



Manual de Prácticas Procesamiento Digital de Imágenes Médicas

División de Ingeniería Eléctrica

Laboratorio de Cómputo para el
Procesamiento de Señales

Introducción a MATLAB

N° de práctica: 1

Nombre completo del alumno		Firma
N° de cuenta:	Fecha de elaboración:	Grupo:

Elaborado por:	Revisado por:	Autorizado por:	Vigente desde:
Dr. Ernesto Moya Albor			Agosto 2015

	<h1 style="text-align: center;">Manual de Prácticas Procesamiento Digital de Imágenes Médicas</h1>	
División de Ingeniería Eléctrica	Laboratorio de Cómputo para el Procesamiento de Señales	

1. Seguridad en la ejecución

	Peligro o Fuente de energía	Riesgo asociado
1		
2		
3		

2. Objetivo

El propósito de este pequeño tutorial es familiarizar al alumno con el paquete de computación científica MATLAB, brindando un breve resumen de los comandos y de la estructura de programación, así como de algunas de las características particulares que se utilizarán en las posteriores prácticas de la asignatura.

3. Introducción

❖ Marco teórico

MATLAB es el nombre abreviado de “**MATrix LABoratory**”. *MATLAB* es un programa para realizar cálculos numéricos con vectores y matrices. Como caso particular puede también trabajar con escalares –tanto reales como complejos–, con cadenas de caracteres y con otras estructuras de información más complejas. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB cuenta con un lenguaje de programación propio.



Manual de Prácticas Procesamiento Digital de Imágenes Médicas

División de Ingeniería Eléctrica

Laboratorio de Cómputo para el
Procesamiento de Señales

❖ Conceptos clave

Operadores aritméticos

Operación aritmética	Operador
Suma	+
Resta	-
Multiplicación matricial	*
Multiplicación de elementos	.*
Potenciación matricial	^
Potenciación de elementos	.^
División matricial izquierda	\
División matricial derecha	/
División izquierda de elementos	.\
División derecha de elementos	./

MATLAB tiene dos diferentes tipos de operaciones aritméticas. Las operaciones aritméticas matriciales están definidas por las reglas del álgebra lineal. Las operaciones aritméticas de elementos son realizadas elemento por elemento, por lo que para realizar las operaciones de multiplicación, potenciación y división de elementos (.*, .^, ./, .\), las matrices involucradas deberán ser de las mismas dimensiones.

Operadores de relación

Realizan comparaciones elemento por elemento.

Comparación	Operador
Menor que	<
Mayor que	>
Menor o igual que	<=
Mayor o igual que	>=
Igual que	==
Diferente que	~=



Manual de Prácticas Procesamiento Digital de Imágenes Médicas

División de Ingeniería Eléctrica

Laboratorio de Cómputo para el
Procesamiento de Señales

Operadores lógicos

Operación	Operador
AND	&&
OR	
NOT	~

Los *scripts* se crean desde el editor de texto que incluye *MATLAB*, también se puede usar algún otro editor como el *notepad* de windows o *vi* de unix/linux, la única condición es que el archivo no contenga formato y que se guarde con la extensión *.m*. Los *scripts* pueden operar sobre los datos existentes en el workspace y/o pueden crear nuevos datos.

Otra forma de realizar cálculos es por medio de una **función**, la cual es un archivo *.m* que a diferencia de un *script* que contiene bloques de comandos de *MATLAB* u otros *scripts* hace uso de sus propias variables y acepta argumentos de entrada.

La sintaxis básica de una función es la siguiente:

```
function [args1, args2, .. , argsn] = name(arge1, arge2, ..., argen)
```

```
%Información de ayuda de la función
```

```
Cuerpo
```

donde:

function.- es la palabra reservada para que *MATLAB* interprete el *script* como una función.

name.- Es el nombre de la función y deberá ser igual al nombre del archivo *.m*.

arge1, arge2, ..., argen.- Son las variables de entrada de la función.

args1, args2, ..., argsn.- Son las variables de salida que la función regresa al terminar su ejecución.

	<h1 style="text-align: center;">Manual de Prácticas Procesamiento Digital de Imágenes Médicas</h1>	
División de Ingeniería Eléctrica	Laboratorio de Cómputo para el Procesamiento de Señales	

Cuerpo.- Son todos los cálculos que se realizarán usando las variables de entrada y almacenando los resultados en las variables de salida.

Un *script* puede operar sobre las variables que se encuentran en el espacio de trabajo y generar nuevas variables si el *script* lo contempla, una función en cambio recibe variables del espacio de trabajo pero al realizar sus operaciones todas las variables son locales sólo a la función (a menos que se especifiquen variables globales).

4. Ejemplos

❖ Ejemplo 1

Un ejemplo de un *script* son los siguientes comandos almacenados en el archivo *inicia.m*:

```
%Programa inicia

clc % Limpia la pantalla

clear all %Elimina todas las variables

close all %Cierra todas las ventanas de figuras
```

el propósito de este *script* es la de limpiar la pantalla, borrar todas las variables del *workspace* y cerrar todas las ventanas de figuras existentes.

Para ejecutar el *script* se teclea su nombre sin la extensión en el *prompt*, por ejemplo para ejecutar *inicia.m*:

```
>> inicia
```

Nota: hay que observar que el directorio actual de trabajo (*Current Directory*) sea el que contenga almacenado al *script*.

Otra forma de ejecutar un *script* es directamente desde la ventana del editor de texto de MATLAB donde se visualiza el *script* usando la opción Run del menú Debug (tecla F5) o por medio del acceso directo, el cual salva el *script* y lo ejecuta mostrando los posibles resultados en el *prompt* (>>).

	<h1 style="text-align: center;">Manual de Prácticas Procesamiento Digital de Imágenes Médicas</h1>	
División de Ingeniería Eléctrica	Laboratorio de Cómputo para el Procesamiento de Señales	

❖ Ejemplo 2

Por ejemplo, se desea realizar un programa que realice la siguiente operación:

- Calcular el número de segundos que ha vivido una persona hasta su último cumpleaños.

Se puede implementar por medio del script *ejemplo1*:

```
%Listado que calcula el número de segundos que una persona ha vivido

inicia

edad=input('Edad en años = ');

s=edad*3.154e7
```

Este *script* usa la función de **MATLAB input** para asignar valores a variables desde el teclado, el texto entre comillas simples se muestra en la pantalla y se espera por un valor por parte del usuario. En este *script* se han creado en el espacio de trabajo las variables **edad** y **s**.

❖ Ejemplo 3

Una forma alternativa de realizar el cálculo del ejemplo 2 es mediante la función de usuario **vida_segundos.m**:

```
function s = vida_segundos(edad)

%
% Funcion que calcula el numero aproximado de segundos que usted ha vivido
%
% s = vida_segundos(edad)
%
% edad.- Edad en años
```



Manual de Prácticas Procesamiento Digital de Imágenes Médicas

División de Ingeniería Eléctrica

Laboratorio de Cómputo para el
Procesamiento de Señales

```
% s.- Numero de segundos que ha vivido
```

```
%
```

```
% 1 año = 3.154x10^7 segundos
```

```
%
```

```
s=edad*3.154e7;
```

Para ejecutar esta función realizamos:

```
>> inicia
```

```
>> s=25;
```

```
>> Res=vida_segundos(s);
```

```
>> s
```

```
s =
```

```
25
```

```
>> Res
```

```
Res =
```

```
788500000
```

Se observa que las variables de la función son válidas sólo dentro de la misma, la variable **s** usada en el espacio de trabajo mantiene su valor después de la ejecución de la función.

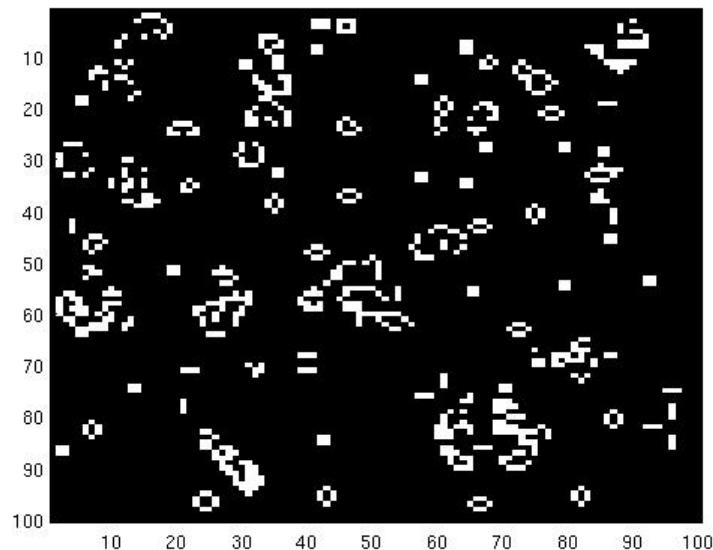
	<h1 style="text-align: center;">Manual de Prácticas Procesamiento Digital de Imágenes Médicas</h1>	
División de Ingeniería Eléctrica	Laboratorio de Cómputo para el Procesamiento de Señales	

5. Ejercicios a realizar

I. Ejercicio 1

El juego de la vida

Realizar un script en MatLab para implementar el juego de la vida diseñado por el matemático británico John Horton Conway en 1970, este juego es un autómata celular que consta de una matriz donde cada casilla puede contener vida (célula blanca) o no contener vida (célula negra):



En base a una serie de reglas se va construyendo una nueva matriz al analizar cada una de las células y decidir que células deberán “morir” y cuales deberán “vivir”.

El primer paso es crear un mundo inicial (mundo1) donde en forma aleatoria se determina que células contienen vida. La probabilidad de que exista vida en una célula determinada puede ser del 50%, con lo cual se puede usar directamente el comando **rand**.

El mundo inicial es analizado elemento por elemento y se determina si la casilla actual tiene vida o no, la condición de la célula en el mundo siguiente (mundo2) dependerá de las siguientes reglas:



Manual de Prácticas Procesamiento Digital de Imágenes Médicas

División de Ingeniería Eléctrica

Laboratorio de Cómputo para el
Procesamiento de Señales

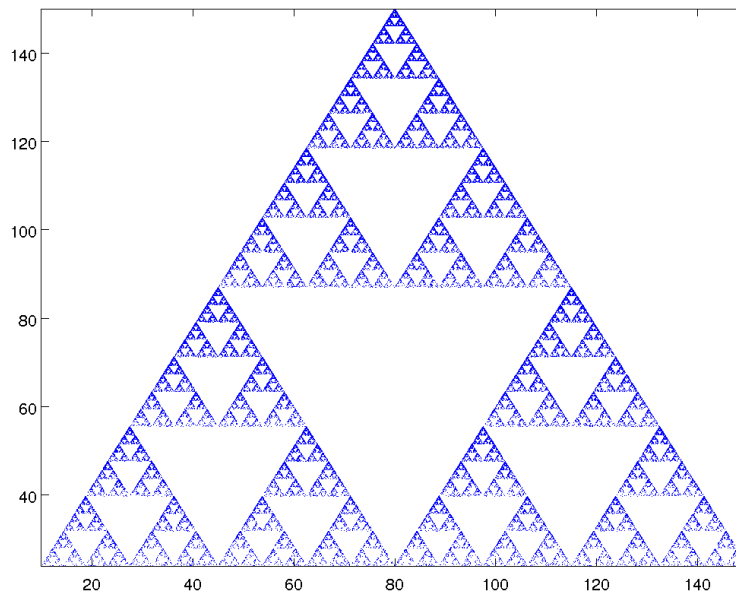
- 1) Una célula muerta vivirá si a su alrededor hay exactamente 3 células vivas.
- 2) Una célula seguirá viva si tiene a su alrededor existen 2 o 3 células vivas.
- 3) En el resto de los casos, la célula muere por falta de población o por superpoblación.

En la siguiente iteración el mundo siguiente (mundo2) se convierte en el mundo inicial (mundo1) y el proceso se repite un número predeterminado de ciclos.

II. Ejercicio 2

Triángulo de Sierpinski

Fractal ideado por el matemático polaco Waclaw Sierpinski en 1919. Partiendo (iteración $n=0$) de la superficie de un triángulo equilátero de lado unitario. Seguidamente (iteración $n=1$) se toman los puntos medios de cada lado y se construyen a partir de ellos un triángulo equilátero invertido de lado $1/2$. Ahora (iteración $n=2$) se repite el proceso con cada uno de los tres triángulos de lado $1/2$ que quedan. Posteriormente se recortan, esta vez, tres triángulos invertidos de lado $1/4$. Si se repite infinitamente el proceso se obtendrá una figura fractal denominada triángulo de Sierpinski.



	<h1 style="text-align: center;">Manual de Prácticas</h1> <h2 style="text-align: center;">Procesamiento Digital de Imágenes Médicas</h2>		
División de Ingeniería Eléctrica	Laboratorio de Cómputo para el Procesamiento de Señales		

Realizar un script en MatLab para dibujar el triángulo de Sierpinski de manera iterativa, donde el único parámetro será el número de puntos a dibujar.

III. Ejercicio 3

Criptografía Visual

1) Realizar un script para encriptar la imagen de dos niveles de gris (blanco y negro) almacenada en el archivo **img.mat** utilizando el siguiente algoritmo visual de criptografía:

a) Se analiza cada pixel de una imagen binaria (conformada por 0's y 1's) y por cada pixel se generarán dos bloques de 2x2 pixeles, el conjunto de todos los bloques formarán dos imágenes de salida del doble del tamaño de la imagen original.







b) Los posibles bloques para codificar cada pixel son:

Nota: El color blanco de cada bloque se representará como el valor de 1 y el negro con 0. Cada bloque tiene su complemento, por ejemplo, los bloques 1 y 2 son complementarios, al igual que los bloques 3 y 4, y 5 y 6.

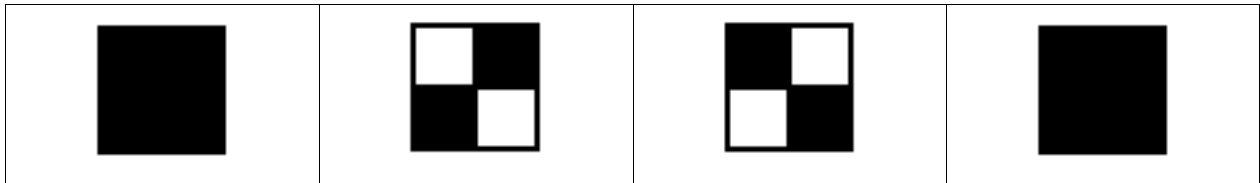
c) Las reglas para asignar los bloques a la imagen de salida son:

- Cuando se trata de un pixel blanco (255) se elegirá aleatoriamente uno de los posibles bloques y se escribirá en la posición correspondiente en ambas imágenes de salida.
- Cuando se trate de un pixel negro (0) se elegirá de forma aleatoria uno de los posibles bloques y se escribirá en la imagen 1 y su bloque complementario en la imagen 2.

Por ejemplo en la siguiente figura se muestra una posible elección de los bloques de salida para un pixel blanco (fila 1) y un pixel negro (fila 2).

Pixel original	Bloque 1	Bloque 2	Bloque 1 AND Bloque 2
			

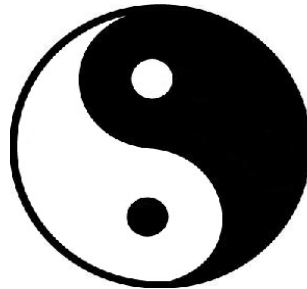
	<h1 style="text-align: center;">Manual de Prácticas Procesamiento Digital de Imágenes Médicas</h1>	
División de Ingeniería Eléctrica	Laboratorio de Cómputo para el Procesamiento de Señales	



La idea básica del algoritmo es “apilar” ambas imágenes, que pueden ser impresas en transparencias o en nuestro caso aplicar una operación AND de tal manera que las siguientes combinaciones nos permitan recuperar el valor de la imagen original:

- Negro AND negro nos da negro (mismo bloque)
- Negro AND blanco nos da negro (bloques complementarios)
- Blanco AND blanco nos da blanco (mismo bloque)

A continuación se muestra un ejemplo de la ejecución del algoritmo utilizando la siguiente imagen:



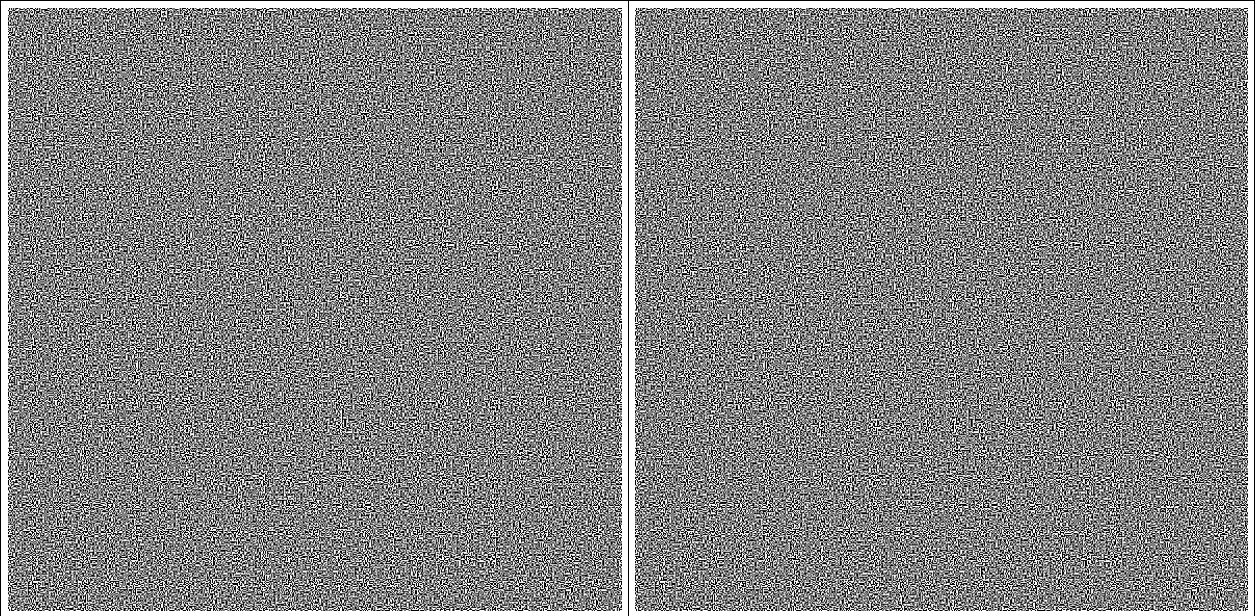
Las imágenes de salida serán:



Manual de Prácticas Procesamiento Digital de Imágenes Médicas

División de Ingeniería Eléctrica

Laboratorio de Cómputo para el
Procesamiento de Señales



Como se observa, las dos imágenes por separado no muestran algún indicio de la forma de la imagen original.

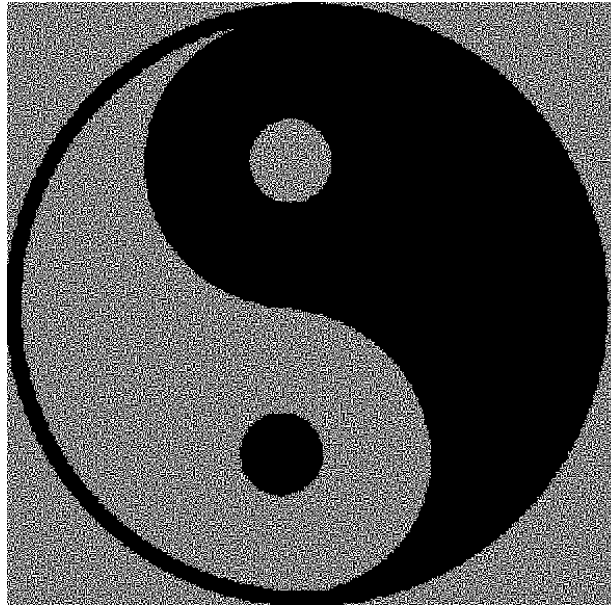
Al realizar la operación AND entre ambas imágenes se obtiene la siguiente imagen del doble de tamaño original:



Manual de Prácticas Procesamiento Digital de Imágenes Médicas

División de Ingeniería Eléctrica

Laboratorio de Cómputo para el
Procesamiento de Señales



A pesar de que el contraste de la imagen resultante es degradado en un 50%, el ojo humano puede identificar el contenido de la imagen secreta fácilmente.

6. Referencias

- ❖ **Digital Image Processing, González, R.C , Woods, P., Addison Wesley, 1992**
- ❖ **MATLAB Documentation:**
<http://www.mathworks.com/help/matlab/>