PROYECTO 1

# Proyecto: desenfoque (y enfoque) de imagen

Este proyecto incluye 3 posibles ejercicios al final. Se debe elegir **uno solo** de ellos para el trabajo.

Matlab permite visualizar imágenes con el comando `image`. Este recibe una matriz de números y la muestra como una imagen. Los diferentes valores en cada entrada de la matriz representan colores. En este ejercicio se tratrán imágenes en escala de grises, para simplficar. Para ello, lo primero que se ha de hacer es indicarlo:

```
> colormap(gray(256));
```

A partir de ahora, las matrices se representarán como imágenes en una escala de 256 niveles de gris. El 0 es negro y el 255 es blanco. Por ejemplo:

```
> A = [0 2 4 8; 16 32 64 128; 255 132 58 190];
> image(A);
```

debería mostrar una imagen de 12 bloques en distintos niveles de gris. Según la pantalla, los negros de la primera fila se distinguirán más o menos.

En el archivo `foto.dat` se encuentra una imagen de un famoso físico del Siglo XX. Para cargarla:

```
> load('foto.dat');
> image(foto);
```

debería aparecer en la pantalla una imagen en escala de grises.

El proceso de *desenfoque gaussiano* (que se puede leer en la Wikipedia) es una técnica que permite eliminar bordes o ruido mediante un desenfoque en el que cada punto pierde algo de su valor y se lo transmite a los adyacentes. La técnica general utiliza más puntos que los meramente adyacentes pero en este trabajo nos limitaremos a estos.

## 1. Preparar la imagen como un vector

Para operar con la imagen en Matlab, es preciso transformarla en un vector. Las dimensiones de `foto` (la imagen en cuestión) pueden conocerse:

```
> size(foto)
```

da el número de filas y columnas de `foto`. Se ha de convertir `foto` en un vector de su número de filas por su número de columnas. Para esto se utiliza el comando `reshape`. Llamemos `vfoto` al vector obtenido.

**Nota:** El comando `reshape` se debe utilizar continuamente en este ejercicio:

- Cuando una matriz `A` (digamos de tamaño $300 \times 200$) quiere transformarse en un vector fila `v` (de 60000 componentes), debe hacerse

```
> v = reshape(A, [1, 300*200]); % que Matlab haga las cuentas
```

- Cuando un vector fila `v` quiere transformarse en una matriz `A` (para ver la imagen), debe hacerse al revés:

```
> A = reshape(v, [300,200]);
> image(A);
```

## 2. El desenfoque

La parte de alrededor de un punto `(i,j)` de la imagen puede considerarse como en la figura 1.

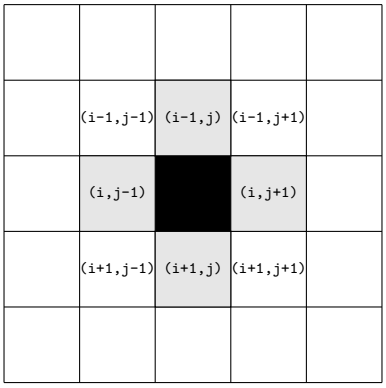


Figura 1. Puntos alrededor del `(i,j)` (en negro).

Para realizar el desenfoque, se transforma el vector `vfoto` de manera que, para cada punto `(i,j)` de la imagen (que corresponderá a una componente de `vfoto`) por ejemplo: el 90% del color se queda en el propio punto `(i,j)`, mientras que cada punto adyacente (de la cruz en gris) recibe el 2% del color de `(i,j)` y cada uno de los puntos de las cuatro esquinas cercanas recibe solo el 0.5% de tal color. Así, un punto completamente negro se transformaría (de manera exagerada) en algo similar a lo que se muestra en la figura 2.

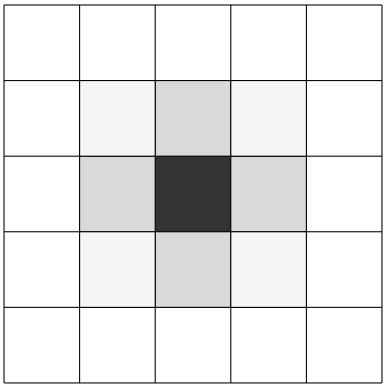

Figura 2. En qué se transforma un punto negro: él pierde intensidad, en la cruz adyacente se gana algo y en las esquinas adyacentes de gana menos. La suma total es 1 (ó 100%, que es lo mismo).

**Nota:** Obsérvese que $90 + 4 \times 2 + 4 \times 0.5 = 100$. ¿Por qué?

Cuando esto se realiza en todos los puntos del vector `vfoto`, la transformación puede describirse mediante una matriz que llamaremos `D` (por "desenfoque").

Para definir matrices tan grandes como las que se van a utilizar (de más o menos $100.000 \times 100.000$ elementos) es preciso usar lo que se conocen como *matrices dispersas*. Para generar una matriz dispersa, se escribe el comando `sparse` antes de la definición de la matriz:

```
> A = 0.9 * sparse(eye(8000,8000)); % matriz 0.9*identidad 8.000 x 8.000
> A = A + diag(sparse(0.2*ones(1,7999)),-1); % 0.2 en la diag -1
> A = A + diag(sparse(0.2*ones(1,7999)),1); % 0.2 en la diag 1
```

## 3. Desenfocar (y enfocar)

Una vez que se ha construido la matriz de desenfoque `D`, puede comprobarse cómo actúa aplicándola sobre el vector de la imagen y volviendo a representar esta:

```
> v1 = D*vfoto';
> v1img = reshape(v1, ...);
> image(v1img);
```

donde el comando `reshape` se ha dejado con unos puntos suspensivos para que el alumno descubra cómo ha de utilizarse.

Está claro que el desenfoque puede iterarse para conseguir una imagen cada vez más desenfocada.

**Ejercicio 1:** ¿Qué forma tiene la matriz `D`? ¿Cómo se puede construir de una manera eficiente? ¿Qué cuidado especial hay que tener para los puntos de la imagen que corresponden a los lados y a las esquinas? Esta es la parte más delicada. Desenfóquese la imagen de prueba con diferentes iteraciones utilizando la matriz `D`. ¿Cómo se calcularía el desenfoque producido al repetir 300 veces la operación correspondiente a `D`? Muéstrese el efecto de desenfocar para diferentes iteraciones. —

**Ejercicio 2:** Si se utilizan los valores que se pusieron arriba como ejemplo, ¿se desenfoca poco ó mucho? ¿por qué? ¿hay algunos valores más razonables que otros? ¿Cómo se calcularía una iteración de 100 desenfoques? ¿Puede servir este método para enfocar una foto desenfocada? ¿Cómo? —

**Ejercicio 3:** ¿Es razonable construir una matriz tan grande? Téngase en cuenta que para una imagen de $300 \times 400$ píxeles en solo 256 grises, la imagen debería tener $(300 \times 400)^2 = 14.400.000.000$ bytes (14 gigabytes), y esa imagen es, hoy día, muy pequeña. ¿Qué método alternativo podría usarse? —

PROYECTO 2

# Espectro de sonido

Para realizar este proyecto se requieren los archivos `.m` siguientes:

- `spectrum.m`, que realiza la operación de pasar de una onda a su espectro.
- `ft_frame.m`, que calcula los coeficientes de Fourier de una trama de una onda (este se utiliza en el anterior).

Ambos han de grabarse en `Documentos/MATLAB`.

Además, se necesitan dos archivos de datos (que se guardarán en el mismo directorio):

- `wood-thrush.mat`, que contiene los datos de una onda (en estéreo) del trino un zorzal del bosque.
- `PianoPhrase.mat`, que contiene los datos de una onda (en estéreo) de una figura de piano.

El proyecto consiste en explicar *qué hacen* los archivos `.m` de arriba teniendo en cuenta lo que sigue.

## 1. Archivos de audio

Matlab utiliza para almacenar datos de audio una matriz de dos filas y tantas columnas como datos de entrada haya. Además, una onda de audio se muestrea a un número de veces por segundo. Con el siguiente comando

```
> load('wood-thrush.mat');
```

se carga una variable llamada `wavedata` que contiene la onda de sonido y otra llamada `samplerate` que indica a qué frecuencia se ha muestreado tal onda

```
> samplerate
samplerate =  22050
```

(a 22050 herzios).

La variable `wavedata` contiene dos canales (izquierdo y derecho), aunque solo se va a utilizar uno

```
> size(wavedata)
ans =
     2    124992
```

que indica que la onda es en estéreo y que hay 124992 datos (por tanto, unos 5.6 segundos de audio).

```
> sound(wavedata(1,:), 22050);
```

reproduce por el altavoz el sonido grabado en el primer canal de la onda.

Si en lugar de `wood-thrush.mat` se utiliza `PianoPhrase.mat`, la mecánica es la misma pero con diferentes datos y nombres de archivos.

## 2. El espectro

Considérese el listado del programa `spectrum.m` y, ejecútese, para el zorzal:

```
> load('wood-thrush.mat');
> plot(wavedata(1,:));
> sp_zorzal = 20*spectrum('wood-thrush.mat', 20, .1);
> % el 20 es para que se distinga bien todo
> image(sp_zorzal);
```

y para el piano (el sonido del piano está grabado a 44100Hz, el doble que el del zorzal; si se quiere escuchar bien, hay que utilizar el comando `sound` con ese dato):

```
> load('PianoPhrase.mat');
> plot(wavedata(1,:));
> sp_piano = 20*spectrum('PianoPhrase.mat', 20, .1);
> image(sp_piano(1:200,:)); % hay muy pocas bandas con sonido
```

**Ejercicio 1:** Explicar los siguientes aspectos del proyecto:

- Compárense las gráficas (los `plot(waveform)`) que corresponden a `wood-thrush.mat` y `PianoPhrase.mat` y explicar las semejanzas y diferencias.
- Explicar qué hace el programa `spectrum` y qué imágenes son las que aparecen al hacer los `image(...)` arriba.
- ¿Qué significan el `20` y el `.1` de los parámetros de `spectrum`?
- ¿Por qué el último comando `image` del piano es diferente del del zorzal? ¿Qué significa el dato que se usa? Si se utiliza un parámetro análogo para el zorzal, ¿se ve algo más claro?

—

PROYECTO 3

## Ley de los gases ideales y no ideales

La ley de los gases ideales está dada por

$$PV = nRT$$

donde $P$ es la presión absoluta, $V$ el volumen, $n$ el número de moles, $R$ la constante universal de los gases y $T$ la temperatura absoluta. Pero el comportamiento de un gas real es mucho más complejo.

A lo largo del tiempo se han propuesto distintas ecuaciones de estado para los gases reales. La más sencilla es la de van der Waals:

$$\left(P + \frac{n^2 a}{V^2}\right)(V - nb) = nRT.$$

Pero es más común utilizar la siguiente forma alternativa en variables reducidas:

$$\left(P_R + \frac{3}{V_R^2}\right)(3V_R - 1) = 8T_R$$

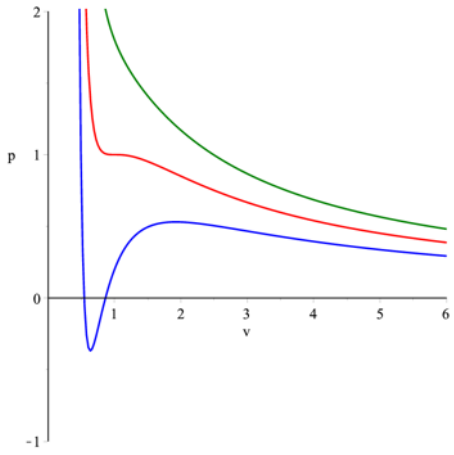


Figura 1. Isotermas de la ecuación de van der Waals (L. Bayón).

que, despejando, da

$$V_R^3 - \frac{1}{3}\left(1 + \frac{8T_R}{P_R}\right)V_R^2 + \frac{3}{P_R}V_R - \frac{1}{P_R} = 0.$$

Las isotermas en un diagrama $P - V$ se pueden representar como en la figura 1.

Para una temperatura fija (cada una de las gráficas) puede ocurrir que para cada valor de la presión haya 3 ó solo 1 valor de volumen correspondiente.

**Ejercicio 1:** Fijando diferentes valores de $T_R$ y $P_R$, resolver la ecuación de van der Waals en varios casos. Se han de encontrar casos de los tres tipos indicados arriba. Aplicando los métodos de solución de ecuaciones no lineales, analizar las dificultades encontradas. Explicar qué ocurre. Ejemplos:

- Estudiar cómo estimar los parámetros iniciales para aplicar el algoritmo de bisección.
- ¿Cómo se puede estimar el valor inicial para aplicar el algoritmo de Newton-Raphson?
- ¿Qué función de iteración se ha de elegir para aplicar el método del punto fijo?

—