

# *Eliminando ruido en imágenes*

## Métodos Numéricos

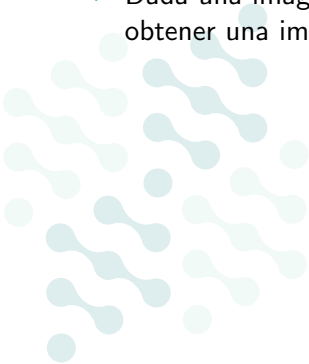
Departamento de Computación, FCEyN, Universidad de Buenos Aires.

8 de septiembre de 2017



# El problema

- Dada una imagen *ruidosa*, extraer el ruido de la misma para obtener una imagen similar a la original



# Oye, espacio cerebritito...

- ▶ ¿Qué es una imagen?
- ▶ ¿Qué significa que sea ruidosa?
- ▶ ¿Cómo puedo extraer ruido de una imagen si no conozco la original?
- ▶ ¿Qué significa que la imagen sea similar a la original?

# Oye, espacio cerebritito...

- ▶ ¿Qué es una imagen?
- ▶ ¿Qué significa que sea ruidosa?
- ▶ ¿Cómo puedo extraer ruido de una imagen si no conozco la original?
- ▶ ¿Qué significa que la imagen sea similar a la original?

# Oye, espacio cerebritito...

- ▶ ¿Qué es una imagen?
- ▶ ¿Qué significa que sea ruidosa?
- ▶ ¿Cómo puedo extraer ruido de una imagen si no conozco la original?
- ▶ ¿Qué significa que la imagen sea similar a la original?

# Oye, espacio cerebritito...

- ▶ ¿Qué es una imagen?
- ▶ ¿Qué significa que sea ruidosa?
- ▶ ¿Cómo puedo extraer ruido de una imagen si no conozco la original?
- ▶ ¿Qué significa que la imagen sea similar a la original?

# ¿Qué es una imagen?

- ▶ Para nosotros es simplemente una matriz de píxeles con valores de una escala de grises
- ▶ La leemos en MATLAB con *imread*
- ▶ La procesamos como cualquier matriz
- ▶ La mostramos con funciones como *imshow*, *image*, *imagesc*
- ▶ Es decir que cualquier matriz podemos verla como una imagen. (Ejemplo: *imagesc(rand(16))*, arte, arte, arte ...)



# ¿Qué es una imagen?

- ▶ Para nosotros es simplemente una matriz de píxeles con valores de una escala de grises
- ▶ La leemos en MATLAB con *imread*
- ▶ La procesamos como cualquier matriz
- ▶ La mostramos con funciones como *imshow*, *image*, *imagesc*
- ▶ Es decir que cualquier matriz podemos verla como una imagen. (Ejemplo: *imagesc(rand(16))*, arte, arte, arte ...)





# ¿Qué es una imagen?

- ▶ Para nosotros es simplemente una matriz de píxeles con valores de una escala de grises
- ▶ La leemos en MATLAB con *imread*
- ▶ La procesamos como cualquier matriz
- ▶ La mostramos con funciones como *imshow*, *image*, *imagesc*
- ▶ Es decir que cualquier matriz podemos verla como una imagen. (Ejemplo: *imagesc(rand(16))*, arte, arte, arte ...)



# ¿Qué es una imagen?

- ▶ Para nosotros es simplemente una matriz de píxeles con valores de una escala de grises
- ▶ La leemos en MATLAB con *imread*
- ▶ La procesamos como cualquier matriz
- ▶ La mostramos con funciones como *imshow*, *image*, *imagesc*
- ▶ Es decir que cualquier matriz podemos verla como una imagen. (Ejemplo: *imagesc(rand(16))*, arte, arte, arte ...)



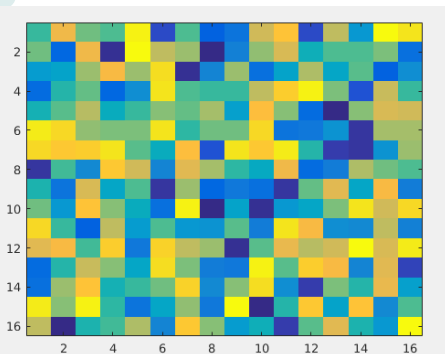
## ¿Qué es una imagen?

- ▶ Para nosotros es simplemente una matriz de píxeles con valores de una escala de grises
- ▶ La leemos en MATLAB con *imread*
- ▶ La procesamos como cualquier matriz
- ▶ La mostramos con funciones como *imshow*, *image*, *imagesc*
- ▶ Es decir que cualquier matriz podemos verla como una imagen. (Ejemplo: *imagesc(rand(16))*, arte, arte, arte ...)



# ¿Qué es una imagen?

- ▶ Para nosotros es simplemente una matriz de píxeles con valores de una escala de grises
- ▶ La leemos en MATLAB con *imread*
- ▶ La procesamos como cualquier matriz
- ▶ La mostramos con funciones como *imshow*, *image*, *imagesc*
- ▶ Es decir que cualquier matriz podemos verla como una imagen. (Ejemplo: *imagesc(rand(16))*), arte, arte, arte ...)



# ¿Qué es el ruido?

- Para nosotros quiere decir que los valores de una imagen difieren del valor real

- Existen varios tipos de ruidos (aditivo, multiplicativo, puntual)
- En MATLAB los simulamos *a mano* o con la funcion *imnoise*

# ¿Qué es el ruido?

- Para nosotros quiere decir que los valores de una imagen difieren del valor real



- Existen varios tipos de ruidos (aditivo, multiplicativo, puntual)
  - En MATLAB los simulamos *a mano* o con la funcion *imnoise*

# ¿Qué es el ruido?

- Para nosotros quiere decir que los valores de una imagen difieren del valor real



- Existen varios tipos de ruidos (aditivo, multiplicativo, puntual)
- En MATLAB los simulamos *a mano* o con la funcion *imnoise*

# Ejemplos

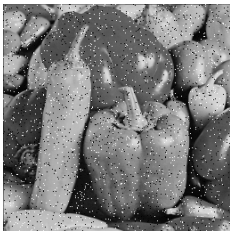


Figura: Salt & Pepper

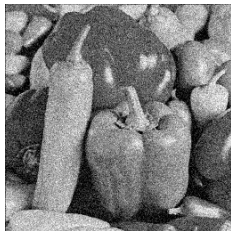


Figura: Gaussiano

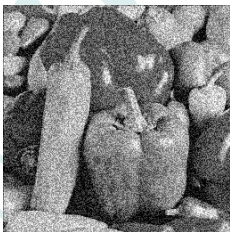


Figura: Uniforme

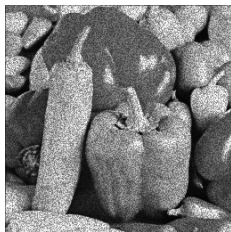


Figura: Speckle



# Diferencias

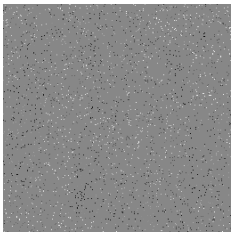


Figura: Salt & Pepper

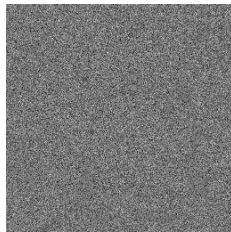


Figura: Gaussiano

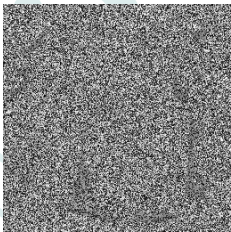


Figura: Uniforme

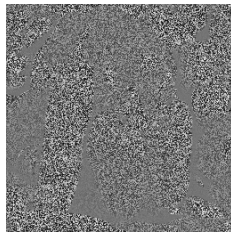


Figura: Speckle

# Shhh, extrayendo el ruido...

¿Cómo hacemos para sacar el ruido de una imagen?

- ▶ Existen muchos métodos
- ▶ Varios de ellos dependen de conocer la distribución de ruido particular de la imagen a procesar
- ▶ Filtro pasabajos
- ▶ Filtros basados en DFT y DCT
- ▶ Filtro de mediana

# Shhh, extrayendo el ruido...

¿Cómo hacemos para sacar el ruido de una imagen?

- ▶ Existen muchos métodos
- ▶ Varios de ellos dependen de conocer la distribución de ruido particular de la imagen a procesar
- ▶ Filtro pasa-bajos
- ▶ Filtros basados en DFT y DCT
- ▶ Filtro de mediana

# Shhh, extrayendo el ruido...

¿Cómo hacemos para sacar el ruido de una imagen?

- ▶ Existen muchos métodos
- ▶ Varios de ellos dependen de conocer la distribución de ruido particular de la imagen a procesar
- ▶ Filtro pasabajos
  - ▶ Filtros basados en DFT y DCT
  - ▶ Filtro de mediana

# Shhh, extrayendo el ruido...

¿Cómo hacemos para sacar el ruido de una imagen?

- ▶ Existen muchos métodos
- ▶ Varios de ellos dependen de conocer la distribución de ruido particular de la imagen a procesar
- ▶ Filtro pasabajos
- ▶ Filtros basados en DFT y DCT
- ▶ Filtro de mediana

# Shhh, extrayendo el ruido...

¿Cómo hacemos para sacar el ruido de una imagen?

- ▶ Existen muchos métodos
- ▶ Varios de ellos dependen de conocer la distribución de ruido particular de la imagen a procesar
- ▶ Filtro pasabajos
- ▶ Filtros basados en DFT y DCT
- ▶ Filtro de mediana

# Nuestro método

Se puede pensar el problema de filtrar una imagen con ruido como la minimización del siguiente funcional:


$$\Pi = \int_{\Omega} \frac{\lambda}{2} |u - \tilde{u}|^2 + \frac{1}{2} \|\nabla u\|^2 d\Omega, \quad (1)$$

donde  $u : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$  describe la imagen filtrada y  $\tilde{u} : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$  la imagen a filtrar (con ruido).

De esta manera, el primer término *pesa* cuánto ruido tiene  $\tilde{u}$  y el segundo *pesa* la suavidad de la imagen obtenida. La constante  $\lambda$  controla la importancia relativa de los dos términos.

## Minimizando...

La minimización del funcional de la ecuación (1) da lugar a la siguiente ecuación diferencial:


$$\lambda(u - \tilde{u}) - \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = 0. \quad (2)$$



## Discretizando...

La solución de la ecuación (2) que representa la imagen filtrada se puede aproximar de manera discreta utilizando el método de diferencias finitas, lo cual conduce al siguiente sistema de ecuaciones:

$$\lambda u_{i,j} - (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}) = \lambda \tilde{u}_{i,j} \quad (3)$$

donde ahora  $u, \tilde{u} : \Omega \subset \mathbb{Z}^2 \rightarrow [0 \dots 255]$  son las versiones discretas de la imagen filtrada y la imagen original, respectivamente.

Viendo la imagen  $u$  como una matriz,  $i, j$  son los índices de fila y columna de cada elemento (píxel) de la matriz y donde el valor 0 representa al color negro y el 255 al blanco.

# Codeando...

```
% Armado del sistema
Lambda = 1;
Dim = size(IR);
NInc = prod(Dim);

% Armado del vector resultado
Utilde = Lambda*double(IR(:));

% Armado de la matriz a resolver
B = -1*ones(NInc,5);
B(:,3) = (Lambda+4)*ones(NInc,1);
d = [-Dim(1) -1 0 1 Dim(1)];
A = spdiags(B,d,NInc,NInc);
```

- ▶  $\lambda u_{i,j} - (u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}) = \lambda \tilde{u}_{i,j}$
- ▶ IR es la imagen con ruido
- ▶ `spdiags(B, d, m, n)` crea una matriz esparsa de  $m \times n$  colocando las columnas de  $B$  como diagonales de acuerdo a las posiciones indicadas en  $d$  como relativas a la diagonal principal

## ¿Quedó mejor?

Una vez que filtramos la imagen, ¿cómo medimos qué tan similar quedó a la original?

El PSNR se define como:

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX_u^2}{ECM} \right)$$

donde  $MAX_u$  define el rango máximo de la imagen (para nuestro caso sería 255) y ECM es el *error cuadrático medio*, definido como:

$$\frac{1}{N} \sum_{i,j} (u_{i,j}^0 - u_{i,j})^2$$

donde  $N$  es la cantidad de píxeles de la imagen,  $u^0$  es la imagen original y  $u$  es la imagen perturbada (o en nuestro caso, la imagen recuperada).

## ¿Quedó mejor?

Una vez que filtramos la imagen, ¿cómo medimos qué tan similar quedó a la original?

El PSNR se define como:

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX_u^2}{ECM} \right)$$

donde  $MAX_u$  define el rango máximo de la imagen (para nuestro caso sería 255) y ECM es el *error cuadrático medio*, definido como:

$$\frac{1}{N} \sum_{i,j} (u_{i,j}^0 - u_{i,j})^2$$

donde  $N$  es la cantidad de píxeles de la imagen,  $u^0$  es la imagen original y  $u$  es la imagen perturbada (o en nuestro caso, la imagen recuperada).

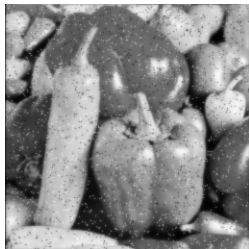
# Ejemplos



(a) Original



(b) Salt & Pepper  
PSNR: 18.3201  
ECM: 957.567

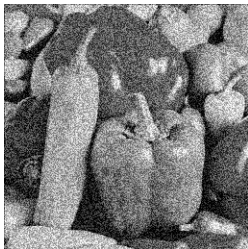


(c) Denoised  $\lambda = 1$   
PSNR: 22.8122  
ECM: 341.34

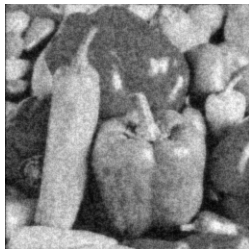
# Ejemplos



(a) Original



(b) Uniforme  
PSNR: 16.8928  
ECM: 1329.86

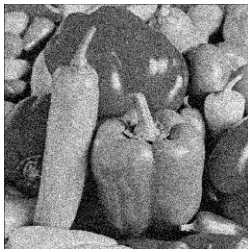


(c) Denoised  $\lambda = 1$   
PSNR: 23.6228  
ECM: 283.32

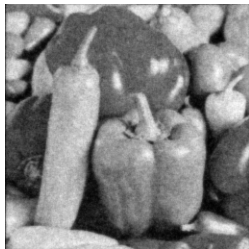
# Ejemplos



(a) Original



(b) Gaussian  
PSNR: 20.1515  
ECM: 627.972

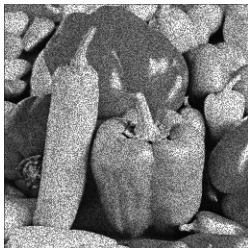


(c) Denoised  $\lambda = 1$   
PSNR: 24.2042  
ECM: 248.407

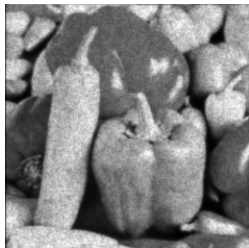
# Ejemplos



(a) Original



(b) Speckle  
PSNR: 18.8881  
ECM: 839.999



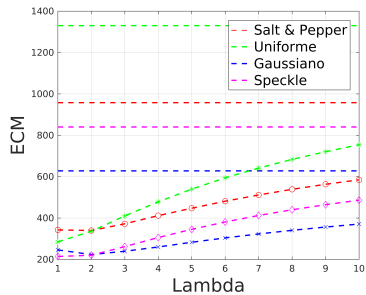
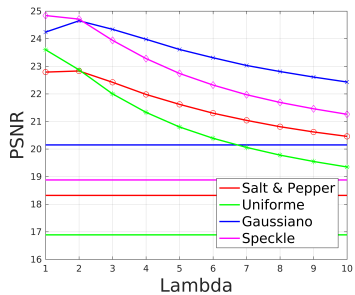
(c) Denoised  $\lambda = 1$   
PSNR: 24.8398  
ECM: 214.28



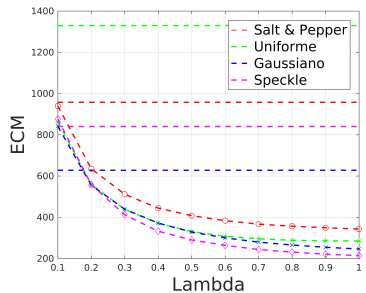
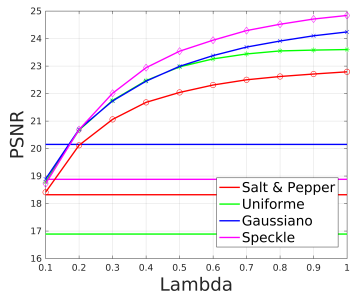
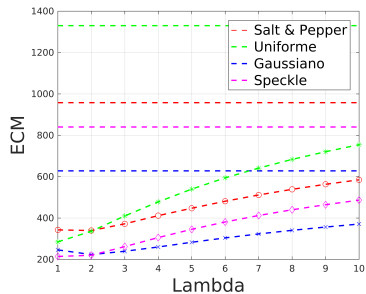
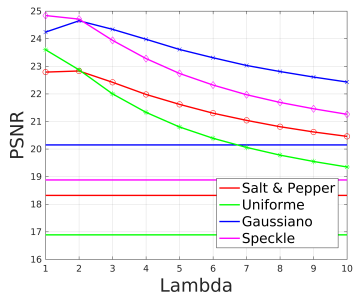
# Experimentos

	Salt & Pepper		Uniforme		Gaussiano		Speckle	
	PSNR	ECM	PSNR	ECM	PSNR	ECM	PSNR	ECM
$\lambda$	18.32	957.56	16.89	1329.86	20.15	627.97	18.88	839.99
1	22.79	342.65	23.60	284.75	24.24	245.98	24.84	214.21
2	22.83	339.45	22.87	335.99	24.65	223.34	24.71	220.08
3	22.42	372.09	22.00	410.15	24.34	239.36	23.94	261.97
4	21.98	411.53	21.33	478.06	23.98	260.00	23.28	305.48
5	21.62	447.31	20.80	539.77	23.61	282.81	22.74	345.38
6	21.30	481.21	20.39	594.10	23.31	303.92	22.32	380.99
7	21.04	511.41	20.06	641.26	23.03	323.12	21.97	412.23
8	20.81	538.75	19.78	683.45	22.81	340.61	21.69	439.89
9	20.62	563.07	19.55	720.33	22.61	356.36	21.46	464.38
10	20.46	584.71	19.35	754.36	22.43	370.87	21.26	486.47

# Experimentos



# Experimentos



# Pasemos al taller

