Procesamiento de imágenes con Matlab

En esta práctica inicial comenzaremos describiendo las características del software, mostraremos las funciones más destacadas ofrecidas por la herramienta para el tratamiento digital de imágenes y las primeras operaciones básicas para trabajar en Matlab.

MATLAB se corresponde con la abreviatura de **MATrix LABoratory** ("laboratorio de matrices"), un software matemático que ofrece un entorno de desarrollo integrado (IDE) y con un lenguaje de programación propio, lenguaje M.

Está disponible para las plataformas: Unix, Windows y Apple.

Ofrece prestaciones básicas tales como:

- Manipulación de matrices

- Representación de datos y funciones

- Implementación de algoritmos

- Creación de interfaces de usuario (GUI)

- Comunicación con programas en otros lenguajes

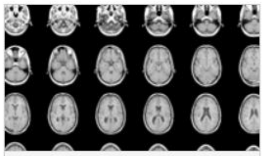
…

Es posible ampliar las capacidades de **MATLAB** utilizando las cajas de herramientas (Toolboxes).

En nuestro caso y por tanto específica para el tratamiento de imágenes utilizaremos la Toolbox "Image Processing".

http://es.mathworks.com/products/image/?refresh=true

Funciones



Exploración y descubrimiento

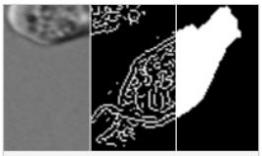
Utilice las funciones y aplicaciones para adquirir, visualizar, analizar y procesar imágenes en muchos tipos de datos.

» Más información

Mejora de la imagen

Aumente la proporción de señal a ruido y acentúe las características de las imágenes modificando los colores o las intensidades de una imagen.

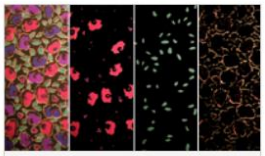
» Más información



Análisis de imágenes

Lleve a cabo análisis de imágenes extrayendo información significativa de las imágenes, por ejemplo localizando formas, contando objetos, identificando colores o midiendo las propiedades de los objetos.

» Más información

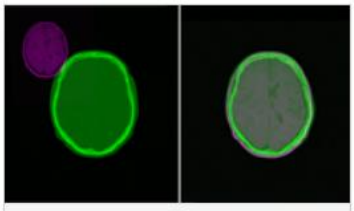


Segmentación de imágenes

Infórmese sobre los distintos planteamientos de la segmentación de imágenes, entre los que se incluyen métodos progresivos, creación automática de umbrales, métodos basados en bordes y métodos morfológicos.

» Más información

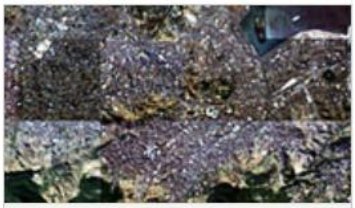
Ver el vídeo 5:11



Registro de imágenes y transformaciones geométricas

Lleve a cabo registro de imágenes, importante en la detección remota, la generación de imágenes médicas y otras aplicaciones en las que las imágenes deben estar alineadas para hacer posible el análisis cuantitativo o la comparación cualitativa.

» Más información



Procesamiento de imágenes grandes y aceleración del rendimiento

Trabaje con imágenes voluminosas que resulta difícil procesar y visualizar mediante los métodos estándar.

» Más información

Soporte para hardware y C/C++

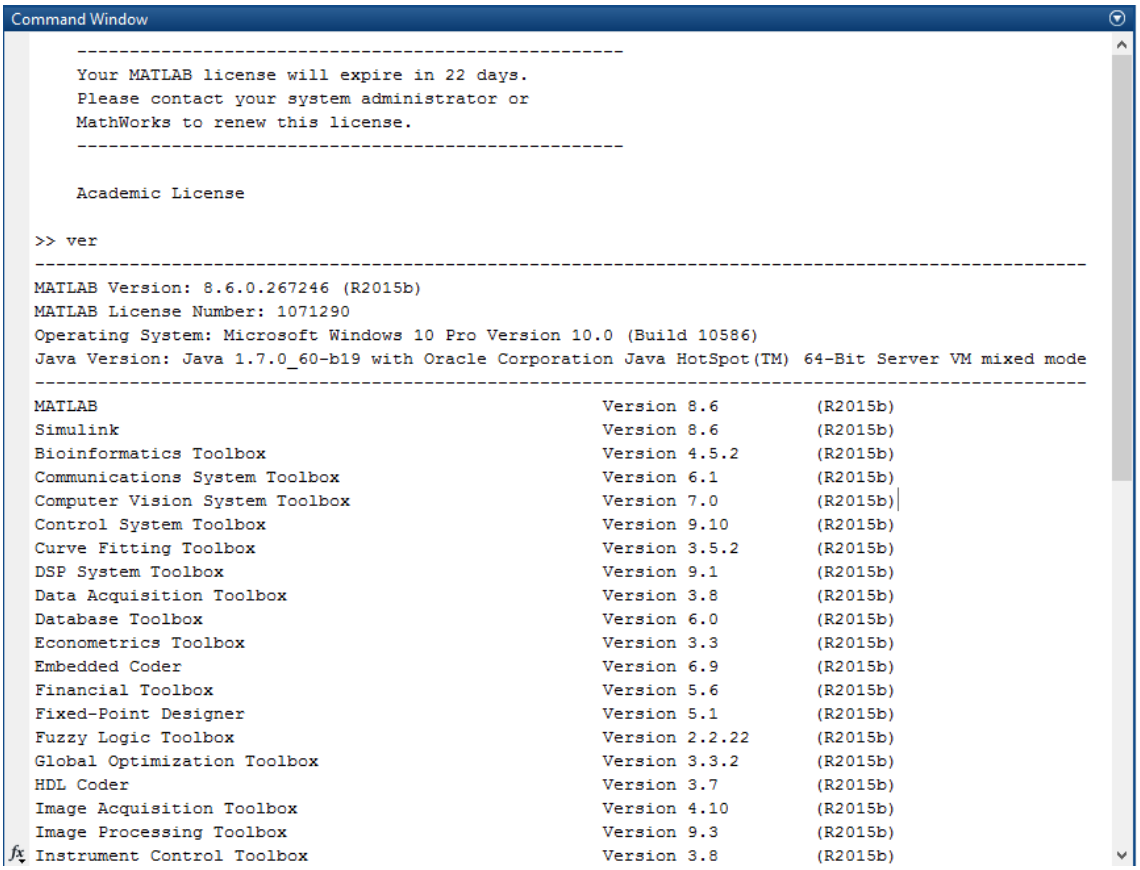
Utilizando Image Processing Toolbox con MATLAB Coder, Vision HDL Toolbox, y HDL Coder, puede trabajar con C/C++ y código HDL directamente con MATLAB. Muchas funciones de procesamiento de imagen soportan generación de código permitiendo ejecutar algoritmos de procesamiento de imagen en PC, FPGAs y ASICs. Esto le permitirá desarrollar sistemas de imagen para industrias como aeroespacial y defensa y médica.

» Más información

1. Primeras operaciones

Una vez instalado y ejecutado el software, la instrucción **ver** indicará las Toolboxes instaladas y las versiones correspondientes:

```
>> ver
```



En este punto es conveniente guardar las imágenes que vamos a utilizar en la carpeta de trabajo actual.

Ya que la primera instrucción que se empleará será la **lectura de ficheros** de imágenes (**imread( )**):

>> imgEnt = imread('cameraman.tif');

imgEnt representa el identificador para la imagen que queremos leer. Para buscar ayuda sobre las funciones o comandos se emplea la instrucción help:

>> help imread

Si queremos obtener más información sobre las variables, esta puede obtenerse visualizando la ventana WORKSPACE. También puede utilizarse la instrucción whos:

>> whos

Podemos comprobar el formato de la imagen, niveles de grises, la clase así como el tamaño de la misma, a modo de ejemplo la información será del tipo (256 valores diferentes (de la clase uint8) y tamaño de 256 x256 píxeles). Para la visualización de la imagen utilizamos la instrucción imshow:

>>imshow(imgEnt);

El tipo de dato matriz que contendrá una imagen, puede ser de varios tipos (según el tipo de dato de cada pixel)

- **double**: Doble precisión, números en punto flotante que varían en un rango aproximado de -10308 a 10308 (8 bytes por elemento)

- **uint8**: Enteros de 8 bits en el rango de [0,255] (1 byte por elemento)

- **uint16**: Enteros de 16 bits en el rango de [0, 65535] (2 bytes por elemento)

- **uint32**: Enteros de 32 bits en el rango de [0, 4294967295] (4 bytes por elemento)

- **int8**: Enteros de 8 bits en el rango de [-128, 127] (1 byte por elemento)

- **int16**: Enteros de 16 bits en el rango de [-32768, 32767] (2 bytes por elemento)

- **int32**: Enteros de 32 bits en el rango de [-2147483648,2147483647] (4 bytes por elemento)

- **logical**: Los valores son 0 ó 1 (1 bit por elemento)

2. Ejercicio:

Repita estas operaciones con otra imagen ‘bacteria.tif’. ¿Cuál es su tamaño?¿Cuántos niveles de grises tiene? Si la imagen fuese en color, normalmente quedará definida por tres matrices correspondiente a los tres colores básicos (rojo, verde y azul).

Vuelva a realizar las mismas operaciones de: a) lectura, b) tamaño y clase de la imagen y c) visualización sobre una imagen de color 'flowers.tif'.

Utilizando la notación de matrices de Matlab se pueden visualizar las tres componentes del color. El operador **:** hace referencia a todos los elementos de esa dimensión, luego el nivel de gris para cada parte del espectro de la luz será definido por (:,:,i).

Indica que todas las filas y las columnas para la componente i, i=1,2 ó 3 según se trate de los colores (rojo, verde o azul):

>>imshow([imgEnt(:,:,1),imgEnt(:,:,2),imgEnt(:,:,3)]);

El operador [ ] permitirá construir una matriz de N x (3*M), siendo N el número de filas y M el número de columnas.

Emplee el comando imtool para ver el nivel de gris de la imagen de 'nombre.tif' y los colores en 'nombre.tif'. Utilice el inspector de valores de los píxeles:

>> imtool('cameraman.tif');

>> imtool('flowers.tif');

Otra opción interesante para tratamiento de imágenes es el formato en binario. Normalmente se emplea el '0' para indicar el fondo y '1' para el objeto. En muchas ocasiones habrá que hacer una conversión.

Conversión entre tipos de datos: Para ciertas operaciones es necesario convertir una imagen de su tipo original a otro tipo de imagen que facilite su procesamiento. Algunos métodos son:

Instrucción	Descripción
gray2ind	Crea una imagen indexada a partir de una imagen de intensidad en escala de gris.
im2bw	Crea una imagen binaria a partir de una imagen de intensidad, imagen indexada o RGB basada en el umbral de luminancia.
ind2rgb	Crea una imagen RGB a partir de una imagen indexada.
rgb2gray	Crea una imagen de intensidad en escala de gris a partir de una imagen RGB
rgb2ind	Crea una imagen indexada a partir de una imagen RGB.

Conversión de imágenes en Matlab

Se emplea por tanto una técnica de umbralización para convertir las imágenes en binarias (im2bw()):

>> imgEntGris = imread('rice.tif');

>>figure(1); imshow(imgEntGris);

>>imgBW = im2bw(imgEntGris);

>> figure(2); imshow(imgBW);

>>impixelinfo;

Realice la misma operación de binarización con la imagen 'coins.png'.

3. Generando un fichero *.m

En este apartado se tratará de realizar la primera función (*.m) de procesamiento de imágenes con Matlab. Consistirá en leer un fichero de imagen 2D, cuyo nombre es pasado por parámetro, se visualizará y se aplicará una umbralización automática, la cual es también visualizada.

Editor - C:\Users\Amparo\Desktop\miPrimerScript.m

miPrimerScript.m

```
1   function miPrimerScript(nombreFich)
2   % Lectura del fichero imgEnt = imread(nombreFich);
3   imgEnt = imread(nombreFich);
4   %Visualización clf; figure(1); imshow(imgEnt);
5   clf;
6   figure(1);
7   imshow(imgEnt);
8   %Umbralización
9   imgBW = im2bw(imgEnt);
10  figure(2);
11  imshow(imgBW);
12
```

4. Construcción de imágenes

Construir una imagen binaria de 120 x 200 píxeles que tenga franjas horizontales de 20 píxeles de anchura, distanciada por cada 20 píxeles:

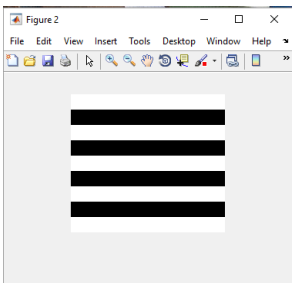
>> imgBW = false([120,200]);

>> for i=1:40:200

imgBW (i:i+19,:)=true;

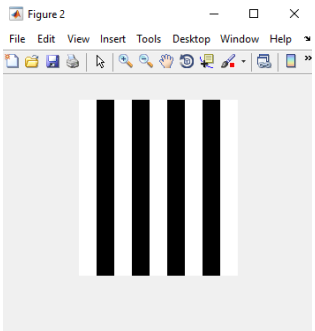
end

>>imshow(imgBW);



Si queremos que las franjas sean verticales sólo habría que emplear el operador traspuesta de las matrices.

>>imshow(imgBW');



5. Ejercicio

Realizar una función que construya y visualice dos imágenes de 256x256 con variación del nivel de gris en filas y columnas.