

Universidad Nacional Autónoma de México  
Facultad de Ingeniería

# Procesamiento Digital de Imágenes

## Práctica 4 - Filtros Espaciales

Integrantes:

Cabrera Sánchez Manuel Salvador  
Torres Ruiz Mateo Alberto  
Velázquez Sánchez José Antonio

## **1. Objetivos:**

- Realizar operaciones de suavizado y de reducción de ruido en imágenes utilizando filtros espaciales de bloque y binomiales.
- Realizar operaciones de detección de bordes en imágenes, tanto limpias como ruidosas, utilizando filtros basados en aproximaciones de gradientes y laplacianos, así como derivadas de primer y segundo orden de funciones Gausianas (binomiales).
- Mejorar la nitidez de las imágenes sin ruido y con ruido usando los filtros unsharp masking.

## **2. Introducción**

Los filtros espaciales tienen como objetivo modificar la contribución de determinados rangos de frecuencias de una imagen. El término espacial se refiere a que el filtro se aplica directamente a la imagen y no a una transformada de la misma, es decir, el nivel de gris de un pixel se obtiene directamente en función del valor de sus vecinos. La convolución es la operación con la cual se hace filtrado espacial.

Los filtros espaciales pueden clasificarse basándose en su linealidad en filtros lineales y en filtros no lineales. A su vez los filtros lineales pueden ser clasificados según las frecuencias que dejen pasar: los filtros paso bajo atenúan o eliminan las componentes de alta frecuencia a la vez que dejan inalteradas las bajas frecuencias; los filtros paso alto atenúan o eliminan las componentes de baja frecuencia con lo que agudizan las componentes de alta frecuencia; los filtros paso banda eliminan regiones elegidas de frecuencias intermedias. A continuación se describe el uso de los diferentes filtros:

- Filtros paso bajas: son utilizados en la reducción de ruido; suavizan y apllanan un poco las imágenes y como consecuencia se reduce o se pierde la nitidez. En inglés son conocidos como Smoothing Spatial Filters.
- Filtros paso altas: estos filtros son utilizados para detectar cambios de luminosidad. Son utilizados en la detección de patrones como bordes o para resaltar detalles de una imagen. En inglés son conocidos como Sharpening Spatial Filters. Los filtros unsharp masking son filtros paso altas usados en el mejoramiento de la nitidez o de la calidad visual de una imagen.
- Filtros paso banda: son utilizados para detectar patrones de ruido. Ya que un filtro paso banda generalmente elimina demasiado contenido de una imagen casi no son usados, sin embargo, los filtros paso banda son útiles para aislar los efectos de ciertas bandas de frecuencias seleccionadas sobre una imagen. De esta manera, estos filtros ayudan a simplificar el análisis de ruido, razonablemente independiente del contenido de la imagen.

### **3. Desarrollo**

**1. Para todos los puntos siguientes, utilizar una imagen sin ruido y otra imagen con ruido. La imagen con ruido se puede generar a partir de la imagen sin ruido usando el siguiente comando de MATLAB:  $J=IMNOISE(I, TIPO...)$ , donde TIPO es una cadena que puede tomar valores ‘gaussian’, ‘localvar’, etc.**

Para esta práctica utilizamos la imagen de Lena (en blanco y negro, de 256x256) y generamos una imagen con ruido gausiano a partir de ella y el comando IMNOISE.

**2. Aplicar los filtros paso bajas de bloque a la imagen sin ruido y a la imagen con ruido usando filtros de orden 3x3, 5x5, 7x7 y 11x11.**

Utilizamos los filtros suavizadores de 3x3, 5x5, 7x7 y 11x11 generando matrices de unos a partir del comando ONES y aplicándolos a las imágenes con y sin ruido a partir del comando IMFILTER.

**3. Aplicar los filtros paso bajas binomiales a la imagen sin ruido y a la imagen con ruido usando filtros de orden 3x3, 5x5, 7x7 y 11x11.**

Utilizamos los filtros binomiales de 3x3, 5x5, 7x7 y 11x11 generando matrices a partir de la multiplicación de vectores de dichos tamaños con los coeficientes del triángulo de Pascal. Posteriormente normalizamos y aplicamos los filtros mediante el comando IMFILTER.

**4. Aplicar a la imagen sin ruido y con ruido los filtros basados en la primera derivada de gausiana o detectores de borde siguientes:**

- a) De bloque [1 -1]
- b) Prewitt en la dirección X y en la dirección Y
- c) Sobel en la dirección X y en la dirección Y
- d) Basados en la primera derivada de Gausiana de orden 5x5, 7x7 y 11x11

Para aplicar estos filtros basados en la primera derivada, definimos las matrices de cada filtro obtenida de los apuntes del curso, y utilizando los coeficientes del triángulo de Pascal modificado para el caso de los filtros de 5x5, 7x7 y 11x11. Posteriormente los aplicamos a cada imagen utilizando el comando IMFILTER.

**5. De igual manera, aplicar a la imagen sin ruido y a la imagen con ruido los filtros basados en la segunda derivada de gausiana siguientes:**

- a) Laplaciano
- b) Basados en la segunda derivada de Gausiana de orden 5x5, 7x7 y 11x11

Para desarrollar esta parte, también definimos primero la matriz correspondiente a cada uno de los filtros y posteriormente fueron aplicados a cada imagen mediante el comando IMFILTER.

**6. Difuminar las imágenes sin ruido y con ruido usando un filtro paso bajas de orden 5x5, de tal manera que se obtenga una imagen sin ruido y con pérdida de nitidez y otra imagen con ruido y pérdida de nitidez. Para cada uno de los siguientes incisos, filtrar las imágenes utilizando el filtro unsharp masking encontrado con los siguientes tipos de filtro paso bajas:**

- a) Filtro paso bajas de bloque de orden 3x3 y 7x7
- b) Filtro paso bajas binomial de orden 3x3 y 7x7

Para desarrollar esta parte, aplicamos un filtro suavizador de 5x5 a la imagen con y sin ruido, generando una matriz de unos a partir del comando ONES y aplicándolo mediante el comando IMFILTER.

## 4. Resultados



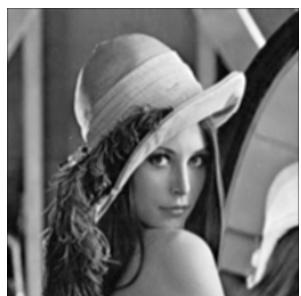
Imagen Original  
(256 x 256)



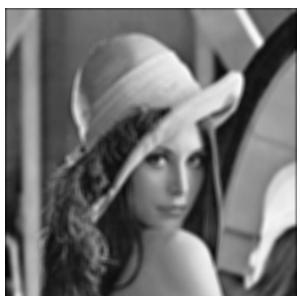
Imagen con ruido  
(256 x 256)

### Filtros Paso-bajas de bloque:

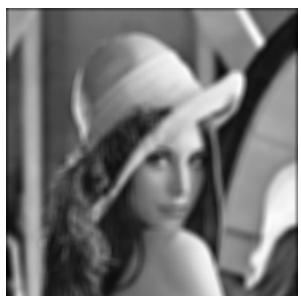
- Para la imagen sin ruido:



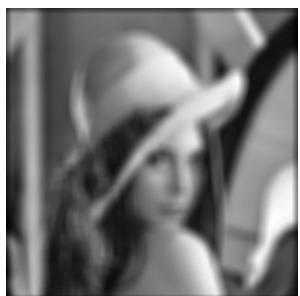
Filtro de 3x3



Filtro de 5x5



Filtro de 7x7



Filtro de 11x11

- Para la imagen con ruido:



Filtro de 3x3



Filtro de 5x5



Filtro de 7x7



Filtro de 11x11

## Filtros Paso-bajas binomiales:

- Para la imagen sin ruido:



Filtro de 3x3

Filtro de 5x5

Filtro de 7x7

Filtro de 11x11

- Para la imagen con ruido:



Filtro de 3x3

Filtro de 5x5

Filtro de 7x7

Filtro de 11x11

### **Filtros basados en la primera derivada:**

### **Filtro de bloque [1 -1]:**

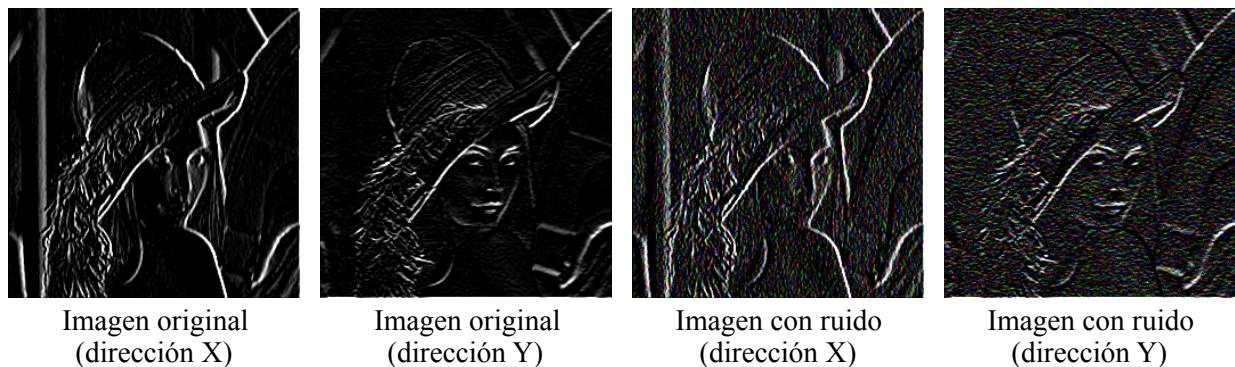


## Imagen Original

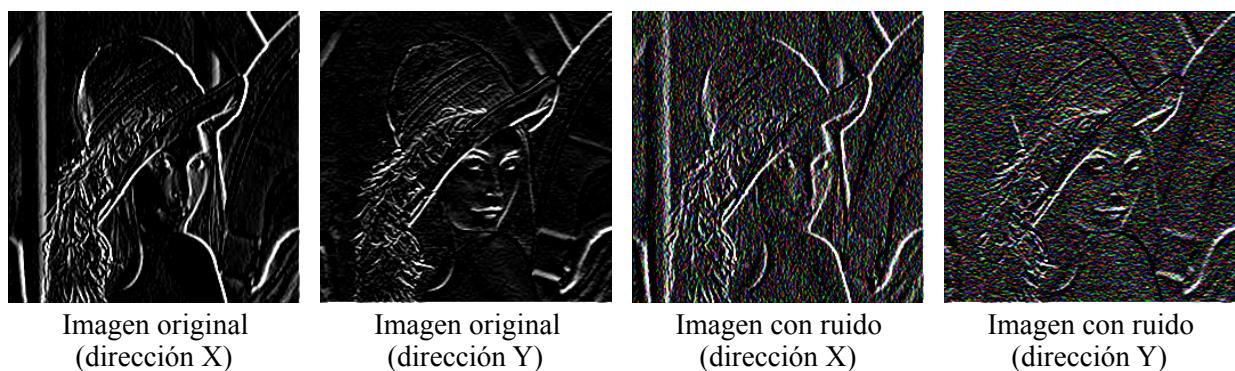


## Imagen con ruido

### Filtro Prewitt:



### Filtro Sobel:



### Filtros de primera derivada:

Para la imagen sin ruido:



Para la imagen con ruido:



Filtro de 5x5



Filtro de 7x7



Filtro de 11x11

Filtros basados en la segunda derivada:

**Filtro Laplaciano:**



Imagen Original



Imagen con ruido

**Filtros de segunda derivada:**

Para la imagen sin ruido:



Filtro de 5x5

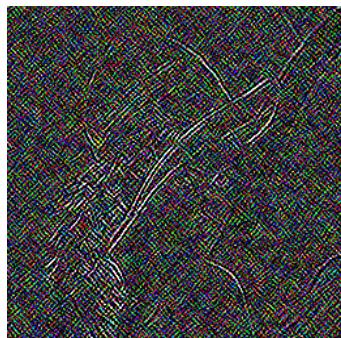


Filtro de 7x7

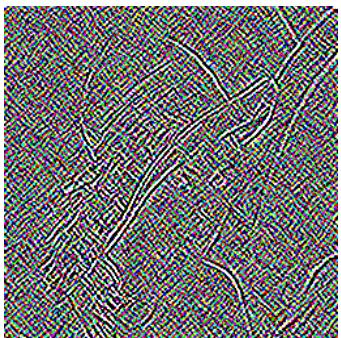


Filtro de 11x11

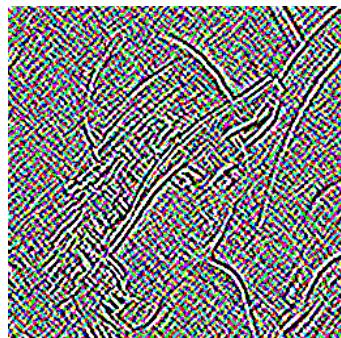
Para la imagen con ruido:



Filtro de 5x5



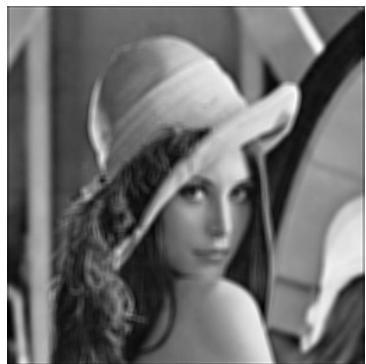
Filtro de 7x7



Filtro de 11x11

**Filtro unsharp masking de bloque:**

Para la imagen sin ruido:



Filtro de 3x3



Filtro de 7x7

Para la imagen con ruido:



Filtro de 3x3



Filtro de 7x7

## 5.- Código

### % Cargando imagen

```
clear  
img = imread('lena.jpg');  
imwrite(mat2gray(img,[0 255]),'lena.png');  
imgr = imnoise(img,'gaussian');  
imwrite(mat2gray(imgr),'lenar.png');
```

### %Filtros paso-bajas de bloque

```
pb3 = (1/9)*ones(3,3);  
imgpb3 = imfilter(img,pb3);  
imgrpb3 = imfilter(imgr,pb3);  
imwrite(mat2gray(imgpb3),'pb3.png');  
imwrite(mat2gray(imgrpb3),'rbp3.png');  
  
pb5 = (1/25)*ones(5,5);  
imgpb5 = imfilter(img,pb5);  
imgrpb5 = imfilter(imgr,pb5);  
imwrite(mat2gray(imgpb5),'pb5.png');  
imwrite(mat2gray(imgrpb5),'rbp5.png');
```

```
pb7 = (1/49)*ones(7,7);  
imgpb7 = imfilter(img,pb7);  
imgrpb7 = imfilter(imgr,pb7);  
imwrite(mat2gray(imgpb7),'pb7.png');  
imwrite(mat2gray(imgrpb7),'rbp7.png');
```

```
pb11 = (1/121)*ones(11,11);  
imgpb11 = imfilter(img,pb11);  
imgrpb11 = imfilter(imgr,pb11);  
imwrite(mat2gray(imgpb11),'pb11.png');  
imwrite(mat2gray(imgrpb11),'rbp11.png');
```

### %Filtros paso-bajas binomiales

```
mb3 = [1;2;1]*[1,2,1];  
fb3 = mb3/sum(mb3(:));  
imgfb3 = imfilter(img,fb3);  
imgrfb3 = imfilter(imgr,fb3);  
imwrite(mat2gray(imgfb3),'fb3.png');  
imwrite(mat2gray(imgrfb3),'rfb3.png');
```

```
mb5 = [1;4;6;4;1]*[1,4,6,4,1];  
fb5 = mb5/sum(mb5(:));  
imgfb5 = imfilter(img,fb5);  
imgrfb5 = imfilter(imgr,fb5);  
imwrite(mat2gray(imgfb5),'fb5.png');  
imwrite(mat2gray(imgrfb5),'rfb5.png');
```

```
mb7 = [1;6;15;20;15;6;1]*[1,6,15,20,15,6,1];  
fb7 = mb7/sum(mb7(:));  
imgfb7 = imfilter(img,fb7);  
imgrfb7 = imfilter(imgr,fb7);  
imwrite(mat2gray(imgfb7),'fb7.png');
```

```
imwrite(mat2gray(imgrfb7),'rfb7.png');  
  
mb11 =  
[1;10;45;120;210;252;210;120;45;10;1]*[1,10,45,  
120,210,252,210,120,45,10,1];  
fb11 = mb11/sum(mb11(:));  
imgfb11 = imfilter(img,fb11);  
imgrfb11 = imfilter(imgr,fb11);  
imwrite(mat2gray(imgfb11),'fb11.png');  
imwrite(mat2gray(imgrfb11),'rfb11.png');
```

### %Filtros detectores de bordes

```
pd3 = [1;0;-1]*[1,0,-1];  
imgpd3 = imfilter(img,pd3);  
imgrpd3 = imfilter(imgr,pd3);  
imwrite(mat2gray(imgpd3),'pd3.png');  
imwrite(mat2gray(imgrpd3),'rpd3.png');
```

```
prewittx = [1,0,-1;1,0,-1;1,0,-1];  
prewitty = [1,1,1;0,0,0;-1,-1,-1];  
imgpx = imfilter(img,prewittx);  
imgpx = imfilter(imgr,prewittx);  
imwrite(mat2gray(imgpx),'prex.png');  
imwrite(mat2gray(imgpx),'rprex.png');  
imgpy = imfilter(img,prewitty);  
imgpy = imfilter(imgr,prewitty);  
imwrite(mat2gray(imgpy),'prey.png');  
imwrite(mat2gray(imgpy),'rprey.png');
```

```
sobelx = [1,0,-1;2,0,-2;1,0,-1];  
sobely = [1,2,1;0,0,0;-1,-2,-1];  
imgsx = imfilter(img,sobelx);  
imgsx = imfilter(imgr,sobelx);  
imwrite(mat2gray(imgsx),'sobx.png');  
imwrite(mat2gray(imgsx),'rsobx.png');  
imgsy = imfilter(img,sobely);  
imgsy = imfilter(imgr,sobely);  
imwrite(mat2gray(imgsy),'soby.png');  
imwrite(mat2gray(imgsy),'rsoby.png');
```

```
pd5 = [1;2;0;-2;-1]*[1,2,0,-2,-1];  
imgpd5 = imfilter(img,pd5);  
imgrpd5 = imfilter(imgr,pd5);  
imwrite(mat2gray(imgpd5),'pd5.png');  
imwrite(mat2gray(imgrpd5),'rpd5.png');
```

```
pd7 = [1;4;5;0;-5;-4;-1]*[1,4,5,0,-5,-4,-1];  
imgpd7 = imfilter(img,pd7);  
imgrpd7 = imfilter(imgr,pd7);  
imwrite(mat2gray(imgpd7),'pd7.png');  
imwrite(mat2gray(imgrpd7),'rpd7.png');
```

```

pd11 =
[1;8;27;48;42;0;-42;-48;-27;-8;-1]*[1,8,27,48,42,0
,-42,-48,-27,-8,-1];
imgpd11 = imfilter(img,pd11);
imgrpd11 = imfilter(imgr,pd11);
imwrite(mat2gray(imgpd11),'pd11.png');
imwrite(mat2gray(imgrpd11),'rpd11.png');

%Filtros basados en la segunda derivada
laplaciano = [-1,-1,-1;-1,8,-1;-1,-1,-1];
imglap = imfilter(img,laplaciano);
imgrlap = imfilter(imgr,laplaciano);
imwrite(mat2gray(imglap),'lap.png');
imwrite(mat2gray(imgrlap),'rlap.png');

sd5 = [1;0;-2;0;1]*[1,0,-2,0,1];
imgsd5 = imfilter(img,sd5);
imgrsd5 = imfilter(imgr,sd5);
imwrite(mat2gray(imgsd5),'sd5.png');
imwrite(mat2gray(imgrsd5),'rsd5.png');

sd7 = [1;2;-1;-4;-1;2;1]*[1,2,-1,-4,-1,2,1];
imgsd7 = imfilter(img,sd7);
imgrsd7 = imfilter(imgr,sd7);
imwrite(mat2gray(imgsd7),'sd7.png');
imwrite(mat2gray(imgrsd7),'rsd7.png');

sd11 =
[1;6;13;8;-14;-28;-14;8;13;6;1]*[1,6,13,8,-14,-28,
-14,8,13,6,1];
imgsd11 = imfilter(img,sd11);

```

```

imgrsd11 = imfilter(imgr,sd11);
imwrite(mat2gray(imgsd11),'sd11.png');
imwrite(mat2gray(imgrsd11),'rsd11.png');

%Filtros unsharp masking
um3 = [(-1/9),(-1/9),(-1/9);(-1/9),(17/9),(-1/9);
(-1/9),(-1/9),(-1/9)];
imgum3 = imfilter(imgpb5,um3);
imgrum3 = imfilter(imgrpb5,um3);
imwrite(mat2gray(imgum3),'um3.png');
imwrite(mat2gray(imgrum3),'rum3.png');

um7 = [(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),
(-1/49),(-1/49);
(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),
(-1/49);
(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),
(-1/49);
(-1/49),(-1/49),(-1/49),(97/49),(-1/49),(-1/49),
(-1/49);
(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),
(-1/49);
(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),(-1/49),
(-1/49)];
imgum7 = imfilter(imgpb5,um7);
imgrum7 = imfilter(imgrpb5,um7);
imwrite(mat2gray(imgum7),'um7.png');
imwrite(mat2gray(imgrum7),'rum7.png');

```

## 6. Conclusiones

- Con el desarrollo de esta práctica pudimos observar claramente el comportamiento de los filtros de bloque sobre una imagen, que causa el efecto de “emborronamiento”. Así podemos comprender por qué estos filtros también reciben el nombre de “suavizadores”.
- Observamos claramente que, al aplicar los filtros suavizadores, el efecto de desenfoque o emborronamiento incrementa proporcionalmente cuando incrementamos el orden de la matriz del filtro. Es decir, un filtro suavizador de orden 11x11 causa un efecto de desenfoque mucho mayor al filtro de orden 3x3, hablando tanto de filtros de bloque como binomiales.
- Al aplicar los filtros basados en la primera derivada de Gaussiana, observamos cómo se comporta el detector de bordes y resalta estas partes de la imagen, dependiendo si es aplicado en la dirección X o en la dirección Y.
- Conocimos la importancia de algunos nuevos comandos en Matlab, tal como IMFILTER para aplicar filtros con matrices previamente definidas a una imagen.

## Referencias

- [1] Gonzalez, R. C. , and Woods, P., Digital Image Processing, Addison Wesley, 2002.
- [2] Universidad de Jaén, Área de ingeniería de sistemas y automática. Detección de bordes en una imagen. 2005.