ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería Eléctrica

"Diseño e Implementación de una Herramienta Didáctica de Software para el Procesamiento Digital Básico de Imágenes para los Estudiantes de la Facultad de Ingeniería en Electricidad y Computación"

Informe de Materia de Graduación

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

ALEX FABIAN GUERRERO ECHEVERRIA

Guayaquil - Ecuador

2010

AGRADECIMIENTO

A Dios por la confianza que deposita todos los días al brindarme la fuerza, el valor y la sabiduría necesaria para culminar la carrera universitaria.

A mi Directora de Tesis MSc. Patricia Chávez, quién fue una guía en cada momento y confió en mí para la realización de este proyecto.

A mis padres, hermanos y a mi hija por el apoyo incondicional a lo largo de la vida universitaria y a mí querida ESPOL que me ha brindado todo.

TRIBUNAL DE SUSTENTACION

Msc. Patricia Chávez	Dr. Xavier Ochoa
Profesora de la Materia	Delegado del Decano

DECLARACIÓN EXPRESA

“La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis de graduación, me corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL”.

(Reglamento de Graduación de la ESPOL).

ALEX FABIAN GUERRERO ECHEVERRÍA

RESUMEN

En el presente documento se explica el manejo y las acciones que realiza la Herramienta Didáctica para el uso estudiantes. El presente software demostrativo básico sirve para la visualización de los efectos sobre una imagen al aplicar las funciones desarrolladas en Matlab R2007a.La herramienta de visualización que permita interactuar con las imágenes al procesarlas y motive a comprender e investigar más aplicaciones que se pueden dar al procesamiento digital de imágenes ha sido dedicada a los estudiantes.

En el capítulo 1 se tratan los conceptos básicos de las imágenes y como son representadas en Matlab así como una introducción a las operaciones que se van a efectuar con la Herramienta Didáctica.

En el capítulo 2 se explica el concepto de filtrado aplicado a las imágenes digitales y se detalla el funcionamiento básico en Matlab de los filtros.

En el capítulo 3 se desarrolla las aplicaciones que tienen en la detección de bordes, y al capítulo 4 como afecta los tipos de ruido a la imagen original y en el capítulo 5 como se obtiene gráficamente la FFT de una imagen.

Se explica el uso de las funciones de Matlab para la implementación de la Herramienta Didáctica utilizada para la captura de imágenes, la cual puede ser a través de dispositivos instalados en la computadora por el puerto USB.

Se encuentra detallado un Manuel de Usuario para que exista una mejor interacción y fácil comprensión al momento de ejecutarla.

INDICE GENERAL

INDICE DE FIGURAS

INTRODUCCION

En el diseño de la Herramienta Didáctica para aplicar el procesamiento de imágenes ingresadas por medio una webcam o agregadas en los formatos establecidos para uso de los estudiantes, se consideró que la implementación se realice un programa que brinde las facilidades para la fácil comprensión y desarrollo. La decisión adoptada fue implementarla en el programa MATLAB (desde la versión 7.4 en adelante) debido a que contiene una Librería de funciones para Procesamiento Digital de Imágenes.

Entre las aplicaciones integradas se encuentra los métodos para filtrado de imágenes, la convolución de funciones, transformada rápida de Fourier, tipos de ruidos que se agregan a las imágenes y la conversión de formatos RGB a CYM.

La Herramienta Didáctica permite realizar una visualización de los diferentes algoritmos agregados como funciones comprobadas en MATLAB para la mejor comprensión de los efectos provocados por la matemática integrada en el tratamiento de las imágenes.

CAPITULO 1

1. CONCEPTOS BASICOS DE IMÁGENES

1.1. Definición de Imágenes en Matlab

Una imagen se define como una función de dos dimensiones, *f(x,y)* donde *x* y *y* son coordenadas espaciales con una amplitud *f* para cada par de coordenadas (x,y), este valor se llama intensidad o nivel de gris de la imagen en ese punto. Cuando los valores de *(x,y)* y la amplitud de *f* son todos finitos y cuantitativos discretos se puede llamar a esa imagen como una imagen digital.

$$f(x,y) = \, Imagen\ en\ dos\ dimensiones$$

Se define en el programa MATLAB a una imagen en 2D en la siguiente clasificación:

- Imagen binaria.- Imagen que contiene pixeles de color blanco y negro representada por una matriz de tipo *unit8* o *double* que contiene 0's y 1's.

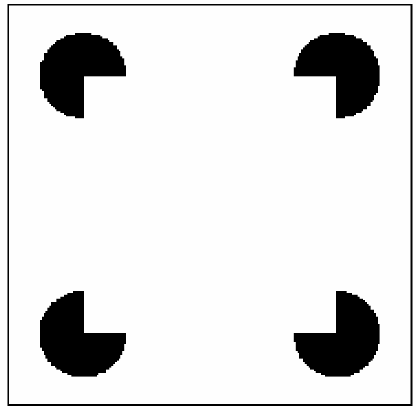


Fig. 1 Ejemplo de imagen binaria

- Imagen en escala de grises.- Imagen que tiene valores de píxeles que corresponden a una escala de grises. Se representa por una matriz de clase *unit8*.

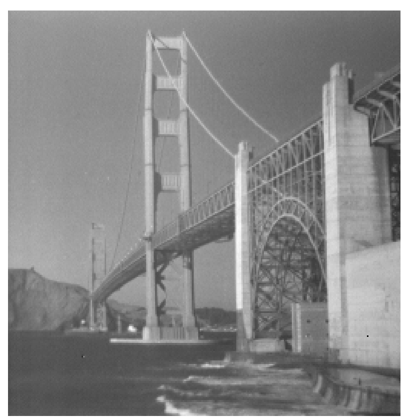


Fig. 2 Ejemplo de imagen en escala gris

- Imagen RGB color.- Imagen cuyos píxeles se especifican por 3 valores, cada uno a su componente de color (rojo, verde y azul) de cada pixel. Se representa por una matriz de mxnx3 ([1]).

Fig. 3 Ejemplo imagen RGB color

La imagen en escala de grises se representa 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 rango de valor que se considera por cada elemento de la matriz es de 0 (negro) a 255 (blanco).

Las imágenes digitales a color los píxeles se cuantifican usando tres componentes independientes uno por cada color primario (**RGB = rojo,**

[1]http://*alojamientos.us.es/gtocoma/pid/tema1-1.ppt*

verde y azul); combinando distintas intensidades de estos tres colores, podemos obtener todos los colores visibles.

Para la herramienta didáctica la imagen que se ingresa por la cámara web se representa por medio de 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, se define el valor de RGB como 1 para el rojo, 2 para el verde y 3 para el azul.

1.2. Conversión de espacios de colores

En algunos casos, son más apropiados modelos diferentes del RGB para algoritmos y aplicaciones específicas. Si se desea otro modelo sólo requiere una conversión matemática simple para obtener el modelo RGB.

En la herramienta didáctica se necesita convertir la imagen original en varios planos, por lo que la fórmula que se aplica es separar a la imagen por planos de RGB definiendo los siguientes:

- Color Rojo (Red) = 1

- Color Verde (Green) = 2
- Color Azul (Blue) = 3

La conversión para cambiar al modelo CMY (Cian-Magenta-Amarillo) es mediante la siguiente fórmula:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} L \\ L \\ L \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

Se aplica conversión de la imagen original al modelo RGB y se muestra cada plano de color aplicando la herramienta didáctica en la figura 4.

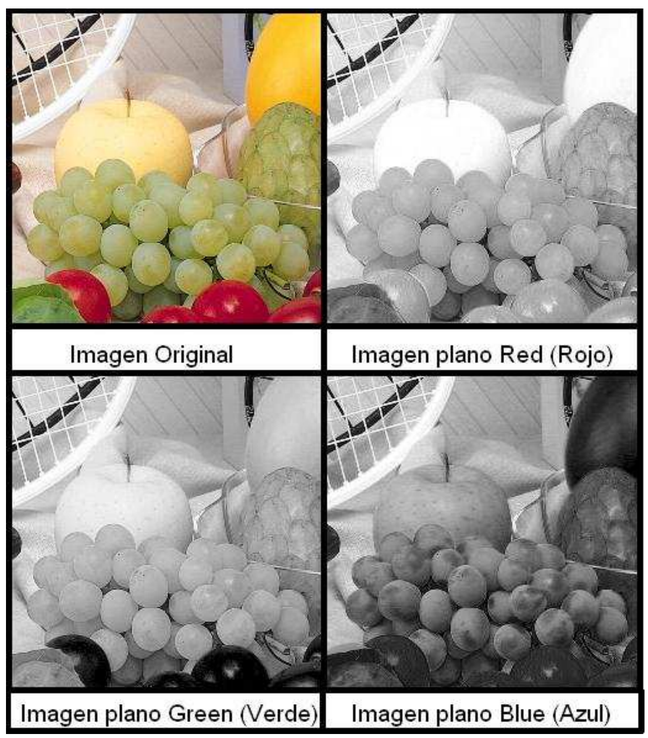


Fig. 4 Planos de imagen original a RGB

Aplicando en la imagen original la fórmula seria:

$$C(m,n) = 1 - \left(\frac{R(m,n)}{255}\right) ; Conversión\ de\ Rojo\ a\ Cian$$

$$M(m,n) = 1 - \left(\frac{G(m,n)}{255}\right) ; Conversión\ de\ Verde\ a\ Magenta$$

$$Y(m,n) = 1 - \left(\frac{B(m,n)}{255}\right);\ Conversión\ de\ Amarillo\ a\ Azul$$

En la figura 5 se aplica la conversión del plano RGB a CMY.

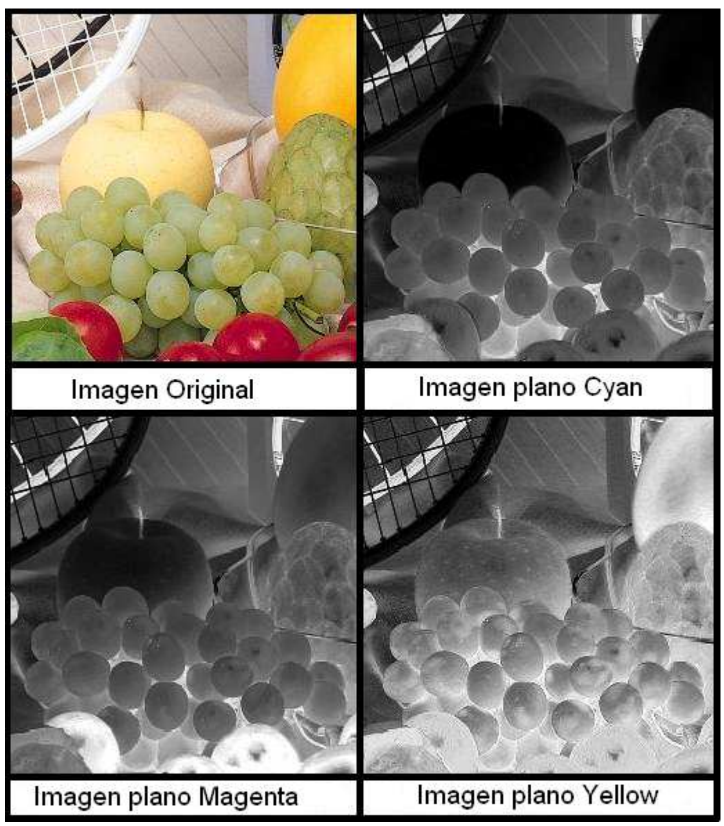


Fig. 5 Plano de Imagen original a CMY

CAPITULO 2

2. FILTRADO DE IMAGENES

El filtrado es parte del proceso que se realiza para restaurar una imagen digital y se efectúa por medio de una computadora digital. El ruido puede presentarse en una imagen por consecuencia del proceso de captura, digitalización y transmisión. Existen distintos tipos de algoritmos que permiten la reducción del ruido razón por la cual se aplica el filtrado a la imagen. En la herramienta didáctica se van a generar filtros pre-definidos en Matlab y se lo va efectuar por medio de la convolución de la imagen original con una máscara predefinida (*fspecial*).

2.1. Filtros espaciales de Imágenes 2D en Matlab

La función en Matlab que permite generar un filtro para aplicarlo sobre una imagen en 2-D es *fspecial*, y contiene filtros predefinidos en 2-D que se detallan a continuación:

- Promedio (Average)

- Disco (Disk)
- Gaussian
- Desenfoque (Unsharp)

En cada uno de los filtros bidimensionales mencionados para que se generen, se invoca a la función *fspecial* y luego se efectúa la convolución entre la imagen original y el filtro seleccionado mediante la función *conv2*. Como resultado final se obtiene la imagen filtrada. Los valores que se seleccionaron en cada uno de los filtros se asumen por default, es decir la función de Matlab va a escogerlo o tiene un valor predeterminado ([2]).

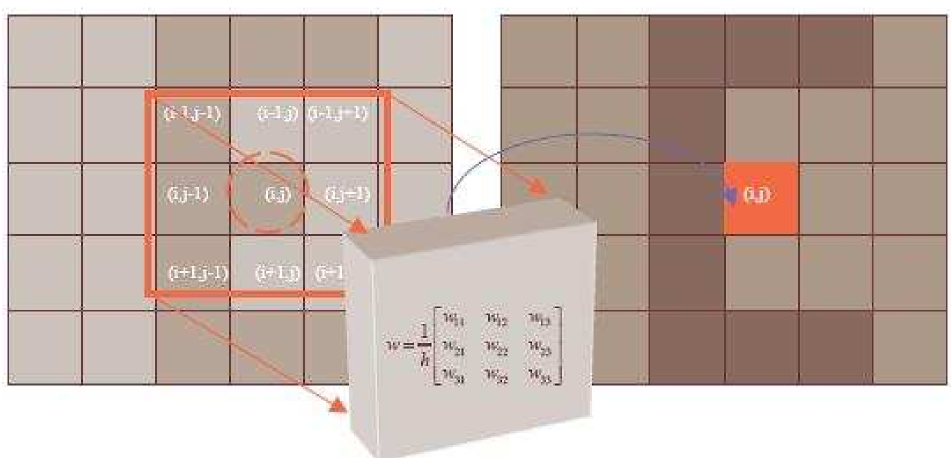


Fig. 6 Máscara espacial generada por fspecial

[2] http://*proton.ucting.udg.mx/tutorial/**vision**/curso**vision**.pdf*

2.1.1. Filtro Promedio (Average)

Al aplicar el filtro promedio se genera una nueva imagen llamada *g(i,j)* cuya intensidad de cada pixel se obtiene promediando los valores de intensidad de la imagen original *f(i,j)* con un entorno de vecindad predefinido (ventana cuadrada o matriz) *w(i,j)* sobre

una imagen digitalizada en dos dimensiones. La operación de filtrado es el producto de la convolución entre la imagen original con el filtro promedio.

$$g(i,j) = w(i,j) * f(i,j)$$

El tamaño del filtro no debe exceder el tamaño de la base de datos. Todos los pixeles son filtrados. Para la herramienta didáctica se definió el parámetro de la ventana de la matriz como un valor por default (3x3).

$$\begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ a7 & a8 & a9 \end{bmatrix} <==== Ventana\ del\ filtro\ 3x3$$

En la figura 7 se muestra el resultado al aplicar el filtro promedio en la imagen original.

Fig. 7 Filtro Promedio

2.1.2. Filtro Disco (Disk)

La función retorna un filtro promedio circular (pillbox) dentro de la matriz cuadrada de lado (2*radio+1). El valor del radio predeterminado es de 5.

Se sigue el esquema del filtro promedio solo que los ahora la máscara es circular.

En la figura 8 se muestra la aplicación del filtro disco en la misma imagen original, note el efecto que causa.

Fig. 8 Filtro Disco

2.1.3. Filtro Gaussiano

El valor de cada punto es el resultado se promediar con distintos pesos de los valores vecinos a ambos lados de dicho punto. Se tiene el problema del difuminado de los bordes. Este tipo de filtro es pasa baja y reduce especialmente el ruido tipo gaussiano. Ruido gaussiano: produce pequeñas variaciones en la imagen.

Para la aplicación en la función *fspecial* se definió el valor de la máscara en una matriz de 3x3 y el valor de sigma = 0,5.

En la figura 9 se muestra la aplicación del gaussiano a la imagen original.

Fig. 9 Filtro Gaussiano

2.1.4. Filtro Desenfoque (Unsharp)

Se usa cuando presenta un desenfoque (difuminado) al captar la imagen por la cámara. Es un filtro pasa alta usado en el mejoramiento de la nitidez y calidad visual de la imagen.

Para la aplicación en la función fspecial el valor de la máscara de desenfoque es una matriz de 3x3 con la negativa de un filtro Laplaciano con un valor de alfa de 0,2 por default.

En la figura 10 se muestra el resultado al aplicar el filtro Unsharp.

Fig. 10 Filtro Desenfoque

CAPITULO 3

3. TIPOS DE RUIDO

Para agregar ruido a la imagen de ingreso en Matlab se considera a la función *imnoise*. En la herramienta didáctica se puede aplicar varias veces los filtros sobre la imagen original *I(x,y)* para poder visualizar el deterioro de la misma. En la sintaxis de comandos de MATLAB definimos a la función *J(x,y)* se define como el resultado de la imagen agregada el ruido. Se detallan los siguientes tipos de Ruido:

- Ruido Gaussiano
- Ruido Poisson
- Ruido Sal & Pimienta (Salt&Pepper)
- Ruido Speckle

3.1. Ruido Gaussiano

El ruido blanco Gaussiano se define como el ruido blanco cuya función de probabilidad conjunta es gaussiana, por lo que tiene media nula y no hay correlación en el tiempo.

El ruido es únicamente el que presenta una distribución de Gauss y es independientemente de que exista una correlación del ruido en el tiempo o no.

Se define ruido blanco como un proceso estocástico que presenta media nula, varianza constante y covarianza nula y si además la distribución es normal, se denomina Ruido Blanco Gaussiano.

La ecuación de la Distribución normal es la siguiente:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{\mu - x}{\sigma}\right)2}$$

El nivel de de gris de todos los pixeles de la imagen es perturbado con una función normal.

En la herramienta didáctica se asume para la función de ruido Gaussiano que el valor predeterminado es cero ruido media *m(μ)* y con una v(σ) *de valor 0,01* de varianza, donde I es la imagen a la cual se va a añadir el ruido.

Los comandos que se usan para invocar a la función son:

$$J = imnoise(I,' gaussian', m, v);$$

Por lo que la función de imnoise se reduce a:

$$J = imnoise(I,' gaussian', 0{,}0.01);$$

En la figura 11 se muestra el resultado al agregar ruido a la imagen original hasta tres veces.

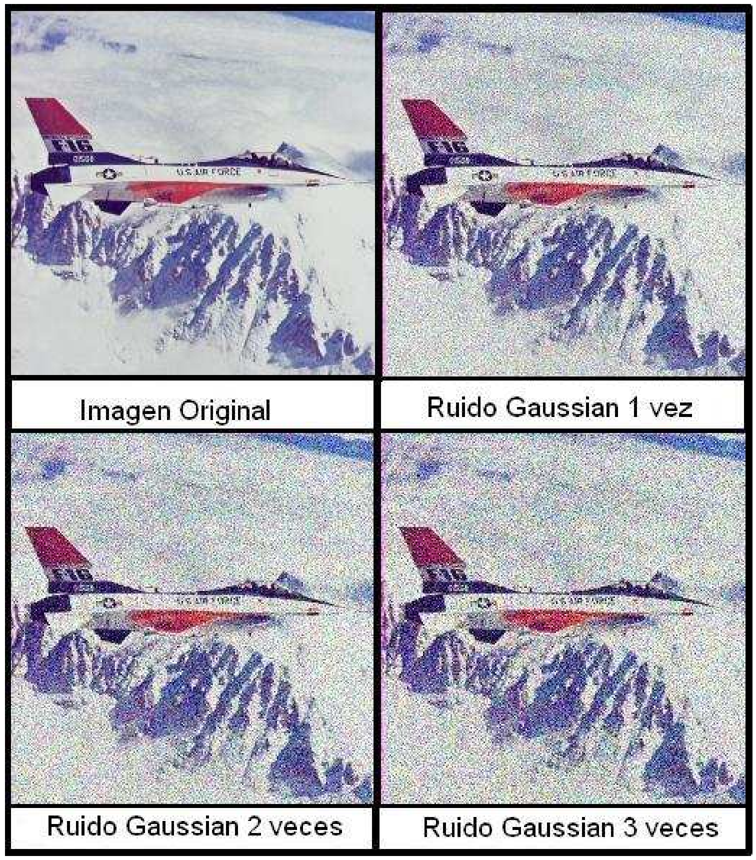


Fig. 11 Aplicación de Ruido Gaussiano

3.2. Ruido Poisson

La Distribución de Poisson está definida por la siguiente fórmula:

$$f(k;\lambda) = \frac{e^{-\lambda}\lambda^{k}}{k!}$$

En la función se genera el ruido de Poisson de los datos de la imagen de ingreso. Si es de doble precisión, entonces los valores de píxel de entrada se interpreta como medio de distribución de Poisson ampliados por 1e12.

$$P_x = e^{-\mu}\left(\frac{\mu^x}{x!}\right)$$

Por ejemplo, si un píxel de entrada tiene el valor de 5.5E-12, entonces el píxel de salida correspondiente se generan de una Distribución de Poisson, con una media de 5,5 y luego reducido por 1e12.

Si es de simple precisión, el factor de escala utilizada es 1E6. Si es uint8 o UInt16, a continuación, los valores de píxel de entrada se utilizan directamente, sin escala.

Por ejemplo, si un píxel de entrada un uint8 tiene el valor 10, entonces el píxel de salida correspondiente se generan de una Distribución de Poisson con 10 media.

$$J = imnoise(I,'poisson');$$

En la figura 12 se muestra como se agrega ruido a la señal.

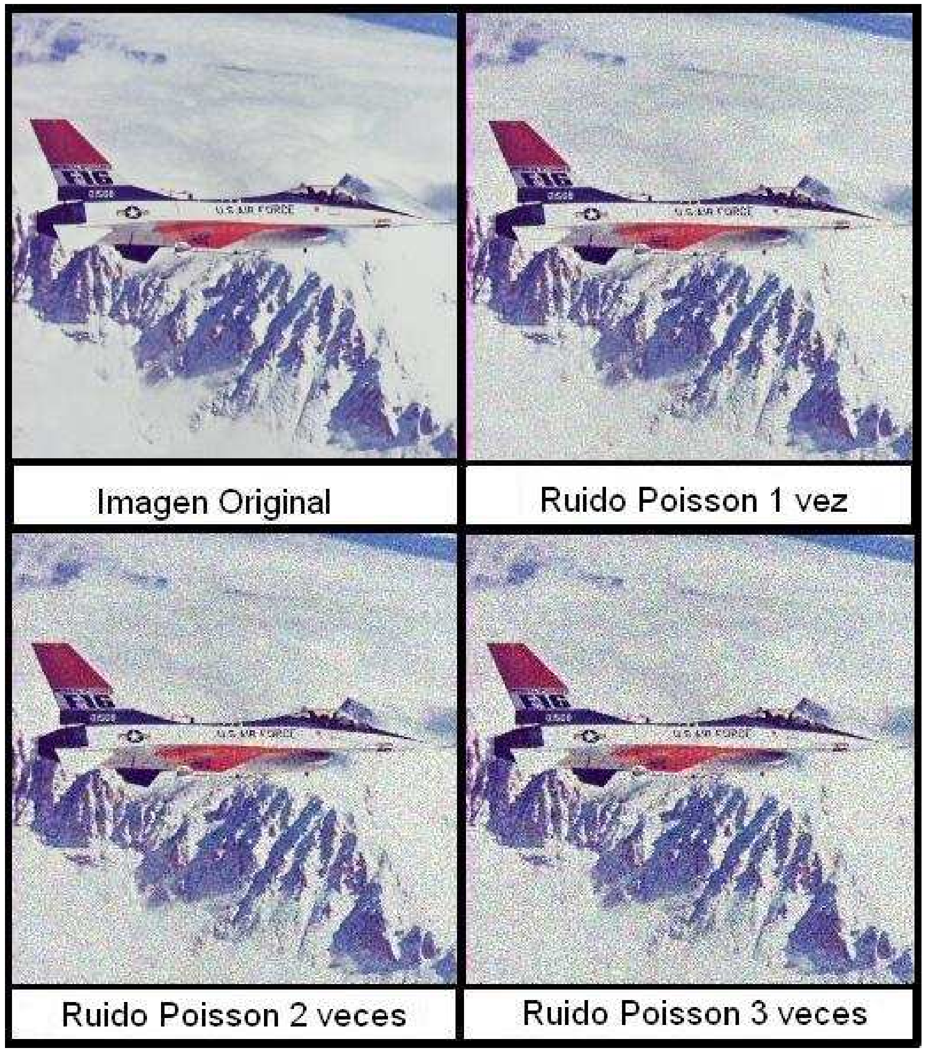


Fig. 12 Aplicación de Ruido Poisson

3.3. Ruido Sal & Pimienta (Salt&Pepper)

Se define como ocurrencias aleatorias de pixeles completamente blancos y completamente negros. Añade el ruido a la imagen donde d es la densidad del ruido. Se afecta aproximadamente a los d*num(I) pixeles. Y en la función se define por defecto un valor de d=0,05.

$$J = imnoise(I,'\,salt\&pepper', d);$$

Por lo que la función se ejecuta

$$J = imnoise(I,'\,salt\&pepper);$$

En la siguiente figura 13 se muestra como se agrega ruido a la señal.

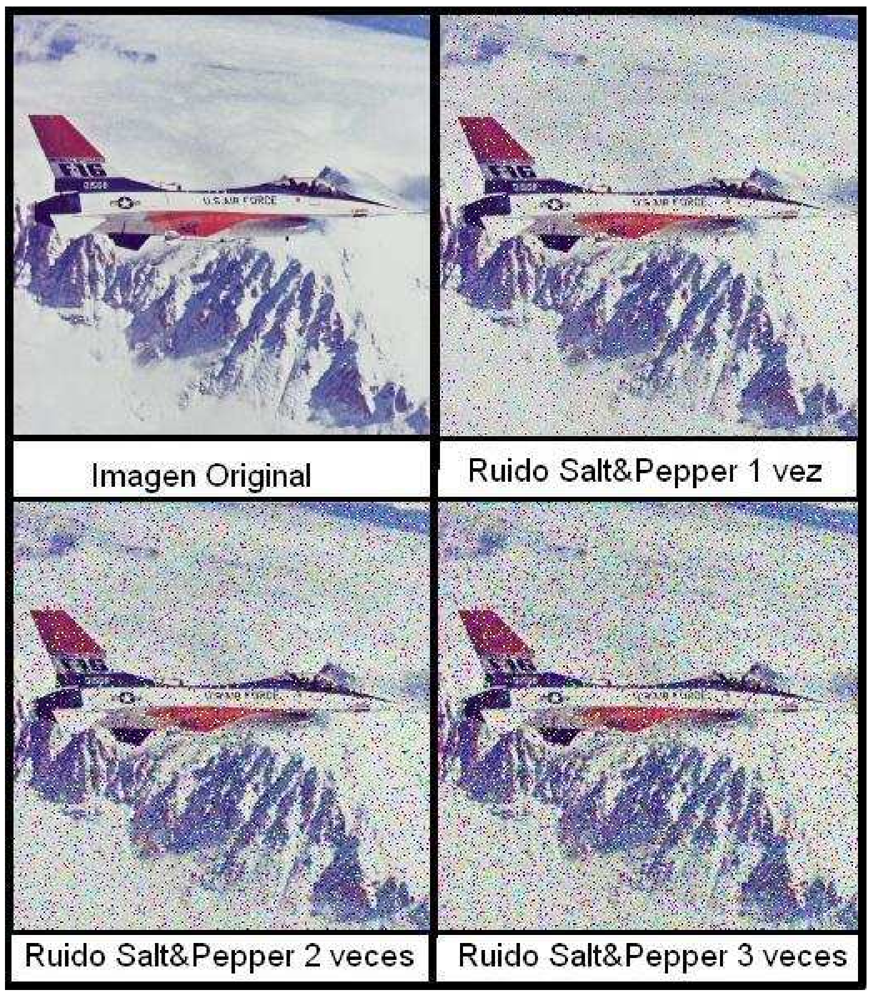


Fig. 13 Aplicación de Ruido Sal & Pimienta

3.4. Ruido Speckle

Se añade el ruido a la multiplicación de la imagen por medio de la siguiente ecuación:

$$J = I + n * I$$

Donde n es de distribución uniforme de ruido aleatorio con media $\mu = 0$ y con varianza v. El valor determinado de v = 0,04. Estos valores se definen por defecto en Matlab.

La sintaxis para invocar a la función es:

$$J = imnoise(I,' speckle', v);$$

Por lo que la función se ejecuta es:

$$J = imnoise(I,' speckle):$$

En la figura 14 se muestra como se agrega ruido a la señal.

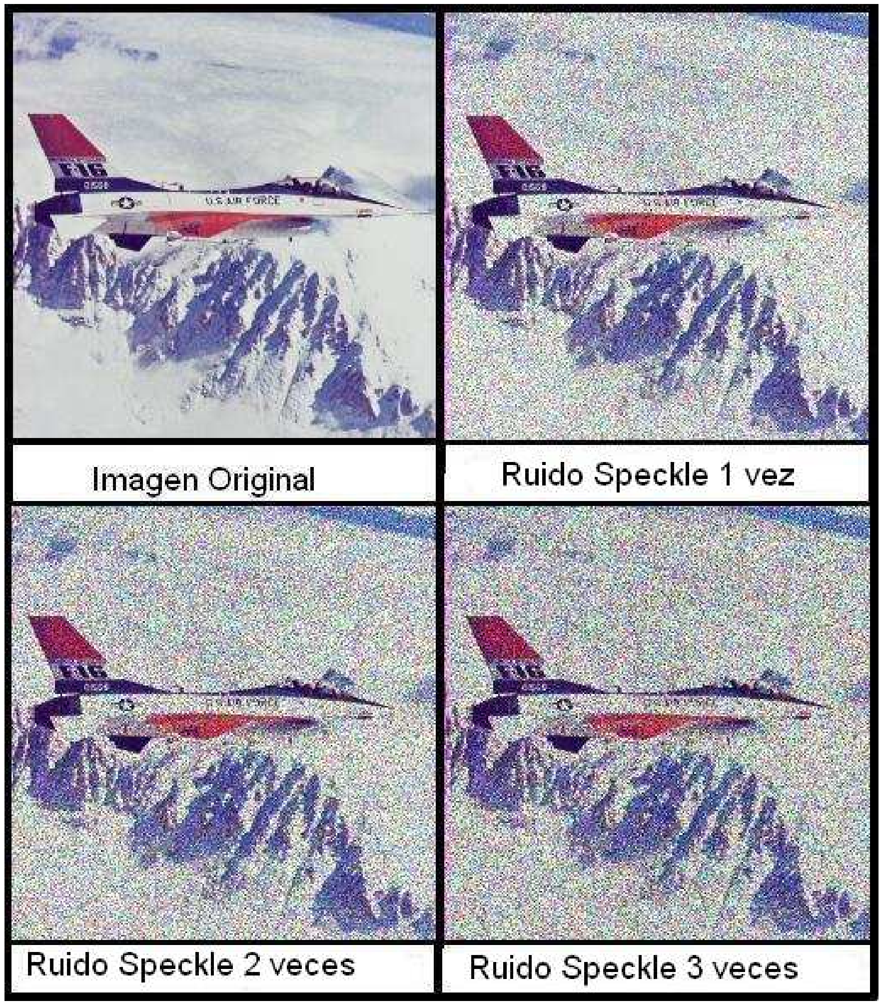


Fig. 14 Aplicación de Ruido Speckle

CAPITULO 4

4. METODOS DE DETECTOR DE BORDES

Los bordes de una imagen digital se definen como transiciones entre dos regiones de niveles de gris significativamente distintos. Al aplicarlos se obtiene una valiosa información sobre las fronteras de los objetos y puede ser utilizada para segmentar la imagen, reconocer objetos, etc.

La mayoría de las técnicas para detectar bordes se basan en distintas aproximaciones discretas de la primera y segunda derivada de los niveles de grises de la imagen.

Para el desarrollo de la herramienta didáctica en Matlab primero se transforma a la imagen original *I(x,y)* a escala de grises por medio de la función *rgb2gray*, este paso se repite para todos los filtros de detector de borde.

I = rgb2gray ('imagen_RGB.jpg');

Donde *I* cambia a una imagen en escala gris y la imagen resultante de cada filtro se la va a definir como *BW(x,y) , Blanco Negro* (Black-White).

La función *rgb2gray* convierte una imagen de RGB en valores de la escala de grises mediante la formación de una suma ponderada de los componentes R, G y B mediante la siguiente fórmula:

$$0.2989 * R + 0.5870 * G + 0.1140 * B$$

Para la detección de bordes en Matlab se considera a la función *edge*, que contiene a los siguientes filtros de detección de Borde:

- Método de Sobel
- Método de Prewitt
- Método de Roberts
- Método de Laplacian of Gaussian (Log)
- Método de Canny

4.1. Método de Sobel

Se define como un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen.

Se encuentra los bordes con la aproximación Sobel a la derivada espacial y retorna el borde en los puntos donde el gradiente I es máximo. Las matrices que se usan para su aproximación son dos kernels de [3x3]; el primero Gx se aplica para los cambios horizontales y el segundo Gy para las verticales mediante la convolución con la imagen original

$$Gx = \begin{bmatrix} +1 & 0 & -1 \\ +2 & +1 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A$$

$$Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 1 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A$$

En cada punto de la imagen, los resultados de las aproximaciones de los gradientes horizontal y vertical pueden ser combinados para obtener la magnitud del gradiente, mediante:

$$G = \sqrt[2]{G_x^2 + G_y^2}$$

Con esta información, podemos calcular también la dirección del gradiente:

$$\theta = arctan\left(\frac{Gy}{Gx}\right)$$

Para generar el filtro se invoca a la función *edge* para el detector de borde *'sobel'* asumiendo los parámetros de umbral (threshold) y la dirección del gradiente (direction) por defecto. Al definirlo de esta manera se considera en horizontal y vertical.

BW = edge (I,'sobel');

Al aplicar el detector Sobel se encuentra los bordes de una imagen de distintos niveles de intensidad y su resultado es una imagen binaria del mismo tamaño que la imagen original en la cual, “1” significa que ha detectado un borde y “0” es que no lo ha detectado.

En la figura 15 se muestra la aplicación de la Detector de borde Sobel.

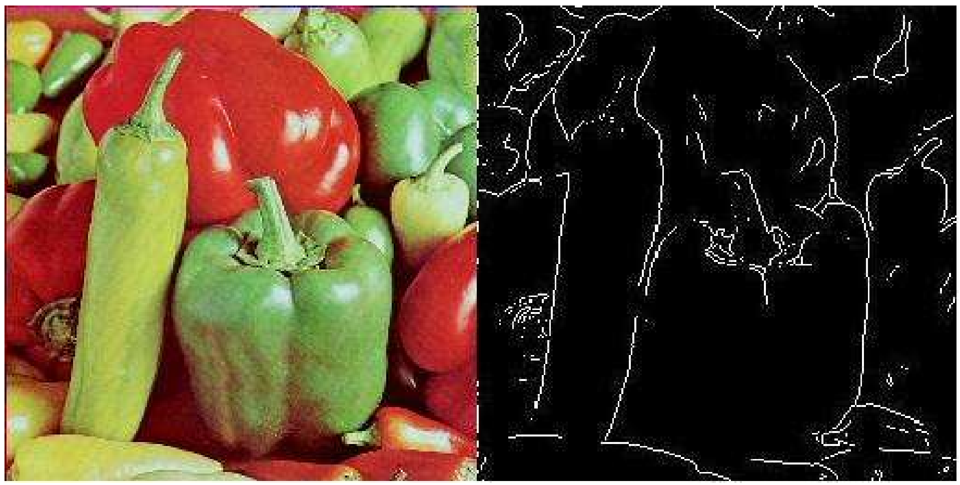


Fig. 15 Detector de borde Sobel

4.2. Método de Prewitt

Se define como la aplicación de 8 matrices pixel a pixel a la imagen. La respuesta es la suma de los bordes bien marcados.

Se marca muy bien los bordes ya que sus matrices actúan desde ocho lados diferentes.

Los nombres de cada matriz se define como un punto cardinal: Norte, Sur, Este, Oeste, Noroeste, Noreste, Suroeste, Sureste.

Las matrices que se usa para su aproximación son dos kernels de [3x3] que se detallan sus valores a continuación:

Gradiente fila = x

1	0	-1
1	0	-1
1	0	-1

Gradiente columna = y

-1	-1	-1
0	1	0
1	1	1

BW = edge(I,'prewitt')

Se invoca a la función *edge* para el detector de borde *'prewitt'* asumiendo los parámetros de umbral (threshold) y la dirección del gradiente (direction) por default. Al definirlo de esta manera se considera en horizontal y vertical.

En la figura 16 se muestra la aplicación de la Detector de borde Prewitt.

Fig. 16 Detector de borde Prewitt

4.3. Método de Roberts

Se define como una de los más antiguos, simples y menos utilizado de los métodos de detección de bordes en la actualidad, a pesar de que su sencillez en la implementación es usado para las aplicaciones hardware de alta velocidad.

Las máscaras utilizadas en este operador para el caso de una matriz de [3x3] es para gradiente fila = Gx

0	0	0
0	0	1
0	-1	0

Y para gradiente columna = Gy

-1	0	0
0	1	0
0	0	0

La respuesta es óptima ante bordes diagonales. El inconveniente de este operador es su extremada sensibilidad al ruido y por tanto se tiene pocas cualidades de detección.

Se invoca a la función *edge* para el detector de borde *'Roberts'* asumiendo los parámetros de umbral (threshold).

BW = edge (I,'roberts');

BW entrega imagen con detección de bordes por Roberts en blanco y negro.

Al aplicar el detector Roberts se encuentra los bordes de una imagen de distintos niveles de intensidad y su resultado se obtiene por medio de la finura del borde que asume el programa.

En la figura 17 se muestra la aplicación del Detector de borde de Roberts

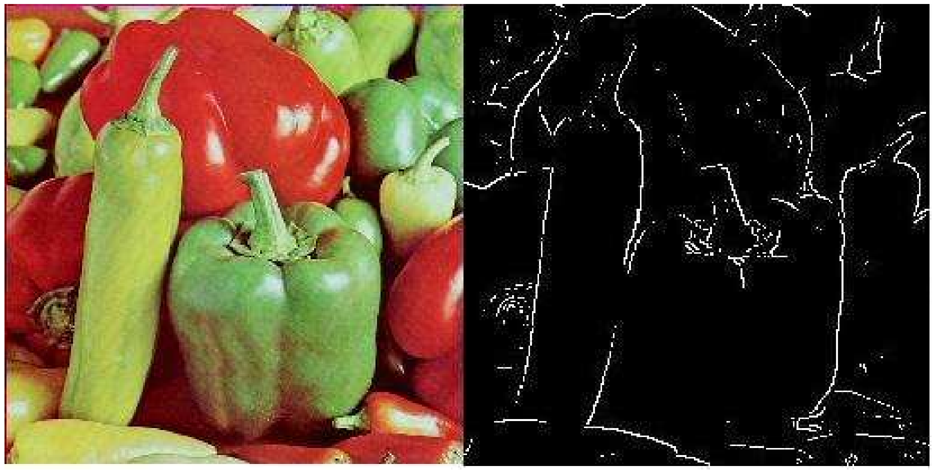


Fig. 17 Detección de borde Roberts

4.4. Método de Laplaciano Gaussiano (Log)

Se define como un filtro de segundo orden porque usa la segunda derivada aplicada al plano y se obtiene mejores resultados en la detección de bordes y en los contornos las curvas de la imagen son más cerradas.

Se lo utiliza por su capacidad de mejorar la nitidez de la imagen. Al ser omnidireccional los bordes que obtenga serán más agudos comparado con otras técnicas.

Se considera como la única desventaja de este filtro la sensibilidad a la presencia de ruido en la imagen, que en algunos casos llega a duplicar los bordes en la imagen final.

BW = edge(I,'log');

BW entrega imagen con detección de bordes por Roberts en blanco y negro.

BW = edge (I, 'log'); especifica el método laplaciano de gaussiana.

La variable de umbral es la sensibilidad del método y sirve para establecer las aristas que no sean más fuertes que dicho valor, en caso de no asignar un valor el programa la elije automáticamente.

BW = edge (I, 'log', umbral)

En la figura 18 se muestra la aplicación del Detector de Borde LoG.

Fig. 18 Detector de borde Laplaciano Gaussiano

4.5. Método de Canny

Se considera para la detección de bordes el uso de la primera derivada, que toma el valor de cero en todas las regiones donde no varía la intensidad y se obtiene un valor constante en toda la transición de la intensidad. Al existir un cambio de intensidad se manifiesta como una alteración brusca en la primera derivada, esta es la característica en la que se basa el algoritmo de Canny.

Para la obtención del gradiente se calcula la magnitud y orientación del vector en cada píxel. La supresión no máxima logra el adelgazamiento del ancho de los bordes obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.

BW = *edge*(I,'canny');

El método de Canny encuentra bordes buscando máximos locales del gradiente de I. El gradiente se calcula utilizando la derivada de un filtro de Gaussiano a la imagen original para que se suavice la imagen. El método utiliza dos umbrales, para detectar los bordes fuertes y débiles, e incluye los bordes débiles en la salida sólo si están conectados a los bordes fuertes. En este método se tiene más probabilidades de detectar ciertos bordes débiles.

En la figura 19 se muestra la aplicación del Detector de Borde Canny.

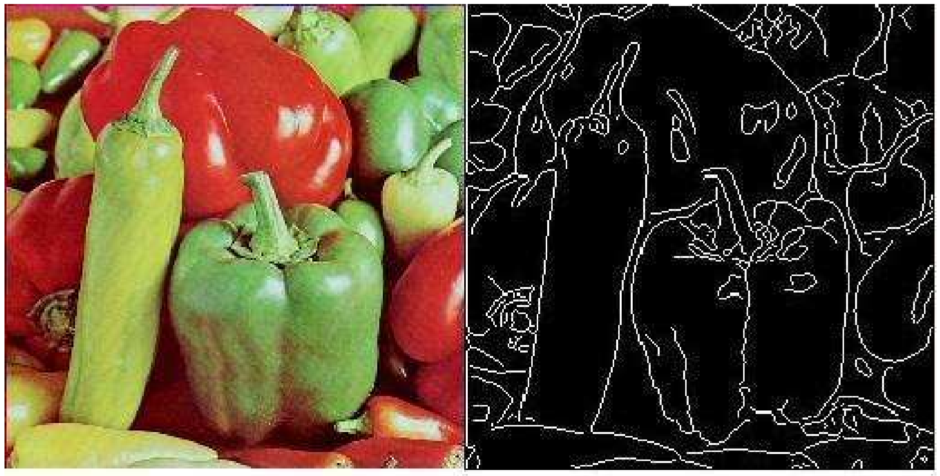


Fig. 19 Detector de borde Canny

CAPITULO 5

5. TRANSFORMADA DE FOURIER

5.1. Definición

Es una operación que transforma una función continua del dominio del tiempo y del dominio espacial en una señal continua en el dominio de la frecuencia.

La imagen original debe de cumplir con las propiedades de la transformada de Fourier para que pueda observar el espectro de la imagen en el dominio de la frecuencia.

Se la define de la siguiente manera:

$$g(\varepsilon) = \frac{1}{\sqrt{2\pi}} \int_{-\alpha}^{+\alpha} f(x) e^{-i\varepsilon x}\, dx$$

Sus aplicaciones son múltiples en diferentes campos como matemáticas, estadística, ingeniería; en particular la usamos para obtener información que no es posible visualizar o medir en el dominio del tiempo. Por ejemplo:

- Análisis y diseño de filtros en base a la densidad espectral.
- Tratamiento digital de imágenes

5.2. Gráfica de la FFT con Matlab

Para la Herramienta didáctica su uso es bastante sencillo, ya que a la imagen original se aplica las siguientes funciones de Matlab.

Se define el colormap para visualizar el espectro de la grafica y se trabaja la imagen solo en un plano, es decir solo en color Rojo.

Se efectuó a la imagen original en un solo plano de colores la Transformada discreta de Fourier en 2D.

Para visualizar defino en la función colormap el espectro que voy a graficar y se muestra con escala de intensidad de colores.

En la figura 20 se presenta gráficamente la FFT aplicada a la imagen original.

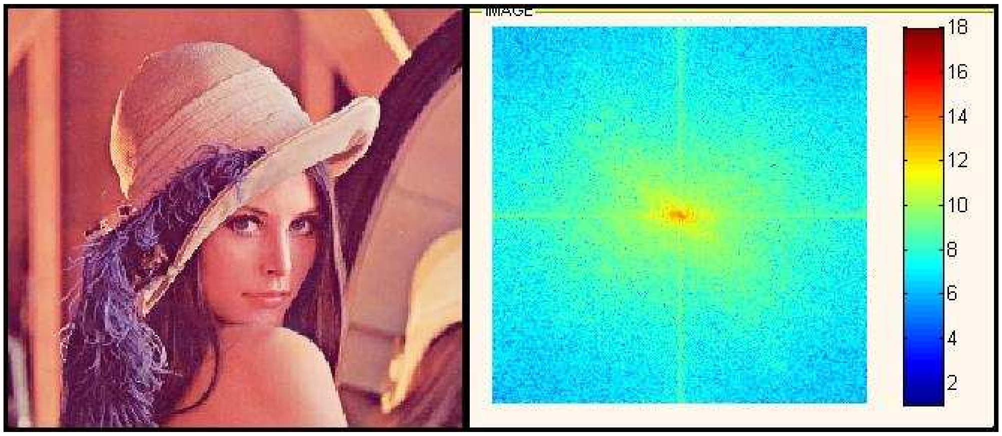


Fig. 20 Función FFT en imagen original

CONCLUSIONES Y RECOMENDACIONES

La interacción gráfica desarrollada en el programa Matlab por medio de implementaciones como la Herramienta didáctica está enfocada a ser una ayuda al estudiante de la facultad de eléctrica para poder visualizar bajo su apreciación el efecto que producen los algoritmos, filtros y transformadas en el procesamiento de imágenes. La facilidad del acceso al uso de la computadora en la facultad de eléctrica promueve al estudiante más tiempo para el desarrollo de aplicaciones en base a funciones de la Librería de Procesamiento de Imágenes y este efecto causa más integración entre el aprendizaje teórico y el software de desarrollo aplicado.

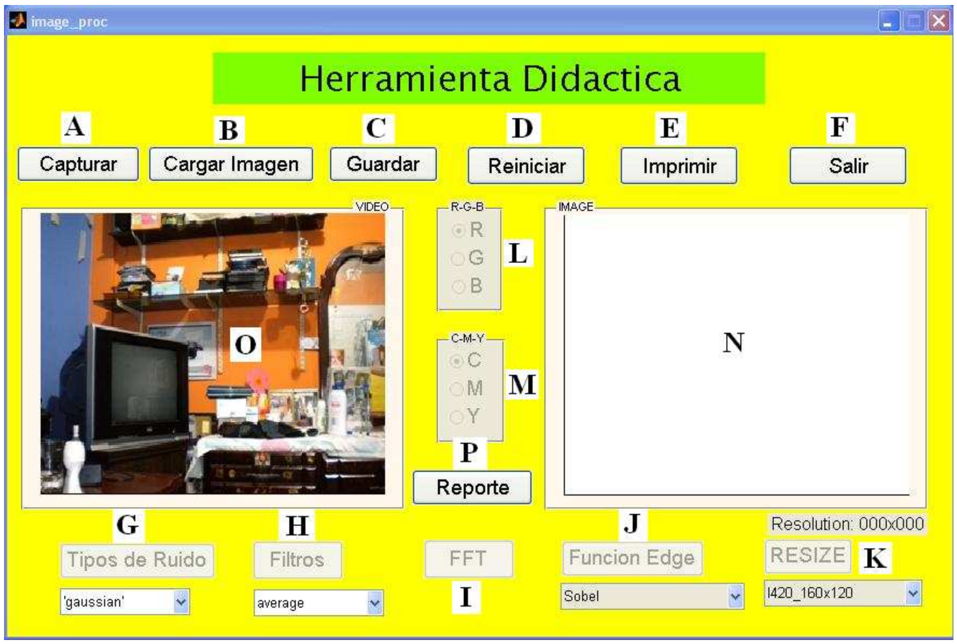
La apreciación visual de cada uno de los procesos efectuados con la herramienta didáctica sobre imágenes es interpretación del estudiante. Y al compartir experiencias con el resto del compañeros sobre las funciones aplicadas motiva al dialogo y reflexión de nuevas ideas para desarrollar e implementar mejoras y optimizaciones.

Se recomienda que la herramienta didáctica se use en las primeras experiencias que tenga el estudiante con el programa Matlab usando la Librería de

Procesamiento de Imágenes para causar un efecto visual en los efectos de la teoría aplicada por medio del software.

ANEXO

ANEXO A: MANUAL DE USUARIO DE HERRAMIENTA DIDACTICA



Descripción de BOTONERAS.

A: Capturar.- Captura la imagen proyectada por la webcam y se coloca en N.

B: Cargar Imagen.- Carga el archivo de una imagen en los siguientes formatos permitidos: (*.jpg), (*.tif) y (*.bmp).

C: Guardar.- Permite guardar la imagen ya procesada por la herramienta didáctica, cada vez que el estudiante requiera efectuarlo.

D: Reiniciar.- Retorna a la imagen original cargada en el inicio.

E: Imprimir.- Permite enviar a la impresora la imagen procesada, en caso de tener instalado el programa para convertir a formato PDF(opcional) sirve de vinculo para efectuarlo.

F: Salir.- Botón para salir de la Herramienta Didáctica.

G: Selecciona los tipos de ruido a usar sobre la imagen original.

H: Selecciona los filtros a usar sobre la imagen original.

I: Realiza la Transformada Rápida de Fourier a la imagen original.

J: Activa la función EDGE para detectar bordes de la imagen original.

K: Cambia la resolución de la nueva imagen que se va a capturar si y solo si por medio de la webcam.

L: Cambia el color a la imagen original en colores RGB (Red, Green, Blue).

M: Cambia el color a la imagen original en colores CMY (Cyan, Magenta, Yellow).

N: Imagen original que va a procesarse con la Herramienta Didáctica.

O: Imagen proyectada en tiempo real de la Webcam.

P: Reporte de las acciones realizadas en el proceso de la imagen original.

Validaciones de las Botoneras

Los botones que se encuentran en la parte superior, enumerados así: A, B, C, D, E, y F funcionan de la misma manera que en el Sistema Operativo Windows, por tal motivo su manejo es sencillo y de conocimiento general. Se encuentra validados los casos que no son congruentes con “mensajes de alerta” que se muestran en el centro de la aplicación cada vez que se cometa un error.

En caso de no contar con cámara web la herramienta didáctica va a mostrar un mensaje de alerta diciendo “No existe cámara conectada”. De ocurrir este caso solo se podrá cargar la imagen pulsando el botón *Cargar Imagen*.

En caso de pulsar P desde el inicio se mostrará el *Reporte* en blanco por el motivo que no se efectuó ningún proceso.

Nota: La función de la Botonera RESIZE es dimensionar la captura de la imagen de acuerdo a las opciones mostradas. Sirve solo si al correr la aplicación esta conectada la webcam.

Método para Cargar la Imagen Original

1. Se captura la imagen de la webcam directamente pulsando el botón *Capturar*. En caso de no tener cámara web conectada a la PC, se muestra un mensaje de alerta.

2. Se puede cargar una imagen pulsando el botón *Cargar Imagen*.
3. Se abre una ventana para ubicar la imagen en la carpeta de origen con formato validado (*.jpg, *.tif y *.bmp), seleccionar y pulsar enter.

4. La imagen se ubicará siempre en el lado derecho en la ubicación N.

Nota: Solo si la imagen original se carga correctamente los botones nombrados desde la letra G, H, I, J, K, L y M se activan. Se considera para las botoneras de procesamiento de la imagen original ya está cargada en la ubicación N de la pantalla para explicar el resto de aplicaciones.

Aplicación de Tipos de Ruido a la Imagen Original

1. Las opciones que ofrece la botonera *Tipo de Ruido* son cuatro: Gaussian, Poisson, Salt&Pepper y Speckle.

2. Seleccione una de las opciones, presione la botonera y observe el cambio de la Imagen Original a una Imagen agregada ruido. Es válido aplicar múltiples opciones de ruido sobre la misma imagen.
3. Si desea volver a la imagen original debe pulsar el botón Reiniciar.

4. Si desea capturar la imagen aplicada con un ruido determinado o varios debe pulsar la botonera *Guardar*, se selecciona la carpeta y coloque el nombre del archivo. Esta opción es válida las veces que requiera el estudiante.

Aplicación de Filtros a la Imagen Original

1. Las opciones que ofrece la botonera *Filtro* son cuatro: Average, Disk, Gaussian y Unsharp.

2. Seleccione una de las opciones, presione la botonera, observe el cambio de la Imagen Original a una Imagen Filtrada. El filtro se aplica solo una vez a la imagen original y se cambia automáticamente en relación al filtro que se elija.

3. Si desea volver a la imagen original debe pulsar el botón Reiniciar.
4. Si desea guardar la imagen aplicada con un filtro determinado debe pulsar la botonera *Guardar*, se selecciona la carpeta y el nombre del archivo. Esta opción es válida las veces que requiera el estudiante.

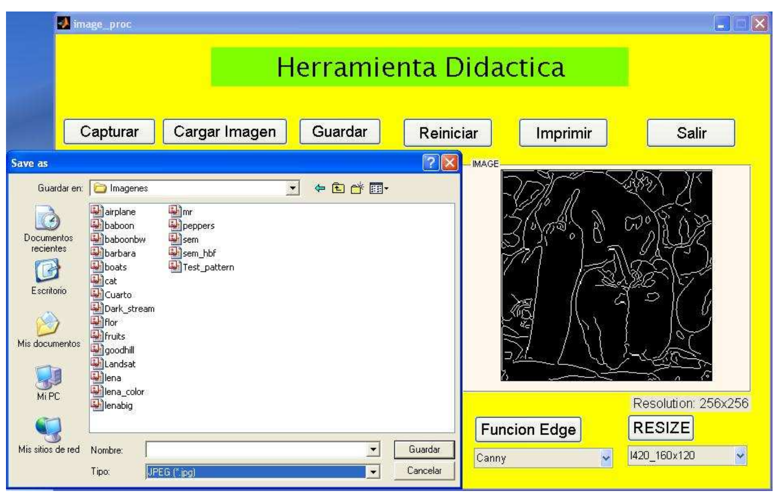
5. Para el ejemplo usamos el Filtro de Unsharp sobre la imagen original.

Aplicación de Detectores de Borde EDGE

1. Las opciones que ofrece la botonera *Edge* son cuatro: Sobel, Prewitt, Roberts, Laplacian of Gaussian y Canny.

2. Seleccione una de las opciones, presione la botonera, observe el cambio de la Imagen Original a la Imagen con la detección de bordes. Se aplica solo una vez a la imagen original y se cambia automáticamente en relación al método escogido.

3. Si desea volver a la imagen original debe pulsar el botón *Reiniciar*.
4. Si desea guardar la imagen aplicada por un método de detector de bordes determinado debe pulsar la botonera *Guardar*, se selecciona la carpeta y el nombre del archivo. Esta opción es válida las veces que requiera el estudiante.



5. Para el ejemplo se uso el método de detección de bordes de Canny sobre la imagen original.

Aplicación de Conversión de Colores

En la parte central de la Herramienta Didáctica se encuentra las botoneras con los tipos de conversiones de colores: RGB y CYM.

1. En el recuadro R-G-B superior central se podrá seleccionar la conversión de la imagen original a R (Red), G (Green) y B (Blue).

2. En el recuadro C-M-Y inferior central se podrá seleccionar la conversión de la imagen original a C (Cyan), M (Magenta) y Y (Yellow).

3. La ubicación inicial de las botoneras están por defecto en R(Red) y en C(Cyan).
4. Seleccione una de las opciones, presione la botonera, observe el cambio de la imagen original en relación al Color seleccionado. Se aplica solo una vez a la imagen original y se cambia automáticamente en relación al método escogido.

5. Si desea volver a la imagen original debe pulsar el botón *Reiniciar*.
6. Si desea guardar la imagen aplicada por Color determinado debe pulsar la botonera *Guardar*, se selecciona la carpeta y el nombre del archivo. Esta opción es válida las veces que requiera el estudiante.

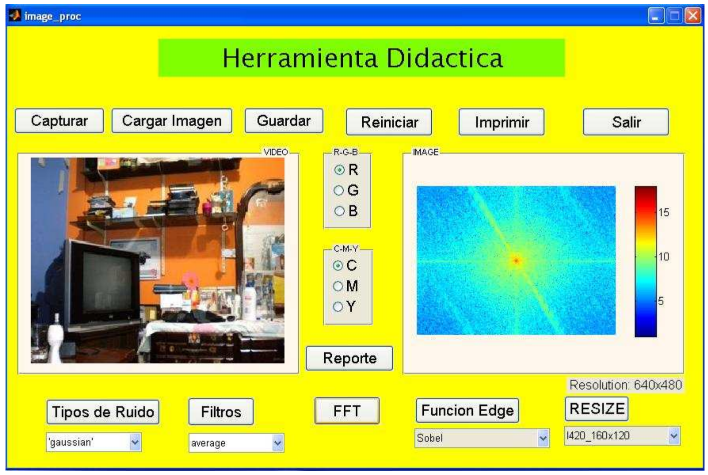
7. Para el ejemplo se uso la conversión de color a Magenta sobre la imagen original.

Aplicación de la Transformada Rápida de Fourier

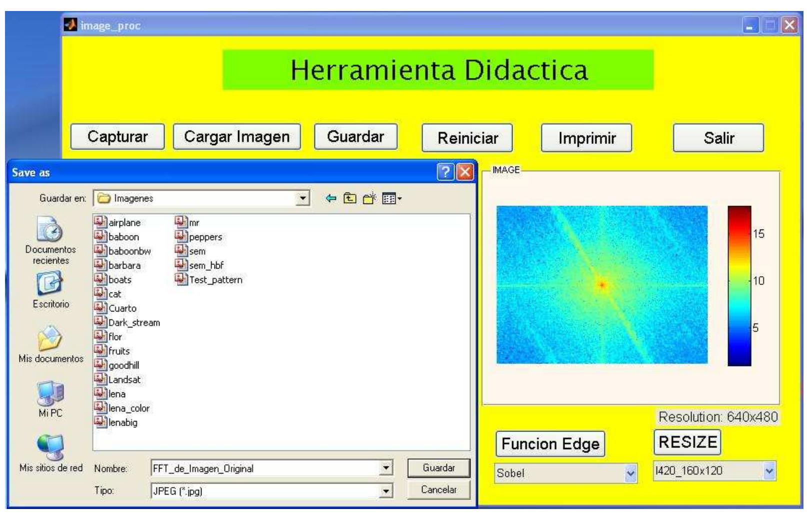
La botonera *FFT* grafica la Transformada Rápida de Fourier de la imagen original.

1. Presione la botonera FFT sobre la imagen original ingresada.

2. Observe de manera grafica la Transformada Rápida de Fourier de la imagen original. En la extrema derecha se encuentra la escala en valores numéricos.



3. Si desea volver a la imagen original debe pulsar el botón *Reiniciar*.
4. Si desea guardar la imagen aplicada por un método de detector de bordes determinado debe pulsar la botonera *Guardar*, se selecciona la carpeta y el nombre del archivo. Esta opción es válida las veces que requiera el estudiante.



Reporte de Procesos

Desde que se inicia la Herramienta didáctica se puede usar la botonera Reporte que su función es mostrar un documento llamado reporte.txt el detalle de las acciones realizadas sobre la imagen original.

1. Si el estudiante desea un reporte de los procesos realizados sobre la imagen original pulsar la botonera Reporte.

2. Se muestra un archivo de nombre reporte.txt que detalla el nombre de las opciones que se utilizo en cada una de las botoneras de manera secuencial.
3. Se guarda el archivo reporte.txt usando el botón Guardar y selecciona la ubicación que va a tener ese archivo.

ANEXO B: CODIGO DE MATLAB

```
% --- Condiciones iniciales del programa.
% Se ejecuta antes de que image-proc se haga visible.
function image_proc_OpeningFcn(hObject, eventdata, handles, varargin)
% Centrado de la GUI en la pantalla de la PC
movegui(hObject,'center');

imaqreset %Se borra el driver de la camara
% Se deshabilita los botones hasta que se abra una imagen o se capture con
% la imagen con la webcam
set(handles.red_s,'Enable','off')
set(handles.green_s,'Enable','off')
set(handles.blue_s,'Enable','off')

set(handles.cyan_b,'Enable','off')
set(handles.magenta_b,'Enable','off')
set(handles.yellow_b,'Enable','off')
%
set(handles.add_noise,'Enable','off')
set(handles.filter_button,'Enable','off')
set(handles.edge_b,'Enable','off') %
set(handles.fft_image,'Enable','off')
set(handles.resize_b,'Enable','off')


% Iniciar objeto de adquisición de video, solo si la cámara USB está
% conectada
if strcmp(get(hObject,'Visible'),'off')
  try
    handles.vidobj = videoinput('winvideo',1,'RGB24_352x288');
    info=imaqhwinfo(handles.vidobj);
    info_a=imaqhwinfo(info.AdaptorName);
    formatos=info_a.DeviceInfo.SupportedFormats;
    set(handles.new_size,'String',char(formatos));
    % Actualiza los handles de la estructura
    start(handles.vidobj);
    guidata(hObject, handles);
    vidRes = get(handles.vidobj, 'VideoResolution');
    nBands = get(handles.vidobj, 'NumberOfBands');
```

```
    hImage = image(zeros(vidRes(2), vidRes(1), nBands), 'Parent',...
               handles.video_cam);
    preview(handles.vidobj,hImage);
  catch
    msgbox('NO HAY CÁMARA CONECTADA. Cargando Profile.jpg.','A V I S O')
    hImage = image(imread('profile.jpg'), 'Parent', handles.video_cam);
    axis off
  end
end
%*-*-*-*-*-*-*-*-*-*-*-*-*-*------------*************------------

% --- Outputs from this function are returned to the command line.
function varargout = image_proc_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;



% --- Obtener imagen de la webcam.
function capture_img_Callback(hObject, eventdata, handles)
try %Si la cámara está conectada
    handles.rgb = getsnapshot(handles.vidobj);
catch %Cargar imagen si no está conectada
    handles.rgb = imread('profile.jpg');
end

% ------Calcular resolución------------------
[ff cc cl]=size(handles.rgb);

% ------Presentar resolución---------------------
set(handles.la_resolution,'String',['Resolution: ',num2str(cc),'x',num2str(ff)]);
% ************************************************************


% Función que contiene todos los formatos de imágenes
function fileTypes = supportedImageTypes
fileTypes = {'*.jpg','JPEG (*.jpg)';'*.tif','TIFF (*.tif)';...
            '*.bmp','Bitmap (*.bmp)';'*.*','All files (*.*)'};


% ------Mostrar resolución---------------------
```

```
set(handles.la_resolution,'String',['Resolution: ',num2str(cc),'x',num2str(ff)]);
% ************************************************************


% ---------------- Añadir ruido--------------------
function add_noise_Callback(hObject, eventdata, handles)
if isempty(handles.rgb)
   msgbox('Capture una imagen','Aviso')
   return
end

img=handles.rgb;%Llamar imagen leída
nois=get(handles.noise_type,'Value');%Tomar el tipo de ruido a aplicar
switch nois
   case 1
      type='gaussian';
   case 2
      type='poisson';
   case 3
      type='salt & pepper';
   otherwise
      type='speckle';
end

J = imnoise(img,type); %Añadir ruido a la imagen
handles.rgb=J; %Salvar la imagen con ruido
axes(handles.photo) %Seleccionar axes a graficar
image(J) %Mostrar imagen procesada
axis off
% Escribir en el reporte
type=[type,'--'];
handles.noise=[handles.noise type];
guidata(hObject,handles)


% --- Executes on selection change in noise_type.
function noise_type_Callback(hObject, eventdata, handles)

% --- Función de filtrado----------.
function filter_button_Callback(hObject, eventdata, handles)
```

```
% img=handles.rgb; % GET IMAGE
orgImage=handles.rgb;
% Tomar el tipo de filtro del pop-up menú
contents = get(handles.filtros,'String');
filtro=contents{get(handles.filtros,'Value')};
filter = fspecial(filtro);
filterImage = conv2(double(orgImage(:,:,1)), filter);     % convolucion

axes(handles.photo) %Seleccionar axes donde graficar
imshow(filterImage, [0 250]); %Mostrar imagen
axis off
% Código para reporte.
handles.filtro=[handles.filtro,' -- ', filtro];%'Low Filter';
guidata(hObject,handles)

----
% ----------------------------------------------------------------------
% ------------------- FFT -----------------------
function fft_image_Callback(hObject, eventdata, handles)
try
   img=handles.rgb;
   % 2D FFT
   fftImage = fftshift(fft2(double(img(:,:,1))));   % 2d fft
   axes(handles.photo) %Seleccionar axes para mostra imagen
   imshow(uint8(log(abs(fftImage))),[]), colormap(jet(64)), colorbar
   axis off
   handles.fft='FFT processing';
catch % Mensaje en caso de que la imagen no tenga FFT
   msgbox('THIS IMAGE DO NOT HAVE FFT','ERROR');
end

guidata(hObject,handles)


function red_s_Callback(hObject, eventdata, handles)
function green_s_Callback(hObject, eventdata, handles)
function blue_s_Callback(hObject, eventdata, handles)
```

```
% ----------------------- R-G-B-------------------------------
function uipanel4_SelectionChangeFcn(hObject, eventdata, handles)
if hObject==handles.red_s
   plane=handles.original(:,:,1); %Rojo
   P='Red';
elseif hObject==handles.green_s
   plane=handles.original(:,:,2); %Verde
   P='Green';
else
   plane=handles.original(:,:,3); %Azul
   P='Blue';
end
axes(handles.photo)%Seleccionar axes para mostra imagen
image(plane)%Mostrar imagen
axis off    %Deshabiliar ejes
colormap gray %Mapa de colores gris
% Para reporte
P=[P,'--'];
handles.RG_B=[handles.RG_B P];
guidata(hObject,handles)



function cyan_b_Callback(hObject, eventdata, handles)
function magenta_b_Callback(hObject, eventdata, handles)
function yellow_b_Callback(hObject, eventdata, handles)

% ------------------------- C-M-Y----------------------------
function uipanel5_SelectionChangeFcn(hObject, eventdata, handles)
imagen=double(handles.original);
% SEPARA LOS PLANOS R, G y B
R=imagen(:,:,1);
G=imagen(:,:,2);
B=imagen(:,:,3);
% CALCULAR LA DIMENSIÓN DE CADA PLANO
[iR jR]=size(R);
[iG jG]=size(G);
[iB jB]=size(B);
if hObject==handles.cyan_b
```

```
   % CÁLCULO DE CYAN*****************
   % 'C' ES DE LA MISMA DIMENSIÓN QUE 'R'. AL INICIO ES UNA MATRIZ
DE '0' PARA
   % LLENARLA CON LOS VALORES DE LA FÓRMULA C(m,n)=1-
(R(m,n)/255);
   C=zeros(iR,jR);
   for m=1:iR % FILAS
      for n=1:jR %COLUNMAS
         C(m,n)=1-(R(m,n)/255);
      end
   end
   P='Cyan';
   S=C;
elseif hObject==handles.magenta_b
   % *********************************
   % CÁLCULO DE MAGENTA*****************
   % 'M' ES DE LA MISMA DIMENSIÓN QUE 'R'. AL INICIO ES UNA MATRIZ
DE '0' PARA
   % LLENARLA CON LOS VALORES DE LA FÓRMULA M(m,n)=1-
(G(m,n)/255);
   M=zeros(iG,jG);
   for m=1:iG
      for n=1:jG
         M(m,n)=1-(G(m,n)/255);
      end
   end
   P='Magenta';
   S=M;
else
   % *********************************
   % CÁLCULO DE YELLOW*****************
   % 'Y' ES DE LA MISMA DIMENSIÓN QUE 'R'. AL INICIO ES UNA MATRIZ
DE '0' PARA
   % LLENARLA CON LOS VALORES DE LA FÓRMULA Y(m,n)=1-
(B(m,n)/255);
   Y=zeros(iB,jB);
   for m=1:iB
      for n=1:jB
         Y(m,n)=1-(B(m,n)/255);
      end
```

```
    end
    S=Y;
    P='Yellow';
end
imshow(S)

% Para reporte
P=[P,'--'];
handles.CM_Y=[handles.CM_Y P];
guidata(hObject,handles)

function new_size_Callback(hObject, eventdata, handles)


% -------------- Cambio en resolución de adquisición------------------------.
function resize_b_Callback(hObject, eventdata, handles)
% Tomar la nueva resolucion del pop-up menu
contents = get(handles.new_size,'String');
new_res=deblank(contents(get(handles.new_size,'Value'),1:end));
% Parar la captura para cambiar la resolución
closepreview(handles.vidobj)
try %re iniciar la captura con la nueva resolución
handles.vidobj = videoinput('winvideo',1,new_res);
start(handles.vidobj);
guidata(hObject, handles);
vidRes = get(handles.vidobj, 'VideoResolution');
nBands = get(handles.vidobj, 'NumberOfBands');
hImage = image(zeros(vidRes(2), vidRes(1), nBands), 'Parent',...
    handles.video_cam);
preview(handles.vidobj,hImage);
msgbox('PLEASE CAPTURE NEW IMAGE','MESSAGE')
catch % Si el formato es no soportado, mostrar mensaje
    msgbox('The FORMAT specified is not supported by this device','ERROR')
end


% --------------------- Bordes -----------------------
function edge_b_Callback(hObject, eventdata, handles)
% Tomar el tipo de borde
contents = get(handles.edge_t,'String');
```

```
edge_mode=contents{get(handles.edge_t,'Value')};
try
if strcmp(edge_mode,'Laplacian of Gaussian')
   edge_mode='log';
end

img=rgb2gray(handles.rgb);%Cambiar imagen a escala de grise
I=edge(img,edge_mode); %Aplicar borde a imagen

axes(handles.photo)%Seleccionar donde graficar imagen
imshow(I) %Mostar imagen
axis off %Deshabilitar etiqueta de ejes
colormap gray
% Para reporte
edge_mode=[edge_mode,'--'];
handles.edg_e=[handles.edg_e edge_mode];
catch %IF DON'T EXIST ANY IMAGE, PRESENT MESSAGE
   msgbox('PLEASE GET A IMAGE','ERROR')
end
guidata(hObject,handles)
% --- Executes on selection change in edge_t.
function edge_t_Callback(hObject, eventdata, handles)


% --- Executes on selection change in filtros.
function filtros_Callback(hObject, eventdata, handles)
```

REFERENCIAS BIBLIOGRÁFICAS

[1].- Gonzalez, R., Woods, R. y Eddins, S., Digital Image Processing Using Matlab, Prentice Hall, 1999.

[2].- Vaseghi, S., Advance Digital Signal Processing and Noise Reduction. Second Edition, John Wiley y Sons, Ltd., 2000.

[3].- Ingle, V. y Proakis, J., Digital Signal Processing Using Matlab V.4, The PWS Bookware Companion Series, 1997.

[4].- Proakis, J., Digital Communications 4th Edition, McGraw Hill, 2001.

[5].- Smith, S., The Scientist and Engineer's Guide to Digital Signal Processing Second Edition, California Technical Publishing, 1999.