**Asignatura:**

Procesamiento Digital de Imágenes

**Profesor:**

Dr. Modesto Guadalupe Medina Melendrez

**Nombre del Estudiante:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Nombre de la Práctica:**

Práctica 1: Captura, grabación y procesamiento básico de imágenes.

**Fecha de entrega:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Competencias:**

El estudiante deberá desarrollar las siguientes competencias específicas:

* Comprender la importancia y ubicación del procesamiento de imágenes como una etapa de pre-procesamiento requerida para resolver diversos problemas de aplicación práctica.
* Adquirir imágenes digitales desde una cámara, almacenarlas y representarlas como arreglos de pixeles a color o en escalas de grises que permitan su posterior procesamiento.
* Utilizar una herramienta de programación para implementar y emplear algoritmos que permitan realizar el procesamiento de imágenes.

El estudiante fortalecerá las siguientes competencias genéricas que son acordes al perfil de egreso:

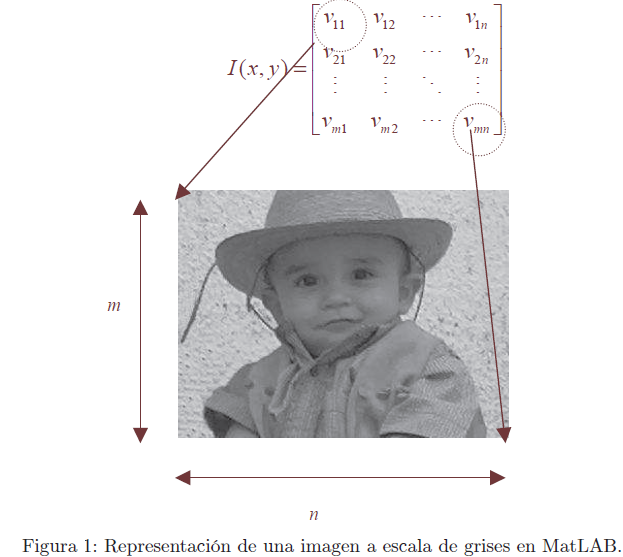
* Conocimiento sobre el área de estudio y la profesión.
* Capacidad de aplicar los conocimientos en la práctica.
* Habilidades en el uso de las tecnologías de la información y de la comunicación.
* Capacidad de trabajo en equipo.
* Capacidad para organizar y planificar el tiempo.
* Capacidad de comunicación oral y escrita.

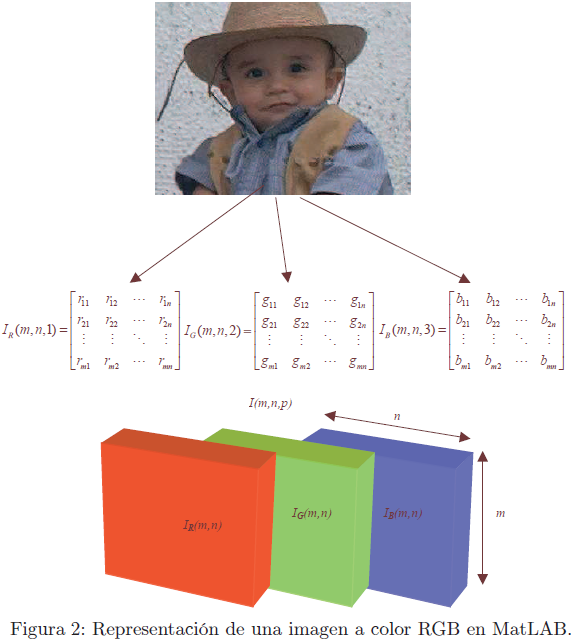
**Introducción:**

La implementación de algoritmos de análisis de imágenes digitales resulta muy costoso en tiempo ya que se requiere de la manipulación de punteros, gestión de memoria, etc. Hacerlo en lenguaje C++ (que por sus características compartidas de alto y bajo nivel lo hacen el más apropiado para la implementación de algoritmos de visión computacional) supondría la inversión de tiempo y sin la seguridad de que lo queremos implementar funcionará. Además utilizar C++ para el periodo de prueba exige un tiempo normal de corrección de errores debidos al proceso de implementación del algoritmo, es decir errores programáticos efectuados por ejemplo al momento de multiplicar dos matrices, etc. Todos estos problemas pueden ser resueltos si la implementación de prueba es realizada en MatLAB utilizando su toolbox de procesamiento de imágenes, con ello el tiempo de implementación se convierte en el mínimo con la confianza de utilizar algoritmos científicamente probados y robustos.

El toolbox de procesamiento de imágenes contiene un conjunto de funciones de los algoritmos más conocidos para trabajar con imágenes binarias, trasformaciones geométricas, morfología y manipulación de color que junto con las funciones ya integradas en matlab permite realizar análisis y trasformaciones de imágenes en el dominio de la frecuencia (trasformada de Fourier y Wavelets). Aunque para utilizar las funciones anteriores es necesario comprender como está constituida una imagen que pueda ser procesada.

En matlab una imagen a escala de grises es representada por medio de una matriz bidimensional de m x n elementos en donde n representa el número de píxeles de ancho y m el número de píxeles de largo. El elemento v11 corresponde al elemento de la esquina superior izquierda (ver figura 1), donde cada elemento de la matriz de la imagen tiene un valor de 0 (negro) a 255 (blanco). Por otro lado una imagen de color RGB (la más usada para la visión computacional, además de ser para matlab la opción default) es representada por una matriz tridimensional m x n x p, donde m y n tienen la misma significación que para el caso de las imágenes de escala de grises mientras p representa el plano, que para RGB que puede ser 1 para el rojo, 2 para el verde y 3 para el azul. La figura 2 muestra detalles de estos conceptos.





En ésta práctica se tratan conceptos básicos de las imágenes y como pueden ser adquiridas, manipuladas y almacenadas empleando MATLAB. Adicionalmente se abordan instrucciones para la conversión entre formatos de color de imágenes y se analizan los resultados.

**Materiales y equipos:**

Se requieren los siguientes equipos y programas.

Equipo:

* PC.
* Cámara.
* Tarjeta de adquisición de imágenes y aditamentos.

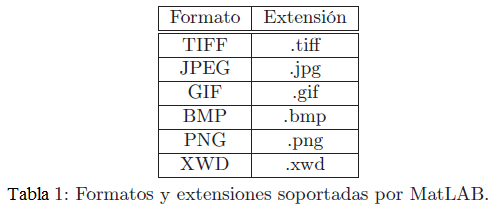
Programas:

Matlab versión 2011b en adelante con el Toolbox Digital Image Processing.

**Marco Teórico:**

Procedimiento para lectura y escritura de imágenes

Para leer imágenes contenidas en un archivo al ambiente de matlab se utiliza la función imread, cuya sintaxis es imread(’nombre del archivo’), donde nombre del archivo es una cadena de caracteres conteniendo el nombre completo de la imagen con su respectiva extensión, los formatos de imágenes que soporta matlab son los mostrados en la tabla 1.



Si se quisiera introducir la imagen contenida en el archivo data.jpg a una variable para su procesamiento en matlab, entonces se tendría que escribir en línea de comandos:

>> image=imread(’data.jpg’);

Con ello la imagen contenida en el archivo data.jpg quedará contenida en la variable image. Una vez que la imagen está contenida en una variable de matlab es posible utilizar las funciones para procesar la imagen.

Para grabar el contenido de una imagen en un archivo se utiliza la función imwrite(variable,’nombre del archivo’), en donde variable representa la variable que contiene a la imagen y nombre del archivo el nombre del archivo con su respectiva extensión de acuerdo a la tabla 1. Suponiendo que la variable image2 contiene la imagen que nos interesa grabar en el archivo dato2.jpg tendríamos que escribir:

>>imwrite(image2, ’data2.jpg’) ;

Después que realizamos un procesamiento con la imagen, es necesario desplegar el resultado obtenido, las funciones imshow(variable) e image(variable) permiten desplegar la imagen en una ventana en el ambiente de trabajo de matlab. Si la variable a desplegar por ejemplo, es face al escribir en la línea de comandos:

>>imshow(face);

o

>>image(face);

Procedimiento para capturar imágenes provenientes de una WEBCAM.

Otra opción es capturar las imágenes directamente desde la cámara para inmediatamente después procesarlas. A continuación se describe el procedimiento para realizar la lectura de imágenes de una webcam.

Teclear “imaqhinfo” en la ventana de comandos para ver información de los adaptores de cámaras instaladas

>> imaqhwinfo

ans =

InstalledAdaptors: {'matrox' 'winvideo'}

MATLABVersion: '7.13 (R2011b)'

ToolboxName: 'Image Acquisition Toolbox'

ToolboxVersion: '4.2 (R2011b)'

Notar el adaptador “winvideo” que es el más utilizado, de tal manera que cualquier webcam será accesada a través de este adaptador. Si la webcam que se quiere accesar es externa, también se podrá accesar si es conectada a través de un puerto USB. Por lo que el primer paso será conectar la webcam a través del puerto USB.

Existen varios tipos de formatos de imágenes que pueden entregar las cámaras (algunas en formato RBG, YUY, entre otras), por lo que es conveniente conocer los formatos soportados por la webcam conectada. Para esto, teclear:

>> dev\_info = imaqhwinfo('winvideo', 1);

>> char(dev\_info.SupportedFormats)

ans =

YUY2\_1024x768

YUY2\_1280x1024

YUY2\_1280x720

YUY2\_1600x1200

YUY2\_160x120

YUY2\_176x144

YUY2\_320x240

YUY2\_352x288

YUY2\_640x480

YUY2\_800x600

Para este ejemplo en particular, se puede observar que todos son formato YUY con diferentes resoluciones. Se debe seleccionar uno de ellos para generar el objeto de entrada de video. La generación de tal objeto se realiza tecleando lo siguiente:

>> vid = videoinput('winvideo',1,'YUY2\_1600x1200');

Se puede observar una previsualización en vivo de las imágenes obtenidas por la webcam tecleando

>> preview(vid);

El siguiente paso sería capturar imágenes o snapshots de la webcam, para esto se teclea

>> A = getsnapshot(vid);

Este comando almacena la imagen capturada en la variable A. En este ejemplo, la imagen A tiene un tamaño 1600x1200x3 de la imagen en el formato YUY, por lo que para desplegarla o utilizarla en un algoritmo de procesamiento de señales tendría que convertirse al formato RGB. La conversión de YUY a RGB se realiza tecleando:

>> B=ycbcr2rgb(A);

El resultado puede visualizarse tecleando:

>> image(B);

o

>> imshow(B);

Procesamiento básico de imágenes

El acceso a píxel de una imagen es una de las operaciones más comunes en visión computacional y en matlab está sumamente simplificado; solo bastará con indexar el píxel de interés en la estructura de la imagen. Consideremos que tenemos una imagen image1 en escala de grises (Figura 3) y deseamos obtener su valor de intensidad en el píxel especificado por m=100 y n=100; solo tendríamos que escribir

> >image1(100,100)

ans =

84

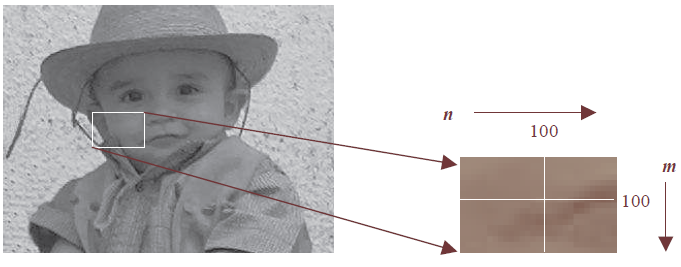


Figura 3: Imagen mostrada al utilizar la función imshow.

De igual forma si se desea cambiar el valor de este píxel a negro, es decir asignarle el valor de 0 lo que tendría que escribirse en línea de comandos es:

>> image1(100,100)=0;

En el caso de imágenes a escala de grises estas solo tienen un plano, constituido por la matriz m x n que contiene los valores de intensidad para cada índice. Sin embargo las imágenes de color cuentan con más de un plano. En el caso de imágenes RGB (tal como se explicó previamente) estas cuentan con 3 planos uno para cada color que representa. Consideremos ahora que la imagen RGB contenida en la variable image2 es la mostrada en la figura 4, y deseamos obtener cada uno de los planos que la componen. Entonces escribiríamos:

> >planeR=image2( :, :,1) ;

> >planeG=image2( :, :,2) ;

> >planeB=image2( :, :,3) ;

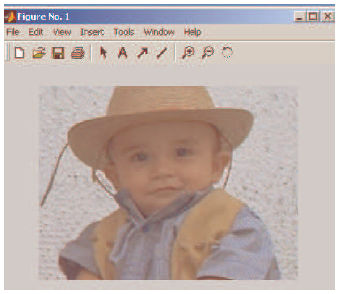


Figura 4. Imagen mostrada al utilizar la función imshow.

Los planos resultantes por los anteriores comandos son mostrados en la figura 5.

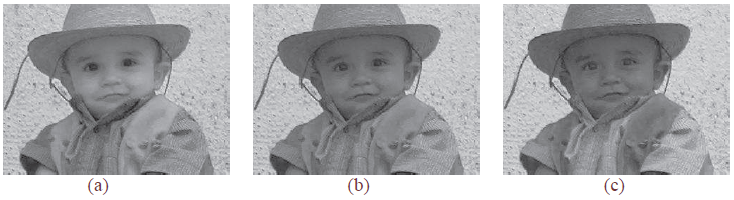


Figura 5. Planos de la imagen a) rojo, b) verde y c) azul.

Si se deseará manipular un píxel de una imagen a color RGB este tendrá un valor para cada uno de sus planos correspondientes. Supongamos que tenemos la imagen RGB contenida en la variable image2 y deseamos obtener el valor del píxel m=100 y n=100 para cada uno de los diferentes planos R, G y B.

Tendríamos que escribir:

>> valueR=image2(100,100,1) ;

>> valueG=image2(100,100,2) ;

>> valueB=image2(100,100,3) ;

Lo cual dará como resultado una tripleta de valores. De igual forma que con el caso de escala de grises podemos modificar este píxel a otro color mediante el cambio de su valor en cada uno de sus respectivos planos; por ejemplo un cambio a color blanco mediante:

>> image2(100,100,1)=255;

>> image2(100,100,2)=255;

>> image2(100,100,3)=255;

En ocasiones resulta preferible saber el color o la intensidad de gris (el valor del píxel) de forma iteractiva, es decir tener la posibilidad de seleccionar un píxel en una región y obtener el valor de este. Esta posibilidad es ofrecida por la función impixel, la cual iterativamente entrega el valor (uno o tres) del píxel seleccionado que aparezca en la ventana desplegada por la función imshow. El formato de esta función es:

value=impixel;

donde value representa un escalar, en el caso de que la imagen sea a escala de grises o bien un vector de 1 x 3 con los valores correspondientes a cada uno de los planos RGB.

Para utilizar esta función es necesario antes, desplegar la imagen con la función imshow. Una vez desplegada se llama a la función y cuando el cursor del ratón este sobre la superficie de la imagen cambiara a una +. Cuando se presione el botón izquierdo del ratón se seleccionara el píxel, el cual podemos seleccionar otra vez en caso de que se allá cometido un error a la hora de posicionar el ratón, ya que la función seguirá activada hasta que se presione la tecla de enter.

Una operación importante en el procesamiento de imágenes es el determinar un perfil de la imagen; es decir convertir un segmento de la imagen a una señal unidimensional para analizar sus cambios. Matlab dispone de la función improfile que permite trazar el segmento interactivamente con el ratón, desplegando después el perfil de la imagen en una gráfica diferente. Esta función necesita que la imagen original sea previamente desplegada mediante la función imshow. Debe de considerarse que si la imagen es a escala de grises, el perfil mostrara solo una señal correspondiente a las fluctuaciones de las intensidades de la imagen, sin embargo si la señal es de color RGB esta mostrará un segmento de señal para cada plano. Para la utilización de esta función solo es necesario escribir en línea de comandos

>> improfile

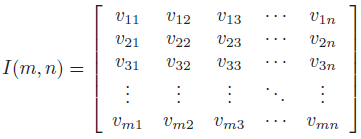
como es una función iterativa en cuanto el ratón se encuentra en la superficie de la imagen el puntero cambiara de símbolo a una +, de esta manera podemos mediante el establecimiento de una línea en la imagen configurar el perfil deseado. La figura 7 muestra una imagen de la operación descrita.



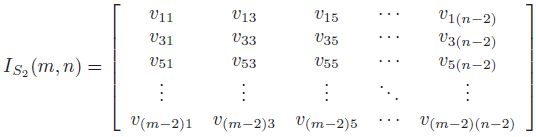
Figura 6: Utilización de la función improfile.

En ocasiones es necesario hacer cálculos que requieren procesar por completo la imagen, en estos casos hacerlo sobre la resolución original de la imagen sería muy costoso. Una alternativa más eficiente, resulta el sub-muestreo de la imagen.

Sub-muestreo significa generar una imagen a partir de tomar muestras periódicas de la imagen original, de tal forma que esta quede más pequeña. Si se considera la imagen I(m,n) definida como:

 (1)

y se desea sub-muestrear la imagen para obtener la mitad de su tamaño original, lo que se puede hacer es generar una nueva imagen compuesta por los elementos tomando uno si y otro no de la imagen original:

 (2)

Considerando que se tiene la imagen RGB en la variable image2 y la Submuestreamos a la mitad tendríamos que escribir el siguiente código en la línea de comandos:

>> imageSub2=image2(1:2:end,1:2:end,:);

Conversión entre modelos de color.

El formato de representación de color ofrecido por las imágenes RGB resulta no apropiado para aplicaciones en las cuales el cambio de iluminación es problema. Matlab dispone de funciones especiales para realizar cambios entre modelos de color.

En ciertas aplicaciones lo que se requiere son imágenes en escalas de grises. Por lo que Matlab proporciona la función rgb2gray que se utiliza para cambiar una imagen en formato RGB a escala de grises, el formato de dicha función es:

imagegray =rgb2gray(imageRGB);

Otro formato de interés, principalmente cuando se trata de transmisión por televisión en el formato NTSC, es el modelo YIQ. Al igual que el modelo RGB, el modelo NTSC está compuesto de tres planos correspondientes al Y, I y Q. Matlab provee el par de funciones rgb2ntsc y ntsc2rgb para realizar las conversiones entre los formatos RGB y NTSC. Los formatos de tales funciones son:

imageNTSC=rgb2ntsc(imageRGB); convierte de RGB a NTSC

imageRGB=ntsc2rgb(imageNTSC); convierte de NTSC a RGB

Por su parte las funciones rgb2hsv y hsv2rgb realizan cambios entre los modelos de color RGB y HSV. Al igual que el modelo RGB, el modelo HSV está compuesto de tres planos correspondientes al H, S y V. El formato de estas funciones es:

imageHSV=rgb2hsv(imageRGB);

imageRGB=hsv2rgb(imageHSV);

Otras opciones de modelos de color estandarizados muy utilizados son el CIE Lab y el CIE xyz, requiriéndose las funciones siguientes para realizar las conversiones entre el modelo RGB y estos modelos de color:

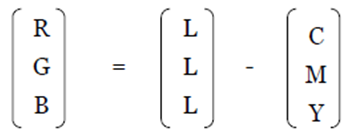
imageLab=rgb2lab(imageRGB);

imageRGB=lab2rgb(imageLab);

imagexyz=rgb2xyz (imageRGB);

imageRGB=xyz2rgb(imagexyz);

También se puede requerir la conversión de una imagen en formato RGB a un formato CMY que es el que utilizan las impresoras. Aunque para esta conversión no existe una función de Matlab para su implementación, se puede generar una función que implemente la siguiente conversión:



**Procedimiento:**

Se realizarán algunas operaciones de captura, lectura, almacenamiento, procesamiento básico y conversión de modelos de color de imágenes. El procedimiento a seguir se lista a continuación:

1. Leer y analizar el contenido del marco teórico.

2. Leer alguna imagen guardada en la PC y desplegarla en Matlab.

3. Sub-muestrear la imagen original para reducir a una novena parte su tamaño original.

4. Guardar en la PC la imagen resultante en dos formatos diferentes (jpg y bmp) y comparar los requerimientos de almacenamiento.

5. Leer alguna imagen desde la webcam de la computadora y desplegarla en Matlab.

6. Agregar franjas verticales blancas de 100 pixeles de ancho cada 200 pixeles horizontales a la imagen original y desplegar la imagen resultante.

7. Leer alguna imagen del medio de su elección (proveniente de la PC o de la webcam) con alto contenido de cromaticidad (gran variedad de colores).

8. Realizar las siguientes conversiones entre modelos de color, desplegar los resultados (diferentes imágenes para cada componente de los modelos CMY, HSV) y analizar los resultados obtenidos:

1. RGB a escalas de grises.

2. RGB a CMY.

3. RGB a HSV.

En el informe se deberán reportar las imágenes resultantes y conclusiones y/o comentarios de las actividades 4, 6 y 8. Finalmente incluya una sección para conclusiones y/o comentarios generales.

**Recomendaciones:**

* El estudiante deberá seguir las pautas de higiene, conducta y seguridad indicadas en el reglamento interno de los laboratorios.
* Seguir las indicaciones dadas en este documento para el desarrollo de la práctica.

**Observaciones:**

* El proceso de revisión se llevará a cabo solo si se entrega el informe escrito de la práctica.
* La evaluación se realizará a partir de la presentación oral del trabajo y la calidad del reporte entregado.
* En el informe se deberán incluir las siguientes secciones: Portada, Introducción, Competencias a desarrollar, Marco Teórico, Procedimiento y Análisis de resultados (con los resultados obtenidos-figuras y códigos), Conclusiones generales y Fuentes de Información.

**Fuentes de información.**

1. Apuntes sobre Procesamiento Digital de Imágenes.
2. Manuales de Image Processing Toolbox de MATLAB.
3. R.C. Gonzalez, R.E. Woods, Digital Image Processing, Ed. Addison Wesley, 1992.