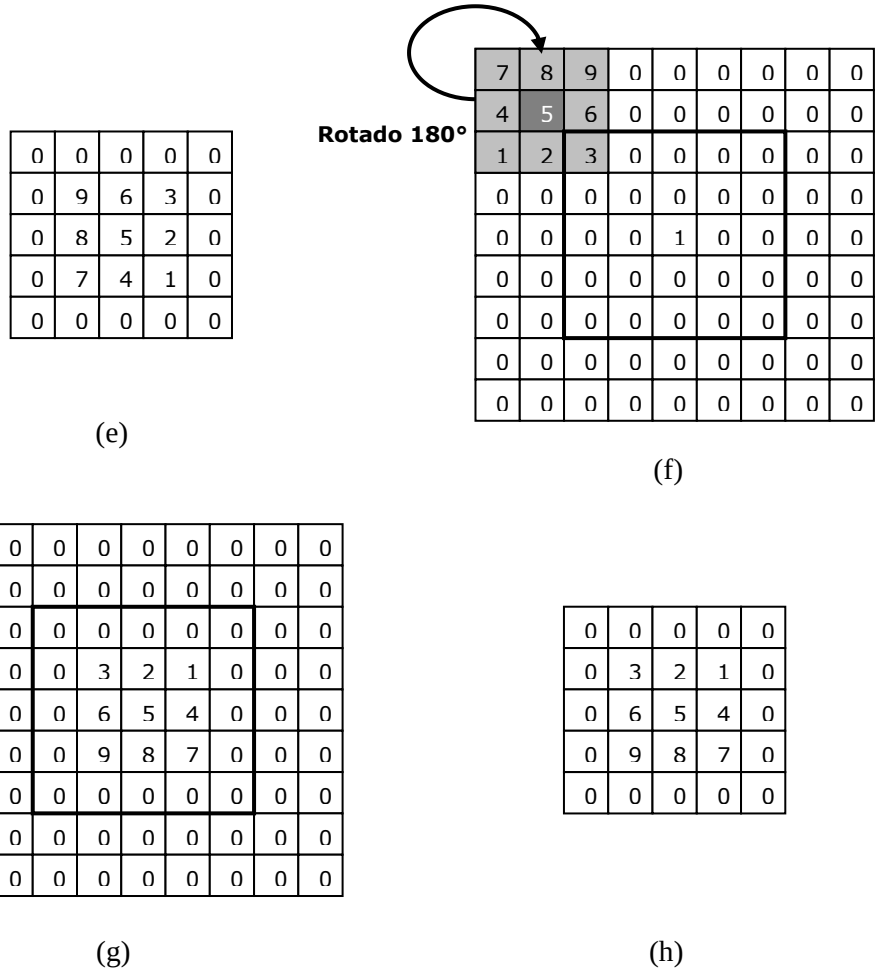


Figura 5.29 Operaciones de correlación y convolución en imágenes. (a) Imagen original f *y kernel* k*, (b) Imagen original a la que se le han añadido ceros con el objetivo de garantizar que exista un correspondiente valor de la imagen para cada coeficiente del kernel, (c) inicio de la correlación del kernel con la imagen a la que se le han añadido ceros, (d) resultado completo de la correlación* `'full'`*, considerando la imagen con los ceros adicionales, (e) resultado de la correlación del mismo tamaño* `'same'` *que la imagen, logrado al no añadir ceros en la imagen original, lo que tiene como efecto obtener una imagen del mismo tamaño que la original, (f) imagen original y kernel rotado 180°, con el que se efectúa la operación de convolución sobre la imagen, (g) resultado completo* `'full'` *de la convolución y (h) resultado de la convolución del mismo tamaño* `'same'` *que la imagen.*

5.8.2 Manejo de las fronteras de la imagen

Un problema importante como ha sido mencionado repetidamente en secciones anteriores es la forma en la que deben ser tratadas las fronteras de la imagen. Cuando se efectúa la operación de correlación o convolución de una imagen con un filtro habrá en las fronteras de la imagen coeficientes del filtro que no tengan un correspondiente valor de intensidad en la imagen. La forma en como este problema se trate tiene algún efecto en el resultado final.

Existen 4 formas en las que puede tratarse este problema:

- Agregando líneas de ceros en las fronteras
- Replicando líneas de valores iguales a las últimas filas o columnas de la imagen.
- Agregando las líneas similares a si la imagen se reflejara en un espejo.
- Agregando líneas como si la imagen se repitiera cíclicamente.

A continuación será tratado brevemente cada una de estas formas de resolver el problema de las fronteras en imágenes y al mismo tiempo se relacionaran estos conceptos a la forma en como MatLAB incorpora las soluciones en sus funciones.

Agregando líneas de ceros en las fronteras. Esta forma es la que ya ha sido mencionada anteriormente como solución al problema de la frontera en imágenes, en la que se agregan líneas de ceros tantas como sean necesarias (dependiendo del tamaño del filtro) para garantizar la correspondencia coeficiente-píxel. En MatLAB esta opción de añadir ceros a la imagen es considerada con la bandera de `'0'`. La figura 5.30 ilustra este proceso.

Figura 5.30 Añadir ceros a la imagen con el objetivo de garantizar la correspondencia de cada coeficiente del filtro por píxel de la imagen.

Replicando líneas de valores iguales a las últimas filas o columnas de la imagen. En esta opción ala imagen se le añaden líneas que son iguales a las líneas frontera de la imagen, lo que en otras palabras significaría aumentar el tamaño de la imagen repitiendo la última fila o columna de la imagen según sea el caso y tantas veces como sea necesario (dependiendo del tamaño del filtro). Esta forma de resolver el problema de la frontera de la imagen es preferida en comparación a la de solo añadir ceros. En MatLAB esta opción es definida por la bandera **`'replicate'`**. La figura 5.31 ilustra este proceso.

Figura 5.31 Añadir la última columna o renglón según el caso a la imagen con el objetivo de garantizar la correspondencia de cada coeficiente del filtro por píxel de la imagen.

Agregando las líneas similares a si la imagen se reflejara en un espejo. En este caso se agregan renglones o columnas a la imagen de forma como si se reflejara la imagen en un espejo en sus fronteras. Esta opción es un poco más complicada de implementar que las anteriores puesto que los renglones o columnas no son fijos como las anteriores sino que varían dependiendo de cuántos será necesario añadir (dependiendo del tamaño del filtro) a la imagen. En MatLAB esta opción es definida por la bandera **`'symmetric'.`** La figura 5.32 ilustra este proceso.

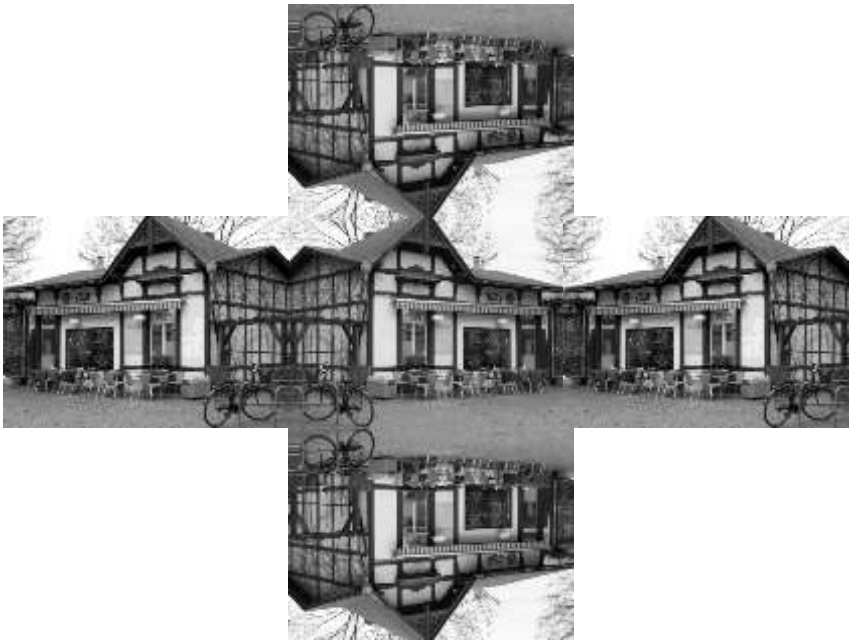


Figura 5.32 Añadir columnas o renglones según el caso a la imagen como si la imagen se reflejara en un espejo en las fronteras de la imagen con el objetivo de garantizar la correspondencia de cada coeficiente del filtro por píxel de la imagen.

Agregando líneas como si la imagen se repitiera cíclicamente. En este caso se agregan renglones y columnas a la imagen considerando como si esta fuera una señal que se repite en todos los sentidos, es decir como si la imagen estuviera empalmada con una imagen arriba y abajo, a un lado y al otro. En MatLAB esta opción es definida por la bandera **`'circular'`.** La figura 5.33 ilustra este proceso.

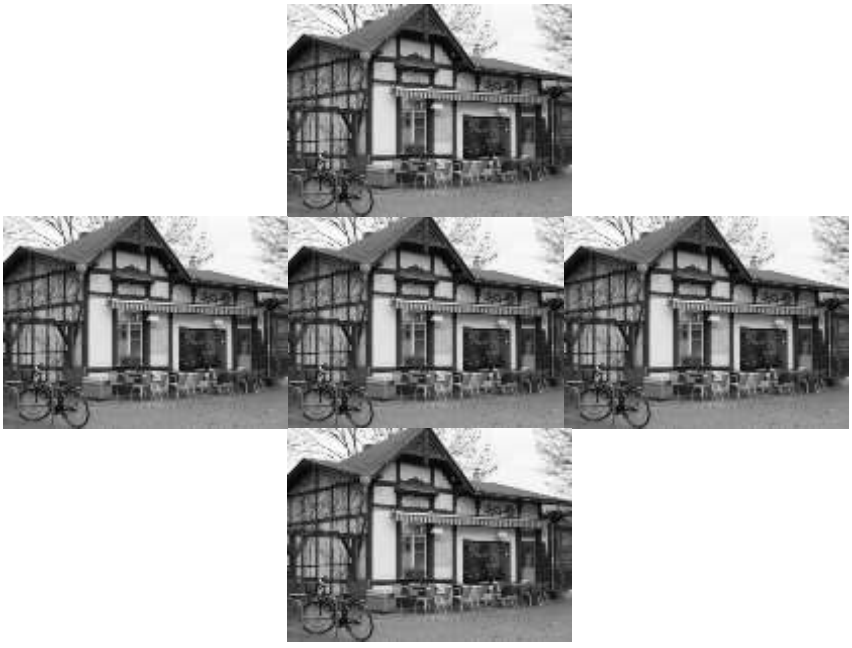


Figura 5.33 Añadir columnas o renglones según el caso a la imagen como si la imagen se repitiera cíclicamente en las fronteras de la imagen con el objetivo de garantizar la correspondencia de cada coeficiente del filtro por píxel de la imagen.

5.8.3 Funciones de MatLAB para la implementación de los filtros lineales espaciales

El toolbox de procesamiento de imágenes de MatLAB implementa para el filtraje espacial lineal de imágenes la función imfilter, la cual tiene la siguiente sintaxis:

Donde `g` es la imagen filtrada y `f` la imagen que se desea filtrar con el filtro `H`. La bandera `modo_filtrado` especifica si la función utilizara para el filtrado correlación (`'corr'`) o convolución (‘conv’). La bandera `opciones_frontera` se refiere a la forma en que se resolvera el problema de frontera por parte de la función, las opciones son `P`, `replicate`, `symetric` y `circular`, cuyos efectos e implicaciones ya fueron tratados en la seccion anterior 5.8.2. La bandera `tamaño_del_resultado` advierte el tamaño del resultado obtenido por el efecto

del filtro, sus opciones pueden ser: `same` y `full`. De igual manera los efectos de estas opciones ya fueron explicados en la sección 5.8.1. La tabla 5.1 muestra un resumen de las banderas y sus posibles opciones.

Opciones	Descripción
Modo de Filtrado	
corr	El filtraje se efectúa usando la correlación. Esta opción es la que se utiliza por *defecto*.
conv	El filtraje se efectúa usando la convolución.
Opciones de Frontera	
0	La imagen se completa añadiendo ceros.
replicate	La imagen se completa añadiendo filas o columnas que son iguales a las filas y columnas de las fronteras de la imagen. Es la opción por *defecto*.
symetric	La imagen se completa con filas y columnas iguales a los obtenidos por la proyección en espejo de los renglones y columnas de la imagen reflejada.
circular	La imagen se completa con filas y columnas iguales a los obtenidos si se considera ala imagen como una señal periódica, de tal forma que la imagen se repite arriba y abajo, aun lado y al otro.
Tamaño del resultado	
full	El resultado de la imagen filtrada es del tamaño de la imagen con las columnas y renglones adicionados en las fronteras para asegurar correspondencia píxel-coeficiente.

same	El resultado de la imagen filtrada es del tamaño de la imagen original. Es la opción por *defecto*.

Tabla 5.1 Descripción resumida de las banderas utilizadas por la función `imfilter` de MatLAB.

Como ha sido mencionado anteriormente, una convolución es lo mismo que una correlación habiendo pre-rotado el filtro 180°, lo contrario también es aplicable una correlación arroja el mismo resultado que una convolución habiendo pre-rotado el filtro 180°. Lo anterior es importante ya que el toolbox de procesamiento de imágenes define varios filtros especiales los cuales han sido pre-rotados para ser utilizados directamente con la correlación (opción por defecto) cuando originalmente fueron planeados para la realizar convolución sobre la imagen.

Cuando se trabaja con algún filtro y se piensa convolucionar con una imagen se tienen dos diferentes opciones una es o bien utilizar la bandera `modo_filtrado` (`conv`) de la función `imfilter`, o bien rotar al filtro 180° y aplicar la función `imfilter` con su opción por defecto (`corr`). Para rotar una matriz (en nuestro caso un filtro) 180° se pude utilizar la función:

Donde `Hr` es el filtro rotado 180° y `H` el filtro que se quiere rotar.

La figura 5.34 muestra la utilización de la función `imfilter` y los efectos de utilizar las diferentes opciones correspondientes a la bandera `opciones_frontera` considerando como filtro `H`, un filtro de suavizado "Box" de 31x31, el cual fue obtenido por la siguiente instrucción en MatLAB:

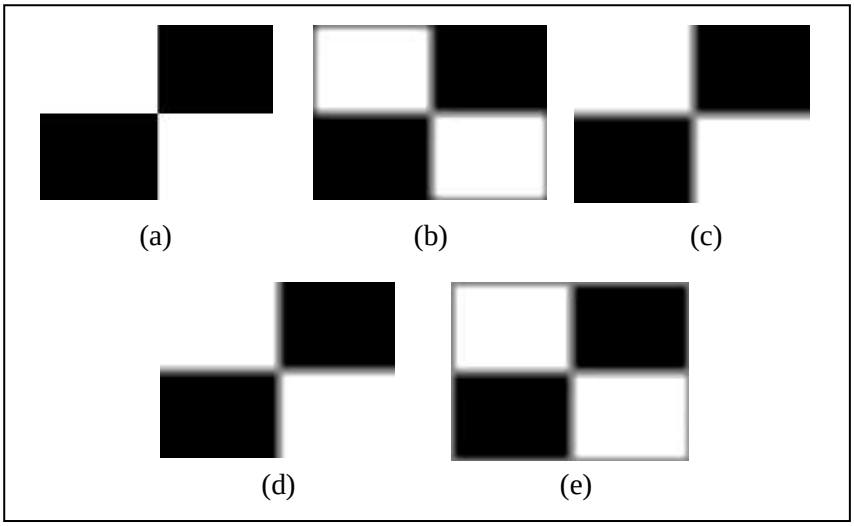


Figura 5.34. Efecto de utilizar las diferentes opciones de la bandera opciones_frontera. (a) Imagen original, (b) efecto en la imagen al utilizar la opción '0', (c) efecto en la imagen al utilizar la opción 'replicate', (d) efecto en la imagen al utilizar la opción 'symmetric' y (e) efecto en la imagen al utilizar la opción 'circular'.

Como puede verse de la figura 5.34(b) la opción `'0'` al añadir renglones y columnas con ceros produce un efecto de suavizado de negro a blanco en regiones donde en la imagen original no tiene píxeles negros, lo mismo sucede en la figura 5.34(e) la cual al añadirle filas y columnas iguales a partes de la imagen (`'circular'`) como si esta se repitiese arriba y abajo, aun lado y al otro se genera el mismo efecto al haber regiones en negro producto de este proceder.

5.8.4 Funciones de MatLAB para el filtraje espacial no lineal

El filtraje no lineal como fue tratado anteriormente se basa al igual que el filtraje lineal en la misma mecánica de desplazamiento de una mascarilla a lo largo de la imagen, sin embargo a diferencia del lineal sus operaciones no son basadas en la suma de las multiplicaciones parciales resultantes entre los coeficientes del filtro y los píxeles de la imagen. Otra marcada diferencia entre el filtraje lineal y el no lineal es que el concepto de mascarilla o coeficientes del filtro deja de ser determinante para convertirse en una operación no lineal efectuada sobre píxeles que se encuentran en una determinada región de influencia, por lo que la mascarilla ahora solo tendrá una significación de que vecinos son los que se consideraran en el procesamiento y no la forma en que éstos participaran en el procesamiento, como es el caso en el filtraje lineal.

El toolbox de procesamiento de imágenes dispone de 2 funciones para realizar filtraje no lineal sobre imágenes. Estas son: `nlfilter` y `colfilt`.

La funcion **`nlfilter`** permite filtrar utilizando una operación no lineal sobre una imagen, su sintaxis general es:

Donde `m` y `n` determina el tamaño de la región de influencia del filtro no lineal que se desplazará a lo largo y ancho de la imagen, `A` la imagen que se filtrará mientras que B contendrá a la imagen filtrada. `fun` define una función de usuario la cual implementa la funcionalidad del filtro, esta función recibe de `nlfilter` un vector de mxn datos y devuelve el resultado de la operación no lineal implementada sobre esos datos. El símbolo `@` es llamado identificador de la función (function handle) el cual es un tipo de dato de MatLAB que contiene información de la función que se esta utilizando como referencia para implementar el cálculo.

Como ejemplo calcularemos el filtro de la mediana utilizando la función `nlfilter`, considerando que la imagen original es la ilustrada en la figura 5.35(a) se le añade ruido del tipo sal y pimienta al 10%, obteniéndose la figura 5.35(b), lo cual lo realizamos con el comando:

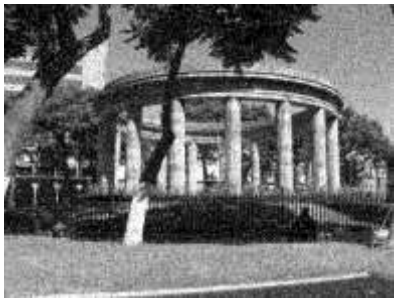
Para el filtraje utilizaremos un filtro de la mediana definido por una vecindad de 5x5, por lo que la función (mediana) que implementará esta operación recibirá de `nlfilter` un vector de 25 datos, sobre los cuales el responderá con el valor de la mediana. La función creada para implementar esta función se encuentra especificada en el programa 5.3. Una vez implementada la función que será utilizada por la función `nlfilter`, se filtra la imagen colocando en línea de comandos:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Función para implementar el filtro no lineal de la mediana
%usando la función nlfilter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 29/12/07
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Se crea la función con el nombre con el que será llamada
%recibe el vector x, que son los píxeles que se encuentran %en la región de influencia mxn
function v=mediana(x)
%se encuentra la mediana del conjunto de datos y ese dato es
%el entregado por la función
v=median(x(:));
```

Programa 5.3 Función implementada para encontrar la mediana de un conjunto de datos, que permite implementar el filtro espacial no lineal de la mediana mediante la función de MatLAB `nlfilter`.

La figura 5.35 ilustra las imágenes utilizadas para el filtrado realizado por `nlfilter` y los efectos obtenidos de este proceso.

(a)



(b)

(c)

Figura5.35 Utilización de la función nlfilter. (a) Imagen original, (b) imagen distorsionada con ruido de sal y pimienta y (c) imagen filtrada por nlfilter usando la función definida en el programa 5.3 que implementa el filtro no lineal espacial de la mediana.

La función **`colfilt`** realiza el filtrado organizando los datos en forma de columna, esta función es preferida en lugar de la función `nlfilter`, ya que desempeña el cálculo en una forma mas rápida.

La función `colfilt`, considerando una región de interés mxn opera sobre una imagen `I` de tamaño MxN generando una matriz A de tamaño mnxMN, en la cual cada columna corresponde a los píxeles que forman el área de interés centrada en una posición de la imagen. Por ejemplo la primera columna de la matriz A estará constituida por los píxeles que forman parte del área de interés centrada en el píxel de la parte superior izquierda (0,0) de la imagen `I` de dimensión MxN. La sintaxis general de esta función es:

Donde m y n es el tamaño de la región de interés considerada o tamaño del filtro, `'sliding'` indica que la mascara mxn se desliza a lo largo de toda la imagen `I` realizando la operación píxel a píxel. `fun` define una función de usuario la cual implementa la funcionalidad del filtro, esta función recibe de `colfilt` la matriz A de mnxMN datos y devuelve como resultado un vector de 1xMN (del tamaño de la imagen `I`) datos producto de la operación no lineal implementada.

Con el objetivo de evidenciar diferencias como ejemplo se calculará de nueva cuenta el filtro de la mediana pero ahora con la función `colfilt`. Al igual que para el caso de la función `nlfilter`, se elige un filtro con una área de interés de 5x5. La función por lo tanto debe de implementarse de tal manera que reciba una matriz de tamaño 25xMN, donde M y N son las dimensiones de la imagen a filtrar y devuelva como resultado un vector de 1xMN que representa las medianas de todos los píxeles de la imagen donde se centro la mascarilla 5x5. El programa 5.4 muestra la implementación de esta función.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Función para implementar el filtro no lineal de la mediana
%usando la función colfilt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 29/12/07
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Se crea la función con el nombre con el que será llamada
%recibe la matriz A de dimensión mnxMN
function v1=mediana1(A)
%se encuentra la mediana de cada columna de la matriz A que son
%25 datos (5x5) y entrega un vector v1 de 1xMN con las medianas
%de toda la imagen, será lo mismo con median(A,1)
V1=median(A);
```

Programa 5.4 Función implementada para encontrar la mediana de una matriz A de mnxMN datos, que permite implementar el filtro espacial no lineal de la mediana mediante la función de MatLAB `colfilt`.

5.9 BLOQUES PARA EL FILTRADO LINEAL ESPACIAL DE LA LIBRERÍA DE PROCESAMIENTO DE IMÁGENES Y VIDEO DE SIMULINK®

La librería de bloques de procesamiento de imágenes y video de Simulink posee varios elementos útiles para el procesamiento espacial lineal de video en tiempo real o imágenes. La idea de esta sección es explicar la funcionalidad de cada uno de los bloques y utilizarlos en algunos ejemplos que permiten mostrar algunos otros requisitos importantes que es necesario considerar para su conexión.

2-D Convolución

El bloque de "2-D Convolucion" realiza la convolución de dos matrices. Asumiendo que la matriz A tiene las dimensiones (M_a, N_a) y la matriz B tiene las dimensiones (M_b, N_b), cuando el bloque calcula la convolución, el cálculo realizado se efectúa en base a la siguiente ecuación.

$$C(i,j) = \sum_{m=0}^{(M_a-1)} \sum_{n=0}^{(N_a-1)} A(m,n) \cdot B(i-m, j-n) \tag{5.38}$$

Donde $0 \le i < M_a + M_b - 1$ y $0 \le j < N_a + N_b - 1$. La Figura 5.36 muestra el esquema del bloque en Simulink. Así mismo en la Tabla 5.2 se describe la funcionalidad entrada/salida de sus conexiones.

Tal y como fue visto en la sección 5.5.1 la convolución puede ser considerada como una operación de filtro normal (correlación) con la excepción de que la matriz $H(i,j)$ de coeficientes debe de ser invertida en sus ejes coordenados $H(-i,-j)$, lo cual puede ser llevado a cabo por una rotación de $H(i,j)$ en 180°.

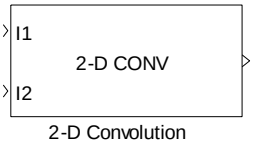


Figura 5.36 Bloque de la librería para el procesamiento de imagen y video de Simulink para realizar la convolución espacial entre dos matrices, normalmente una imagen y una ventana donde se encuentra contenido la funcionalidad del filtro.

Conexión	Descripción
I1	Matriz o una imagen a escala de grises, o bien un plano de la imagen RGB.
I2	Matriz, ventana, kernel o filtro
Salida	La imagen o matriz resultado de la operación de I1 y I2.

Tabla 5.2 Funcionalidad de las conexiones entrada/salida del bloque 2-D Convolution.

El bloque tiene la funcionalidad de detectar el tipo de dato utilizado a sus entradas generando de esta manera una salida coherente con el tipo de datos de entrada.

Las dimensiones de la matriz o imagen de salida se encuentran determinadas por el parámetro de configuración "Output Size". Asumiendo que las dimensiones de la conexión I1 son (Ma, Na) y para la la conexión I2 son (Mb, Nb). Si se elige el parámetro de configuración "Output Size" a `Full`, la salida es la imagen producida por la convolución de I1 e I2, con dimensiones (Ma+Mb-1, Na+Nb-1). Si por el contrario se elige el parámetro de configuración "Output Size" a `Same as input port I1`, la salida es la parte central de la convolución verificada entre I1 e I2, omitiendo las partes laterales o bordes de la imagen o matriz resultado de la convolución completa, por consiguiente las dimensiones de la matriz o imagen resultado son los mismos de los de I1. Por último el parámetro de configuración "Output Size" puede también tener el valor de `valid`, con lo que la imagen o matriz de salida será el resultado de la convolución entre I1 e I2 sin considerar añadir ceros en los bordes de la matriz I1, de tal forma que pueda realizarse la multiplicación de los coeficientes de I2.

En la convolución, el valor de un elemento de la salida es computado como la suma de las multiplicaciones de los elementos vecinos. Por ejemplo, supóngase que la primera matriz del bloque I1 es definida como:

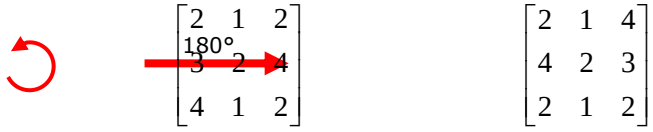
$$I1=\begin{bmatrix} 2 & 3 & 1 & 3 \\ 6 & 7 & 2 & 1 \\ 5 & 6 & 1 & 0 \\ 5 & 4 & 8 & 1 \end{bmatrix} \tag{5.39}$$

La segunda matriz de entrada al bloque I2 la cual normalmente representa la ventana que define la matriz de coeficientes y las vecindades tomadas en cuenta para el cálculo de los elementos de salida de la convolución, es definida como:

$$I2=\begin{bmatrix} 2 & 1 & 2 \\ 3 & 2 & 4 \\ 4 & 1 & 2 \end{bmatrix} \tag{5.40}$$

Considerando estas dos matrices, el proceso para calcular el elemento (2,2) de la salida resultado de la convolución entre I1 e I2 es el siguiente:

1. Tal como se observa en la ecuación 5.38 se rota la matriz I2 180° con respecto a su punto central.



2. Se desplaza el elemento central de la matriz I2 rotada de tal forma que su elemento central coincida con el elemento (2,2) de la matriz I1.

3. Se multiplica cada elemento de la matriz I2 rotada por los elementos con los que coinciden de la matriz I2, que se encuentra debajo de ella.

4. Se suman los productos individuales del paso 3, para calcular el valor total de (2,2).

La Figura 5.37 muestra un esquema de la forma en como se desarrolla el procesamiento de la convolución para el elemento (2,2), considerando como I1 e I2 las matrices anteriormente descritas.

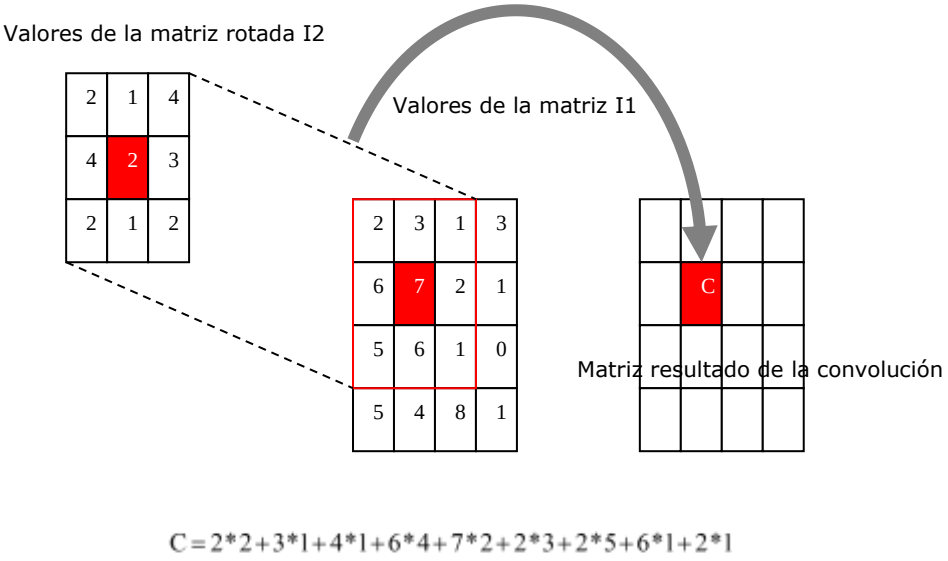


Figura 5.37 Representación de la convolución realizada entre las matrices I1 e I2, considerando que se calcula el elemento (2,2), según el proceso anteriormente tratado. Del proceso puede notarse como la ventana de convolución, la cual normalmente es utilizada para realizar un procesamiento espacial en la imagen, debe ser rotada 180° antes de realizar la multiplicación.

2-D Correlation

El bloque de "2-D correlation" ubicado en la categoría de "Statistics" de la librería de bloques para el procesamiento de imágenes y video permite computar la correlación cruzada de dos matrices. Si se asume que se tiene una matriz A de dimensiones (M_a, N_a) y una matriz B de dimensiones (M_b, N_b) cuando el bloque calcula la correlación, el cálculo realizado se efectúa en base a la siguiente ecuación:

$$C(i,j) = \sum_{m=0}^{(M_a-1)} \sum_{n=0}^{(N_a-1)} A(m,n) \cdot B(m+i, n+j) \tag{5.41}$$

Donde $0 \le i < M_a + M_b - 1$ y $0 \le j < N_a + N_b - 1$. La Figura 5.38 muestra el esquema del bloque en Simulink. Así mismo en la Tabla 5.1 se describe la funcionalidad entrada/salida de sus conexiones, las cuales coinciden también con los correspondientes a los utilizados por el bloque de la convolución.

Figura 5.38 Bloque de la librería para el procesamiento de imagen y video de Simulink para realizar la correlación cruzada espacial entre dos matrices, normalmente una imagen y una ventana donde se encuentra contenido la funcionalidad del filtro.

Las generalidades de los tipos de datos utilizados y el tamaño del resultado obtenido por la operación de este bloque son iguales a los manejados por el bloque de convolución ya explicado anteriormente.

Como ya ha sido mencionado anteriormente la diferencia entre las operaciones de convolución y correlación radica únicamente en la rotación de 180° que experimenta el filtro o matriz de coeficientes en la convolución antes de realizar la operación de desplazar el filtro sobre la imagen, mientras que en el caso de la correlación no se realiza rotación de la mascarilla.

Una opción importante que puede ser configurado tanto en el bloque de "2-D correlation" y "2-D convolution" es el de salida normalizada (Normalizad Output), muy útil particularmente cuando en la operación (ya sea convolución o correlación) se utiliza un filtro cuyos coeficientes son todos positivos, lo que normalmente hace que el resultado exceda el valor permisible manejado por la imagen y mascarilla utilizadas en la operación, de tal forma que una normalización del resultado es necesaria. La figura 5.39 muestra la mascarilla de configuración del bloque de "2-D correlation".

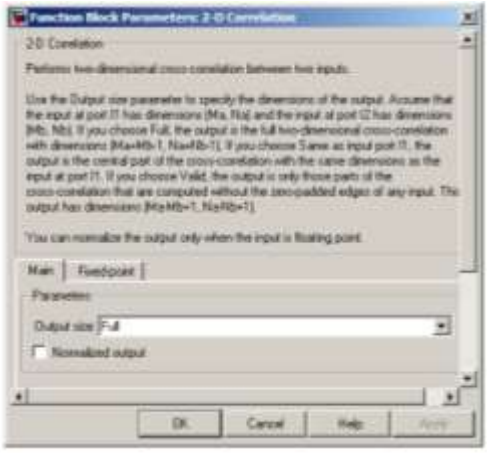


Figura 5.39 Mascarilla de configuración del bloque "2-D correlation".

Por lo tanto si se considera como el filtro la matriz definida como:

$$H = \begin{bmatrix} h_{11} & \cdots & h_{1n} \\ \vdots & \vdots & \vdots \\ h_{m1} & \cdots & h_{mn} \end{bmatrix} \tag{5.42}$$

Y si se define la sección de la imagen con la que se alinea el filtro como:

$$I_s = \begin{bmatrix} I_{11} & \cdots & I_{1n} \\ \vdots & \vdots & \vdots \\ I_{m1} & \cdots & I_{mn} \end{bmatrix} \tag{5.43}$$

La salida individual de cada una de las operaciones filtro-imagen desempeñada por los bloques "2-D correlation" o "2-D convolution, si se selecciona la opción salida normalizada (Normalizad Output), es dividida por el factor:

$$Fac = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} h_{ij}^2 \cdot \sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^2} \tag{5.44}$$

5.9.1 Ejemplos de Filtrado lineal en Simulink

En esta sección se desarrollan 2 ejemplos que ilustran la forma en como utilizar los bloques de "2-D correlation" y "2-D convolution" para el filtraje lineal de frames de video. En el primero de ellos se implementara el filtro Gaussiano definido por el kernel ilustrado en la figura 5.9(c), para ello se utilizara el bloque "2-D correlation". En el segundo ejemplo se implementara el filtro de Laplace definido por la mascarilla mostrada en la figura 5.10(c), en este caso se utilizara el bloque de "2-D convolution".

<u>Ejercicio 1.</u> Filtro Gaussiano.

Para poder procesar video utilizando las herramientas de bloques de Simulink en un programa el primer paso es disponer de una fuente de video. En capítulos anteriores ha sido utilizada la Webcam como elemento que permite introducir los frames de video para su procesamiento en el programa, sin embargo en este ejemplo utilizaremos un bloque que permite introducir en su configuración por defecto un archivo de video de demostración el cual puede utilizarse en los casos que no se posea una Webcam para la realización de los programas para el procesamiento de video de Simulink.

Bloque "From Multimedia File".

Este bloque ubicado en la categoría de fuentes (Sources) de la librería para el procesamiento de imagen y video de Simulink permite utilizar archivos de video avi como fuente de video a programas en Simulink. La figura 5.40 muestra una representación del bloque "From Multimedia File". Cuando el bloque se coloca por primera vez sobre el programa, el bloque se encuentra configurado para reproducir un archivo de video demo, el cual puede ser utilizado para procesar video en programas de Simulink cuando no se dispone de alguna otra fuente como la Webcam.

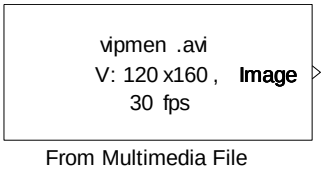


Figure 5.40 Bloque "From Multimedia File" de la librería para el procesamiento de imagen y video de Simulink.

Bloque "Image From Workspace"

Este bloque perteneciente a la categoría de Funetes (Source) permite introducir una imagen o matriz del workspace o línea de comandos de MatLAB a un programa de Simulink, este bloque será utilizado intensivamente en este libro ya que la necesidad de introducir una matriz o imagen de forma constante en un programa en Simulink es importante. La figura 5.41 muestra una representación del bloque y su mascarilla de configuración. El único elemento importante de configuración (1) de este bloque es colocar el nombre de la variable que en línea de comandos contiene a la matriz o la imagen.

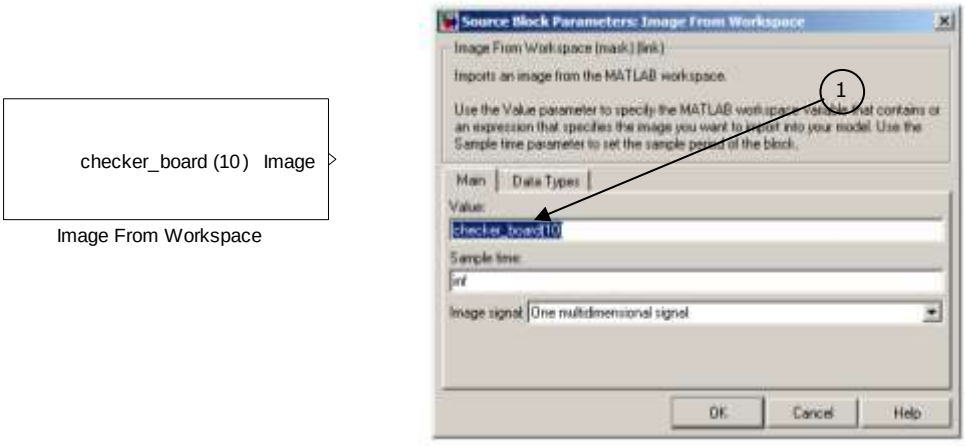


Figura 5.41 Bloque "Image From Workspace" y su mascarilla de configuración.

Una vez que se ha explicado y analizado la configuración de los principales bloques funcionales que representan la temática de este capitulo y algunos otros que permiten realizar operaciones auxiliares importantes se configurara el programa en Simulink mostrado en la figura 5.42.

Como se muestra este programa tiene como elemento fuente de video el bloque "From Multimedia File" que es su configuración inicial tiene un video de demostración que consiste en el encuentro de dos personas en un corredor. Las imágenes o frames entregadas por la fuente son convertidas a imágenes de intensidad (escala de grises) a través del bloque "Color Space Conversion" que se encuentra en la categoría de Conversiones (Conversions). Este bloque habrá que configurarlo a través de su mascarilla de configuración para definirlo como convertidor de RGB a imagen de intensidad.

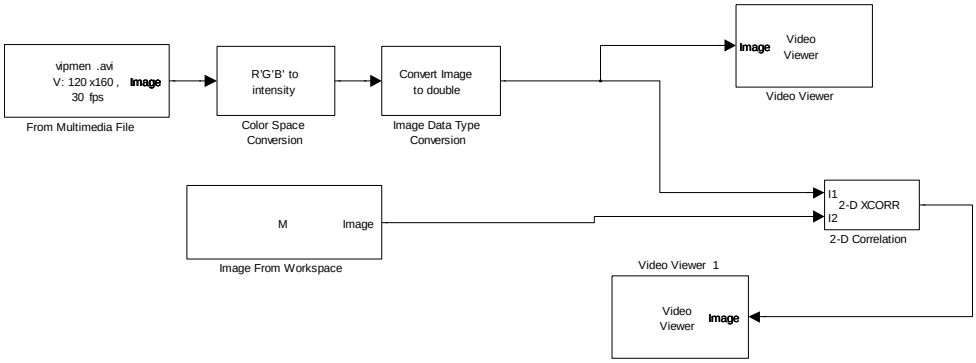


Figura 5.42 Programa que implementa el filtro Gaussiano, mediante la utilización de bloques de la librería de procesamiento de imagen y video de Simulink.

Posteriormente el tipo de dato de la imagen trasformada a escala de grises es convertido de `uint8` a `double`, mediante la utilización del bloque "Image Data Type Conversión". La conversión se presume necesaria por el hecho de que en el bloque "2-D correlation" al realizar su operación mediante multiplicación requerirá que el tipo de entrada de la imagen sea double. El bloque "2-D correlation" opera por lo tanto con la imagen de intensidad convertida a `double` y el kernel `M` que expresa al filtro Gaussiano (véase figura 5.9(c)) definido por línea de comandos como:

Como fue mencionado ya anteriormente al tratarse de un filtro con una matriz de coeficientes puramente positiva será necesario una normalización del resultado, por lo que en la mascarilla de configuración del bloque "2-D correlation" se selecciona la opción Normalizad Output para que el resultado de cada correlación sea dividido por el factor definido en la ecuación 5.44. De igual manera el bloque de "2-D correlation" debe de ser configurado para que el tamaño de la operación sea igual que el de la imagen que se correlaciona, eligiendo la opción `Same as input port I1`.

Los resultados son mostrados tanto del frame original como el procesado por la acción del filtro mediante el bloque "Video Viewer" perteneciente a la categoría monitores (Sinks) que permite observar el video procesado o los frames en forma consecutiva.

Para probar la inmunidad del filtro Gaussiano al ruido y al mismo tiempo su propiedad de no degradar los bordes de la imagen como producto de la suavización de los artefactos presentes en la imagen se le añade al programa en Simulink una función de MatLAB que recibe como entrada una matriz, que en este caso es la imagen trasformada a escala de grises y convertida a `double` y devuelve una matriz (o imagen) de tipo de dato `double` con ruido en la imagen. En el formato de ruido añadido a la imagen la posición del píxel es escogida aleatoriamente así como también el grado de distorsión del mismo, el cual pude ir de ±25. La función implementada en MatLAB para añadir el ruido es mostrada en el Programa 5.4.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%Función para añadir ruido a una imagen en
%Simulink
%El tipo de ruido es aleatorio en la posición
%así %como
%en la distorsión del píxel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 02/1/08
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

%La función recibe la imagen en la Matriz A y
%devuelve la imagen B con el ruido añadido
function B = ruidopro(A)
%Se hace una copia de la imagen A en B
B=A;
%Se obtiene el tamaño de la imagen
[m n]=size(A);
%Se distorsionan 300 puntos de la imagen
for v=1:300
    %Se obtienen las posiciones del píxel a
    %Distorsionar
    x=round((m-1)*rand);
    y=round((n-1)*rand);
    %Se protege si el valor obtenido es cero
    if(x==0)
        x=1;
    end
    %Se protege si el valor obtenido es cero
    if(y==0)
        y=1;
    end
    %Una vez obtenido la posición del píxel se
distorsiona
    %para lo cual se le suma o resta 25 (0.099
en el caso double)
    pix=B(x,y)+((rand-0.5)*2*0.099);
    %Se protege el dato si excede los limites
mínimo y máximo
    if(pix<0)
        pix=0;
    end
    if(pix>1)
        pix=1;
```

```
        end

        %Se modifica finalmente el píxel de la imagen
    que devuelve
        %la función
        B(x,y)=pix;
     end
```

Programa 5.4 Programa que añade ruido a una imagen en Simulink utilizado para probar los efectos del filtro Gaussiano.

Como puede verse en el programa 5.4 la posición del píxel que se elige aleatoriamente debe ser protegida cuando el valor arrojado por la función aleatoria es cero, ya que las matrices y vectores en MatLAB son indexadas a partir de 1.

Cuando una imagen es convertida del formato `uint8` (normal para el despliegue de imágenes) a `double` los rangos de variación del tipo de dato de la imagen cambian de [0 255] a [0 1] por lo que no debe sorprender el cambio de escala que se ve en el programa 5.4. Considerando lo anterior la distorsión aleatoria que se realiza sobre el píxel seleccionado que se describía de -25 a 25 sucederá en el cambio de tipo de dato de -0.099 a 0.099. Para calcular la distorsión aleatoria se utiliza la función `rand` la cual calcula un número aleatorio igualmente distribuido en el intervalo 0 a 1. Para ampliarlo al intervalo -1 a 1 se realiza la transformación desempeñada por:

$$(\texttt{rand} - 0.5) \cdot 2 \tag{5.45}$$

Donde el valor calculado dentro del paréntesis variara de -0.5 a 0.5 y multiplicado por dos de -1 a 1.

Para añadir el ruido solo se incluye al programa mostrado en la figura 5.43 el bloque “MATLAB Fcn” de la categoría funciones definidas por el usuario (User-defined functions) y configurarlo escribiendo el nombre de la función que se ejecutara en ese bloque que en el caso aquí mostrado será “ruidopro”. El programa por lo tanto mostrará ahora la forma ilustrada en la figura 5.43.

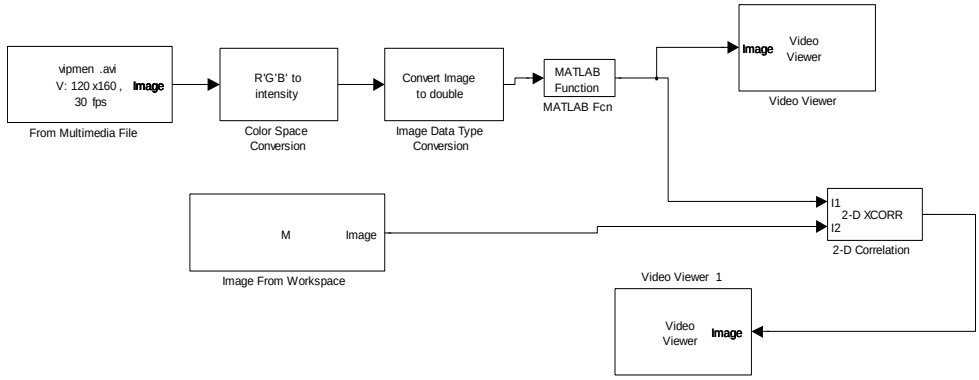


Fig.ura 5.43 Programa que implementa el filtro Gaussiano ante el efecto de ruido añadido por la función mostrada en el programa 5.4, mediante la utilización de bloques de la librería de procesamiento de imagen y video de Simulink.

Los resultados obtenidos muestran como el filtro Gaussiano permite eliminar el ruido provocado por la función que lo añadía artificialmente sin degradar substancialmente la significación de los bordes. La figura 5.44 muestra el efecto de ejecutar el programa mostrado en la figura 5.43.

<u>Ejercicio 2.</u> Filtro de Laplace.

En este ejercicio implementaremos en Simulink el filtro definido en la figura 5.10(c) llamado filtro de Laplace o del sombrero mexicano definido en la ecuación 5.10. La diferencia en este caso es que se utilizara para la implementación del filtro el bloque de "2-D convolution". El programa para la implementación de este filtro es prácticamente igual al mostrado en la figura 5.42 con la diferencia de que en lugar del bloque "2-D correlation" se tendrá "2-D convolution".

Un aspecto importante como ya ha sido tratado es que la convolución es una operación equivalente a la correlación con la excepción de que la matriz de coeficientes debe ser rotada 180°, sin embargo debido a que el filtro de Laplace es isotrópico (invariante a la rotación) no será necesario rotarlo pues dará el mismo resultado. La figura 5.44 muestra el programa en Simulink que implementa el filtro de Laplace. De igual manera el bloque de "2-D convolution" debe de ser configurado para que el tamaño de la operación sea igual que el de la imagen que se convoluciona, eligiendo la opción `Same as input port I1.`

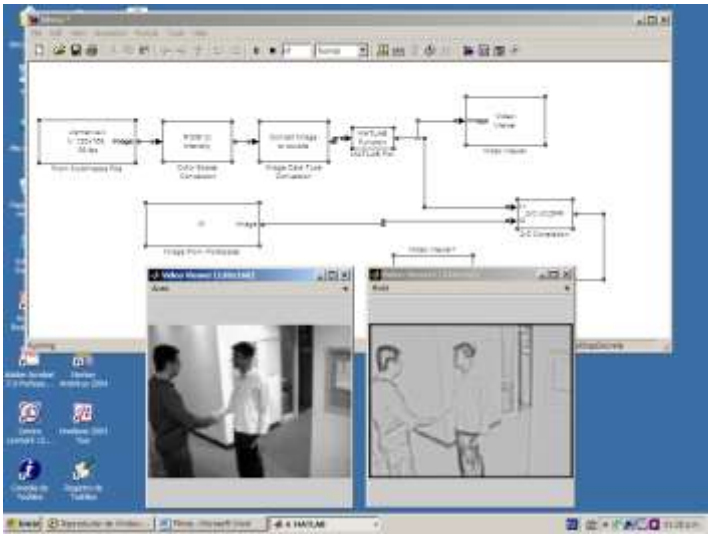


Figura 5.44 Resultado de ejecutar el programa mostrado en la figura 5.43.

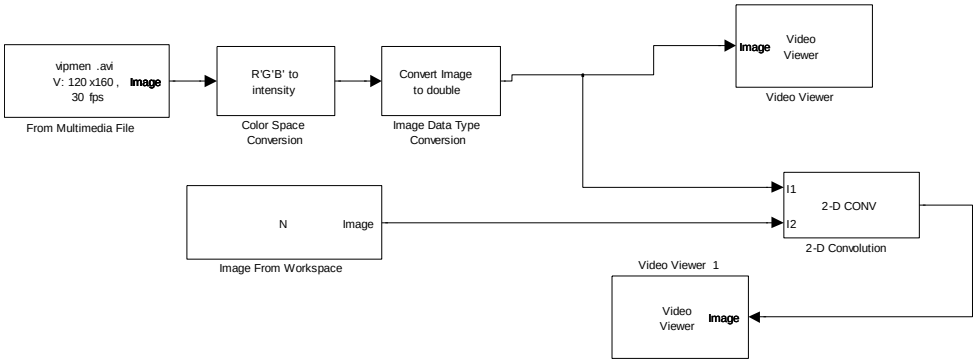


Figura 5.45 Programa que implementa el filtro de Laplace, mediante la utilización de bloques de la librería de procesamiento de imagen y video de Simulink.

Únicamente lo que resta es crear la variable `N` en línea de comandos que define al filtro de Laplace (véase figura 5.10(c)), dicha matriz de coeficientes no se rotará al ser isotrópico, por lo que será necesariamente definir solamente:

```
0 -1 -2 -1 0;
```

0 0 -1 0 0];

5.10 BLOQUES PARA EL FILTRADO NO LINEAL ESPACIAL DE LA LIBRERÍA DE PROCESAMIENTO DE IMÁGENES Y VIDEO DE SIMULINK®

La librería de bloques para el procesamiento de imágenes y video de Simulink no posee más que un bloque para poder realizar filtraje no lineal espacial sobre video o imágenes. Aunque es posible implementar cualquier otro filtro mediante la incorporación de una función de usuario como fue visto en la sección anterior, cuando se uso una función para añadir ruido ala imagen, en esta sección solo se explicará el filtro ya implementado por la librería de Simulink, que es el filtro de la mediana.

El Bloque "Median Filter"

Este bloque perteneciente a la categoría de filtrado (Filtering) implementa la funcionalidad del filtro de la mediana explicado ya en la sección 5.6.2 de este mismo capítulo. La figura 5.43 muestra la descripción del bloque "Median Filter".

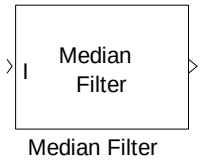


Figura 5.46 Bloque "Median Filter" de la librería para el procesamiento de imagen y video de Simulink.

Los principales elementos de configuración de este bloque es el tamaño del filtro el cual definirá el tamaño de la región de influencia sobre la cual se considera el cálculo de la mediana y el tamaño del resultado. La figura 5.46 muestra la mascarilla de configuración de este bloque.

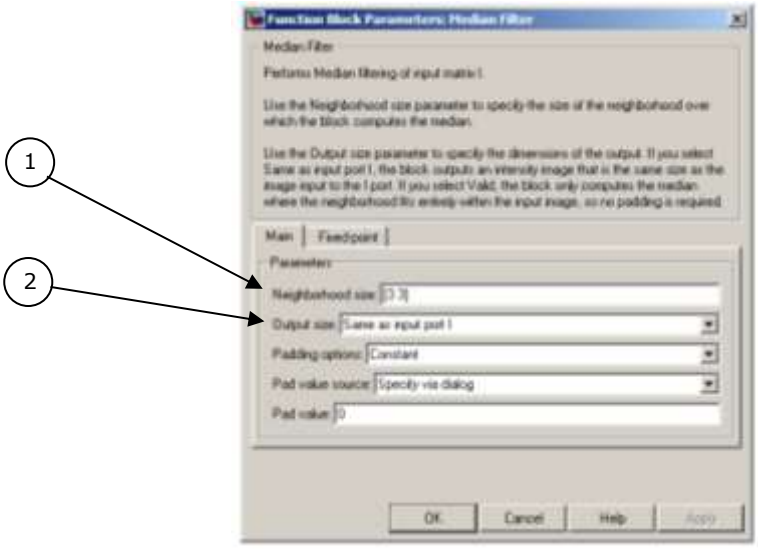


Figura 5.47 Mascarilla de configuración del Bloque "Median Filter"

La consideración del tamaño de la región de influencia del filtro es configurada en el campo (1) tamaño de la vecindad (Neighborhood size), mientras que el tamaño del resultado es configurado en el campo (2) tamaño de la salida (Output size). El tamaño de la región de influencia como ya fue explicado en la sección 5.6.2 es un factor determinante en el filtraje de estructuras o artefactos en la imagen mientras que el tamaño del resultado solo expresa el criterio a seguir con respecto al problema de correspondencia del píxel de la imagen y coeficiente del filtro en las fronteras de la imagen por lo que este campo normalmente es configurado a `Same as input port I`.

5.10.1 Ejemplo de Filtrado no lineal en Simulink

En esta sección se mostrara la forma de utilizar el filtro de la mediana para la eliminación del ruido de sal y pimienta añadido artificialmente a frames de video. El programa que se realizará es muy similar al mostrado en la figura 5.43 donde de igual manera se añade ruido a las imágenes provenientes de la fuente de video. El programa por lo tanto tiene la apariencia mostrado en la figura 5.47.

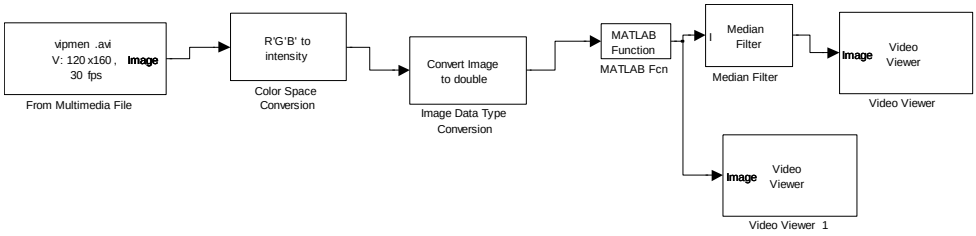


Figure 5.48 Programa que implementa el filtro de la mediana ante el efecto del ruido de sal y pimienta añadido por la función mostrada en el programa 5.5, mediante la utilización de bloques de la librería de procesamiento de imagen y video de Simulink.

El bloque de la mediana se configura de tal forma que el tamaño del filtro sea de 3x3. En este programa el bloque “MATLAB Fcn” que funcionalmente inyecta artificialmente el ruido de sal y pimienta, implementa la función “ruidosp”, la cual se encuentra descrita en el programa 5.5.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Función para añadir ruido a una imagen en Simulink
%El tipo de ruido el de sal y pimienta
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 02/1/08
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%La función recibe la imagen en la Matriz A y
%devuelve la imagen B con el ruido añadido
function B = ruidosp(A)
%Se hace una copia de la imagen A en B
B=A;
%Se obtiene el tamaño de la imagen
[m n]=size(A);
%Se distorsionan 300 puntos de la imagen con ruido
%de sal
for v=1:300
    %Se obtienen las posiciones del píxel a
    %Distorsionar
    x=round((m-1)*rand);
    y=round((n-1)*rand);
     %Se protege si el valor obtenido es cero
    if(x==0)
        x=1;
    end
    %Se protege si el valor obtenido es cero
    if(y==0)
        y=1;
    end
    %Se distorsiona el píxel
    B(x,y)=1;
```

```
 end
 %Se distorsionan 300 puntos de la imagen con
ruido
 %de pimienta
 for v=1:300
     x=round((m-1)*rand);
     y=round((n-1)*rand);

     if(x==0)
         x=1;
     end
     if(y==0)
         y=1;
     end
     B(x,y)=0;
 end
```

Programa 5.5 Programa que añade ruido de sal y pimienta a una imagen en Simulink utilizado para probar los efectos del filtro de la mediana.

El bloque "MATLAB Fcn" procesa los datos de las imágenes a partir de que el tipo de dato de estas ha sido cambiado a `double`, por lo que el intervalo de variación de 0 a 255 fue cambiado de 0 a 1. Considerando lo anterior el píxel de sal será añadido al definirlo en la función como 1, mientras que el píxel de pimienta será añadido en la función al definirlo como 0.

Al ejecutar el programa mostrado en la figura 5.47 muestra cómo el resultado del filtro sobre la imagen ruidosa hace ilógica la idea de que la imagen original contenía alguna especie de ruido. La figura 5.48 muestra el resultado de haber ejecutado el programa.

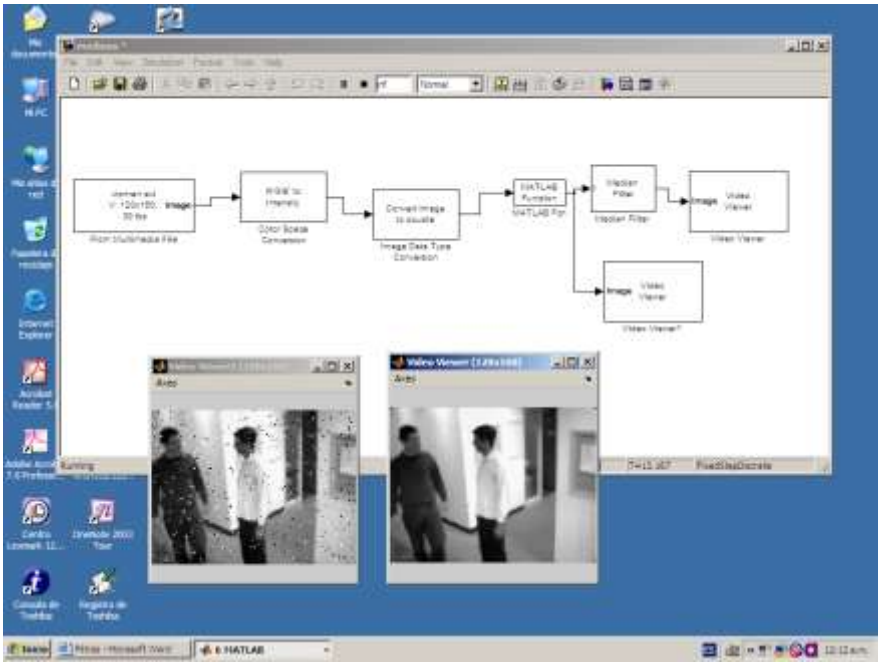


Figura 5.49 Resultado de ejecutar el programa mostrado en la figura 5.47.

5.11 FILTRO BINARIO

Este filtro esta ideado para funcionar sobre imágenes binarias, el cual consiste de una serie de ecuaciones booleanas que permiten eliminar ruido de zonas blancas y de zonas negras. También el filtro posee ecuaciones para reconstruir esquinas de figuras que tienen ángulos definidos. A continuación para explicar el funcionamiento de las ecuaciones del filtro se toma como base la imagen binaria creada artificialmente definida en la figura 5.49.

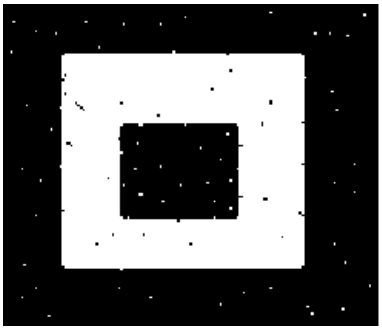


Figura 5.50 Imagen binaria utilizada para ejemplificar la operación del filtro binario.

Usando la ventana 3x3 mostrada en la figura 5.50 como referencia de procesamiento, se realiza el calculo para el píxel central P_0 de la mascara. El nuevo valor será obtenido a partir de *p* y de acuerdo a los píxeles *a,b,c,d,e,f,g,h* que lo rodean. El proceso de cálculo es similar al utilizado en los filtros lineales con la excepción de que en este caso se utilizan operaciones lógicas, actuando por consiguiente sobre píxeles que solo pueden tener los valores de 0 y 1.

a	*b*	*c*
d	***p***	*e*
f	*g*	*h*

Figura 5.51 Ventana de procesamiento usado en la operación del filtro binario.

El primer paso es quitar el ruido o artefactos negros de las zonas blancas, para ello se usa la ecuación:

$$P_0 = p + \big(b \cdot g \cdot \big(d + e\big)\big) + \big(d \cdot e \cdot \big(b + g\big)\big) \tag{5.46}$$

El resultado de este procesamiento es mostrado en la figura 5.51(a). El segundo paso es quitar el ruido o artefactos blancos de las zonas negras, para lo cual se usa la ecuación definida como:

$$P_0 = p \cdot (((a+b+d)\cdot(e+g+h)) + ((b+c+e)\cdot(d+f+g))) \tag{5.47}$$

El resultado de este procesamiento es mostrado en la figura 5.51(b). Para reconstruir las esquinas de la figura blanca, se deben de usar cuatro diferentes ecuaciones, una por cada esquina de la figura. Esto es:

Para la esquina superior derecha:

$$P_0 = \overline{p} \cdot d \cdot f \cdot g \cdot (\overline{a+b+c+e+h}) + p \tag{5.48}$$

Para la esquina inferior derecha:

$$P_0 = \overline{p} \cdot a \cdot b \cdot d \cdot (\overline{c+e+f+g+h}) + p \tag{5.49}$$

Para la esquina superior izquierda:

$$P_0 = \overline{p} \cdot e \cdot g \cdot h \cdot (\overline{a+b+c+d+f}) + p \tag{5.50}$$

Para la esquina inferior izquierda:

$$P_0 = \overline{p} \cdot b \cdot c \cdot e \cdot (\overline{a+d+f+g+h}) + p \tag{5.51}$$

El resultado de estas 4 operaciones es mostrado en la figura 5.51(c). Por ultimo a partir de las últimas cuatro ecuaciones se deducen otras cuatro que puedan ser utilizadas para la regeneración de las esquinas en las figuras negras:

Para la esquina superior derecha:

$$P_0 = p \cdot (d+f+g) + (\overline{a \cdot b \cdot c \cdot e \cdot h \cdot p}) \tag{5.52}$$

Para la esquina inferior derecha:

$$P_0 = p \cdot (a+b+d) + (\overline{c \cdot e \cdot f \cdot g \cdot h \cdot p}) \tag{5.53}$$

Para la esquina superior izquierda:

$$P_0 = p \cdot (e+g+h) + (\overline{a \cdot b \cdot c \cdot d \cdot f \cdot p}) \tag{5.54}$$

Para la esquina inferior izquierda:

$$P_0 = p \cdot (b+c+e) + (\overline{a \cdot d \cdot f \cdot g \cdot h \cdot p}) \tag{5.55}$$

El resultado de estas 4 operaciones, que definen el resultado final del procesamiento del filtro binario, es mostrado en la figura 5.51(d).

(a) (b) (c) (d)

Figura 5.51 Operación del Filtro binario. (a) Eliminación de los artefactos negros, (b) eliminación de los artefactos blancos, (c) regeneración de esquinas de elementos blancos y (d) regeneración de esquinas en objetos negros.

5.11.1 Implementación del filtro binario en MatLAB.

En esta sección se muestra la forma en la que puede ser implementado el filtro binario para operar sobre imágenes binarias. En la implementación del filtro se dividió su operación en 4 diferentes funciones, las cuales deben de ser aplicadas secuencialmente.

La primera función llamada `remove_black_noise` elimina los artefactos o puntos negros que son considerados como ruido. Esta función se muestra en el programa 5.6.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%Función para eliminar los artefactos o puntos negros
%de una imagen binaria, dentro de la operación del
%filtro binario
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 30/03/08
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function Ifiltered = remove_black_noise(Ibin)
Ifiltered = Ibin;
%Obtiene el tamaño de la imagen
[height, width] = size(Ifiltered);
%Se calcula el proceso para toda la imagen
for m= 2: (height-1)
for n= 2:  (width-1)
%Se obtienen los píxeles binarios correspondientes
%a una mascara 3x3
b= Ifiltered(m-1,n);
d= Ifiltered(m,n-1);
p= Ifiltered(m,n);
e= Ifiltered(m,n+1);
g= Ifiltered(m+1,n);
%se aplica la ecuación booleana definida en 5.46
Ifiltered(m,n)= p | (b&g&(d|e))|(d&e&(b|g));
end
end

return
```

Programa 5.6 Función que elimina los artefactos negros considerados ruido, como parte de la operación del filtro binario.

La función llamada `remove_white_point_noise` elimina los artefactos o puntos blancos que son considerados como ruido. Esta función se muestra en el programa 5.7.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%Función para eliminar los artefactos o puntos blancos
%de una imagen binaria, dentro de la operación del
%filtro binario
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 30/03/08
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function Ifiltered = remove_white_point_noise(Ibin)
Ifiltered = Ibin;
%Obtiene el tamaño de la imagen
[height, width] = size(Ifiltered);
%Se calcula el proceso para toda la imagen
for m= 2: (height-1)
for n= 2:  (width-1)
%Se obtienen los píxeles binarios correspondientes
%a una mascara 3x3
a= Ifiltered(m-1,n-1);
b= Ifiltered(m-1,n);
c= Ifiltered(m-1,n+1);
d= Ifiltered(m,n-1);
p= Ifiltered(m,n);
e= Ifiltered(m,n+1);
f= Ifiltered(m+1,n-1);
g= Ifiltered(m+1,n);
h= Ifiltered(m+1,n+1);
%se aplica la ecuación booleana definida en 5.47
Ifiltered(m,n)=  p  &(  (  (a|b|d)&(e|g|h)  )  |  (
(b|c|e)&(d|f|g) ) );
end
end

return
```

Programa 5.7 Función que elimina los artefactos blancos considerados ruido, como parte de la operación del filtro binario.

La función llamada `rebuild_white_corners` reestablece las esquinas en los objetos blancos que poseen ángulos definidos. Esta función se muestra en el programa 5.8.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%Función para reconstruir las esquinas de los objetos
%blancos que tienen ángulos definidos, dentro de la
%operación del filtro binario
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 30/03/08
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function Ifiltered = rebuild_white_corners(Ibin)
Ifiltered = Ibin;
[height, width] = size(Ifiltered);
for m= 2: (height-1)
for n= 2:  (width-1)
a= Ifiltered(m-1,n-1);
b= Ifiltered(m-1,n);
c= Ifiltered(m-1,n+1);
d= Ifiltered(m,n-1);
p= Ifiltered(m,n);
e= Ifiltered(m,n+1);
f= Ifiltered(m+1,n-1);
g= Ifiltered(m+1,n);
h= Ifiltered(m+1,n+1);
%se aplica la ecuación lógica definida en 5.48
Ifiltered(m,n)=     ((not(p))     &     (d&f&g)     &
not(a|b|c|e|h))|p;
end
end
for m= 2: (height-1)
for n= 2:  (width-1)
a= Ifiltered(m-1,n-1);
b= Ifiltered(m-1,n);
c= Ifiltered(m-1,n+1);
d= Ifiltered(m,n-1);
p= Ifiltered(m,n);
e= Ifiltered(m,n+1);
f= Ifiltered(m+1,n-1);
g= Ifiltered(m+1,n);
h= Ifiltered(m+1,n+1);
%se aplica la ecuación lógica definida en 5.49
Ifiltered(m,n)=((not(p))&(a&b&d)&(not(c|e|f|g|h)))|p
;
end
end
for m= 2: (height-1)
for n= 2:  (width-1)
a= Ifiltered(m-1,n-1);
b= Ifiltered(m-1,n);
c= Ifiltered(m-1,n+1);
```

```
        d= Ifiltered(m,n-1);
        p= Ifiltered(m,n);
        e= Ifiltered(m,n+1);
        f= Ifiltered(m+1,n-1);
        g= Ifiltered(m+1,n);
        h= Ifiltered(m+1,n+1);
        %se aplica la ecuación lógica definida en 5.50
        Ifiltered(m,n)=((not(p))&(e&g&h)&(not(a|b|c|d|f)))|p
      ;
        end
        end
        for m= 2: (height-1)
        for n= 2:  (width-1)
        a= Ifiltered(m-1,n-1);
        b= Ifiltered(m-1,n);
        c= Ifiltered(m-1,n+1);
        d= Ifiltered(m,n-1);
        p= Ifiltered(m,n);
        e= Ifiltered(m,n+1);
        f= Ifiltered(m+1,n-1);
        g= Ifiltered(m+1,n);
        h= Ifiltered(m+1,n+1);
        %se aplica la ecuación lógica definida en 5.51
        Ifiltered(m,n)=((not(p))&(b&c&e)&(not(a|d|f|g|h)))|p
      ;
        end
        end
        return
```

Programa 5.8 Función que reconstruye las esquinas de los objetos blancos, como parte de la operación del filtro binario.

La función llamada `rebuild_black_corners` reestablece las esquinas en los objetos negros que poseen ángulos definidos. Esta función se muestra en el programa 5.9.

```
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %
        %
        %Función para reconstruir las esquinas de los objetos
        %negros que tienen ángulos definidos, dentro de la
        %operación del filtro binario
        %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %
        %
        % Erik Cuevas, Daniel Zaldivar, Marco Pérez
        % Versión: 30/03/08
        %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %
        function Ifiltered = rebuild_black_corners(Ibin)
        Ifiltered = Ibin;
```

```
[height, width] = size(Ifiltered);
for m= 2: (height-1)
for n= 2:  (width-1)
a= Ifiltered(m-1,n-1);
b= Ifiltered(m-1,n);
c= Ifiltered(m-1,n+1);
d= Ifiltered(m,n-1);
p= Ifiltered(m,n);
e= Ifiltered(m,n+1);
f= Ifiltered(m+1,n-1);
g= Ifiltered(m+1,n);
h= Ifiltered(m+1,n+1);
%se aplica la ecuación lógica definida en 5.52
Ifiltered(m,n)= p& ( (d|f|g) + not(a&b&c&e&h&p) );
end
end

for m= 2: (height-1)
for n= 2:  (width-1)
a= Ifiltered(m-1,n-1);
b= Ifiltered(m-1,n);
c= Ifiltered(m-1,n+1);
d= Ifiltered(m,n-1);
p= Ifiltered(m,n);
e= Ifiltered(m,n+1);
f= Ifiltered(m+1,n-1);
g= Ifiltered(m+1,n);
h= Ifiltered(m+1,n+1);
%se aplica la ecuación lógica definida en 5.53
Ifiltered(m,n)= p & ( (a|b|d) | not(c&e&f&g&h&p) );
end
end

for m= 2: (height-1)
for n= 2:  (width-1)
a= Ifiltered(m-1,n-1);
b= Ifiltered(m-1,n);
c= Ifiltered(m-1,n+1);
d= Ifiltered(m,n-1);
p= Ifiltered(m,n);
e= Ifiltered(m,n+1);
f= Ifiltered(m+1,n-1);
g= Ifiltered(m+1,n);
h= Ifiltered(m+1,n+1);
%se aplica la ecuación lógica definida en 5.54
Ifiltered(m,n)= p & ( (e|g|h) + not(a&b&c&d&f&p) );
end
end


for m= 2: (height-1)
for n= 2:  (width-1)
a= Ifiltered(m-1,n-1);
b= Ifiltered(m-1,n);
c= Ifiltered(m-1,n+1);
```

```
        d= Ifiltered(m,n-1);
        p= Ifiltered(m,n);
        e= Ifiltered(m,n+1);
        f= Ifiltered(m+1,n-1);
        g= Ifiltered(m+1,n);
        h= Ifiltered(m+1,n+1);
        %se aplica la ecuación lógica definida en 5.55
        Ifiltered(m,n)= p & ( (b|c|e) | not(a&d&f&g&h&p) ) ;
        end
        end

        return
```

Programa 5.9 Función que reconstruye las esquinas de los objetos negros, como parte de la operación del filtro binario.