Prácticas de Visión Artificial

Práctica 0: Introducción al MATLAB y al procesado de imágenes

En esta primera sesión vamos a hacer una introducción al lenguaje de programación y entorno MATLAB y vamos a dar los primeros pasos en el procesado de imágenes. Los ejercicios del apartado "Primero pasos con MATLAB" son para familiarizarse con el entorno MATLAB y por lo tanto no se entregan.

1 Primeros pasos con MATLAB

Para la realización de la práctica se aconseja leer previamente el tutorial de introducción al Matlab. En algunos ejercicios de esta práctica, se hará referencia al apartado del tutorial donde se puede obtener más información.

- http://www.mathworks.es/help/pdf_doc/matlab/getstart.pdf

1.1 Descripción del producto

MATLAB es un lenguaje de alto nivel que permite desarrollar algoritmos, visualizar y analizar los datos y realizar cálculos numéricos. Su entorno de desarrollo permite un fácil acceso a las variables y a la visualización de los resultados.

*** (Opcional) Si quieres ver más información: 1. Quick Start -> Product Description**

1.2 Entorno MATLAB

Al iniciar MATLAB veremos que la aplicación muestra diversas ventanas. "Command Window" es una línea de comandos donde podremos introducir comandos. En Workspace se guardan todas las variables creadas con la línea de comandos o mediante scripts/funciones MATLAB.

*** Seguir los pasos descritos en Quick Start -> Desktop Basics**

1.3 Las funciones Help y Lookfor

*** Ver: 1. Quick Start -> Help and Documentation**

A la hora de obtener información sobre las funciones incluidas en MATLAB, la función help es muy útil, ya que da información sobre la función y parámetros aceptados. Ejecuta en la línea de comandos "help max" a modo de ejemplo.

En algunos casos queremos utilizar una acción, pero no conocemos el nombre de la función en MATLAB, en este caso una opción es utilizar el comando lookfor. Ejecuta "lookfor mean" a modo de ejemplo.

2 Introducción al MATLAB

Para la realización de cada uno de los apartados debe crearse una función .m. Ver cómo crear una función en Matlab.

Ver: 1.

- **Quick Start -> Programming and Scripts y**
- **Programming -> Scripts and Functions**

2.1 Creación de vectores y matrices

a) Ver cómo crear vectores y matrices.

2.1.1 Crea un script que crea un vector a que sea una fila de 10 elementos con valores aleatorios entre 0 y 1 y realiza las siguientes operaciones sobre el vector creado.

*** Ver: 1. Quick Start -> Matrices and Arrays**
*** Ver: 1. Quick Start -> Array Indexing**
*** Ver: 1. Quick Start -> Calling Functions**

Exercises:

a) Crea un vector a de 7 elementos aleatorios (help: rand) y accede a la tercera posición del vector.
b) Obtén el vector b formado por las 5 primeras posiciones del vector a.
c) Obtén el vector c formado por las posiciones impares del vector a.
d) Suma a todos los elementos del vector c la constante 0.7. Después suma todos los elementos resultantes del nuevo vector al vector c.
e) Obtén un vector d formado por los valores de b mayores o iguales a 0.4 .
f) Observar los valores de las variables creadas en el Workspace.

2.1.2 Crea una función llamada ex212.m que define 2 matrices 'A' y 'B', la primera de 2 filas y 3 columnas (2x3): [1 2 3; 4 5 6] y la segunda de (3x2): [2 4; 5 7; 9 3]. A continuación realiza las siguientes operaciones.

a) Multiplica las 2 matrices.
b) Convierte la matriz A en una nueva matriz A2 de 3x2 utilizando la función RESHAPE (utiliza la función **help** para conocer la función RESHAPE).
c) ¿Cómo puedes convertir la matriz A2 en un vector utilizando reshape?
d) Suma las matrices A2 y B y guarda el resultado en la variable C.
e) Multiplica, elemento por elemento, las matrices A2 y B.

2.1.3 Crea una función llamada ex213.m que define una matriz 'A' de 20x25 (help: rand) con números aleatorios. A continuación realiza las siguientes operaciones.

a) ¿Qué hace la línea: B=[A,A]? Utiliza la función size() para comprobar las dimensiones de B.
b) ¿Qué hace la línea: C=[A;A]? Comprueba e imprime las dimensiones de C.
c) Construye la submatriz D a partir de las primeras 5 filas y 6 columnas de A.
d) Suma todos los valores de la última fila de D. ¿Cómo se puede hacer la suma sin utilizar explícitamente el número de la última fila?

e) Pon todos los elementos de D, mayores que 0.8 a 0.
f) Pon todos los elementos de A, mayores que 0.8 y menores que 0.1 a 1 dcreando la matriz D. (help: Logical indexing applying multiple conditions en: http://es.mathworks.com/help/matlab/matlab_prog/find-array-elements-that-meet-a-condition.html#bt_xjhz)
g) Obtén el mínimo elemento de la submatriz de D conteniendo sus primeras 4 filas y 5 columnas. ¿En qué fila y columna está el mínimo elemento?
h) Utiliza el comando find() para comprobar la posición del elemento mínimo.

2.2 Visualización de datos

*** Ver: 1. Quick Start -> Workspace Variables**
*** Ver: 1. Quick Start -> 2-D and 3-D Plots**

Crea una función llamada ex22.m que:

a) Crea un vector de 50 posiciones con valores entre 0 y 4*pi para guardar la variable 'x' utilizando la función linspace(). Crea la función y=cos(x).
b) Utiliza el Workspace de MATLAB para ver el valor de los elementos de los vectores creados. ¿Qué valor aparece en la posición 7 del vector 'y'?
c) Crea un gráfico para visualizar la función coseno creada anteriormente (help: plot()).
d) Aplica un offset de +2 en cada una de las posiciones de 'y' y guarda el resultado en la variable y1. Dibuja los dos vectores en la misma figura (help: hold on, hold off) con diferentes colores (help: consulta los parámetros del plot()).
e) Crea un nuevo vector 'y2', donde cada posición contiene la raíz cuadrada de la posición correspondiente en 'y'. Consejo: Si no encuentras la función raíz cuadrada, utiliza el comando **LOOKFOR**.
f) Visualiza en la misma figura, en diferentes colores los gráficos de las funciones 'y' y 'y2'.
g) Visualiza en 2 gráficos contenidos en una única ventana donde el primer gráfico visualiza las funciones 'y' e 'y1', y el segundo: 'y' e 'y2' (help: consulta los comandos figure y subplot() de Matlab).

2.3 Scripts y funciones

*** Ver: 1. Quick Start -> Programming and Scripts**
* Ayuda: Las funciones zeros() y ones() son útiles a la hora de inicializar matrices.
Crea una función llamada ex23() que:

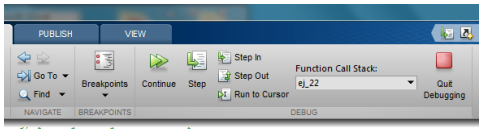
a) Crea dentro de ex23() una función divide2() que divida entre 2 el valor de entrada, siempre que éste sea mayor que 5. En caso contrario retornar el valor de entrada.

b) Repite el ejercicio, utilizando como argumento de entrada un vector fila en lugar de un único valor. Utiliza un bucle *for* para recorrer el vector. Llama a la función divideVector2().
c) MATLAB permite operar sobre vectores y matrices evitando la utilización de bucles. Optimiza el algoritmo anterior eliminando el bucle for. Llama a la función "divideVector2_opt.m".
d) Testea la función ex23.m pasando los valores 2 y 7 a la función divide2() y el vector x = [4 22 3.5 5.5 7] como argumento de entrada a las funciones creadas en b) y c). Guarda en distintas variables los resultados obtenidos.

2.4 Debug

*** Ejercicio sin entrega.**

MATLAB incluye métodos para ejecutar el código línea a línea y ver en cada paso el valor de las variables existentes.



Abrir el script ex23.m y situar un breakpoint dentro del código de la función. A continuación ejecutar el código y observar cómo la ejecución se detiene en el breakpoint creado. Ejecutar el comando "step" para avanzar línea a línea y utilizar "step in" para seguir línea a línea en una de las funciones creadas. Observar los valores de las variables en el workspace. Finalmente, ejecutar "continue" para avanzar hasta el final del código o hasta el próximo breakpoint.

3 Introducción al procesado de imágenes

El imageprocessingtoolbox de MATLAB permite abrir imágenes, manipularlas, convertir el espacio de colores, realizar distintos tipos de filtrado, etc. En esta primera parte, vamos a realizar una serie de ejercicios para familiarizarnos con la manipulación de imágenes en el entorno MATLAB.

* Nota: Algunos lectores de PDF o navegadores pueden convertir los guiones contenidos en el enlace como dobles guiones e impedir así que se habra el enlace.

El imageprocessingtoolbox define **cuatro tipos básicos de imágenes**:

- Binary (con valores de 0s y 1s, interpretados como negro y blanco, respectivamente),
- Grayscale (valores uint8, uint16, double,.. cuyos píxeles definen diferentes valores de intensidad),
- Truecolor o RGB Image (formadas por 3 matrices, llamadas comúnmente canales, que contienen los valores de intensidad R, G y B).
- Indexed (valores logical, uint8, uint16, double,.. cuyos píxeles son índices en un mapa de colores e.d. colormap. Cada índice del mapa de colores especifica una intensidad de color R, G y B.

Tipos de imágenes:
http://www.mathworks.es/es/help/images/image-types-in-the-toolbox.html
Mostrar diferentes tipos de imágenes
http://www.mathworks.es/es/help/images/displaying-different-image-types.html
Convertir imágenes de un tipo a otro:
http://www.mathworks.es/es/help/images/converting-between-image-types.html
Más información:
http://www.mathworks.es/es/help/images/image-import-and-export.html
http://www.mathworks.es/es/help/images/image-enhancement-and-analysis.html
https://en.wikibooks.org/wiki/MATLAB_Programming/Advanced_Topics/Toolboxes_and_Extensions/Image_Processing_Toolbox

3.1 Creación de imágenes

* Las imágenes en MATLAB no son nada más que matrices.

a) Crear una función ex31() que crea una imagen im1 de color negro de tamaño 25x25 de un sólo canal de tipo 8-bit unsigned. Help: crea la imagen como una matriz de tipo 'uint8'.
b) Obtener información sobre el tipo de imagen creada y su tamaño e imprimirla. Help: Funciones class, size y display.
c) Visualizar la imagen con MATLAB y guardarla como *void_img.jpg* (fig. 1 a). Ayuda: imshow() e imwrite().
d) Crear una imagen im2 de tamaño 256x256 de tipo 'double', donde cada píxel tenga valor 1. Repetir el procedimiento pero en este caso la imagen debe ser de tipo 8-bit unsigned ('uint8'). Visualizar ambas imágenes en una figura utilizando subplot. Poner los títulos de las imágenes usando title(). ¿Por qué el color de las 2 imágenes no es el mismo? ¿Cómo podemos utilizar 'imshow' para que la imagen 8-bit unsigned se visualice de color blanco (help: mirar los parámetros de imshow())?
e) Crear una imagen im3 de tamaño 256x256 de un sólo canal de tipo 8-bit unsigned mitad blanca y mitad negra, tal como se muestra en la figura 1 b). ¿Qué valor se ha de asignar a los píxeles a la izquierda para que sean blancos?
f) Visualizar la imagen con MATLAB y guardarla como *edge_img.jpg*.

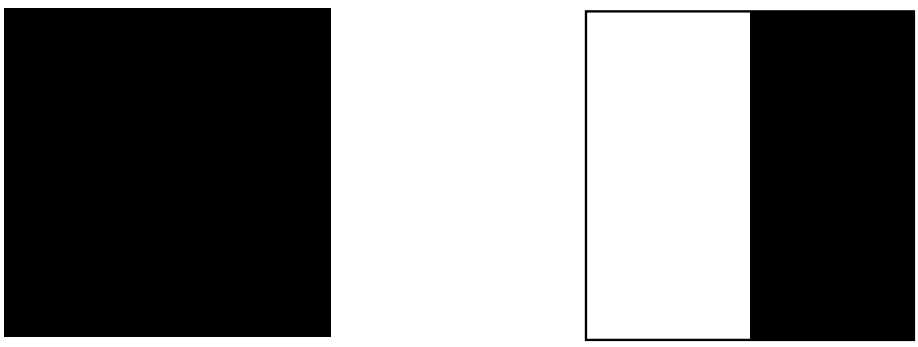


Figura 1 Crear imágenes a partir de matrices

3.2 Tratamiento de imágenes en escala de grises

Dada la imagen en escala de grises hand.jpg, crear la función *ex32*() que implemente los siguientes puntos:

a) Abrir el archivo hand.jpg (help: imread()) y guardarlo en una variable.
b) Visualizar la imagen con MATLAB.
c) Imprimir por pantalla los valores de los niveles de gris de la fila 200 en el rango de columnas [310, 330]
d) Convertir la imagen original de manera que resulte especular respecto al eje vertical (fig. 2 a) -> fig. 2 b)). Ayuda: Existen funciones en MatLab para realizar dicha operación.
e) Rotar la imagen original 90º en dirección de las agujas del reloj.
f) Guardar las imágenes creadas en d) y e) con el nombre *hand_flip.jpg* y *hand_rot.jpg*, respectivamente.

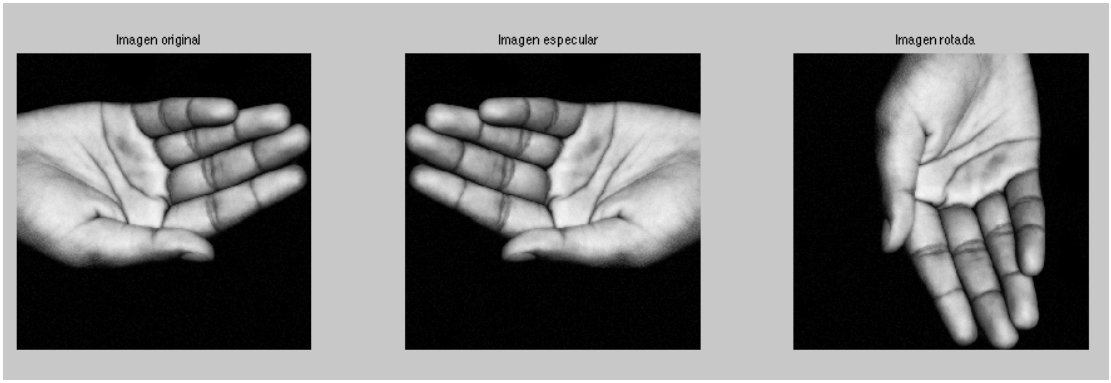


Figura 2 La imagen original hand.jpg (a), su especular (b) y su rotada (c).

Nota: Utilizar los comandos clear all, y close all para limpiar el workspace y cerrar todas las figuras innecesarias.