MATLAB Y SIMULINK

CAPITULO 2

Hasta hace pocos años la comunidad de procesamiento de imágenes y visión por computadora era un grupo relativamente pequeño de personas, las cuales tenían acceso a herramientas de procesamiento muy caras, o bien se caracterizaban por ser expertos en algún lenguaje de programación. Actualmente existe una gran cantidad de librerías que facilitan el proceso de diseño del sistema de procesamiento de imagen, ya sea para aplicaciones de inspección de manufactura, hasta para la navegación de un robot móvil. Ejemplos de este tipo de herramientas es **OpenCV** para utilizarse con lenguaje **C** o **ImajeJ** para ser utilizada con **Java**.

Aun a pesar de la oferta existente de librerías y software desarrollado por empresas e institutos de investigación, sigue siendo compleja la implementación de algoritmos de visión, esto se debe principalmente a que la curva de aprendizaje de la utilización de las librerías así como la caprichosa configuración de sus ambientes hace complicado uso. Lo anterior puede ser entendible ya que la utilización de una librería no solo involucra el dominar los tipos de datos y parámetros de las funciones que implementan, sino que demanda un conocimiento aceptable del lenguaje en el que la librería se encuentra implementada.

MatLAB es un programa que permite realizar cálculos técnicos y científicos. MatLAB con ya varios años en el mercado y una gran cantidad de funciones implementadas para diferentes disciplinas científicas se ha convertido prácticamente en un estándar de programación y desarrollo rápido de aplicaciones. La unidad de procesamiento de MatLAB es la matriz, por lo que su utilización en el procesamiento de imágenes (matrices) es una extensión natural de su lenguaje de programación.

Simulink puede ser considerado como un ambiente de simulación de sistemas incorporado al programa MatLAB, el cual permite analizar la respuesta en el tiempo de sistemas dinámicos complejos. Con la incorporación de la herramienta llamada “Video and Image Processing Blockset” Mathworks1 logro lo que parecía imposible generar un ambiente de implementación de algoritmos de procesamiento de imagen, que tuviera la capacidad de operan en tiempo real. Lo anterior se denota como increíble, debido a que las herramientas desarrolladas son muy sencillas de manejar y extender, ya que para ello solo es necesario realizar código utilizando el lenguaje de programación de MatLAB.

En este capítulo se aborda de manera condensada algunos aspectos importantes del manejo de imágenes y de funciones incluidas en el toolbox de procesamiento de imagen de MatLAB que serán de uso extensivo a lo largo del libro, de igual manera se dan los detalles generales de la programación en MatLAB. El material tratado en este capítulo es considerado como básico e indispensable para el desarrollo de los algoritmos propuestos a lo largo del libro, mientras que la manera de desarrollar aplicaciones en Simulink, utilizando las librerías de video y procesamiento de imágenes, es tratada por su extensión de manera separada.

[1] Compañía de marca registrada que desarrollo el programa MatLAB y el entorno Simulink

2.1 CONSIDERACIONES INICIALES

Una imagen es considerada una función bidimensional descrita en forma matricial como:

$$I(x,y)=\begin{bmatrix} I(1,1) & I(2,1) & \dots & I(N,1) \\ I(1,2) & I(2,2) & \dots & I(N,2) \\ \vdots & \vdots & \ddots & \vdots \\ I(1,M) & I(2,M) & \dots & I(N,M) \end{bmatrix} \tag{2.1}$$

Donde el desplazamiento en forma horizontal trae consigo un aumento en el índice x mientras que los desplazamientos en el sentido vertical aumentan el índice y, además M y N representan las dimensiones de la imagen. La forma de representar imágenes en MatLAB representa una adaptación del modelo descrito en la ecuación 2.1 a la manera de indexar los datos matricialmente, esto significa que los índices intercambian posiciones, con lo anterior es posible adoptar la notación matricial típica de renglón-columna. Considerando lo anterior las imágenes en MatLAB serán representadas de la siguiente manera:

$$\mathtt{I}=\begin{bmatrix} \mathtt{I(1,1)} & \mathtt{I(1,2)} & \dots & \mathtt{I(1,N)} \\ \mathtt{I(2,1)} & \mathtt{I(2,2)} & \dots & \mathtt{I(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathtt{I(M,1)} & \mathtt{I(M,2)} & \dots & \mathtt{I(M,N)} \end{bmatrix} \tag{2.2}$$

2.2 LECTURA, DESPLIEGUE Y ESCRITURA DE IMÁGENES

Las imágenes son leídas de algún dispositivo de almacenamiento al ambiente de MatLAB usando la función `imread`. La sintaxis general de esta función es:

```
A=imread('nombre_del_archivo')
```

Donde `nombre_del_archivo` es una cadena de caracteres que determina el nombre de la imagen con su respectiva dirección de búsqueda, mientras que la variable `A` almacenara la imagen obtenida. El tipo de imagen que se puede cargar mediante la función `imread` corresponde a los tipos descritos en la Tabla 2.1.

Una vez que la imagen ha sido cargada en una variable, podría ser conveniente indagar sobre sus dimensiones, lo que puede ser realizado mediante la función `size`. De tal manera que para indagar el tamaño de la imagen almacenada en `A`, se escribiría:

```
[M N]=size(A)
```

Donde la variable `M` almacenara el número de renglones de la imagen, mientras que `N` el número de columnas.

Tipo de imagen	Descripción	Extensión
TIFF	Tagged Image file format	.tif o .tiff
JPEG	Joint photographic experts group	.jpg o .jpeg
GIF	Graphics interchange format	.gif
BMP	Windows Bitmap	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

Tabla 2.1 Tipos de imágenes soportadas por la función imread.

Para desplegar imágenes en MatLAB como si se trazar de graficas generadas por un el comando `plot`, se utiliza la función `imshow`. De esta manera si se deseara desplegar la imagen almacenada en la variable `A`, se escribiría en línea de comandos:

```
imshow(A)
```

Después de que se han realizado procesamientos sobre la imagen seria conveniente guardar la imagen resultante, lo cual se realiza mediante la función `imwrite`. Su sintaxis se define como:

```
imwrite(B,'nombre_del_archivo')
```

Donde `B` representa la imagen que se pretende guardar, mientras que `nombre del archivo` representa la cadena de caracteres que define un nombre valido para la imagen junto con una extensión congruente con las definidas en la tabla 2.1.

2.3 TIPOS DE DATOS

Aunque normalmente se utilizan números enteros para indexar las coordenadas de la imagen, los valores de los píxeles no se encuentran restringidos a valores enteros del intervalo clásico de despliegue [0,255], sino que estos pueden asumir diferentes valores o tipos, inclusive números complejos (en el caso de la transformada de Fourier).

Todos los cálculos numéricos desempeñados por MatLAB se realizan considerando que los números son del tipo **`double`** (o números de coma flotante), por lo que muchas de las operaciones realizadas sobre imagen tienen (o tendrán) también este formato. El tipo de dato **`uint8`** (enteros en el intervalo [0,255]) se presenta normalmente cuando se realiza la carga de una imagen al ambiente de MatLAB por medio de la función `imread`. Otro tipo de dato popular para la representación de imágenes es el **`logical`**, el cual permite representar en forma compacta imágenes binarias, por lo que este tipo de dato solo tiene dos valores posibles 0 y 1. Otro tipo de datos común que es muy utilizado en el ambiente Simulink es el **`single`**, el cual permite representar un número real con una precisión simple, siendo su intervalo de representación $\left[10^{-38}, 10^{38}\right]$.

2.4 TIPOS DE IMÁGENES EN MATLAB

El toolbox de procesamiento de imágenes de MatLAB permite la manipulación de 4 tipos de imágenes:

1. Imágenes de intensidad o escala de grises
2. Imágenes Binarias
3. Imágenes indexadas
4. Imágenes de color RGB

En esta sección solo se analiza el caso de las dos primeras por ser consideradas las mas comunes y básicas en el procesamiento, los demás tipos de imágenes serán descritas en los subsecuentes capítulos.

Imágenes a escala de grises

Una imagen a escala de grises es una matriz cuyos valores han sido escalados para representar un determinado número de intervalos. Si la imagen es del tipo `uint8` entonces los datos que la conforman se encuentran en el intervalo [0, 255]. Si la imagen es del tipo `double`, entonces los datos que la constituyen son del tipo flotante y se encuentran en el intervalo [0,1].

Imágenes binarias

Una imagen binaria del tipo `logical` se representa en MatLAB como un arreglo que solo contiene unos y ceros. Estos ceros y unos son especiales, por que no implican valores numéricos, sino más bien banderas que indican el estado de falso (0) o verdadero (1).

2.5 CONVERSIÓN ENTRE DIFERENTES TIPOS DE DATOS Y DE IMÁGENES

La conversión de tipos de datos que constituyen a las imágenes es una tarea frecuente en el procesamiento de imágenes, debido a que no todos los tipos de datos son convenientes para realizar un tipo de operación especifica.

Conversión entre tipos de datos

La conversión entre tipos de datos se realiza de manera sencilla en MatLAB utilizando la siguiente sintaxis general:

```
B=tipo_de_dato(A)
```

Donde `tipo_de_dato` es uno de los tipos de datos `uint8`, `double`, `logical`. De esta manera el tipo de datos original atribuido a `A` se convertirá al tipo de dato designado por `tipo_de_dato` y será grabado en la matriz `B`.

Un ejemplo de esta conversión seria, si se considera que la imagen `A` es del tipo de datos `uint8` y se deseara convertir al tipo `double` se escribiría en línea de comandos:

```
B=double(A)
```

Conversión entre diferentes tipos de imágenes

El toolbox de procesamiento de imágenes posee varias funciones que permiten realizar los escalamientos necesarios para convertir diferentes tipos de imágenes.

La función `im2uint8` indaga el tipo de dato de la imagen de entrada y realiza a partir de este las operaciones necesarias para convertirla al intervalo de valores definido para `uint8`, el cual se encuentra entre 0 y 255.

Para ejemplificar esta función se considera una imagen de 2x2 del tipo `double` definida en MatLAB como:

```
A =

   -0.2000    0.8000
    1.0000    0.2000
```

Si se desempeñara la conversión se escribiría:

```
B=im2uint8(A)
```

Resultando:

```
B =
     0   204
   255    51
```

Del resultado puede ser observado como se le asigna al menor numero de A el menor numero representable en el tipo `uint8` (0) y al máximo valor de `A` se le asigna el máximo representable (255), los demás valores se escalan linealmente en el intervalo [0,255].

Una función que será frecuentemente utilizada a lo largo del libro es `mat2gray`. La función `mat2gray` escala linealmente una imagen del tipo `double` cuyo intervalo de valores puede ser cualquiera a otra imagen del tipo `double` pero con un intervalo de valores que van de 0 a 1. En el procesamiento de imágenes existen diferentes tipos de algoritmos los cuales al procesar una imagen generan una imagen resultante cuyos valores pueden variar en intervalos bastante amplios, por lo que normalmente se dificulta su análisis y visualización, es por ello que la función `mat2gray` será utilizada siempre en aquellos casos donde sea necesario visualizar la información contenida, en un intervalo representable a escala de grises. Para ejemplificar esta función se considera una imagen de 2x2 del tipo `double` definida en MatLAB como:

```
A =

   -100      20
    200    1000
```

Si se desempeñara la conversión se escribiría:

```
B=mat2gray(A)
```

Resultando:

```
B =

         0    0.5100
    0.6000    1.0000
```

2.6 INDEXADO DE VECTORES Y MATRICES

En las operaciones de procesamiento de imágenes es frecuente la discriminación y búsqueda de elementos o píxeles contenidos en la imagen, por lo que las operaciones de indexado deben ser consideradas importantes. MatLAB soporta poderosos métodos de indexado para matrices y vectores, que simplifica la elaboración de programas y mejora la eficiencia de los algoritmos.

Indexado de vectores

Un vector renglón de dimensión 1x *N* , puede ser indexado utilizando solamente un índice numérico, de esta manera si se considera a este vector como `v`, se elegirá el primer elemento si se indexa como `v(1)`, de la misma forma se elegirá el elemento numero 2 del vector si se indexa `v(2)`. Esto puede ejemplificarse si se considera:

```
>> v=[1 2 3 4 5 6 7];
```

Y se elige `v(2)`, se tiene

```
>> v(2)
ans =
     2
```

Para acceder a bloques de elementos en un vector MatLAB utiliza la notación de los dos puntos `v(n1:n2)`, los cuales indican una secuencia de índices que van desde el primer numero especificado `n1`, hasta el segundo `n2`. Por lo que si se tiene:

```
>> v(2:4)
ans =
     2     3     4
```

Puede también especificarse el final de la secuencia utilizando la palabra reservada `end`, de tal forma que se indica el último índice considerado en el vector. Esto en el ejemplo significaría:

```
>> v(2:end)
ans =
     2     3     4     5     6     7
```

También puede seleccionarse en la notación de los dos puntos que el incremento realizado del primer índice `n1` y el segundo `n2`, no sea unitario. Por consiguiente si se quisiera elegir solo los índices pares del vector se escribiría:

```
>> v(2:2:end)
ans =
     2     4     6
```

Indexado de matrices

Para el indexado de matrices es necesario 2 números, uno para elegir los renglones y otro para las columnas. De esta manera si se utiliza como ejemplo la matriz 3x3 definida como:

```
>> A=[1 2 3; 4 5 6; 7 8 9];

A =

     1     2     3
     4     5     6
     7     8     9
```

Considerando la matriz `A`, puede seleccionarse el elemento correspondiente al renglón 3 y columna 2, utilizando:

```
>> A(3,2)

ans =

     8
```

Es posible utilizar la notación de los dos puntos para indexar de forma total, es decir los dos puntos en el contexto de matrices se considera como un comodín que indica todo o la totalidad. Por lo que si se quisiera seleccionar todo el renglón 2, se tendría:

```
>> A(2,:)

ans =

     4     5     6
```

De igual manera si se quisiera indexar la columna 1, se escribiría

```
>> A(:,1)

ans =

     1
     4
     7
```

Ejemplos de indexado en imágenes.

Todas las formas de indexar elementos en matrices pueden ser usadas para elegir elementos, reducir o bien seleccionar bloques en imágenes. A continuación se presentan una serie de ejercicios de indexados realizados sobre imágenes, de igual forma se presenta los resultados obtenidos de forma grafica.

Si se considera como imagen ejemplo asignada a la variable `A`, a la mostrada en la figura 2.1(a) y se desearía muestrear la imagen de tal manera que se tomara un píxel de la imagen y otro no en sentido horizontal y vertical, se escribiría en línea de comandos:

```
>>B=A(1:2:end,1:2:end);
```

Con lo anterior se crearía una nueva imagen `B` cuyas dimensiones serian la mitad de `A`, ya que solamente la mitad de los píxeles serán seleccionados para formar `B`. El resultado de esta operación

se muestra en la figura 2.1(b). Se podría también al igual que las operaciones sobre vectores seleccionar un bloque de la imagen `A` para formar una nueva imagen `C`. De esta manera la sentencia:

```
>> C=A(1:100,1:100);
```

Seleccionara un bloque de 100 píxeles tanto en el sentido horizontal como vertical. El resultado de esta operación se muestra en la figura 2.1(c).

Cuando se manipulan bloques de imágenes resulta importante poder cambiar el valor de intensidad de un bloque seleccionado, ya sea para eliminar su información, o bien para formar una mascara para un procesamiento posterior. Lo anterior puede ser fácilmente realizado indexando el bloque de la imagen y confiriéndole el valor deseado, de esta manera la instrucción:

```
>> A(1:100,1:100)=0;
```

Colocaría todos los píxeles del bloque de 100 píxeles cuadrados en negro. El resultado de esta operación se observa en la figura 2.1(d).

(a)

(b)

(c)

(d)

2.7 OPERACIONES SOBRE MATRICES COMPLETAS

De manera frecuente es necesario desempeñar operaciones que engloben la totalidad de los elementos de la imagen, un ejemplo típico es el encontrar ya sea el píxel máximo o mínimo presente en la imagen. El problema de utilizar de forma sencilla la función `max` o `min` de MatLAB aplicada a la imagen o matriz es que devolverá el valor máximo o mínimo de cada columna, por lo que el valor absoluto es encontrado mediante la operación concatenada de máximos o mínimos. Esto es, si se tiene una matriz definida como:

```
M =

     1     2     3
     4     5     6
     7     8     9
```

Y si se colocara:

```
>> max(M)

ans =

     7     8     9
```

Se obtendría el valor máximo de cada columna, de tal forma que si se elige la forma concatenada:

```
>> max(max(M))

ans =

     9
```

Se obtiene el máximo global. La forma de operar en forma global con operaciones puede utilizarse con muchas operaciones como será visto a lo largo del libro.

PROGRAMACIÓN EN MATLAB

MatLAB no solamente puede ser usado como una potente calculadora en línea de comandos, sino que es posible realizar programas utilizando estructuras y lineamientos propios de un lenguaje de programación. La ventaja de utilizar MatLAB para realizar programas es que se puede sacar ventaja de las cientos de funciones ya programadas creadas para el procesamiento de imágenes o bien de otras funciones realizadas como herramientas para otras áreas afines como el procesamiento digital de señales, además se puede de igual manera utilizar los poderosos modos de indexado de matrices mostrados en secciones anteriores. A todas estas ventajas se añade que la forma de programar en MatLAB es prácticamente intuitiva y sencilla, ya que no es necesario declarar variables (a menos que se quiera que tengan valides en varias funciones) ni preocuparse por el tipo de su contenido.

En la programación de MatLAB se pueden crear principalmente dos tipos de programas los llamados **.m** y las **funciones**. Este tipo de programas debe ser considerado como un script, el cual ejecuta una serie de instrucciones en MatLAB, tal como se hiciera secuencialmente desde la consola de MatLAB. Gran parte de los programas mostrados en el libro son .m, la razón de esto es que representa la forma mas rápida de codificar y probar algoritmos. Como ejemplo de un archivo .m, se muestra el código del programa 2.1, que determina el píxel máximo y mínimo de una imagen.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programa que encuentra el valor del píxel máximo
%y mínimo de una imagen considerada como A
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 05/06/08
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Se encuentra el valor máximo
Vmax=max(max(A));
%Se encuentra el valor minino
Vmin=min(min(A));
```

Una función en MatLAB es un programa que a diferencia de los .m tiene una estructura definida que permite la comunicación ya sea con línea de comandos u otros programas. Esto se debe a que este segmento de código permite la transferencia de datos para que opere sobre ellos y también envía información de los resultados obtenidos como efecto de su operación. Una función tiene por lo tanto en MatLAB la siguiente estructura:

```
function[salidas]=nombre(entradas)
```

La palabra reservada `function` al inicio del programa indica que se codifica una función. Entre corchetes se establece los datos `resultados` o `salidas` que la función regresara como resultado de operar sobre los datos recibidos en `entradas` o `datos`. Como ejemplo se estructurara el programa mostrado en 2.1 como .m pero ahora como función, el código se muestra en programa 2.2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Función que determina los valores máximos y mínimos
%de los pixeles de una imagen recibida como entrada
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Erik Cuevas, Daniel Zaldivar, Marco Pérez
% Versión: 05/06/08
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Vmax Vmin] = maxmin_imagen(B)
% se determinan los valores máximo u mínimo
Vmax=max(max(B));
Vmin=min(min(B));
```

HOMEWORK

- Descarga una imagen a color y transformala a escala de grises
- Investiga cual es el comando el cual me da las caracteristicas de la imagen por ejemplo las dimenciones
- Porque es famosa la imagen de lena.jpg