

Procesamiento de imágenes con Matlab



Dr. Javier González Jiménez

Dpto. de Ingeniería
de Sistemas y Automática

¿Que es Matlab?

- Lenguaje de **alto nivel** para cálculo matricial, análisis numérico y computación científica
- **Características:**
 - No hay **declaracion de variables**
 - Gestión de **memoria automática** (aunque ayuda la reserva de la misma “allocations”)
 - Lenguaje “**vectorizado**”: Se puede utilizar bucles `for`, pero se puede y debe EVITAR (menos eficiente)
 - Puede llamar a **programas externos** hechos a medida (dll)
- Una **ayuda** fantástica
 - Con “`help`” se tiene un listado de los temas
 - Con “`help <tema>`” ayuda sobre el tema

Introducción de variables

- Son siempre **matrices**
 - $V = [10, 4.5, 1];$
 - $M = [3, 4; -6, 5];$
- Sin el **punto y coma** al final se vuelca a pantalla la variable (mal asunto para imagenes!)
- La **coma** separa elementos de la misma fila. El **punto y coma** separa filas
- **size**: devuelve el numero de fila y columnas

Algunas matrices típicas:

- Todo ceros o unos: `zeros`, `ones`
- Identidad: `eye`
- Aleatoria: `rand` (uniforme), `randn` (normal)

Crear matrices grandes a partir de una mas pequeña:

`repmat (A, m, n)`: Monta una matriz grande con $m \times n$ copias de A

Indexado (origen: arriba a la izquierda, fila-columna): $A(1, 1)$

Rango de elementos: $m:n$ o $m:i:n$ (i es el incremento)

$A(1:10, 3) \rightarrow$ primeros 10 elementos (filas) de la columna 3

$A(:, 1) \rightarrow$ todos los elementos (filas) de la columna 1

Búsqueda de elementos en una matriz:

find: Índices de elementos no-cero

e.g., $I = \text{find}(A > 100)$: devuelve índices de elementos de A
mayores que 100

Cálculo y manipulación simples:

- Transpuesta (`'`), inversa (`inv`)
- Aritmética de matrices: `+`, `-`, `*`, `/`, `^`
- Aritmética de elementos de matrices : `.*`, `./`, `.^`
- Funciones: `sin`, `cos`, etc.

Cálculo y manipulación mas compleja:

- En sistemas lineales: `A\b` solves $A \cdot x = b$
- descomposicion QR : `qr`
- Descomposicion de valor singular: `svd`
- Autovalores (Eigenvalues): `eig`

Estructuras de control:

- `if/while expresion booleana end`

relaciones: `==`, `>`, `|`, `&`

Ejemplo: (usar coma para separar sentencias en la misma linea)

```
if a==b & isprime(n), M = inv(K); else M = K; end
```

- `for variable = expression statements end`

Ejemplo: `for i=1:2:100, s = s / 10; end`

M-Files

- Es un fichero de **código** Matlab
- Es un fichero de texto con **extensión “.m”**
- Usar **path** o **addpath** para decir a Matlab donde esta el código
- Dos posibilidades: script y función
 - **Script:**
 - coleccion de comandos
 - no se le pasan parámetros
 - variables globales
 - **Funcion:**
 - Tiene argumentos de entrada y devuelve valores (con return).
 - La primera líenea del fichero
 - `function y = foo(A)`
 - `function [x, y] = foo2(a, M, N)`
- **Comentarios:** comienzan `%`

Dibujos

- Vectores (2-D): `plot(x, y)`

Ejemplo: `plot(0:0.01:2*pi, sin(0:0.01:2*pi))`

- Matrices (3-D): `plot3(x, y, z)` (curva en el espacio)

Ejemplo: `t=0:pi/50:10*pi; plot3(sin(t), cos(t), t);`

- Superficies

`Meshgrid`: construye una reticula

`mesh`: dibuja una malla en la reticula definida en `meshgrid`

Ejemplo: `[X,Y] = meshgrid(-2:.2:2, -2:.2:2);`
`Z = X .* exp(-X.^2 - Y.^2);`
`mesh(Z);`

`surf`: version “solida” de `mesh`

Toolbox de Procesamiento de Imágenes

- **Ayuda:** `help images`

<code>im = imread (ruta)</code>	Lee un archivo imagen desde la 'ruta' que se le pasa como parámetro y lo devuelve en 'im'.
<code>imshow (imagen)</code>	Muestra la 'imagen' que se pasa por parámetro en una ventana.
<code>his = imhist (imagen)</code>	Muestra el histograma de 'imagen' y lo devuelve en 'his'.

<code>imwrite (imagen, ruta, tipo)</code>	Almacena una imagen en la 'ruta' (incluye directorio y el nombre del archivo) con formato especificado por 'tipo' que se les pasa como parámetros.
---	--

<code>im = rgb2gray (imagen)</code>	Convierte la 'imagen' de entrada en formato RGB a niveles de grises.
<code>im = imnoise (imagen, tipo_ruido)</code>	Añade a la 'imagen' ruido especificado por el parámetro 'tipo_ruido'.
<code>mask = fspecial (tipo_mask)</code>	Crea una mascara del 'tipo_mask' de 3x3 (5x5 para 'Log').
<code>mask = conv2(imagen, operador, tipo)</code>	Realiza la operación de convolución entre 'imagen' y 'operador', teniendo en cuenta el borde de la imagen o no dependiendo del parámetro 'tipo'.
<code>im = medfilt2(imagen)</code>	Realiza el suavizado de la mediana a la 'imagen'

Toolbox de Procesamiento de Imágenes (cont.)

<code>im = im2bw (imagen, umbral)</code>	Binariza la 'imagen' con 'umbral' y lo devuelve en 'im'.
<code>im = histeq (imagen, h_esp)</code>	Modifica la 'imagen' , al histograma especificado por 'h_esp' o lo iguala si se omite este parámetro y devuelve la nueva imagen en 'im'.

<code>im = edge(imagen, tipo, umbral)</code>	Aplica el detector de bordes especificado por 'tipo' a la 'imagen' con criterio de selección especificado por 'umbral' o el de defecto si se omite.
--	---

<code>puntos= ginput(n_puntos)</code>	Selecciona 'n_puntos' de la imagen activa y almacena sus coordenadas en el vector puntos.
<code>Perfil = improfile</code>	Permite seleccionar de manera interactiva un perfil de intensidades de la imagen activa.

<code>colormap(paleta)</code>	Cambia la paleta activa a la indicada por el parámetro 'paleta'.
<code>brighten(factor)</code>	Incrementa la intensidad de los píxeles de la imagen activa en una proporción dada por 'factor', entre [0..1]
<code>imagesc(im)</code>	Escala los valores la imagen para que ocupen la paleta actual por completo.

Procesamiento de Imágenes. Practica 1

- **Leer** imagen: `I=imread('im1.jpg'), imshow(I)`
- **Guardar** imágenes: `imwrite(I, 'newim.jpg')`
- Representacion
 - Niveles de gris: Matriz de *uint8*
 - Color: “Stack” de 3 matrices *uint8* para R, G, y B
- **Conversion**: `I2 = double(I1)`
- **Cambio de paleta** (`colormap`): `colormap (<paleta>)`
 - paletas predefinidas (gray, bone, hot, cool, prism, jet, ...)
- Leer coord. del **raton** en la imagen: `[X,Y]=ginput(N)`

Procesamiento de Imágenes. Practica 1

Reducir Escala y Tomar Subimágenes

```
im1 = im (1:2:512,1:2:512);%reduccion escala  
im2 = im (1:128,1:128);%subimagen
```

Aplicación de funciones de transformación (lut)

```
<imagen salida> = <vector LUT> (<imagen entrada>)  
    rampai = 255:-1:0;  
    im2 = ramapi(im);
```

Calculo de histograma

```
<histograma> = imhist (<imagen>,256,1,255);  
    h = imhist (im,256,1,255);  
    bar (h);% diagrama de barras del vector de entrada
```

Procesamiento de Imágenes. Practica 1

Construir y aplicar una LUT

```
lut = makelut(fun,n)  
A = applylut(BW,lut)
```

Especificación de histograma

```
[J,T]=histeq(<imagen>,[hgram]);
```

hgram es el vector de histograma deseado para la imagen resultado

Ejemplo:

```
t=imread('imagen.bmp');  
hgram=imhist(t);  
[c,T2]=histeq(a,hgram);
```

Procesamiento de Imágenes. Practica 1

Transformada de Fourier (TdF)

- Cálculo de la transformada directa e inversa

$$\begin{aligned}\langle F \rangle &= \textit{fft2}(\langle \textit{imagen} \rangle); \\ \langle \textit{imagen} \rangle &= \textit{ifft2}(\langle F \rangle);\end{aligned}$$

Visualizar espectro de Fourier

```
imf = fft2 (im);  
imfs = fftshift (imf);  
image (imesp);
```