E S P E
ESCUELA POLITECNICA DEL EJERCITO
CAMINO A LA EXCELENCIA

ESCUELA POLITÉCNICA DEL EJÉRCITO
SEDE LATACUNGA

CARRERA DE INGENIERÍA ELECTRÓNICA
E INSTRUMENTACIÓN

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN PARA EL MEJORAMIENTO DE IMÁGENES A COLOR

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA ELECTRÓNICA EN INSTRUMENTACIÓN

PAULINA ALEJANDRA VINUEZA GARZON

Latacunga –Ecuador

2007

AGRADECIMIENTO

A mis padres por su dedicación, cariño, confianza, apoyo incondicional que supieron darme durante el desarrollo de mi vida estudiantil y ahora profesional.

A cada uno de los profesores de la Escuela Politécnica del Ejército que supieron impartir sus conocimientos sin ningún egoísmo, en especial al Ing. Armando Álvarez por ser más que un profesor un amigo, también a aquellas personas que de una u otra manera contribuyeron con la realización de este proyecto.

DEDICATORIA

A Dios, a mis padres y hermanas, por estar siempre presente en cada momento importante de mi vida.

INDICE

CAPITULO I

IMÁGENES DIGITALES

1.1 INTRODUCCIÓN.

El procesamiento de imágenes tiene como objetivo mejorar el aspecto de las imágenes y hacer más evidentes en ellas ciertos detalles que se desean hacer notar. La imagen puede haber sido generada de muchas maneras, por ejemplo, fotográficamente, o electrónicamente, por medio de monitores de televisión. El procesamiento de las imágenes se puede en general hacer por medio de métodos ópticos, o bien por medio de métodos digitales, en una computadora.

Píxeles y Colores.- La propiedad más importante de una imagen digital son los píxeles o puntos de pantalla ya que el tamaño de la imagen digital se mide por píxeles, contándose los píxeles que conforman el alto y el ancho de la imagen.
Profundidad de color, la otra propiedad de las imágenes digitales se refiere al número de colores diferentes que se pueden visualizar en una misma imagen.

Imagen.- El término imagen monocroma o simplemente imagen, se refiere a una función bidimensional representando intensidad de luz, donde x e y son las coordenadas espaciales y el valor de f en cualquier punto (x,y) es proporcional al brillo (o nivel de gris) de la imagen en ese punto.

Imagen digital.- Es una imagen que ha sido discretizada tanto en coordenadas espaciales como en brillo. Se puede considerar una imagen digital como una matriz cuyos índices de filas y columnas identifican un punto en la imagen y el correspondiente elemento de matriz identifica el valor de gris en ese punto.

Una imagen digital se compone de puntos o píxeles dispuestos en filas y columnas. En función del tipo de imagen digital, el punto puede ser blanco o de cualquier otro color.

Figura 1.1 Imagen Digital

La función de imagen digital representa un valor de brillo en ese punto, pero cuando se quiere representar una imagen en color, esa interpretación puede cambiar. Por ejemplo, en función de la división de colores en un cubo RGB, podemos descomponer toda la imagen en tres bandas, cada una de ellas representando el brillo de rojo, de verde y de azul respectivamente.

El color visualizado en un punto concreto será el resultante de combinar los valores de ese punto en las tres bandas, en la proporción indicada por esos valores. O bien se puede tener una tabla de colores (una paleta) y hacer que los valores de la imagen estén 'apuntando' a los colores de esa paleta. De cualquier forma se puede distinguir: blanco y negro, escala de grises, color indexado 16, color indexado 256, color real RGB y color real CYM.

Los tipos de datos se diferencian por la resolución en bits (número de bits de información de la imagen por píxel) y por el número de canales que comprende la

imagen. La resolución en bits determina cuántos colores o niveles de gris puede representar cada píxel.

1.2 REPRESENTACIÓN DE IMÁGENES DIGITALES.

1.2.1 SEÑALES Y SISTEMAS BIDIMENSIONALES[1]

El sentido de la visión se constituye en la principal fuente de información que disponen los seres humanos para percibir y adquirir conocimiento acerca del mundo a su alrededor. Sin embargo, el hombre, en su avance tecnológico, ha inventado una serie de equipos que le han permitido detectar radiaciones electromagnéticas con longitudes de onda fuera del espectro de percepción de la visión humana y con niveles de energía abajo de los que la visión humana es capaz de percibir. Además, otras fuentes han sido utilizadas en la generación de imágenes, tales como ondas acústicas y Rayo-x. Así el concepto de imagen de que trataremos es más amplio que el empleado en el estudio de la visión humana.

En el contexto del Procesamiento de Señales, el concepto de señal es el de una función que contiene información sobre el estado o comportamiento de un sistema físico. Las señales pueden ser clasificadas según distintos criterios. Como funciones, pueden ser representadas a través de operadores matemáticos como $f : X \to Y$, donde X es el dominio y Y es el rango de la función y

$$y = f(x_1, x_2, ..., x_n), (x_1, x_2, ..., x_n) \in X, y \in Y.$$

Debido al dominio X las señales se dividen en analógicas o de dominio discreto. En el primer caso, la variable independiente es continua, o sea, es definida sobre un continuo dominio. En contrapartida, en las señales de dominio discreto la variable

[1] Tomado del folleto de una Maestría en Electrónica ESPE Matriz Autor Ing. Atcocela, pag 1.

independiente asume solamente valores discretos, en general enteros. Estas últimas son también conocidas como secuencias.

En lo que se refiere al rango Y, las señales se clasifican en señales de amplitud continua y de amplitud discreta, según el mismo criterio de la clasificación debido al dominio. Una señal digital es de dominio y amplitud discretos.

Otra posible clasificación es por la dimensión del dominio. Así tenemos señales unidimensionales (1-D), bidimensionales (2-D), etc.

Finalmente, una señal puede ser clasificada debido a los intervalos de valores que sus variables independientes pueden asumir, que pueden ser finitos o infinitos. En particular, para las secuencias se dice que es de soporte finito cuando tales intervalos son todos finitos, en caso contrario, es una secuencia de soporte infinito.

1.2.1.1 SEÑALES DE IMAGEN

Se puede ahora conceptuar lo que es una señal de imagen y el tipo de señal de que trataremos. Imagen es una señal bidimensional. En este texto, se considera imágenes de espacio discreto, amplitud continua o discreta y, en general, de soporte finito. La notación adoptada será $x(n_1,n_2)$, donde x es una función de espacio discreto bidimensional con argumentos enteros n_1 y n_2. Los puntos (n_1,n_2)reciben la denominación de píxeles[2]. La forma gráfica adoptada para representar las secuencias 2-D puede ser entendida a través del siguiente ejemplo:

$$x(n_1, n_2) = \begin{cases} 1, -1 \le n_1 \le 0 y 0 \le n_2 \le 2 \\ 2, \quad 1 \le n_1 \le 0 y 0 \le n_2 \le 2 \\ 0, \quad caso\, contrario \end{cases}$$

[2] Tomado del folleto de una Maestría en Electrónica ESPE Matriz Autor Ing. Atcocela, pag 2.

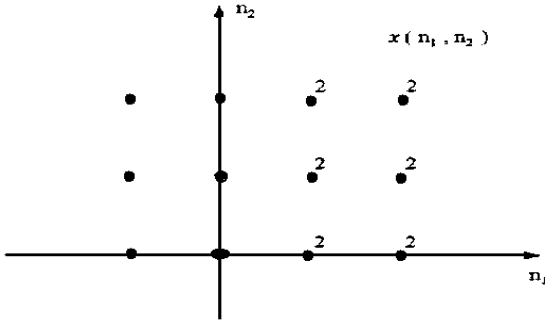


Figura 1.2: Representación Gráfica de una Secuencia 2-D

Así, son representados solamente los valores no nulos. Las magnitudes son puestas a la derecha y arriba de los respectivos puntos, excepto cuando su valor es 1. Las coordenadas de las variables independientes son omitidas para evitarse confusión con las magnitudes. Cuando es necesaria mayor claridad, los puntos con amplitud nula serán representados con círculos vacíos, como en el ejemplo de abajo:

$$x(n_1, n_2) = \begin{cases} 1, -1 \le n_1 \le 1 y 0 \le n_2 \le 1 \\ 2, \quad n_1 = 3\, y 0 \le n_2 \le 1 \\ 0, \quad caso\, contrario \end{cases}$$

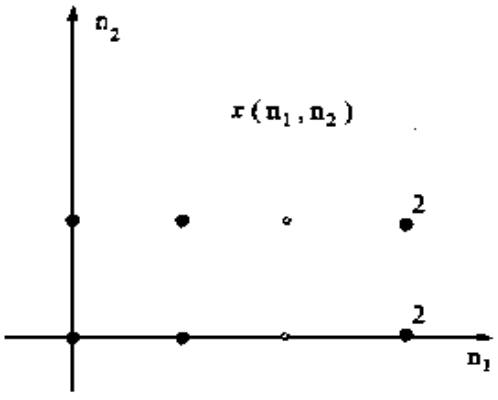


Figura 1.3: Representación Gráfica de una Secuencia 2-D

Algunas secuencias son particularmente importantes en procesamiento de señales bidimensionales, como los impulsos, los escalones, las secuencias exponenciales y las secuencias separables.

- **Secuencia Impulso Unitario**

$$\delta(n_1, n_2) = \begin{cases} 1, n_1 = n_2 = 0 \\ 0, caso contrario \end{cases}$$

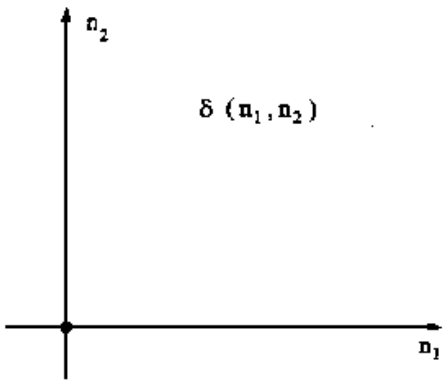


Figura 1.4: Representación Gráfica de $\delta(n_1,n_2)$

El impulso unitario desempeña papel similar a la función delta de Kronecker en procesamiento 1-D. Asi, cualquier señal 2-D puede ser representada a través de una doble sumatoria de impulsos unitarios desplazados y ponderados:

$$x(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2).\delta(n_1 - k_1, n_2 - k_2)$$

- **Secuencia Impulso Línea**

$$\delta_B(f(n_1, n_2)) = \begin{cases} 1, & f(n_1, n_2) = 0 \\ 0, & caso contrario \end{cases}$$

Por ejemplo, si $f(n_1,n_2)= n_1- 2n_2$, entonces

$$\delta_B(n_1 - 2n_2) = \begin{cases} 1, \; n_1 = 2n_2 = 0 \\ 0, \; caso\, contrario \end{cases}$$

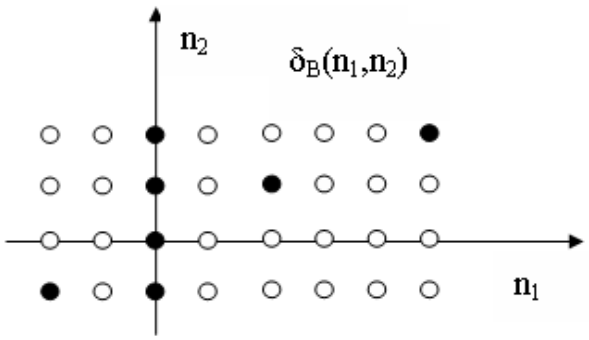


Figura 1.5: Representación Gráfica de $\delta_B(n_1,n_2)$

- **Secuencia Escalón Unitario**

$$u(n_1, n_2) = \begin{cases} 1, n_1 \geq 0 y n_2 \geq 0 \\ 0, \; caso\, contrario \end{cases}$$

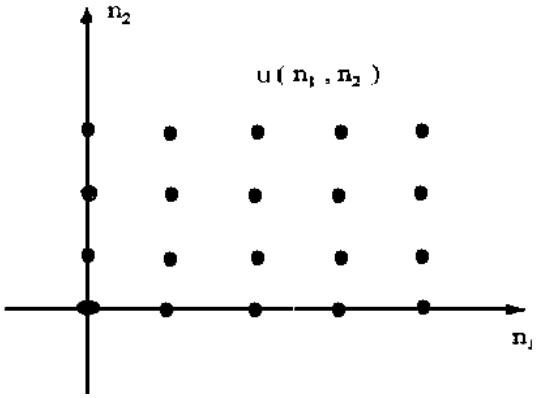


Figura 1.6: Representación Grafica de $u(n_1,n_2)$

Las secuencias impulso unitario y escalón unitario pueden ser relacionadas como:

$$u(n_1, n_2) = \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} \delta(n_1 - k_1, n_2 - k_2) = \sum_{l_1=-\infty}^{n_1} \sum_{l_2=-\infty}^{n_2} \delta(l_1, l_2)$$

y $\delta(n_1, n_2) = u(n_1, n_2) - u(n_1 - 1, n_2) - u(n_1, n_2 - 1) + u(n_1 - 1, n_2 - 1)$

- **Secuencie Exponencial**

  Son secuencias del tipo $x(n_1, n_2) = A\alpha^{n_1}\beta^{n_2}$, donde A, α y β son constantes reales o complejas

- **Secuencias Separables**

  Una secuencia 2-D es dicha separable si puede ser expresada como el producto de dos funciones 1-D, siendo una función de n_1 y la otra función de n_2:

  $$f(n_1, n_2) = g(n_1).g(n_2).$$

  Las secuencias impulso unitario, escalón unitario y exponenciales son secuencias separables.

1.2.2 SISTEMAS BIDIMENSIONALES DE ESPACIO DISCRETO

Un sistema es definido matemáticamente como una transformación u operador que mapea cada señal de entrada en una única señal de salida. Si las señales son bidimensionales y de espacio discreto, el sistema recibe la misma denominación[3].

Su representación es:

[3] Tomado del folleto de una Maestría en Electrónica ESPE Matriz Autor Ing. Atcocela, pag. 5

$$y(n_1, n_2) = T\{x(n_1, n_2)\}$$

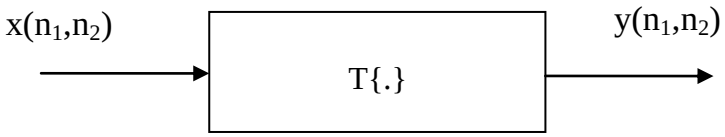


Figura 1.7: Representación Gráfica de un Sistema

Los sistemas pueden ser divididos en clases de acuerdo con sus características. Las principales características, en sistemas que tratan de imágenes son la linealidad, la invariancia al desplazamiento y la estabilidad, las cuales son analizadas a seguir. Además, es de interés la región de soporte de su respuesta al impulso.

- **Linealidad**

  Un sistema es lineal si, y solamente si, satisface el Principio de la superposición, es decir, si:

  $$x_1(n_1, n_2) \text{ y } x_2(n_1, n_2)$$

  son dos posibles entradas del sistema,

  $$y_1(n_1, n_2) = T\{x_1(n_1, n_2)\} \text{ y } y_2(n_1, n_2) = T\{x_2(n_1, n_2)\}$$

  Sus respectivas salidas, a1 y a2 escalares arbitrarios,

  Entonces:

  $$T\{a_1 x_1(n_1, n_2) + a_2 x_2(n_1, n_2)\} = a_1 y_1(n_1, n_2) + a_2 y_2(n_1, n_2)$$

Generalizando el Principio de Superposición, se tiene que, si:

xk(n1,n2)

Son posibles entradas del sistema, $y_k(n_1, n_2) = T\{x_k(n_1, n_2)\}$

Sus respectivas salidas, ak escalares arbitrarios,

$$x(n_1, n_2) = \sum_k a_k x_k(n_1, n_2)$$

Entonces:

$$y(n_1, n_2) = T\left\{\sum_k a_k x_k(n_1, n_2)\right\} = \sum_k a_k y_k(n_1, n_2)$$

- **Invarianza al Desplazamiento**

Un sistema es dicho invariante al desplazamiento si a un desplazamiento espacial de la secuencia de entrada corresponde un desplazamiento idéntico de la secuencia de salida, o sea:

$$y(n_1, n_2) = T\,x(n_1, n_2) \Rightarrow \forall.(n_1, m_2) \in Z^2, y(n_1 - m_1, n_2 - m_2) = T\,x(n_1 - m_1, n_2 - m_2)$$

En procesamiento de imágenes, esto significa que si la imagen de entrada es desplazada, la imagen, de salida no cambia de aspecto, solamente de posición.

- **Estabilidad**

El concepto de estabilidad es variable de acuerdo con la aplicación. Un sistema es estable si, y solamente si, toda y cualquier entrada limitada produce una salida limitada matemáticamente, esto es expresado como:

$$\forall(n_1,n_2),\exists B_x \in R tal\ que\ |x(n_1,n_2)| \leq B_x < \infty \Rightarrow \forall(n_1,n_2),\exists B_y \in R tal\ que\ |T\,x(n_1,n_2)| \leq B_y < \infty$$

La estabilidad es una propiedad deseable en procesamiento digital de imágenes para evitarse el "overflow" y otras dificultades en el procesamiento.

1.3 DESCRIPCIÓN.

1.3.1 TIPOS DE IMÁGENES

La forma de manejar los datos, en un archivo de imagen, ha dado dos principales modos de manipular la información para visualizar imágenes digitales. Éstos son las imágenes de mapa de bits y las imágenes vectoriales. Ya que, cada uno de estos modos se adapta mejor a algún tipo de imagen es necesario saber, antes de conocer los tipos de formatos de imagen, como funcionan tanto imágenes vectoriales como imágenes de mapa de bits.

1.3.1.1 MAPAS DE BITS

Las imágenes de mapa de bits están formadas por una rejilla de celdas. A cada una de estas celdas, que se denominan píxeles, se le asigna un valor de color y luminancia propios. Por esto, cuando vemos todo el conjunto de celdas, tenemos la ilusión de una imagen de tono continuo.

El píxel es una unidad de información, no una unidad de medida, ya que no se corresponde con un tamaño concreto. Un píxel puede ser muy pequeño -0.1mm.- o muy grande -1 cm.-.

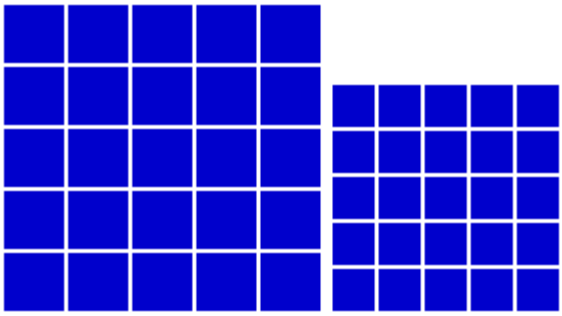


Figura 1.8 Dos rejillas de cinco píxeles

Cuando se crea una imagen de mapa de bits se genera una rejilla específica de píxeles. Por esto, al modificar su tamaño, se transforma, a su vez, la distribución y coloración de los píxeles, por lo que los objetos, dentro de la imagen, suelen deformarse. Esto es porque los objetos pierden o ganan algunos de los píxeles que los definen. Gracias a esta característica, que siempre hay que tener en cuenta, las imágenes de mapa de bits se crean con un tamaño determinado y pierden calidad si se modifican sus dimensiones.

1.3.1.2 IMÁGENES VECTORIALES

Los elementos constituyentes del vector, en una imagen vectorial, son las curvas de Bézier. Desarrolladas por Pierre Bézier. Una curva Bézier se define por cuatro puntos. Los puntos inicial y final de la curva -nodos o puntos de anclaje- y dos puntos de control –manecillas o manejadores-, estos últimos como sus nombres lo indican sirven para definir la forma de la curva y no aparecerán en la imagen final. Así, para modificar la curva sólo se tiene que mover alguno de los nodos.

Las imágenes vectoriales tienen el inconveniente de tener dificultades en tratar algunas cosas de forma natural como las sombras, luces, entre otros y cuando son muy grandes o muy complejas se vuelven extremadamente difíciles de manejar para la capacidad de un computador[4].

Dichas imágenes no serán analizadas en el desarrollo del tema.

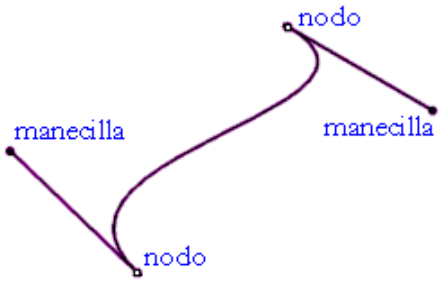


Figura 1.9 Curva de Bézier

1.3.1.3 IMÁGENES EN BLANCO Y NEGRO

Una imagen en **blanco y negro** es una imagen de 1 bit. Esto significa que cada punto o píxel de la imagen sólo puede tener dos valores: blanco y negro.

[4] http://www.gusgsm.com/imagen_vectorial

Figura 1.8 Imagen en Blanco y Negro

1.3.1.4 IMÁGENES EN ESCALA DE GRISES

Una imagen de **escala de grises** es de 8 bits, lo que significa que cada punto puede tener un valor entre 256. Por lo tanto, un píxel en una imagen de escala de grises puede ser de color negro, blanco, o de un tono de gris entre 254 valores posibles.

Cada píxel de una imagen en escala de grises puede ser uno de los 256 valores distintos de gris, del negro (cero) al blanco (255). Este tipo de datos muestra suaves cambios de tono utilizando tonos intermedios de gris.

Figura 1.9 Imagen en Escala de Grises

La resolución de una imagen en escala de grises determina el tamaño de los píxeles y, en consecuencia, el número de píxeles de una imagen. Cuanto mayor sea la resolución, los cambios de tono de gris serán más suaves y por tanto más exacta la representación de la imagen (las imágenes de alta resolución también utilizan más memoria).

1.3.1.5 IMAGEN A COLOR

Cada píxel de una imagen a color consta de tres valores rojo verde y azul, se puede representar por 16.7 millones de colores posibles, un píxel tiene 3 bytes, un byte por color.

Figura 1.10 Imagen a Color

1.3.1.6 IMÁGENES MULTICANAL

La imagen es multicanal cuando los píxeles representan magnitudes en distintos dominios físicos. Por ejemplo una imagen a color es una imagen con 3 canales; canal R (rojo) canal G (verde) y canal B (azul).

En algunas aplicaciones (imágenes de satélite, visión nocturna) suelen usarse canales para frecuencias no visibles, infrarrojo, ultravioleta, etc.

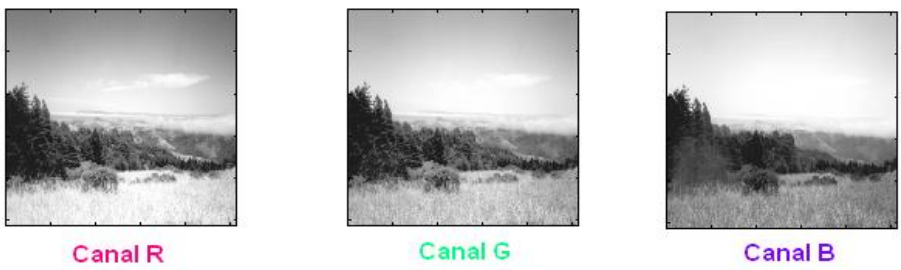


Figura 1.11 Imágenes Multicanal

1.3.2 TIPOS DE ARCHIVOS[5]

1.3.2.1 BMP

El formato BMP (Bit Map) es el formato de las imágenes de mapa de bits de Windows. Su uso fue muy extendido, pero los archivos son muy grandes dado la escasa compresión que alcanzan. Este formato BMP admite los modos de color RGB, Color indexado, Escala de grises y Mapa de bits.

1.3.2.2 GIF

El formato GIF corresponde a las siglas de Graphics Interchange Format propiedad de CompuServe y fue desarrollado en 1987. El formato GIF es preferible para las imágenes de tonos no continuos o cuando hay grandes áreas de un mismo color ya que utiliza una paleta de color indexado que puede tener un máximo de 256 colores.

Una de sus mayores ventajas es que se puede elegir uno o varios colores de la paleta para que sean transparentes y podamos ver los elementos que se encuentren por debajo de estos. También es uno de los pocos formatos de imagen con el que podemos mostrar animaciones porque hace que distintos frames se ejecuten secuencialmente. Además, es un formato de compresión diseñado para disminuir el tiempo de transferencia de una imagen raster por las líneas telefónicas, independientemente del hardware que se este utilizando.

1.3.2.3 JPG o JPEG

[5] http://www.wikilearning.com/formatos_de_imagenes_i-wkccp-12911-2.htm

Este formato toma su nombre de Joint Photographic Experts Group, asociación que desarrollo el método de codificación de la compresión. Por lo tanto, no es específicamente un formato de imagen sino de compresión.

Se utiliza usualmente para almacenar fotografías y otras imágenes de tono continuo, gracias a que utiliza un sistema de compresión que de forma eficiente reduce el tamaño de estos archivos. En contraste con GIF, JPEG guarda toda la información referente al color con millones de colores (RGB) sin obtener archivos excesivamente grandes. Además, los navegadores actuales reconocen y muestran con fidelidad este formato. Tiene una escala de pérdida de información que nos permite nivelar la calidad de la imagen que obtendremos, por lo que, una imagen de 24 bits salvada con JPEG puede ser reducida hasta alrededor de la vigésima parte de su tamaño original aplicando el nivel máximo de compresión. Claro esta que mientras mayor compresión tengamos, mayor será la pérdida de información que habrá en la imagen.

1.3.2.4 TIFF

El formato TIFF Tag Image File Format es desarrollado por Aldus Corporation en 1986, específicamente para guardar imágenes desde el escáner y programas para creación de imágenes y retoque fotográfico. Se utiliza para imágenes de mapa de bits y es admitido prácticamente por todas las aplicaciones de autoedición y tratamiento de imágenes. Es probablemente el formato de mapa de bits más versátil, seguro y con mayor soporte. Ya que es capaz de describir los datos de una imagen desde 2 colores hasta color completo en varios espacios de tonos.

1.3.2.5 PSD

Este es el formato de Adobe Photoshop y, por lo mismo, es el único que admite todas las funciones que este programa contiene. Sin embargo, su uso se centra en la manipulación de la imagen y no tanto para ser empleado en publicaciones digitales. Presenta grandes ventajas para la edición, ya que al guardar con este formato se mantiene las capas (en estas se manipula los diferentes elementos de una imagen por separado) que se haya utilizado en la manipulación de la imagen.

1.3.2.6 PNG

PNG son las siglas del grupo que lo desarrollo Portable Networks Graphics pensando en un formato ideal para su distribución en Internet y por los problemas de patente del algoritmo de compresión LZW que emplean por lo regular las imágenes GIF. PNG posee ventajas respecto a los otros formatos más comunes en Internet -JPG y GIF-. Ya que fue desarrollado especialmente para su distribución en red posee gran parte de las ventajas de un GIF y de un JPG. Por ejemplo, permite altos niveles de comprensión, además, permite utilizar la técnica de la indexación para crear colores transparentes, semitransparencias o transparencias degradadas. Finalmente, no está limitado a una paleta de 256 colores, sino que puede utilizar millones de colores. Su única limitación es que no podemos crear ficheros animados.

1.4 FUNDAMENTOS DE COLOR EN IMÁGENES.

En 1666, Isaac Newton descubrió que cuando un rayo de luz del sol pasa a través de un prisma, el rayo de luz que emerge de él, no es blanco, pero consiste en un espectro continuo de colores que van del rojo y terminan en el violeta. Como se muestra en la figura 1.12.

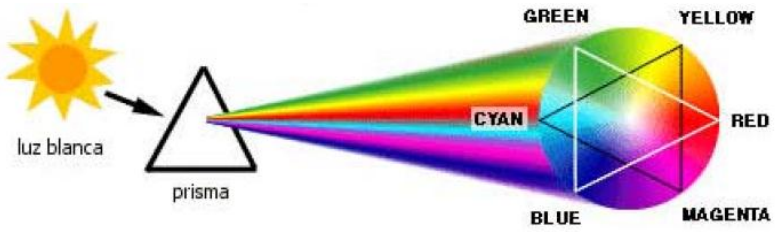


Figura 1.12 Prisma Óptico

El espectro de color puede ser dividido en seis regiones anchas: de color violeta, azul, verde, amarillo, anaranjado, y rojo. Ningún color en el espectro termina abruptamente, sino que cada color cambia del uno al otro suavemente.

Básicamente, los colores que los seres humanos percibimos de un objeto están determinados por la naturaleza de la luz reflejada del objeto. Como se muestra en la figura 1.13, la luz visible está compuesta de una banda relativamente estrecha de frecuencias en el espectro electromagnético.

Un cuerpo que refleja luz que esta balanceada equilibrado en todas las longitudes de onda visibles aparece blanco para el observador. Sin embargo, un cuerpo que favorece la reflectancia en un rango limitado del espectro visible exhibe algunas sombras de color. Por ejemplo, los objetos verdes reflejan la luz con longitudes de onda en los 500 a 570 nm(10^{-9} m), mientras que absorben la mayoría de la energía a otras longitudes de onda.

La caracterización de luz es esencial para la ciencia de color. Si la luz es acromática (sin color), su único atributo es su intensidad, o cantidad. La luz acromática es la que se percibe en un monitor blanco y negro.

La luz cromática se expande en el espectro electromagnético de aproximadamente 400 a 700nm. Se utilizan tres cantidades básicas para describirla: radianza, luminosidad, y brillo.

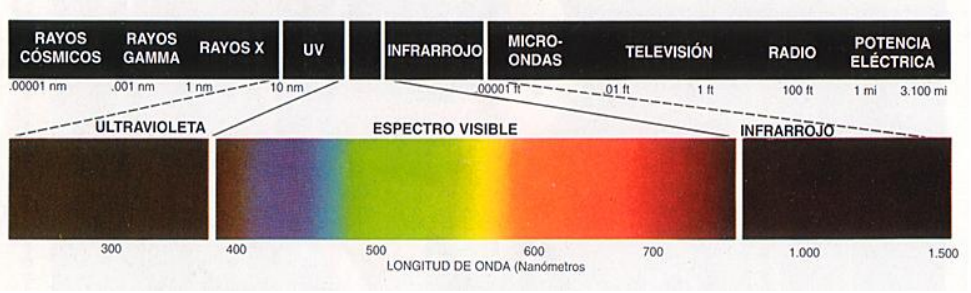


Figura. 1.13. Longitudes de onda que comprenden e[6]l rango visible del espectro electromagnético[6]

La radianza es la cantidad total de energía que fluye de la fuente de luz, y es normalmente medida en vatios (W). La luminosidad, se mide en lumens (lm), y da una medida de la cantidad de energía que un observador percibe de una fuente de luz. Por ejemplo la luz emitida por una fuente que opera en la región infrarroja lejana del espectro podría tener una cantidad significativa de energía (radianza), pero un observador apenas lo percibiría; su luminosidad sería casi cero.

Finalmente, el brillo es un descriptor subjetivo que es prácticamente imposible medir. Incluye la noción acromática de intensidad y es uno de los factores importantes para describir la sensación del color.

Debiendo a la estructura del ojo humano, todos los colores se ven como las combinaciones inconstantes de los tres llamados colores primarios: rojo (R), verde (G), y azul (B). las longitudes de onda específicas de los tres colores primarios son: azul = 435.8nm, verde = 546.1nm, y rojo = 700nm. Sin embargo, ningún color solo puede llamarse rojo, verde o azul. Así, teniendo tres longitudes de onda específicas con el propósito de la estandarización hace que estos tres componentes de RGB fijos que actúan exclusivamente pueden generar todo los colores del espectro.

Los colores primarios pueden agregarse o sumarse para producir los colores secundarios de luz magenta (rojo más azul), celeste (verde más azul), y amarillo (rojo

[6] http://campusvirtual.uma.es/tdi/www_netscape/TEMAS/Tdi_30/index2.php

más verde). Mezclando los tres colores primarios, o los secundarios con los primarios opuestos con las intensidades correctas produce la luz blanca, como se muestra en la figura 1.14(a), qué también ilustra los tres colores primarios y sus combinaciones para producir los colores secundarios.

Diferenciar entre los colores primarios de luz y los colores primarios de pigmentos o colorantes es importante. En el último, un color primario se define como uno que substrae o absorbe un color primario de luz y refleja o transmite los otros dos. Por consiguiente los colores primarios de pigmentos son magenta, celeste, y amarillo, y los colores secundarios son el rojo, verde y azul. Estos colores se muestran en el figura 1.14 (b) una combinación apropiada de los tres colores primarios de pigmento, o los secundario con sus opuestos primarios, produce el negro.

Las características que se usan para distinguir un color de otro son brillo, el tono y la saturación, como ya se dijo el brillo involucra la noción cromática de intensidad. El tono es un atributo asociado con la longitud de onda dominante en la mezcla de luz, representa el color dominante percibido por el observador. La saturación se refiere a la pureza relativa de la cantidad de luz blanca mezclada con el tono, los colores puros del espectro están completamente saturados. Colores como el rosa (rojo con blanco) o lavanda (violeta con blanco) están menos saturados, con el grado de saturación siendo inversamente proporcional a la cantidad de luz blanca añadida

a) Colores aditivos primarios (luz)

b) Colores sustractivos primarios (pigmentos)

Figura 1.14 Colores primarios aditivos y sustractivos

El tono y la saturación tomadas en conjunto se llaman cromaticidad y por lo tanto el color en particular se llaman valores triestimulos y se denotan por X,Y y Z respectivamente. Un color se especifica entonces por los coeficientes triestimulo definidos como:

$$x = \frac{X}{X+Y+Z};$$

$$y = \frac{Y}{X+Y+Z};$$

$$z = \frac{Z}{X+Y+Z}$$

Por lo tanto x+y+z=1.

Otro método de especificar el color es utilizando el diagrama de cromaticidad (CIE)* que muestra la composición del color como función de x(rojo) y y(verde). Para cada valor de x y y el correspondiente valor de z(azul) se obtiene de la ecuación anterior x+y+z=1: haciendo que z= 1-(x+y)

Las posiciones de varios colores del espectro – desde violeta a 380nm a rojo 780nm- se indican alrededor del límite del diagrama cromático con forma de dedo pulgar. El diagrama de cromaticidad que se obtiene con el sistema de coordenadas XYZ es lo más amplio posible y tiene el siguiente aspecto

Los tres extremos representan los colores puros. Cualquier punto que no este en los bordes representa alguna mezcla de color. El punto de energía igual es la luz blanca.
Cualquier punto colocado exactamente en el borde corresponde a un color completamente saturado. Conforme nos acercamos del borde al punto de energía igual al color se le agrega más luz blanca y por lo tanto esta menos saturado.

No todos los colores del diagrama de cromaticidad pueden ser representados. El área triangular representa los colores típicos de un monitor a color. El área irregular representa los colores de dispositivos de impresión como se indica en la figura 1.16.

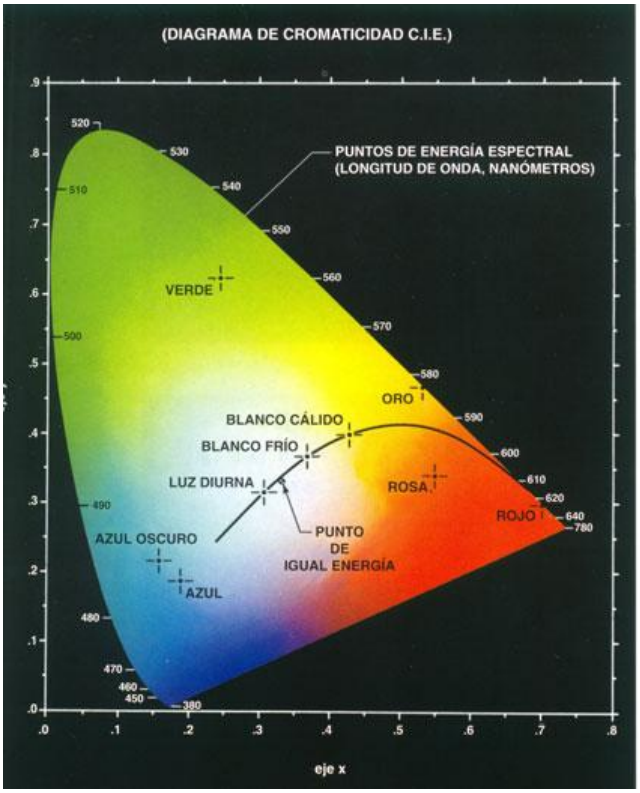


Figura 1.15: Diagrama de cromaticidad[7]

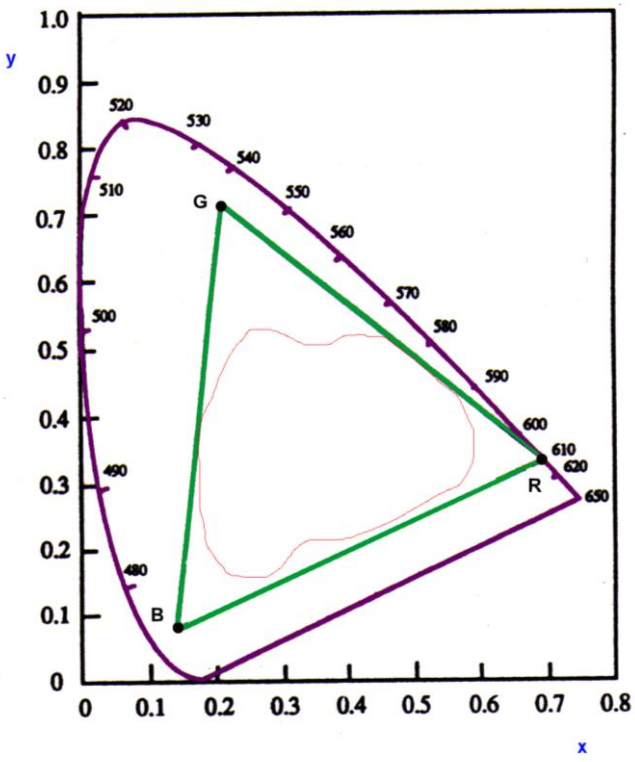


Figura 1.16 Regiones del diagrama de cromaticidad

[7] http://pub.ufasta.edu.ar/SISD/vision/CIE.htm

1.5 MODELOS DE COLOR EN IMÁGENES

Se puede considerar el modelo de color como el contenedor en que se coloca la información sobre cada píxel de una imagen. Así, se guarda una cantidad pequeña de datos de color en un contenedor muy grande, pero no podrá almacenar una gran cantidad de datos de color en un contenedor muy pequeño.

Un color o matiz primario es una tonalidad en el modelo correspondiente que no puede ser generado a partir de otros colores usados en el modelo. En ese sentido, los colores primarios son la base para crear nuevos tonos con base en sus mezclas. Cualquier color generado a partir de la mezcla de dos tonos primarios es un color secundario.

1.5.1 MODELO RGB

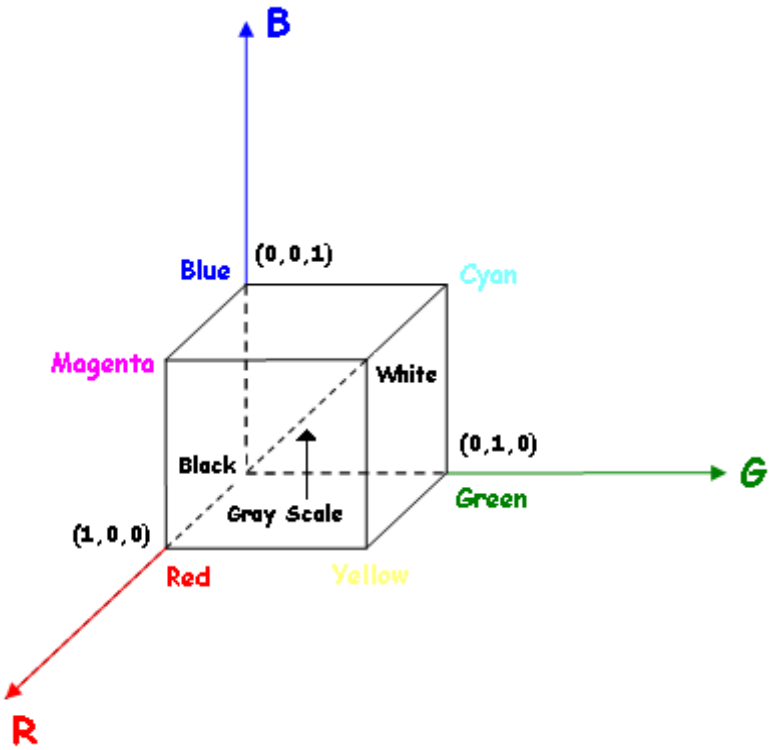
Es un modelo de color basado en la síntesis aditiva, en el cual es posible representar un color mediante la mezcla por adición de los tres colores luz primarios: rojo, verde y azul (***R**ed, **G**reen, **B**lue*)

Este modelo se basa en el sistema de coordenadas cartesianas, los colores de subespacio de interés es un cubo mostrado en la figura 1.7, en el cual los valores del RGB son tres coordenadas: celeste (cyan), violeta (magenta), y amarillo (yellow); negro (black) esta en el origen y blanco (white) esta en la coordenada mas lejana desde el origen. En este modelo la escala de grises (gray scale) se extiende desde el negro hacia el blanco solo la línea punteada, estos 2 puntos, y los colores son puntos que están dentro del cubo, definido por vectores desde el origen. Por conveniencia, se asume que todo color tiene un valor normalizado como el que el

cubo muestra en la figura 1.7, es la unidad de cubo, que es todo los valores de R G y B son asumidos dentro de un rango [0, 1][8]

En el modelo RGB, los colores primarios son el rojo, verde y azul, mientras los secundarios son el magenta (rojo + azul), el cian (azul + verde) y el amarillo (rojo + verde).

El modo RGB asigna un valor de intensidad a cada píxel que oscile entre 0 (negro) y 255 (blanco) para cada uno de los componentes RGB de una imagen en color. Por ejemplo, un color rojo brillante podría tener un valor R de 246, un valor G de 20 y un valor B de 50. El rojo más brillante que se puede conseguir es el R: 255, G: 0, B: 0. Cuando los valores de los tres componentes son idénticos, se obtiene un matiz de gris. Si el valor de todos los componentes es de 255, el resultado será blanco puro y será negro puro si todos los componentes tienen un valor 0.[9]



[8] Digital Image Processing Autores Gonzales R y Woods R. pag 226
[9] www.fotonostra.com

Figura 1.17 Tetraedro de color RGB. Los puntos a lo largo de la diagonal principal tienen valores de gris, desde negro al origen al punto (1,1,1)

1.5.2 MODELO DE COLOR CMY

Como se ha mencionado cyan magenta, y amarillo son los colores secundarios de luz o, alternativamente, los colores primarios de pigmentos. Por ejemplo, cuando una superficie cubierta con el pigmento celeste se ilumina con la luz blanca, ninguna luz roja se refleja de la superficie. Esto es, el celeste substrae la luz roja de la luz blanca reflejada, qué está compuesto de cantidades iguales de rojo de la luz verde, y azul.[10]

El modelo de color CMY (Cyan Magenta Yellow), es un modelo de color basado en la síntesis sustractiva, según la cual, la mezcla a partes iguales de los tres primarios (cian, magenta y amarillo), en su máxima intensidad, resulta en negro. Si, por otra parte, mezclamos en parejas los colores primarios, obtenemos los colores secundarios, que se corresponden con los primarios de la síntesis aditiva de color (rojo, verde y azul violeta o RGB).

La mayoría de los dispositivos que depositan los pigmentos de color en el papel como las copiadoras, impresoras, requieran datos de CMY de entrada o se realice una conversión RGB internamente a CMY. Esta conversión se puede realizar usando una operación simple:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad 1.1$$

Los sistemas RGB, CMYK se encuentran relacionados, ya que los colores primarios de uno son los secundarios del otro (los colores secundarios son los obtenidos por mezcla directa de los primarios).

[10] Digital Image Processin Autores Gonzales R y Woods R. pag 227

1.5.3 MODELO DE COLOR HSI

El sistema HSI, o HSV como también se le conoce, es una buena herramienta para desarrollar algoritmos de procesamiento de imágenes basadas en alguna propiedad del sentido de percepción del color del ser humano. La mayoría de estos espacios son transformaciones lineales de RGB, por lo que dependen del dispositivo y los programas gráficos.

En este sistema con lo que se trabaja es con las características tono (H), saturación (S) e intensidad (I).

La representación gráfica de estas tres propiedades genera un espacio en forma de doble cono invertido.

En el perímetro del disco están situados los colores azul, magenta, rojo, amarillo, verde y cyan, separados 60º uno de otro. Cada punto del perímetro describe un color que es mezcla de los dos adyacentes. Un punto que no esté en el perímetro contendrá una mezcla de todos. Por lo tanto, estos puntos describen colores pastel que contienen una cierta cantidad de blanco. La distancia al centro (radio) indicará la saturación del color.

El brillo depende de la altura en el doble cono, y es un valor entre 0 y 1. El brillo es la intensidad del color; el punto medio del disco central describe un blanco de intensidad media.

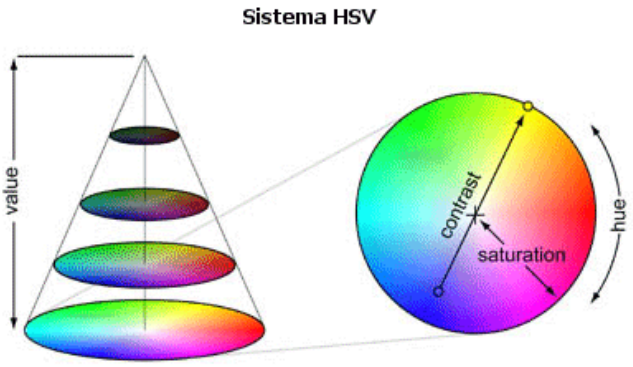


Figura 1.18 Modelo HSI

El matiz (Hue) hace referencia al color como tal, por ejemplo el matiz de la sangre es rojo. La saturación o intensidad indica la concentración de color en el objeto. La saturación de rojo de una fresa es mayor que la del rojo de unos labios. Por su parte, el brillo (Value) denota la cantidad de claridad que tiene el color (tonalidad más o menos oscura). Cuando se habla de brillo hacemos referencia al proceso mediante el cual se añade o se quita blanco a un color. Más adelante estudiaremos con detalle estos conceptos.

1.6 APLICACIONES.

El Procesamiento Digital de Imágenes ha sido ampliamente utilizado por diversas disciplinas tales como: biología, procesado de documentos, medicina, automatización en la industria, restauración de imágenes fotográficas, fotografía satelital, teledetección, biometría.

1.6.1 BIOLOGÍA.

En los campos de la biología y biomedicina, el procesamiento de imágenes se usa para analizar visualmente las muestras biológicas. Hay varios casos donde el análisis de muestras ha sido completamente automatizado usando procesamiento de imágenes. De cara la realización de un buen análisis, se mejoran usando algoritmos como "contraste-balance" o "edge-sharpening", aquellas características inidentificables y/o poco nítidas de la imagen .La identificación automática, clasificación, categorización y análisis de ADN pueden ser realizadas por el procesamiento de imágenes.

1.6.2 PROCESADO DE DOCUMENTOS.

Una recolección y procesado automático de documentos e imágenes se presenta útil para bancos y compañías de seguros. Estos documentos están digitalmente comprimidos y guardados. Se detectan e identifican automáticamente la información impresa sobre cheques y otros documentos contables.

1.6.3 AUTOMATIZACIÓN EN LA INDUSTRIA.

El procesamiento de imágenes se usa para la inspección y supervisión automática en líneas de producción de grandes empresas. Este sistema reduce mucho el error humano al tiempo que proporciona estabilidad y precisión en la producción.

1.6.4 DIAGNÓSTICO A PARTIR DE IMÁGENES MÉDICAS.

Los rayos X y las proyecciones CT médicas se digitalizan para examinar áreas internas del cuerpo. Un número determinado de proyecciones CT se combinan y digitalizan automáticamente para producir imágenes 3-dimensionales.

1.6.5 TELEDETECCIÓN.

Un satélite fotografía la corteza exterior de la Tierra a intervalos regulares y las imágenes se usan para analizar condiciones de crecimiento del cultivo, distribución de la vegetación y para exploraciones de recursos naturales. También la corteza de

la Tierra puede ser convertida en un modelo tridimensional usando imágenes 2D tomadas por satélites.

1.6.6 EFECTOS DE VIDEO/FILM.

La industria cinematográfica usa una extensa variedad de técnicas de procesado de imágenes para producir efectos visuales especiales. Las imágenes irreales y otras escenas costosas son artificialmente producidas usando técnicas de Informática Gráfica o/y Procesamiento de Imágenes Digitales. Las principales técnicas son:

- Morphing.- es una técnica de efectos especiales que visualmente transforma un material en otro de manera espontánea produciéndose una transición natural. Esta técnica se usa ampliamente en películas y publicidad.

- Composiciones de imágenes.-Esta técnica combina un determinado número de imágenes para conformar una sola. Muchos tipos de imágenes, tales como imágenes animadas por ordenador, pueden ser mezcladas conjuntamente.

CAPITULO II

OPERACIONES CON IMÁGENES DIGITALES

2.1 OPERACIONES BÁSICAS DE IMÁGENES

2.1.1 OPERACIONES ARITMÉTICAS

Las operaciones aritméticas[11] entre dos píxeles x e y se denotan como sigue:

La suma: x + y.
La substracción: x – y.
La multiplicación: x*y (también: xy y x x y).
La división: x ÷ y.

Las operaciones aritméticas en las imágenes enteras son acarreadas fuera del píxel por píxel. El uso principal de la suma de imágenes, es para reducir el ruido promedio en una imagen. La substracción de la imagen es una herramienta básica en imágenes médicas la cual se usa para quitar la información del fondo estática. Uno de los principales usos de la multiplicación de la imagen o de la división, es corregir el nivel de gris.

Las operaciones aritméticas involucran sólo una situación del píxel espacial en un momento, para que pueden hacerse o realizarse "en el lugar", en el sentido que el resultado de realizar una operación aritmética a la posición (x, y) puede guardarse en esa posición en una de las imágenes existentes, cuando esa posición no se visitará de nuevo.

2.1.2 OPERACIONES LÓGICAS

Las operaciones lógicas usadas en el procesamiento de imágenes son AND, OR y COMPLEMENTO que se denotan de la siguiente manera[12]:

[11] Digital Image Processing, Autores Gonzalez R. y Woods R. pag.47
[12] Digital Image Processing, Autor Gonzalez R. y Woods R. pag.47

AND: x AND y

OR: x OR y

COMPLEMENTO: NOT y

Estas operaciones son funcionalmente completas en el sentido que pueden combinarse para formar cualquier otra operación lógica. Las operaciones lógicas solo se pueden aplicarse a imágenes binarias, por cuanto las operaciones aritméticas se aplican a los píxeles con multivalores. Las operaciones lógicas son herramientas básicas en el procesamiento de imágenes binarias, dónde se usan para las tareas como enmascarar, el descubrimiento del rasgo, y análisis de la forma.

Las operaciones lógicas se realizan con imágenes enteras píxel por píxel. La operación AND de dos variables binarias es 1 solamente cuando las dos variables son 1, resultado a cualquier situación un resultando AND la imagen sólo es 1 si el correspondiente píxel en las dos imágenes de entrada son 1. Como las operaciones lógicas incluyen solamente una posición del píxel en el tiempo, que pueden hacer en el mismo lugar, como en el caso de las operaciones aritméticas. En la figura se muestra varios ejemplos de operaciones lógicas donde el negro indica 1 y el blanco indica 0. La operación XOR produce un 1, cuando uno o el otro píxel (pero no ambos) es 1, y produce 0 de lo contrario. La operación OR es 1 cuando el uno o el otro píxel es 1 o ambos píxeles son 1.

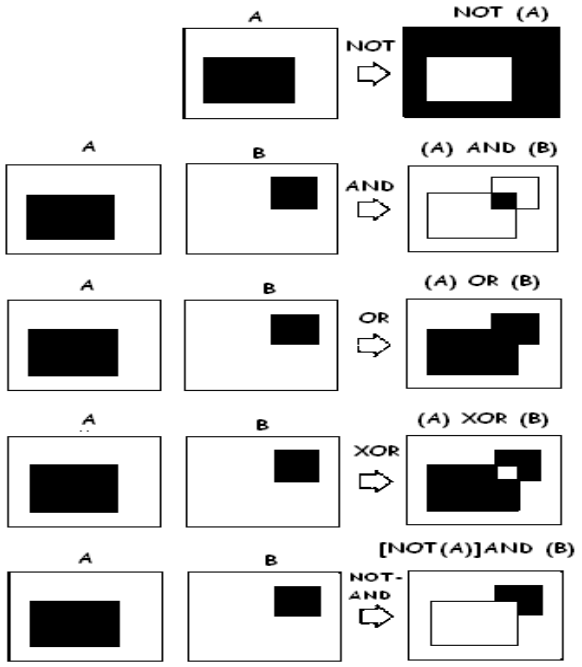


Figura 2.1 Algunos ejemplos de operaciones lógicas en imágenes binarias

Imagen x | Imagen y | x AND y

x NAND y | x OR y | x XOR y

Figura 2.2 Operaciones lógicas entre imágenes

2.2 TRANSFORMACIONES LÓGICAS Y GEOMÉTRICAS.

2.2.1 TRANSFORMACIONES LÓGICAS

La imagen de salida solo posee dos niveles de gris 0 y 255, considerando dichos niveles como el 0 lógico y el 1 lógico respectivamente, obtenemos una imagen binaria lógica.

Con una imagen de estas características es posible realizar sobre ella todo tipo de operaciones lógicas, entre ellas negación or y xor. También dadas dos imágenes, es posible realizar sobre ellas otros tipos de operaciones relacionales tales como $> < \geq$, $\leq$, por ejemplo dadas las imágenes A y B, una nueva imagen C =A $\leq$ B se obtiene realizando la comparación dada píxel a píxel y obteniendo el valor de 1 para C en aquellos píxeles donde se cumple la relación y 0 donde no se cumple.

2.2.2 TRANSFORMACIONES GEOMÉTRICAS

Las transformaciones geométricas nos permiten cambiar la posición, orientación y tamaño de objetos. Se puede definir una transformación geométrica como una función matemática que transforma un punto o conjunto de puntos en otro punto o conjunto de puntos, dentro del sistema de referencia global de la escena.

Es más sencillo representar estas funciones matemáticas como operaciones de matrices, la multiplicación de matrices nos ayuda a realizar las transformaciones geométricas. Para realizar estas operaciones con matrices se emplean coordenadas homogéneas. De esta forma podemos representar la translación, el escalado, la rotación como la multiplicación de un vector (x,y,z) por una matriz de transformación. En particular esta operación representa la combinación del escalado, la translación y la rotación como la multiplicación de dos matrices.

Todas las transformaciones son expresadas en 3D sistema de coordenadas cartesiana, en el cual un punto de coordenadas denota (X;Y;Z) en algunos casos de imágenes en 2D (x,y) que representa las coordenadas de un píxel.

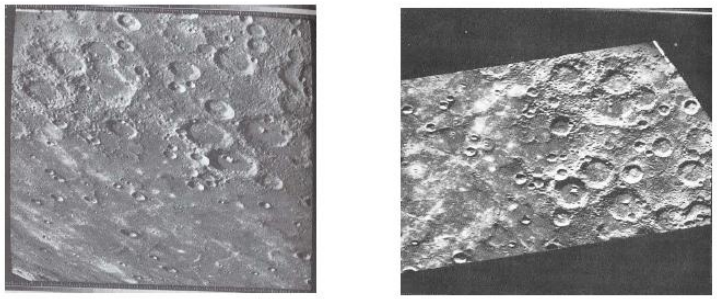


Figura 2.3 Transformaciones Geométricas

2.2.2.1 TRASLACIÓN

La operación de traslación consiste en modificar la posición de un punto en el plano definido por las coordenadas (x, y), mediante un movimiento en línea recta desde la posición actual (posición inicial) a la posición final[13].

Trasladar un punto con coordenadas (X,Y,Z) a una nueva localidad usando desplazamientos (X_0, Y_0, Z_0). La traslación es sumamente fácil usando la ecuación:

$$\begin{aligned} X^* &= X + X_0 \\ Y^* &= Y + Y_0 \\ Z^* &= Z + Z_0 \end{aligned} \qquad (2.2.1)$$

Donde (X*, Y*, Z*) son las coordenadas de un nuevo punto, esta ecuación puede ser expresada en forma de matriz de la siguiente manera:

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (2.2.2)$$

[13] Digital Image Processing Autor Gonzalez R. y Woods R. pag. 52

A menudo es útil encadenar varias transformaciones para producir un resultado compuesto, como la traslación, seguida por el escalar y después la rotación. El uso de matrices cuadrado simplifica la representación rotatoria de este proceso considerablemente. Con esta perspectiva puede escribirse como sigue:

$$\begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (2.2.3)$$

En términos de los valores de X*, Y*, y Z*, ecuaciones (2.2.2 y 2.2.3) son equivalentes en la cual definimos la siguiente matriz

$$\mathbf{v^*=Av} \qquad (2.2.4)$$

Donde A es una matriz transformada 4x4, v es el vector columna contenida por las coordenadas originales

$$v = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (2.2.5)$$

Y v* es el vector columna cuyas componentes son coordenadas trasformadas

$$v^* = \begin{bmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{bmatrix} \qquad (2.2.6)$$

Con esta anotación, la matriz usada para la traslación es

$$T = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.2.7)$$

Y el proceso de traslación es completando usando la ecuación (2.2.4) v*=Tv.

2.2.2.2 ESCALADO

Escalando los factores Sx, Sy, y Sz a lo largo de los ejes X, Y y Z están dados por la matriz transformación[14]:

$$T = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.2.8)$$

2.2.2.3 ROTACIÓN

Las transformaciones usando la rotación 3-D es más compleja que las transformaciones descritas anteriormente. La forma mas simple de estas transformaciones es la de rotación de un punto sobre los ejes de coordenadas.[14] La rotación de un punto sobre otro punto arbitrario en el espacio requiere de tres transformaciones.

1. Traslada un punto arbitrario al origen.

[14] Digital Image Processing Autor Gonzalez R. y Woods R. pag. 53-54

2. Realiza la rotación.
3. Traslada el punto de atrás a la posición original.

Con referencia a la figura 2.4 la rotación de un punto sobre el eje de coordenadas Z por el ángulo θ es determinado usando la siguiente transformación:

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.2.9)$$

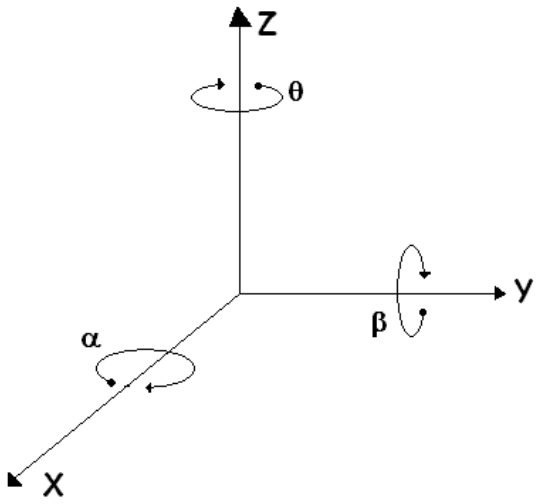


Figura 2.4 Rotación de un punto sobre cada eje de coordenadas

El ángulo de rotación θ es en el sentido de las agujas del reloj, cuando observamos desde el origen a un punto en el eje +Z, esta transformación afecta solamente los valores de coordenadas X y Y.

La rotación de un punto sobre el eje X por un ángulo α es ejecutado usando la transformación.

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.2.10)$$

Finalmente, la rotación de un punto sobre el eje Y y un ángulo β es determinado usando la transformación

$$R_\beta = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.2.11)$$

2.3 INTERPOLACIÓN, DESPLAZAMIENTO Y AMPLIFICACIÓN DE IMÁGENES.

2.3.1 INTERPOLACIÓN

La interpolación se basa en tratar las áreas de una imagen como grupos de puntos coloreados en diferentes configuraciones, la interpolación aprovecha la tendencia que tiene el ojo de desenfocar los puntos de diferentes colores promediando sus efectos y fusionándolos en una única tonalidad o color percibido. Según el coeficiente entre puntos blancos y negros dentro de una determinada superficie, el efecto global será una determinada tonalidad de gris. Del mismo modo, los puntos rojos intercalados con los blancos crean la ilusión de distintos matices de rosado. La interpolación se utiliza para dar realismo a los gráficos informatizados y para suavizar los bordes irregulares de las líneas curvas y diagonales a baja resolución.

La interpolación puede considerarse como el cálculo del valor de intensidad de un píxel, en una posición cualquiera, como función de los píxeles que le rodean (y que

ocuparan las posiciones enteras de la rejilla de destino). Por ejemplo si queremos calcular el valor del píxel de coordenadas (x,y) dado en la figura 2.5, en función de los valores de los píxeles de la rejilla.

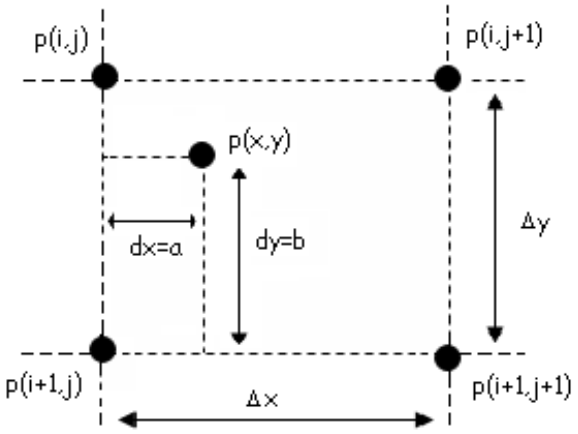


Figura 2.5 Esquema gráfico de la interpolación bilineal

Una forma de hacerlo es suponer que el píxel toma el mismo valor que el más cercano de entre los cuatro que le rodean. Para decir cual es el más cercano se puede utilizar la distancia Euclídea. En la figura tomaría el valor de p(i,j). Otra posibilidad es asociarle la intensidad media asociada a los dos píxeles mas cercanos, una en x y el otro en y. en la misma figura tomaría el valor medio de p(i,j) y p(i+1,j).

2.3.1.1 INTERPOLACIÓN BILINEAL

Otra forma de interpolar con mejores resultados pero con mayor coste computacional es la interpolación bilineal. La cual asigna al píxel en cuestión un valor medio ponderado de las intensidades de los cuatro píxeles que le rodean.[15]

[15] Visión por Computador Imágenes Digitales y Aplicaciones. Autor Pajares G pag. 83

Los factores de ponderación vienen dados por la distancia entre el píxel y los del entorno. Los factores de ponderación se calculan de la manera siguiente,

$$a_1 = \left(1 - \frac{dx}{\Delta x}\right)\left(1 - \frac{dy}{\Delta y}\right); \qquad a_2 = \frac{dx}{\Delta x}\left(1 - \frac{dy}{\Delta y}\right);$$
$$a_3 = \left(1 - \frac{dx}{\Delta x}\right)\frac{dy}{\Delta y} \qquad a_4 = \frac{dy}{\Delta y}\frac{dx}{\Delta x} \tag{2.3.1}$$

Donde $0 \leq dx \leq 1$, $0 \leq dy \leq 1$, $\Delta x = 1$ e $\Delta y = 1$, por lo que obtendríamos.

$$a_1 = (1 - dx)(1 - dy); \quad a_2 = dx(1 - dy);$$
$$a_3 = (1 - dx)dy \qquad a_4 = dxdy \tag{2.3.2}$$

Finalmente, el valor del píxel interpolado, en función de los cuatro de su entorno queda:

$$p(x, y) = a_1 p(i, j) + a_2 p(i, j+1) + a_3 p(i+1, j) + a_4 p(i+1, j+1) \tag{2.3.3}$$

La interpolación del vecino más cercano esta dado por:

$$f_1(x, y) = g_s(alrededor(x), alrededor(y))$$

Se asigna al punto (x, y) el valor de brillo del punto más cercano g en la trama discreta.

En la figura 2.6 se muestra cómo el nuevo brillo es asignado. Las líneas punteadas muestran cómo la transformación del plano inverso traza las rejas de la imagen de salida en la imagen de entrada, las líneas sólidas muestran las rejas de la imagen de entrada.

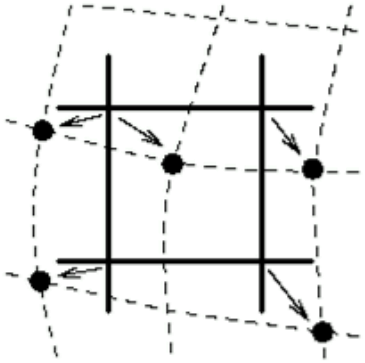


Figura 2.6 Interpolación del vecino mas cercano

2.3.2 DESPLAZAMIENTO

El desplazamiento consiste en reemplazar el píxel $(x+x_d,y+y_d)$ en la imagen resultante por el (x,y) de la imagen original.

2.3.3 AMPLIFICACION DE IMÁGENES

Se trata de seleccionar una pequeña porción de la imagen (subimagen), separada del resto de la imagen original y realizar un zoom mediante una expansión. Este proceso de expansión puede hacerse de muchas formas, pero generalmente se utilizan dos maneras:[16]

1. Consiste en repetir los valores de los píxeles previos, para crear un efecto de bloques.

2. Es más complicado y se utiliza interpolación bilineal, el método de hacerlo mas fácil es encontrar el valor medio entre dos píxeles y usar dicho valor como del píxel entre los dos. Se puede hacer eso por filas de la siguiente manera.

[16] Visión por Computador Imágenes Digitales y Aplicaciones. Autor Pajares G pag. 85

$$\begin{bmatrix} 5 & 3 & 5 \\ 2 & 12 & 4 \\ 6 & 8 & 10 \end{bmatrix} \qquad \begin{bmatrix} 5 & 4 & 3 & 4 & 5 \\ 2 & 7 & 12 & 8 & 4 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

Imagen original Imagen con filas expandidas

$$\begin{bmatrix} 5 & 4 & 3 & 4 & 5 \\ 3.5 & 5.5 & 7.5 & 6 & 4.5 \\ 2 & 7 & 12 & 8 & 4 \\ 4 & 7 & 10 & 8.5 & 7 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

Imagen con filas y columnas expandidas.

Este método nos permite ampliar una imagen de dimensión NxN a otra dimensión (2N-1)x(2N*1) y puede repetirse tantas veces como se desee. Este es el procedimiento para obtener la imagen de la siguiente figura a partir de la subimagen correspondiente.

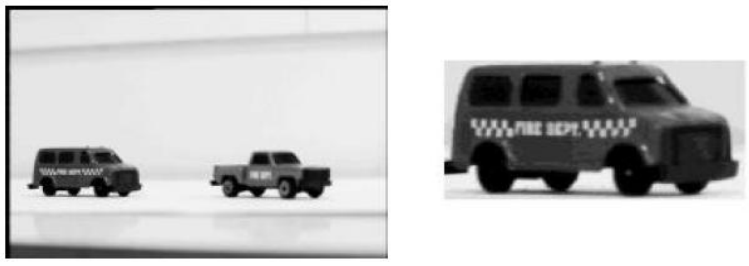


Figura 2.4 a) Imagen original b) Imagen Amplificada

Otro método que consigue un resultado similar, requiere un promedio del entorno de vecindad. Con este método de expansión, se requiere en dos pasos.

1. Extender la imagen añadiendo filas y columnas de ceros entre las filas y columnas existentes:

$$I = \begin{bmatrix} 2 & 6 & 4 \\ 8 & 10 & 2 \\ 4 & 6 & 8 \end{bmatrix} \rightarrow I_{\exp} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 6 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 10 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 6 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2. Realizar el promediado, llamado también convolución.

$$I = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} \rightarrow I_{amp} = h * I_{\exp} \begin{bmatrix} 0.5 & 1 & 2 & 3 & 2.5 & 2 & 1 \\ 1 & 2 & 4 & 6 & 5 & 4 & 2 \\ - & - & - & - & - & - & - \\ - & - & - & - & - & - & - \\ - & - & - & - & - & - & - \\ - & - & - & - & - & - & - \\ - & - & - & - & - & - & - \end{bmatrix}$$

CAPITULO III

MEJORAMIENTO DE IMAGEN

3.1 INTRODUCCIÓN.

El principal objetivo de las técnicas de mejoramiento de imagen, es procesar una imagen con el fin de hacerla más adecuada para una determinada aplicación o procesamiento posterior. Los métodos de mejora de imagen se pueden dividir en dos campos diferentes: métodos en el dominio frecuencial y métodos en el dominio espacial. Los primeros se basan en modificar la transformada de Fourier de la imagen, mientras que los segundos se basan en manipulaciones directas sobre los píxeles de la imagen.

3.2 SUAVIZADO Y REALZADO.

Las operaciones de suavizado y realzado son útiles para reducir el ruido y otros efectos no deseados que pueden estar presentes en una imagen digital como resultado del muestreo, cuantización y transmisión o por perturbaciones en el sistema tales como partículas de polvo en el sistema óptico. El realzado de imágenes resulta de interés para distinguir mejor los detalles subyacentes en la imagen original, pero que no se aprecian con la nitidez que sería deseable.

3.2.1 PROMEDIADO DEL ENTORNO DE VECINDAD

Es una técnica directa en el dominio espacial. Dada una imagen g(x,y), se obtiene una imagen suavizada f(x,y) cuya intensidad para cada punto (x,y) se obtiene promediando los valores de intensidad de los píxeles de g incluidos en el entorno de

vecindad predefinido de (x,y), la operación puede sintetizarse en la siguiente expresión:

$$f(x, j) = \frac{1}{P} \sum_{(m,n)\in S} g(x, y) \tag{3.1}$$

Para todos los píxeles (x,y) de g(x,y). S es el conjunto de coordenadas de los puntos situados en el entorno de vecindad de (x,y) incluido el propio (x,y) y P es el número total de puntos del entorno de vecindad.

Por ejemplo si el entorno de vecindad S es una ventana de dimensión 3x3 la operación realizada a través de (3.1) sería la misma que si la imagen hubiese sido filtrada con el núcleo del filtro pasa bajo siguiente.

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.2}$$

En el siguiente ejemplo, el píxel de ruido (intensidad 1) se suaviza como aparece en la imagen resultante

$$\begin{bmatrix} 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & \bullet 1 & 8 & 8 \\ 18 & 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \end{bmatrix} \Rightarrow \text{suavizado} \Rightarrow \begin{bmatrix} 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & \bullet 7 & 8 & 8 \\ 18 & 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 & 8 \end{bmatrix}$$

Un caso particular del promediado es el suavizado guassiano en el cual el núcleo es una gaussiana 2-D de media 0 y desviación estándar σ

$$G(x, y) = e^{-(x^2+y^2)/2\sigma^2} \tag{3.3}$$

El suavizado gaussiano puede implementarse eficientemente gracias al hecho de que el núcleo es separable:

$$I_G = I * G = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} G(h,k) I(i-k, j-k) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} e^{\frac{h^2+k^2}{s\sigma^2}} I(i-k, j-k)$$

$$= \sum_{h=-m/2}^{m/2} e^{\frac{h^2}{s\sigma^2}} \sum_{k=-m/2}^{m/2} e^{\frac{k^2}{s\sigma^2}} I(i-k, j-k) \qquad (3.4)$$

Esto significa que la convolución de una imagen I con el núcleo gaussiano 2-D se puede realizar por convolución de todas las filas y luego todas las columnas con una Gaussiana 1-D con idénticaσ.

3.2.2 SUAVIZADO BINARIO DE IMÁGENES

Las imágenes binarias están formadas por 2 únicos valores lógicos “0” y “1”. El ruido en este caso produce efectos tales como contornos irregulares pequeños huecos, esquinas perdidas y puntos aislados.[17]

La idea básica de este método consiste en evaluar una función booleana específica sobre un entorno de vecindad centrado sobre un píxel x y dependiendo de la configuración espacial y los valores binarios de sus vecinos, asignarle a x un “0” o un “1”. La configuración espacial, es una mascara 3x3, el procedimiento de suavizado tiene los siguientes efectos:

1. Rellenar los pequeños huecos de un píxel en zonas oscuras.
2. Rellena los pequeños cortes y muestra en segmentos de lados rectos.
3. Elimina los unos aislados.
4. Elimina las pequeñas protuberancias a lo largo de los segmentos de los lados rectos.
5. Repone los puntos perdidos de las esquinas.

[17] Visión por Computador Imágenes Digitales y Aplicaciones. Autor Pajares G pag. 105

3.3 HISTOGRAMA DE UNA IMAGEN.

El histograma de una imagen es una herramienta visual de gran aceptación y utilidad para el estudio de imágenes digitales. Con una simple mirada puede proporcionar una idea muy aproximada de la distribución de niveles de gris, el contraste que presenta la imagen y alguna pista del método más adecuado para manipularla.

El histograma es un gráfico que ofrece una descripción global de la apariencia de la imagen. En el eje de abscisas se representa el rango de valores de píxeles de la imagen, mientras que en el eje de ordenadas se representa el rango de valores que pueden tomar esos píxeles.

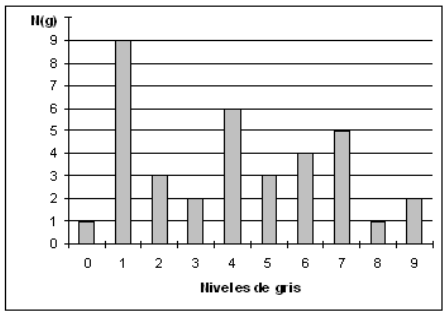
El histograma es una de las formas más comunes de representar la distribución de los niveles de gris de una imagen. El histograma provee la información de cuantos píxeles poseen un determinado nivel de gris en la imagen en un intervalo definido entre 0 (negro) y 255 (blanco) para una imagen de 8 bits, ofreciendo datos importantes como la intensidad media y la dispersión de los valores de los niveles de gris, siendo esta última la medida de contraste de la imagen. Cuanto mayor es la dispersión a lo largo del eje que representa los niveles de gris mayor es el contraste de la imagen.

El histograma de una imagen es una función discreta que representa el número de píxeles en la imagen en función de los niveles de intensidad, g, la probabilidad P(g) de ocurrencia de un determinado nivel g se define como,

$$P(g) = \frac{N(g)}{M} \tag{3.11}$$

Donde M es el número de píxeles en la imagen y N(g) es el número de píxeles en el nivel de intensidad g. Como con cualquier distribución de probabilidad todos los valores de P(g) son menores o iguales que 1 y la suma de todos los valores de P(g) es 1, por ejemplo dada la imagen de la figura 3.1(a) con 10 niveles de gris del 0 al 9; su histograma se muestra en (b).

$$\begin{bmatrix} 0 & 3 & 3 & 4 & 4 & 4 \\ 1 & 1 & 1 & 4 & 4 & 5 \\ 1 & 1 & 1 & 4 & 5 & 5 \\ 1 & 1 & 1 & 7 & 6 & 6 \\ 2 & 2 & 7 & 7 & 6 & 6 \\ 8 & 2 & 7 & 7 & 9 & 9 \end{bmatrix}$$



(a) **(b)**

Figura 3.1(a) Imagen con 10 niveles de gris del 0 al 9; (b) Su histograma

3.4 BRILLO CONTRASTE Y CORRECCIÓN GAMMA.

En general un histograma con una distribución de los niveles de gris concentrada en una determinada zona presenta un contraste muy bajo, mientras que un histograma con una amplia distribución de los niveles de gris concentrados en la parte baja del rango corresponde a una imagen oscura, un histograma con los valores concentrados en la parte alta del mismo corresponde a una imagen brillante. El histograma puede ser modificado mediante una serie de funciones que expandan los niveles de gris, los compriman o los desplacen.

3.4.1 BRILLO

El brillo nos indica el nivel de gris o intensidad que poseen los píxeles de una imagen, es decir, es el valor de ***f(x,y)*** en cada punto de la imagen. Valores grandes de brillo representan tonos claros, mientras que valores pequeños representan tonos oscuros.

Cuando se incrementa el brillo, el valor de cada píxel se acerca más a 255 (blanco), y cuando se disminuye, el valor de cada píxel se reduce más cerca del 0 (negro).

Variar el nivel de brillo de una imagen consiste en sumar o restar a todos los píxeles un nivel constante de gris. Por ejemplo en una imagen de 8 bits, el nivel de brillo a sumar o restar tomará valores comprendidos entre [0,255]. La expresión que calcula el brillo de una imagen de niveles de grises es la siguiente:

$$g(x,y) = f(x,y) + brillo \qquad (3.12)$$

El resultado de aplicar esta ecuación, aumentando el brillo de la imagen es que, dicha imagen se vuelve más clara y el histograma resultante aparece desplazado hacia la derecha en el valor de brillo, que es el parámetro que se le pasa a la función. En cambio, al aplicar en este caso la función reduciendo el brillo de la imagen, lo que produce es una imagen más oscura, y un histograma desplazado hacia la izquierda también en el valor de brillo.

3.4.2 CONTRASTE

El contraste es un término usado para denotar el grado de diferencia entre los componentes más oscuros y los más claros en una imagen, también muestra las variaciones locales del brillo, este muestra la separación entre los niveles de gris. El contraste entre dos objetos se puede definir como la razón entre sus niveles de gris medios.

La variación en el contraste aparece como una compresión del histograma en caso de una reducción, o una expansión en caso de aumento de contraste. Al igual que el brillo, variando el contraste se puede provocar una pérdida de información de la imagen.

Una imagen tiene menor contraste, si contiene solamente transiciones bruscas entre el blanco y el negro, o contiene valores dentro de una gama estrecha. Una imagen tiene un buen contraste si se compone de una amplia gama de valores desde el negro al blanco.

3.4.3 CORRECCIÓN GAMMA

La corrección gamma es una forma especial de aumento de contraste, diseñada para mejorar el contraste en áreas muy claras o muy oscuras. Esto se logra modificando los valores medios, particularmente los medios-bajos, sin afectar el blanco (255) ni el negro (0). Puede utilizarse para mejorar el aspecto de una imagen, o para compensar el rendimiento de diferentes dispositivos frente a una imagen.[18]

Las funciones de corrección gamma, se usan frecuentemente en el procesado de imágenes para compensar respuestas no lineales en sensores de imágenes, pantallas y películas. Se aplica, por ejemplo, en las pantallas CRT (Tubos de Rayos Catódicos), para que la luz emitida no sea proporcional al voltaje recibido.

La expresión matemática de esta transformación es:

$$g(x,y) = f(x,y)^{1/\gamma} \qquad (3.13)$$

donde el parámetro γ toma normalmente valores comprendidos entre 0 y 2.

La corrección gamma define el grado de blanco o intensidad en una imagen; los valores más bajos producen una imagen más oscura mientras que los más altos, una imagen más clara. Un valor de 1.0 se traduce en una transformación nula, un valor gamma menor que 1.0 pero mayor que 0.0 crea curvas exponenciales que oscurecen la imagen. Valores gamma mayores que 1.0 producen curvas logarítmicas que aclaran la imagen.
Por ejemplo, un valor gama de 1 es equivalente a la curva de intensidad original, un aumento en este valor, aclara la imagen y aumenta el contraste en las áreas más oscuras. Un valor menor a 1 la oscurece y enfatiza el contraste en las áreas mas claras.

[18] http://pub.ufasta.edu.ar/SISD/vision/Gamma.htm

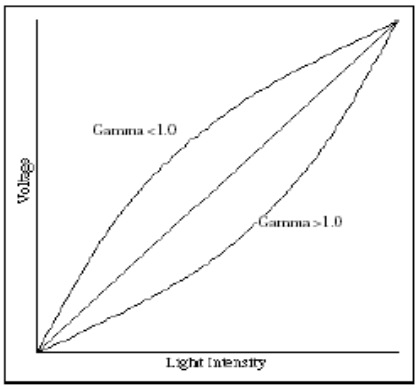


Figura 3.2 Corrección Gamma

Las imágenes que no son corregidas pueden parecer muy oscuras o muy claras para reproducir los colores correctamente también de la corrección gamma ya que esta no solo cambia el brillo sino también las relaciones entre rojo verde y azul.

3.5 MANEJO DE HISTOGRAMA

3.5.1 CONTRACCIÓN DEL HISTOGRAMA

Esta técnica no produce realmente un realzado de la imagen, por el contrario, produce una disminución del contraste de la imagen, no obstante para estudiar su efecto y la mejora que la expansión del histograma produce se introduce esta técnica.[19]

La función que la define es la siguiente:

$$g(x,y) = \left[\frac{C_{MAX} - C_{MIN}}{f(x,y)_{MAX} - f(x,y)_{MIN}} \right] \left[f(x,y) - f(x,y)_{MIN} \right] + C_{MAX} \qquad (3.14)$$

[19] Visión por Computador Imágenes Digitales y Aplicaciones. Autor Pajares G pag. 102

Donde f(x,y) es el nivel de gris de la imagen de entrada; f(x,y) MAX es el mayor valor del nivel de gris en la imagen de entrada f; f(x,y) MIN es el menor valor del nivel de gris en la imagen de entrada f; CMAX y CMIN corresponden al máximo y mínimo valores deseados en la compresión del histograma.

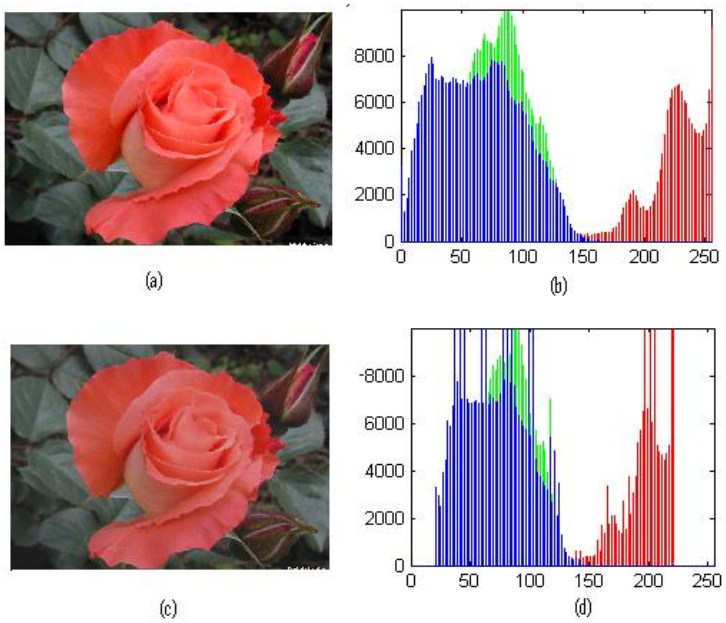


Figura 3.3 Contracción del histograma.

En la figura 3.3 (a) la imagen original, (b) su histograma, (c) la imagen resultante tras aplicar la contracción (d) su histograma.

3.5.2 EXPANSIÓN DEL HISTOGRAMA

Es la operación opuesta a la contracción del histograma. Una función para expandir los niveles de gris de un histograma se puede definir como sigue:

$$g(x,y) = \left[\frac{f(x,y) - f(x,y)_{MIN}}{f(x,y)_{MAX} - f(x,y)_{MIN}} \right] [MAX - MIN] + MIN \qquad (3.15)$$

Donde f(x,y) es el nivel de gris de la imagen de entrada; $f(x,y)_{MAX}$ es el mayor valor del nivel de gris en la imagen de entrada f; $f(x,y)_{MIN}$ es el menor valor del nivel de gris en la imagen de entrada f; MAX y MIN corresponden al máximo y mínimo valores posibles de los niveles de gris (para una imagen de 8 bits sería 0 y 255).[20]

Esta ecuación toma una imagen de entrada f y expande el histograma a lo largo del rango de valores completo de los niveles de gris. Esto tiene el efecto de incrementar el contraste de una imagen de bajo contraste. Si se desea que la expansión no cubra el rango total posible de niveles de gris, se pueden especificar diferentes valores para MAX y MIN.

Si muchos valores de una imagen caen dentro de un pequeño rango y existe un cierto número de valores extremos, entonces el histograma abarca todo el rango de valores y una expansión pura del histograma no mejora la imagen. En este caso suele ser una práctica habitual recortar los niveles de gris en los extremos a los valores de gris mas bajo y más alto del rango (para una imagen de 8 bits serían 0 y 255).

La figura 3.4 (a) imagen original, en (b) su histograma, (c) muestra la imagen resultante tras aplicar una expansión del histograma a la imagen, en (d) el histograma resultante de dicha expansión.

[20] Visión por Computador Imágenes Digitales y Aplicaciones. Autor Pajares G pag. 103

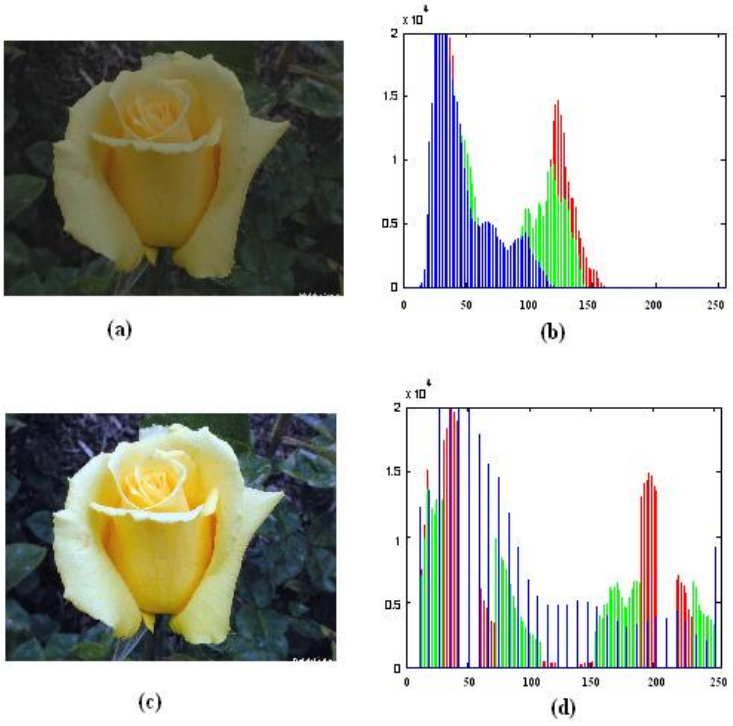


Fig 3.4 Expansión del histograma

3.5.3 DESPLAZAMIENTO DEL HISTOGRAMA

El desplazamiento del histograma se usa para aclarar u oscurecer una imagen, manteniendo la relación entre los valores de los niveles de gris. Esta operación puede llevarse a cabo por la simple adición o sustracción de un número fijo a todos los valores del nivel de gris:[21]

$$g(x,y) = f(x,y) + DES \qquad (3.16)$$

Donde DES es el valor para desplazar el histograma.

En esta ecuación se asume que los valores que sobrepasen el máximo y el mínimo se redondean al máximo y mínimo posibles permitidos. Un valor *DES* positivo incrementa el brillo de la imagen, mientras un valor negativo la oscurece.

[21] Visión por Computador Imágenes Digitales y Aplicaciones. Autor Pajares G pag. 104.

En la figura 3.5, en (a) imagen original, en (b) su histograma, en (c) imagen resultante tras aplicar el desplazamiento hacia la izquierda(mejora de contraste), en (d) su histograma resultante tras aplicar dicho desplazamiento, (e) imagen resultante tras aplicar el desplazamiento hacia la derecha (mejora del brillo), en (f) su histograma.

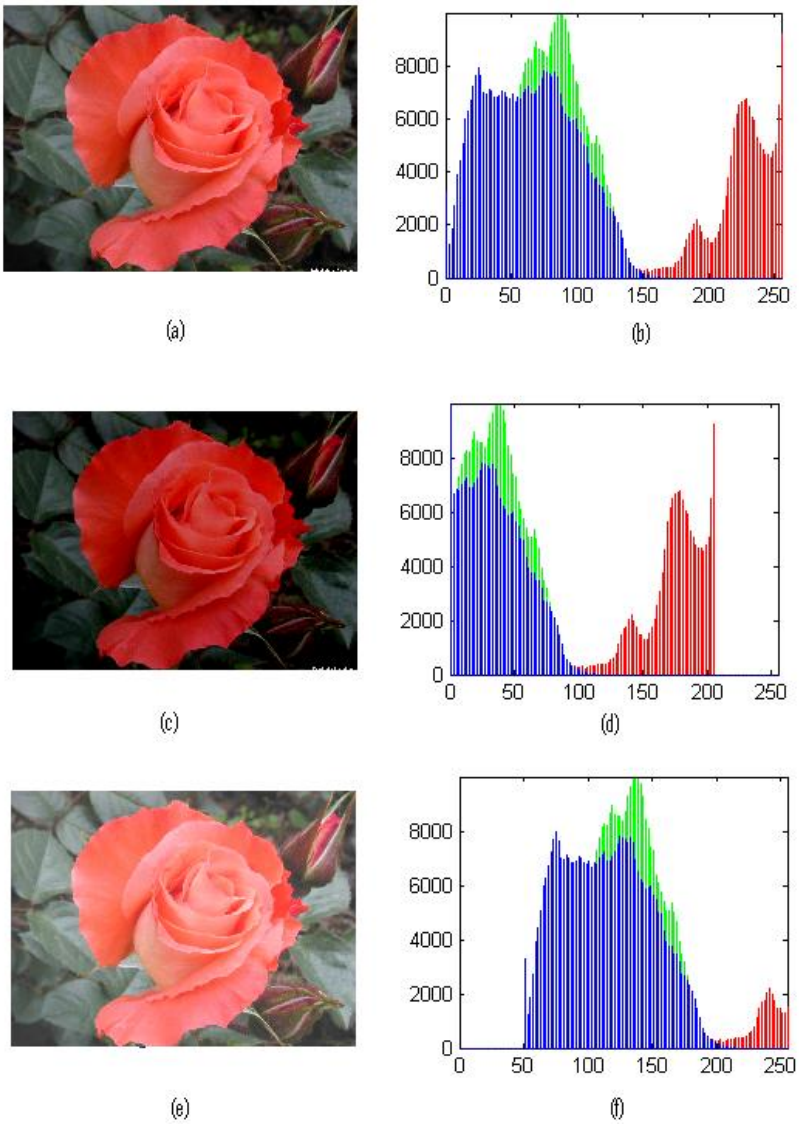


Figura 3.5 Desplazamiento del Histograma

3.5.4 ECUALIZACIÓN DEL HISTOGRAMA

Es una de las técnicas más utilizadas para la mejora del contraste de la imagen original es la de igualación de histogramas. Esta técnica realza la imagen original mediante una determinada transformación o modificación del histograma denominada igualación o ecualización indistintamente. Se trata de encontrar una función F(g) que realce el contraste general en la imagen original expandiendo la distribución de los niveles de gris.[22]

Dicha expansión debe ser lo más suave posible en el sentido de que idealmente debería haber el mismo número de píxeles por niveles de gris, es decir el objetivo es distribuir los niveles de gris de una manera uniforme a lo largo de todo el rango de valores de niveles de gris.

Se puede deducir la función de F(g) mediante la simple inspección del histograma original, pero es deseable una función analítica.

Dado que el número de píxeles en una imagen de dimensión NxM es precisamente este producto y el número de niveles de gris sobre el cual se va a realizar la expansión es N_g, un histograma ideal sería plano, con el mismo número de píxeles en cada nivel de gris, es decir

$$número ideal de píxeles en cada nivel de gris = \frac{NxM}{N_g} \qquad (3.17)$$

Por otra parte a partir del histograma podemos definir la función de densidad de probabilidad como sigue:

1.- Suponiendo por cada nivel de gris en el rango 0 a 255, se cumple:

$$\sum_{g=0}^{g=255} N(g) = NxM \qquad (3.18)$$

[22] Visión por Computador Imágenes Digitales y Aplicaciones. Autor Pajares G pag. 105.

2.- La probabilidad por cada nivel de gris g viene dada por

$$p(g) = \frac{N(g)}{NxM}; \qquad g = 0,1,2,.....255 \tag{3.19}$$

Se define fácilmente que $\sum_{g=0}^{g=255} p(g) = 1.$ (3.20)

3.- La función de densidad de probabilidad resulta ser:

$$P_x(x) \cong \sum_{g=0}^{g=x} p(g) \tag{3.21}$$

4.- Se trata de realizar una transformación entre funciones de densidad de probabilidad $P_x(x)$ y $P_y(y)$, si se impone la condición de que la función de transformación es monótona creciente, para cada valor de x e y se cumple.

$$\int_0^{F(g)} P_y(y)dy = \int_0^{g} P_x(x)dx = \sum_{g=0}^{x} p(g) \tag{3.22}$$

Esta es la expresión general que se utiliza para generar las ecualizaciones.

3.5.4.1 ECUALIZACIÓN UNIFORME

En cualquier histograma, se pretende modificar de tal forma que en la imagen resultante los niveles de gris se repartan de forma equitativa en todo el rango de valores establecidos, es decir 0 a 255. La función de densidad de probabilidad de una distribución uniforme es:

$$P_y(y) = \frac{1}{255}; \quad 0 \le y \le 255 \tag{3.23}$$

Aplicando la expresión general:

$$\int_0^{F(g)} \frac{1}{255} dy = \frac{F(g)}{255} = \sum_{g=0}^{g} p(g) \tag{3.24}$$

Por lo que despejando F(g) se obtiene.

$$F(g) = 255 \sum_{g=0}^{g} p(g) \tag{3.25}$$

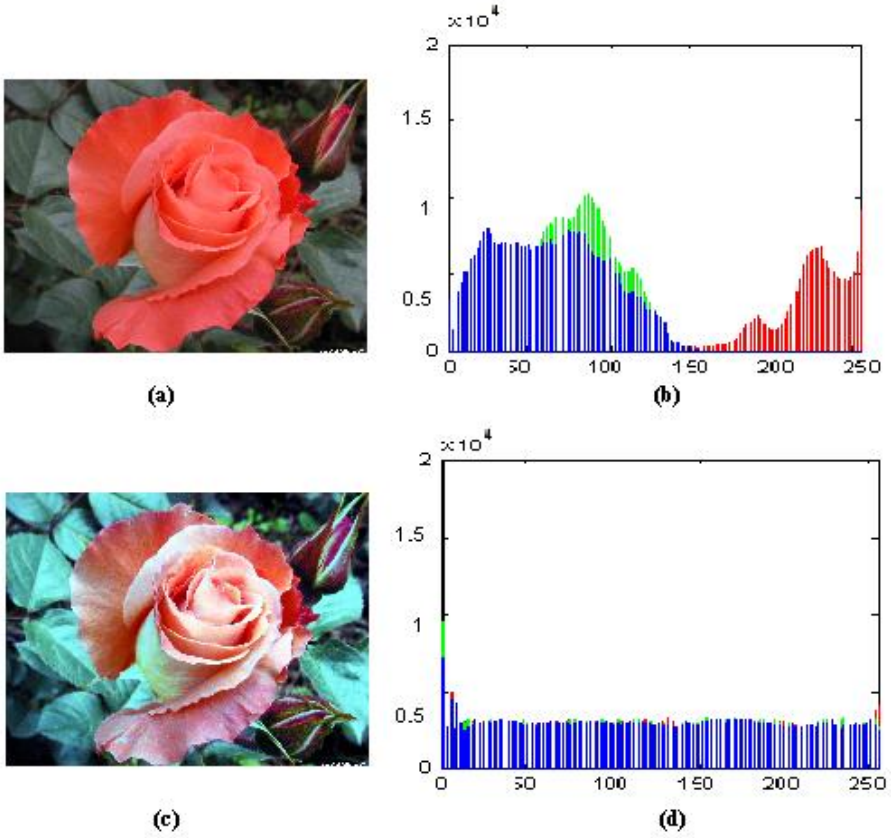


Figura 3.7 Desplazamiento del Histograma

En la figura 3.7 en (a) la imagen original, (b) su histograma, (c) la imagen resultante tras aplicar la ecualización (d) su histograma.

CAPITULO IV

DESARROLLO DE LA INTERFAZ

4.1 RECOLECCIÓN DE INFORMACIÓN

Las técnicas de mejoramiento de imágenes se usan para mejorar una imagen, donde "Mejorar" es definido objetivamente, como *incrementar la relación señal -a- ruido*, y subjetivamente, como *los cambios que modifican los colores o las intensidades.*[23]

Una imagen compuesta de m filas y n columnas de diferentes puntos de color, Matlab guarda como una matriz de mxn. Para imágenes, como imágenes de truecolor o RGB, requieren una matriz tridimensional, donde el primer plano en la tercera dimensión representa las intensidades de los píxeles rojos, el segundo plano representa las intensidades de los píxeles verdes, y el tercer plano representa las intensidades de píxeles azules. Esta conversión hace que MATLAB pueda trabajar con este tipo de imágenes, para las aplicaciones de procesamiento de imágenes a color.[23]

Una imagen RGB, también llamada imagen en color verdadero (truecolor), son almacenadas en Matlab como una matriz mxnx3 que definen los componentes rojo, verde y azul de cada píxel. En estas imágenes, el color se forma mediante la

[23] Matlab Help

combinación de las intensidades almacenadas en cada plano de color para cada píxel.[24] [25]

Una matriz que representa a una imagen RGB puede ser de tipo double, uint8 o uint16, en una matriz RGB de tipo double, cada componente de color es un valor entre cero y uno.

Un píxel con componentes de color (0,0,0) es un punto negro, y un píxel con componentes de color (1,1,1) es un punto blanco. Los tres componentes de color para cada píxel son guardados a lo largo de la tercera dimensión de la matriz de datos. Por ejemplo, los componentes de color rojos, verdes, y azules del píxel (10,5) son guardados en RGB (10,5,1), RGB (10,5,2), y RGB (10,5,3), respectivamente.[25]

En la siguiente tabla se describe el tipo de imágenes con las que trabaja Matlab.[26]

Tipo	Descripción
Double	Doble precisión, números en punto flotante que varían en un rango aproximado de -10^{308} a 10^{308} (8 bytes por elemento)
uint8	Enteros de 8 bits en el rango de [0,255] (1 byte por elemento)
uint16	Enteros de 16 bits en el rango de [0, 65535] (2 bytes por elemento)
uint32	Enteros de 32 bits en el rango de [0, 4294967295] (4 bytes por elemento)
int8	Enteros de 8 bits en el rango de [-128, 127] (1 byte por elemento)
int16	Enteros de 16 bits en el rango de [-32768, 32767] (2 bytes por elemento)
int32	Enteros de 32 bits en el rango de [-2147483648,2147483647] (4 bytes por elemento)
Single	Número en punto flotante de precisión simple, con valores aproximadamente en el rango de -10 a 10 (4 bytes por elemento).
Char	Caracteres (2 byte por elemento)
logical	Los valores son 0 ó 1 (1 byte por elemento)

Tabla 4.1 Tipos de imágenes con MATLAB

[24] http://www.cs.dartmouth.edu/farid/tutorials/fip.pdf
[25] Image Processing Toolbox, Matlab Help
[26] Grupo de Visión Artificial Autor García Pérez David

Los formatos de imágenes para leer, escribir, y visualizar, para una fácil manipulación y análisis con Matlab son los siguientes.[27]

- BMP (Microsoft Windows Bitmap)
- JPEG (Joint Photographic Experts Group)
- PCX (Paintbrush)
- PNG (Portable Network Graphics)
- TIFF (Tagged Image File Format)

Para el desarrollo del tema se utilizaran los formatos BMP, TIFF, JPEG.

La GUIA de Matlab, contiene un conjunto de herramientas para crear interfaces gráficas parecidas a las de Windows, las mismas que simplifican el proceso de creación y de programación.

Para la programación es necesario conocer lo siguiente:[28]

- Controles de flujo: if, switch and case, for, while, continue, and break.

- Estructura de datos: matrices multidimensionales, caracteres y datos de textos.

- Scripts y Funciones.- scripts y funciones, variables globales, argumentos para funciones, la función *eval* para evaluar expresiones de textos, funciones de referencia usando handles.

4.2 ANÁLISIS DE REQUERIMIENTOS.

Para el diseño e implementación de la interfaz se necesita un computador Pentium 4 con memoria RAM de 1Gb, una tarjeta de video, puesto que el microprocesador esta

[27] http://www.gts.tsc.uvigo.es/pi/practicas/intro_toolbox.pdf.
[28] Matlab Help

destinado para la parte matemática, el paquete Matlab, el compilador Borlan C o a su vez el compilador interno de Matlab.

Se requiere manejar imágenes tridimensionales, donde los valores de los píxeles pueden ser enteros, reales (que valen entre 0 y 255). Por ello, se hará uso del programa Matlab ya que este permite trabajar con este tipo de imágenes y una fácil manipulación de información de la imagen.

4.3 DISEÑO DE LA INTERFAZ EN MATLAB.

Para el diseño de la interfaz grafica, se debe considerar lo siguiente: [29][30]

- Que la interfaz debe ser amigable y de fácil manipulación para el usuario.
- El primer paso para implementar un GUI es programar el callback del objeto usado para construir la interfaz.
- Las estructuras handles dan un fácil acceso del manejador en el GUI.
- Todos los códigos incluyendo callbacks están contenidos en M-File.

Se diseña la interfaz a través del GUIDE de Matlab, usando ejes, textos, botones, botones de opción, slider, en las figuras siguientes las pantallas diseñadas para la aplicación, en la figura 4.1 se indica el diseño de la pantalla inicial o de ingreso al software.

La pantalla de la figura 4.2 esta diseñada con textos estáticos y botones.

La pantalla mostrada en la figura 4.3 esta diseñada con textos estáticos, ejes donde se visualizaran las imágenes a tratarse y la resultante, textos editables, botones de opción y botones ejecutables.

La pantalla de la figura 4.4, esta diseñada con textos estáticos, ejes, paneles y botones.

[29]Diseño e Implementación de un sistema de adquisición de datos para Instrumentación Virtual utilizando un microcontrolador PIC Autor Meythaler Naranjo A. Pag 136

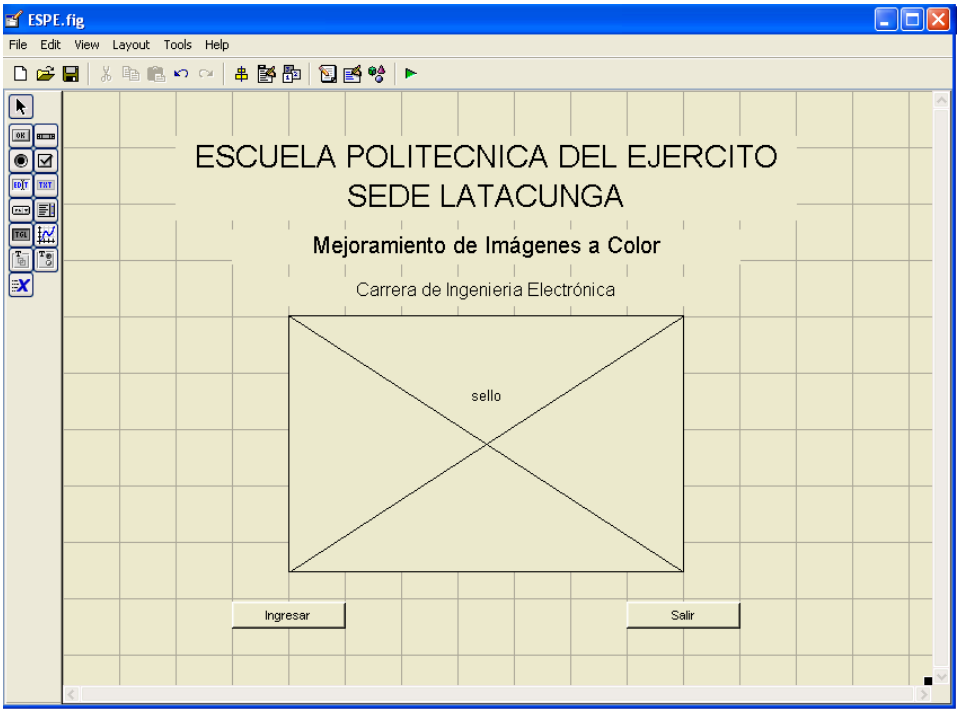


Figura 4.1 Pantalla Principal

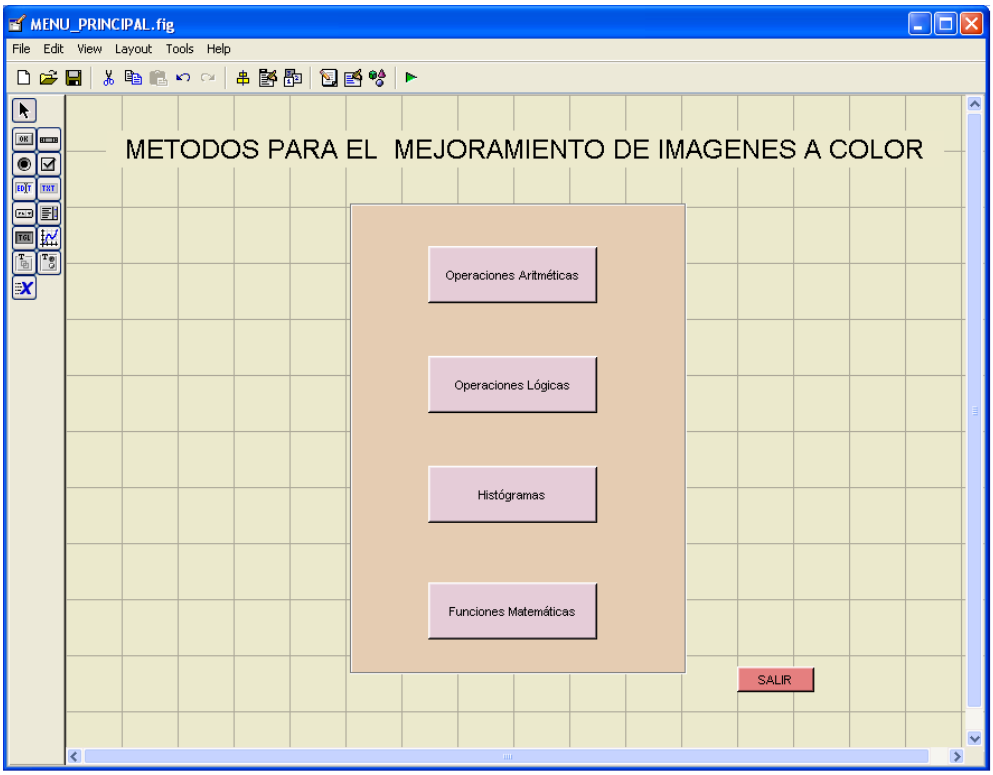


Figura 4.2 Pantalla Menú Principal

Figura 4.3 Pantalla Operaciones Aritméticas

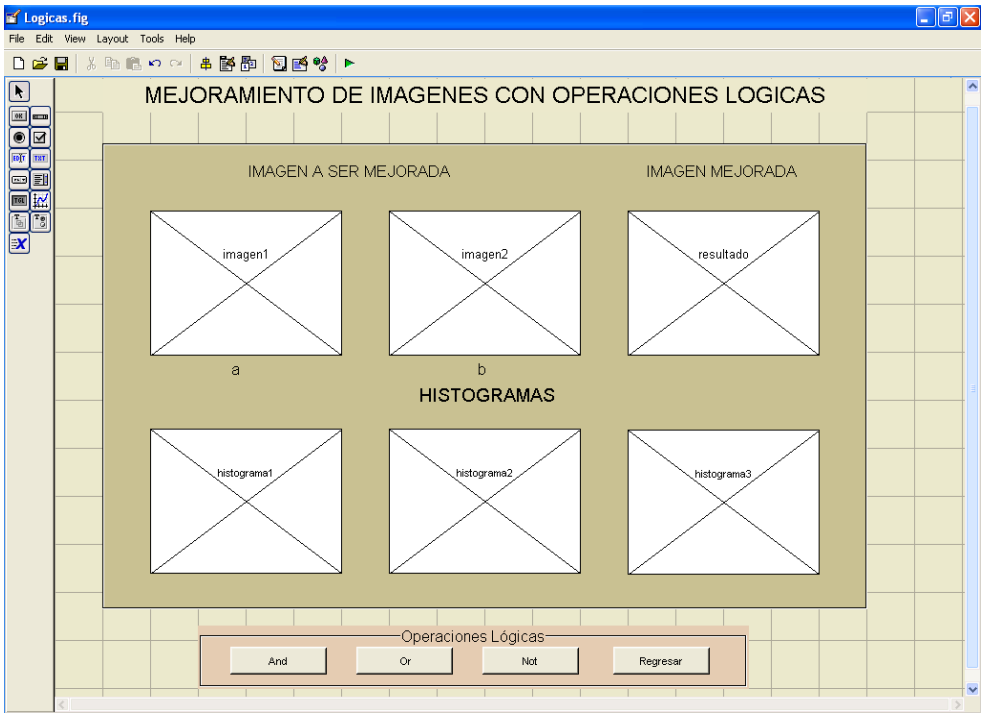


Figura 4.4 Pantalla Operaciones Lógicas

En la ventana mostrada en la figura 4.5, esta diseñada con textos estáticos, slider, botones de opción, ejes, y botones de presionar.

La ventana que se muestra en la figura 4.6 esta diseñada con textos estáticos, ejes, y botones, slider, paneles.

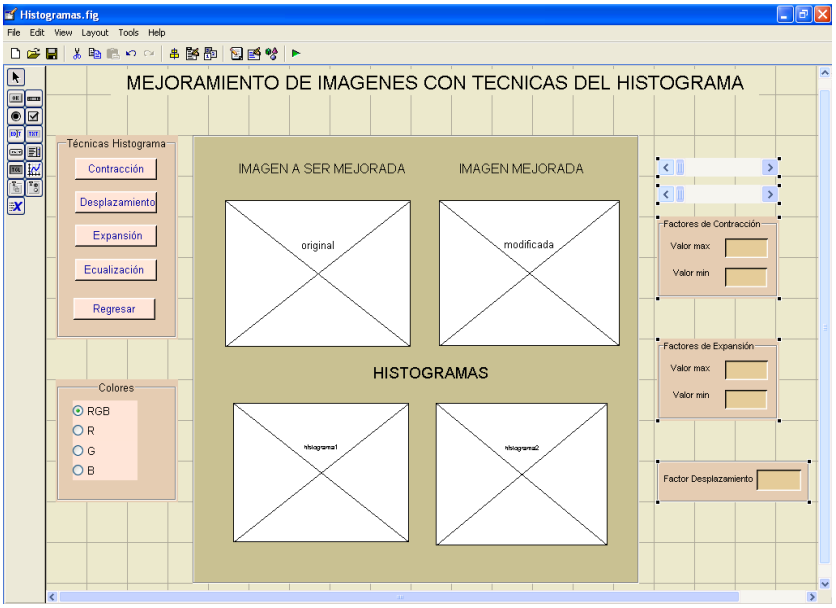


Figura 4.5 Pantalla Técnicas de Histogramas

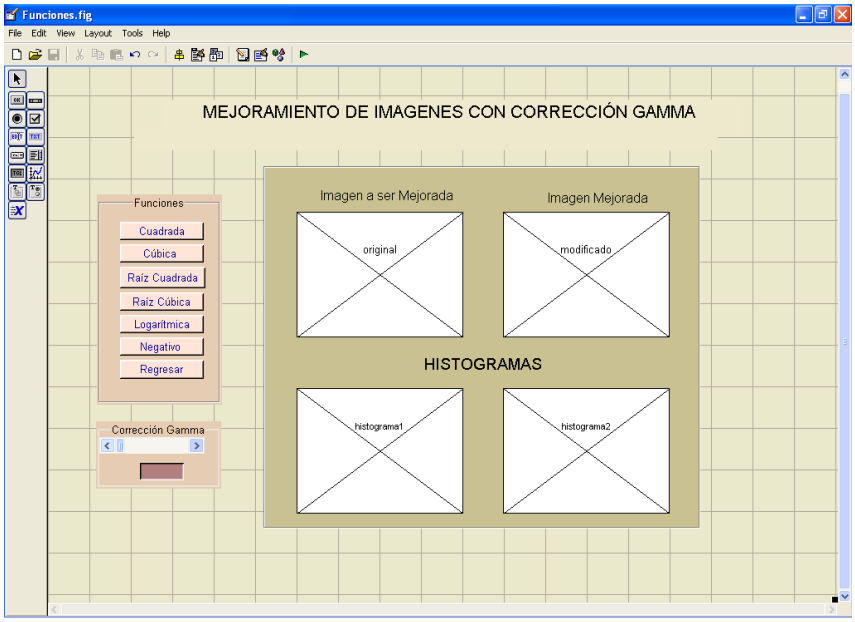


Figura 4.6 Pantalla Corrección Gamma

Para ayuda del diseño de las pantallas anteriores y mayor comprensión para la implementación se detallan a continuación los diagramas de flujo de cada pantalla.

Diagrama de flujo pantalla Principal

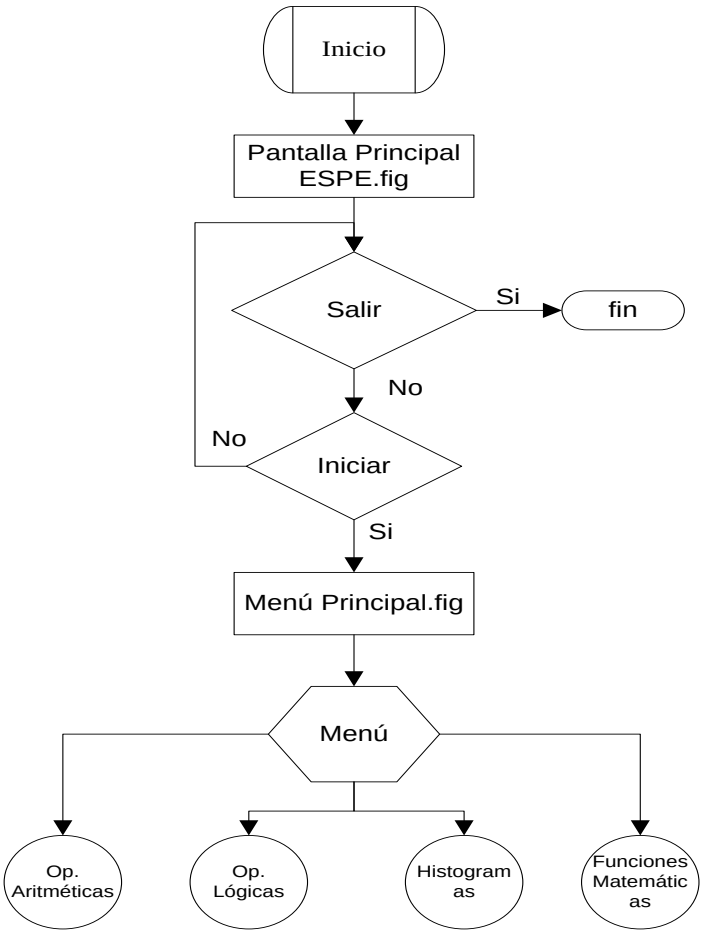


Figura 4.7 Diagrama de flujo de la Pantalla Principal

Diagrama de flujo pantalla Operaciones Aritméticas

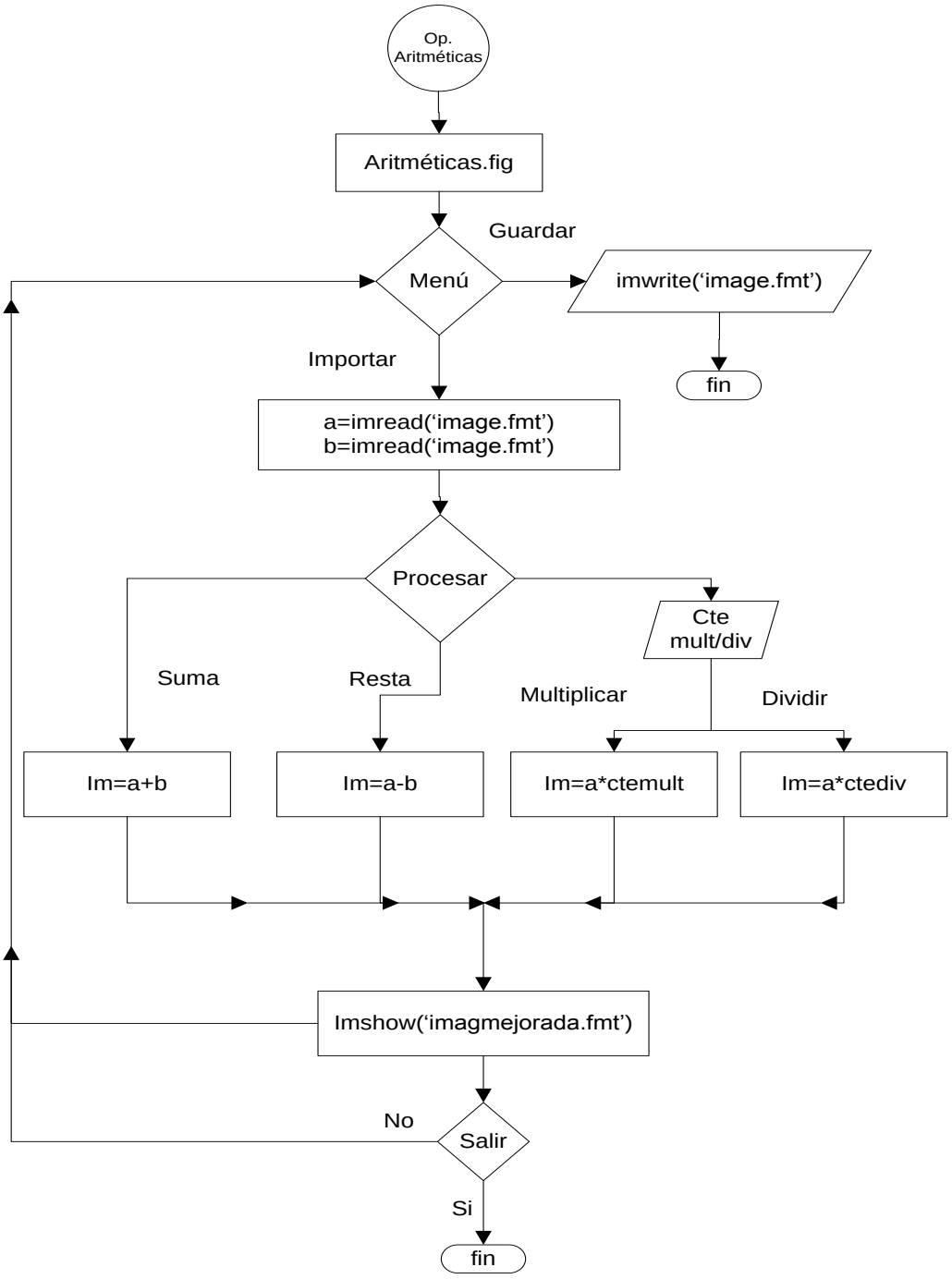


Figura 4.8 Diagrama de flujo de la Ventana Operaciones Aritméticas

Diagrama de flujo pantalla Operaciones Lógicas

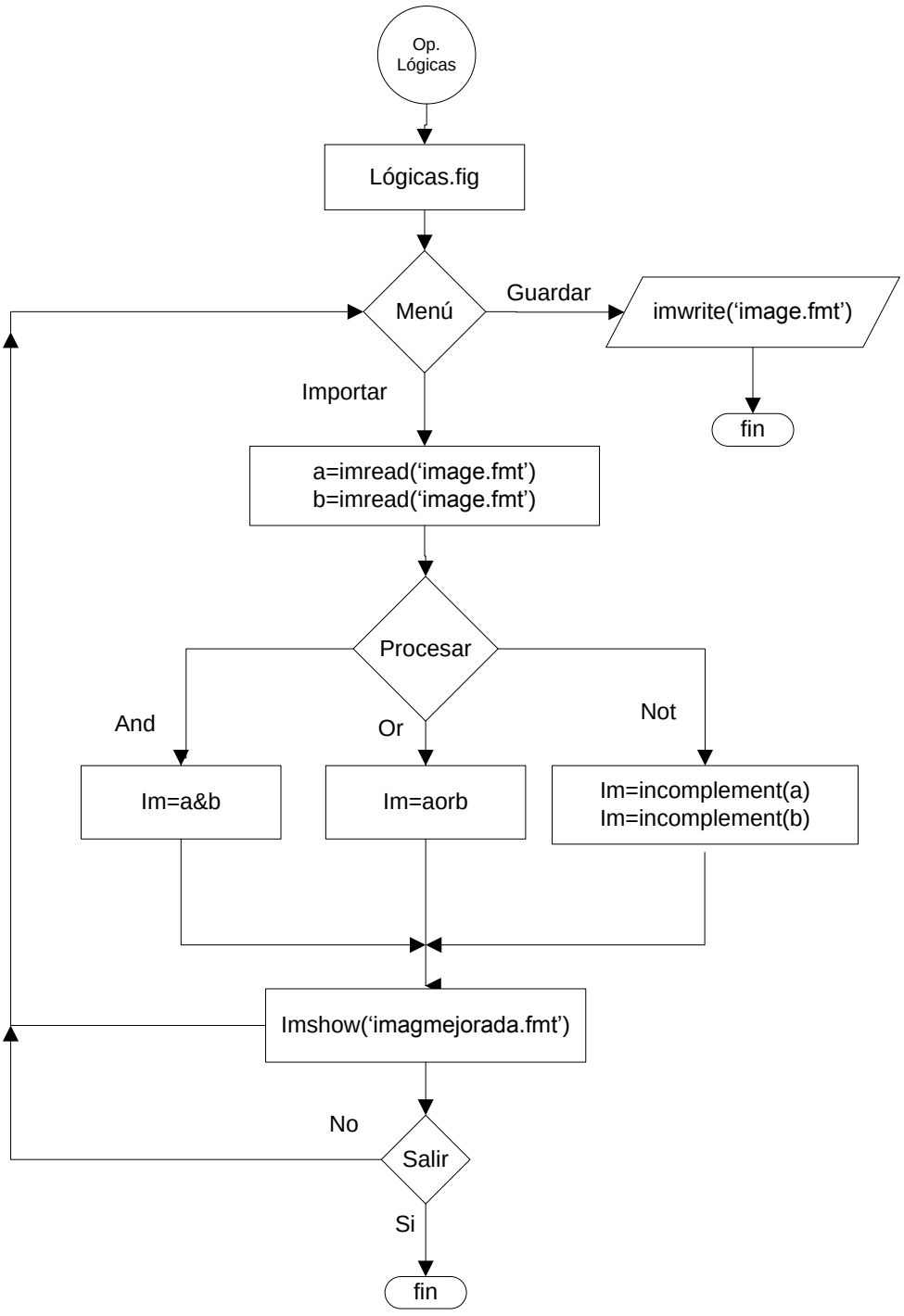


Figura 4.9 Diagrama de flujo de la Ventana Operaciones Lógicas

Diagrama de flujo pantalla Histogramas

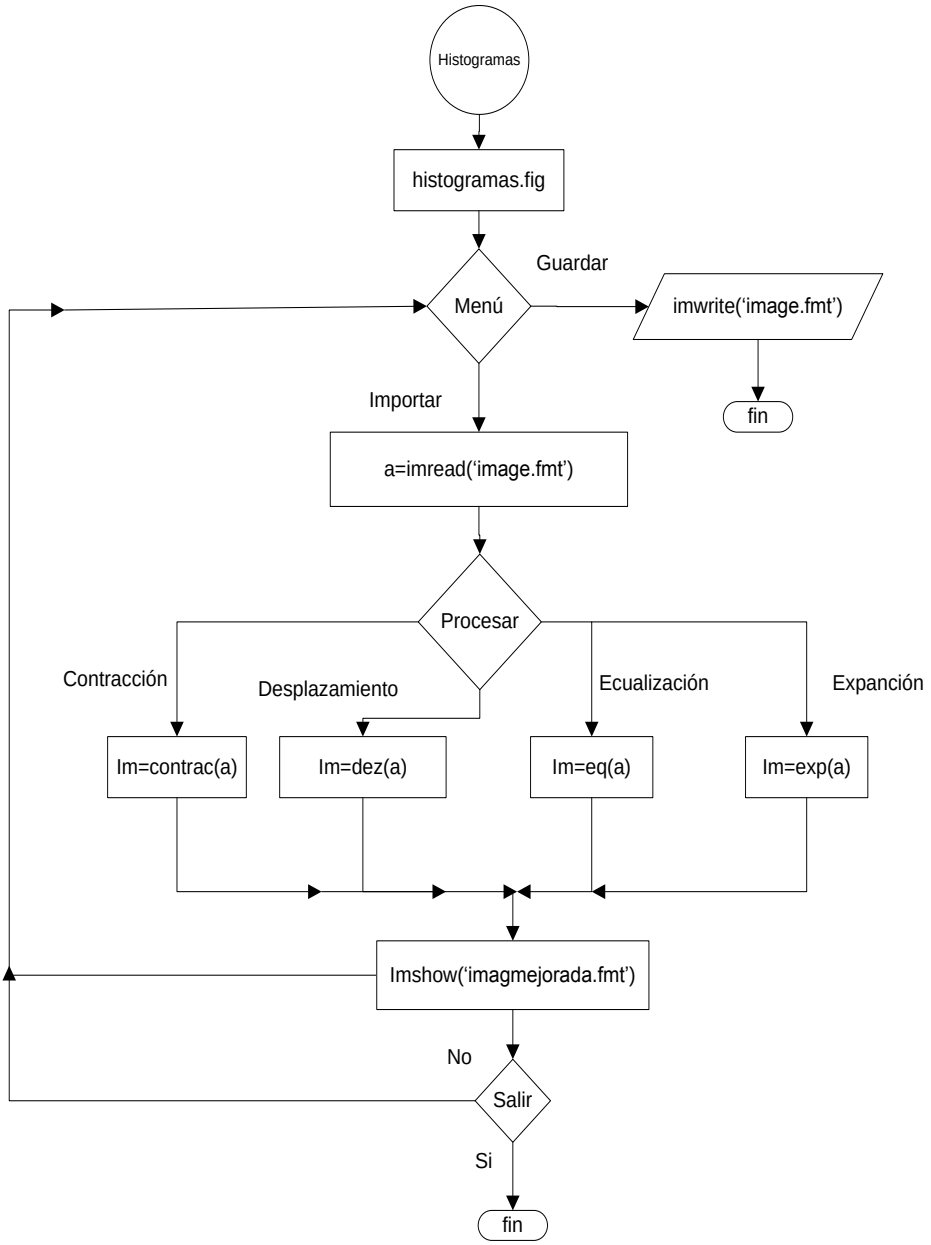


Figura 4.10 Diagrama de flujo de la Ventana Técnicas del Histograma

Diagrama de flujo pantalla Corrección gamma

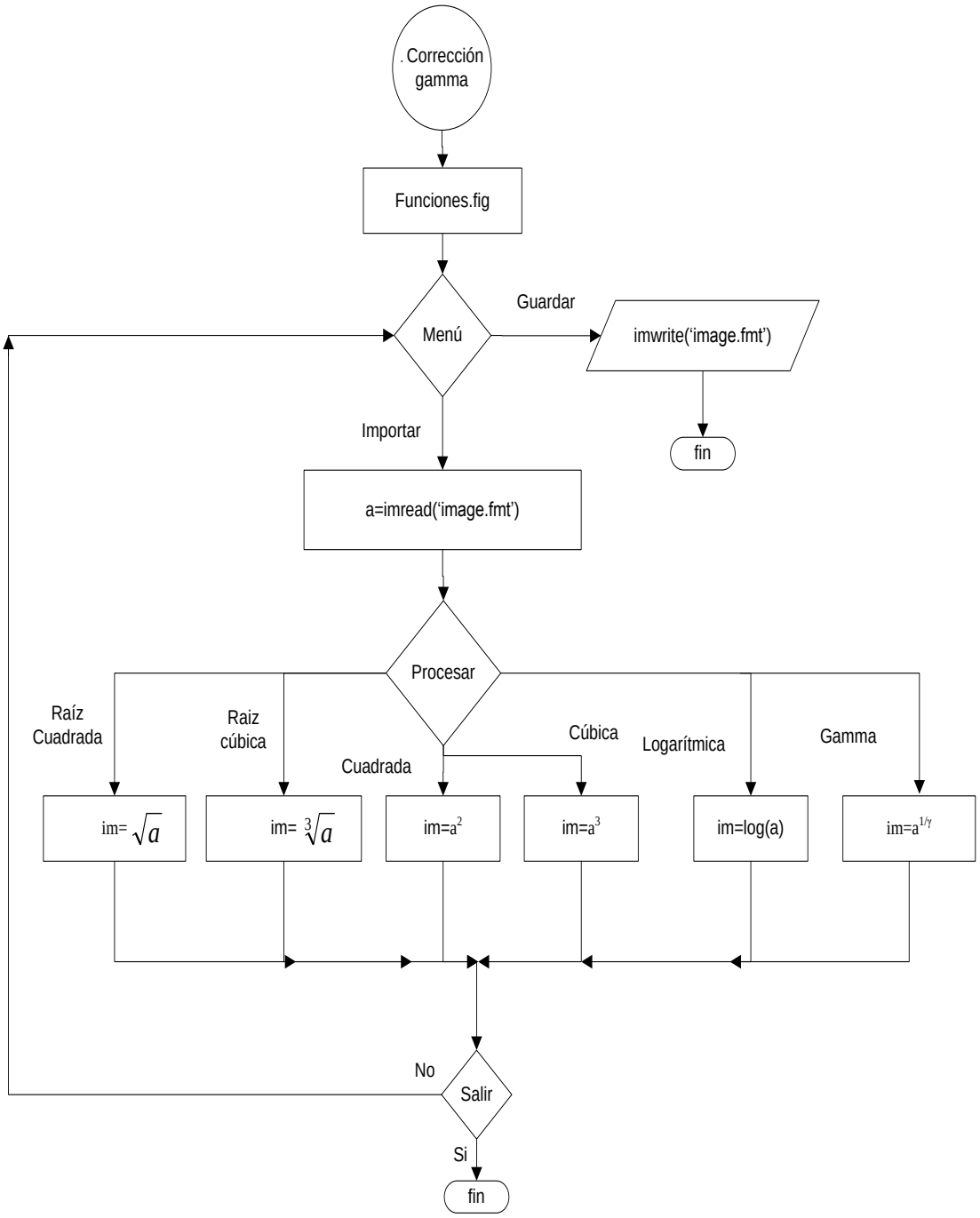


Figura 4.11 Diagrama de flujo de la Ventana Corrección Gamma

4.4 IMPLEMENTACIÓN DE LA INTERFAZ

Para la implementación de la interfaz, se realiza el diseño de la interfaz gráfica "Mejoramiento de imágenes a Color", implementando las pantallas para la interfaz de usuario.

4.4.1 VENTANA PRINCIPAL

Aparecen los datos informativos de la aplicación la misma que esta realizada con textos estáticos, y dos botones, uno de ellos permite ingresar al siguiente menú de la interfaz, mientras que el otro botón permite salir de la misma.

Figura 4.12 Ventana Principal

4.4.2 VENTANA MENÚ PRINCIPAL

En esta ventana aparecen los métodos por los cuales se quiere mejorar la imagen a color, ya sea utilizando operaciones aritméticas o lógicas, la función gamma, así como también las técnicas del histograma que es el tema central del desarrollo de la tesis.

En esta ventana al presionar el botón operaciones aritméticas permite ingresar a la siguiente pantalla donde se mejorará la imagen usando este método, lo mismo sucede con los botones operaciones lógicas, histogramas, y corrección gamma.

En la figura 4.8 se muestra el diseño de la ventana

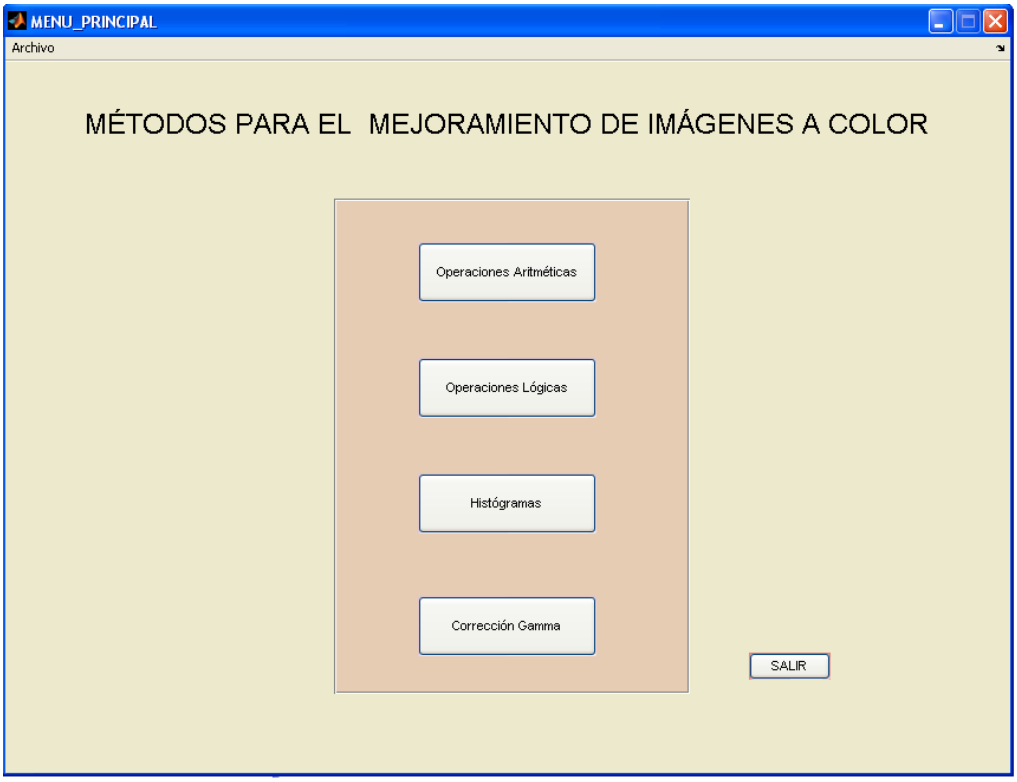


Figura 4.13 Menú Principal

4.4.3 VENTANA MEJORAMIENTO DE IMÁGENES CON OPERACIONES ARITMÉTICAS

En esta ventana se va a mejorar la imagen a través de las operaciones matemáticas como son suma, resta, multiplicación y división, además se puede realizar estas operaciones escogiendo cualquier tono ya sea rojo, verde o azul o la combinación de los tres RGB, que es el color verdadero de la imagen.

Suma.- Al presionar este botón realiza la adición de la misma imagen o de imágenes diferentes.

Resta.- Al presionar este botón realiza la sustracción de la imagen original con una imagen ya mejorada para realizar otro mejoramiento de la misma imagen.

Multiplicación.- al presionar dicho botón, realiza la multiplicación de la imagen de la caja a o de la imagen de la caja b, por un factor multiplicativo o por una constante.

División.- al presionar dicho botón, realiza la división de la imagen de la caja a o la imagen de la caja b, por un factor de división o por una constante.

Regresar.- con este botón se ingresa al Menú Principal.

Colores.- con este grupo de opciones se puede escoger RGB, tono rojo, verde, azul, para realizar una suma resta multiplicación o división solamente entre los tonos seleccionados.

En la figura 4.9 se muestra la ventana.

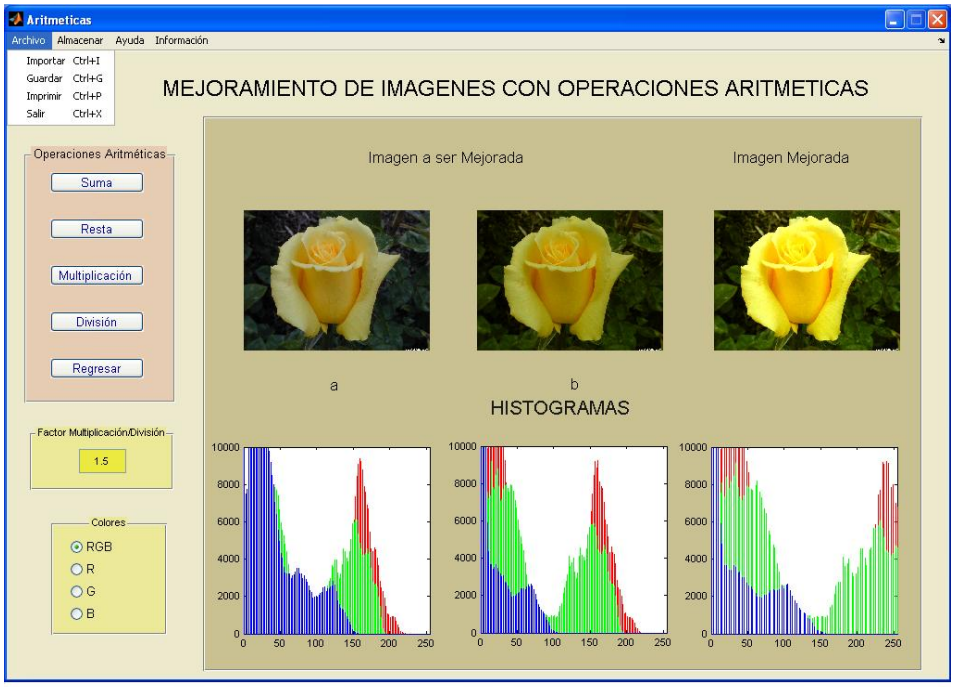


Figura 4.14 Operaciones Aritméticas

4.4.4 VENTANA MEJORAMIENTO DE IMÁGENES CON OPERACIONES LÓGICAS

En esta ventana permite mejorar la imagen usando las operaciones lógicas como son AND y OR, y también realizar el NOT o complemento de la imagen.

And.-al presionar este botón se realiza la operación lógica And entre dos imágenes, una de las imágenes es la que va ser tratada y la otra es una imagen ya mejorada por otro método, para así obtener una imagen de mejor calidad.

Or.- al presionar este botón se realiza la operación lógica Or entre dos imágenes, una de las imágenes es la original y la otra es una imagen ya mejorada por otro método, para así obtener una imagen de mejor calidad.

Not.- al presionar este botón se realiza la operación lógica Not o el negativo de la imagen, para observar detalles que no se puede apreciar en la imagen original.

En la figura 4.10 se muestra el diseño de la ventana.

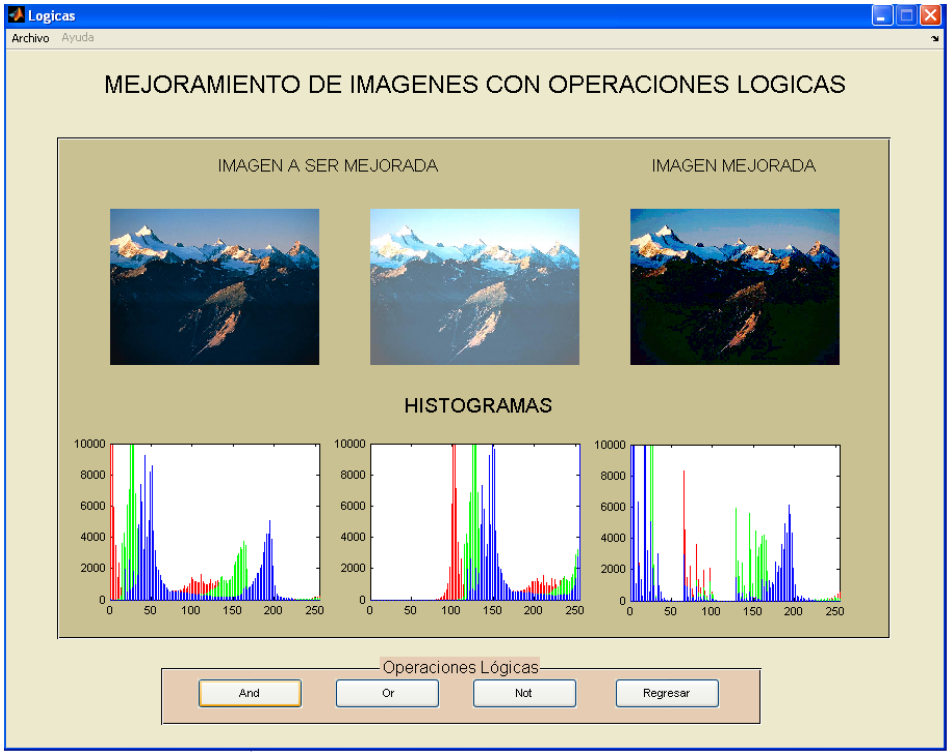


Figura 4.15 Operaciones Lógicas.

4.4.5 VENTANA MEJORAMIENTO DE IMÁGENES CON TÉCNICAS HISTOGRAMAS

En esta ventana permite visualizar las técnicas del histograma con las que se realizará el mejoramiento de la imagen a color.

En la figura 4.11 se muestra el diseño de la ventana.

Contracción.- Permite realizar la contracción del histograma con factores de contracción entre 255 (máximo valor) y 0 (mínimo valor), al contraer el histograma se disminuye el rango dinámico de la distribución de niveles de gris de la imagen.

Desplazamiento.- Permite realizar el desplazamiento del histograma, el desplazamiento se emplea para aclarar u oscurecer una imagen, pero manteniendo la relación entre los valores de los niveles de gris.

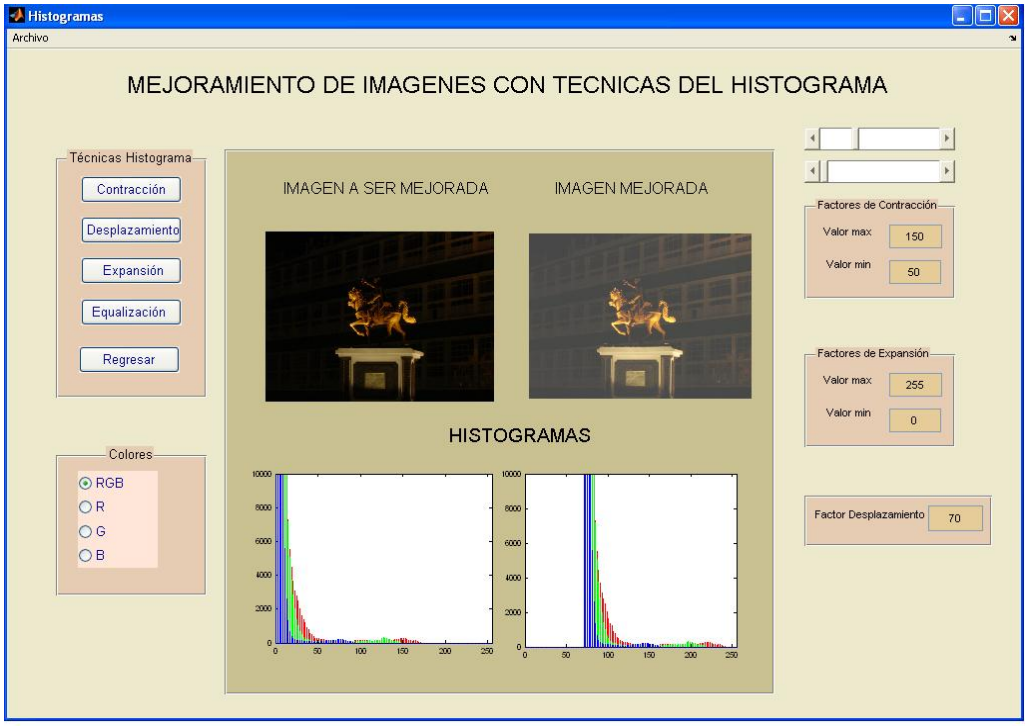


Figura 4.16 Ventana Histogramas

Expansión.- Permite realizar la expansión del histograma con factores de expansión entre 255 (máximo valor) y 0 (mínimo valor), con esta técnica se expande el histograma a lo largo del rango de valores completo de los niveles de gris, con la expansión del histograma no hay mejoramiento de imagen mas bien una degradación de la misma.

Ecualización.- Permite realizar la ecualización del histograma, con la ecualización o igualación se busca mejorar la imagen distribuyendo los niveles de gris a lo largo del rango de los valores establecidos.

Colores.- con este grupo de opciones se puede escoger RGB, tono rojo, verde, azul, para realizar una suma resta multiplicación o división solamente entre los tonos seleccionados.

4.4.6 VENTANA MEJORAMIENTO DE IMÁGENES CON CORRECCIÓN GAMMA

En esta pantalla permite visualizar los operaciones de segundo grado y corrección gamma con las que se realiza el mejoramiento de la imagen a color.

Corrección Gamma.- La corrección gamma, mejora el contraste en áreas muy claras o en áreas muy oscuras, la misma que define el grado de blanco o intensidad en una imagen.

Funciones Matemáticas.

- **Raíz Cuadrada y Raíz Cúbica**.- Permite realzar las áreas oscuras de la imagen a ser mejorada.

- **Función Cuadrada y Cúbica.-** Permite realzar las áreas claras de la imagen a ser mejorada.

- **Logarítmica.-** Mediante el manejo de esta técnica se realza las áreas oscuras de la imagen.

- **Negativo.-** Permite obtener la imagen invertida de una imagen dada, en la cual se puede observar información o aspectos que resultan difíciles de percibir, los datos pueden estar más claros en la imagen invertida.

En la figura 4.12 se muestra el diseño de la ventana

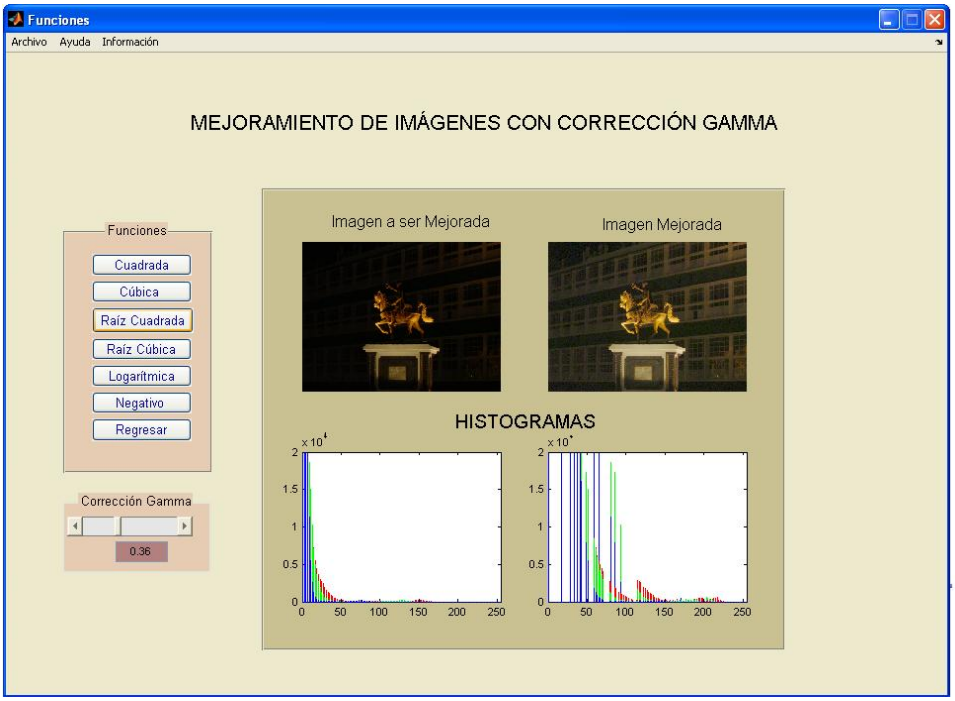
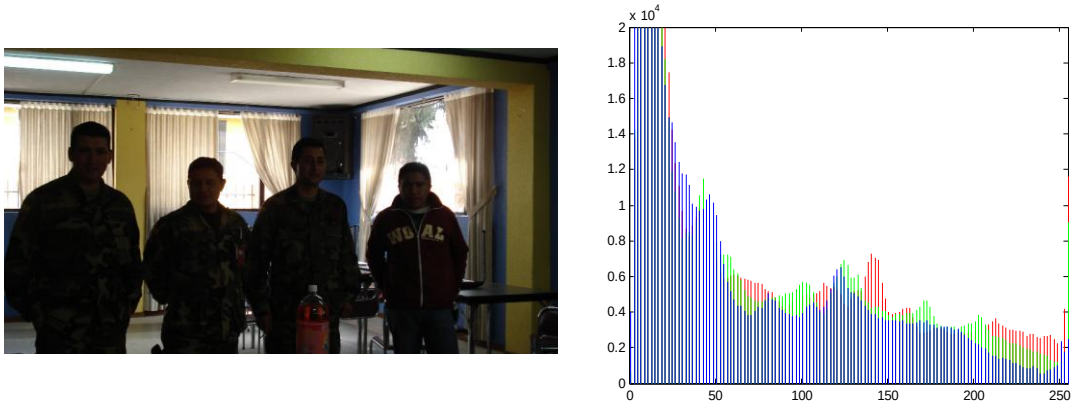


Figura 4.17 Ventana Corrección Gamma

4.5 PRUEBAS Y VALIDACIÓN DE LA INTERFAZ

Para la validación de la interfaz tomaré un grupo de imágenes para probar y validar el funcionamiento de la interfaz diseñada e implementada, para lo cual se tiene las siguientes imágenes:

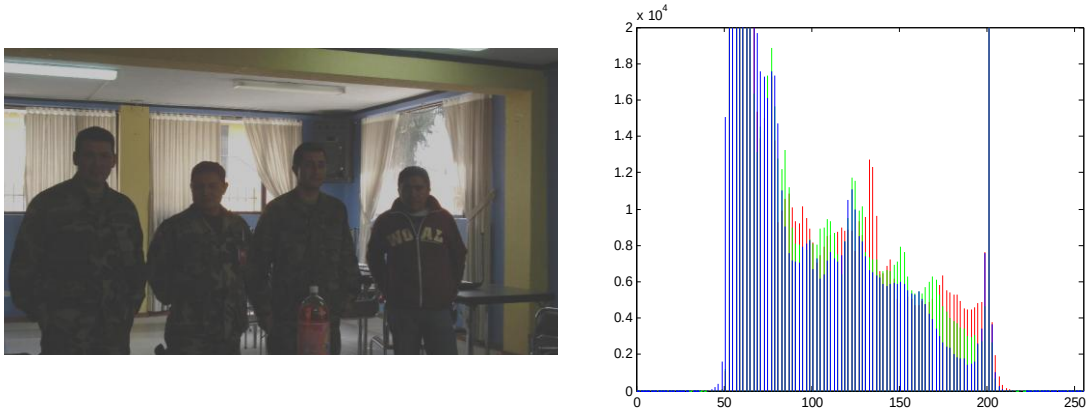
- En la figura 4.18 (a), 4.24 (a), 4.28 (a)se muestra la imagen original a ser tratada por los distintos métodos desarrollados, y en (b) el histograma.



(a) **(b)**

Figura 4.18(a) Imagen Original, (b) Su Histograma.

La figura 4.19 (a) muestra la imagen resultante tras aplicar una contracción de histograma a la imagen de la figura 4.18 (a); (b) el histograma resultante de la contracción.

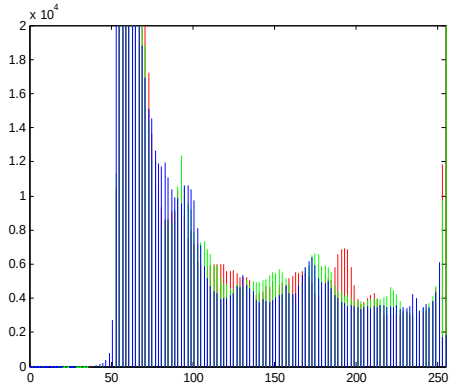


(a) **(b)**

Figura 4.19 (a) Imagen resultante tras aplicar una contracción del histograma a la imagen de la figura 4.18 (a); (b) el histograma resultante.

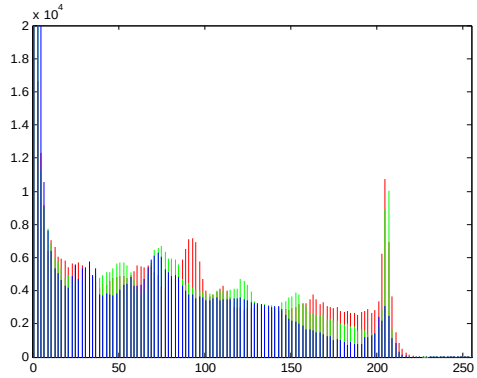
La figura 4.20 (a) muestran las imágenes resultantes después de aplicar un desplazamiento de histograma hacia la derecha e izquierda respectivamente, los histogramas correspondientes en (b) y (d).

(a)



(b)

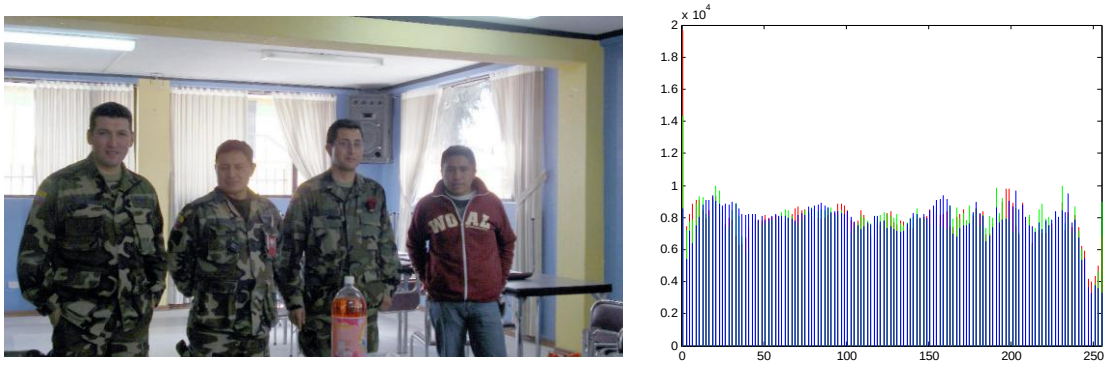
(c)



(d)

Figura 4.20 (a) y (c) Muestran las imágenes resultantes después de emplear un desplazamiento del histograma de la imagen de la figura 4.18 (a) hacia la parte clara del mismo (derecha), y oscura (izquierda) respectivamente; (b) y (d) los histogramas resultantes de dichos desplazamientos.

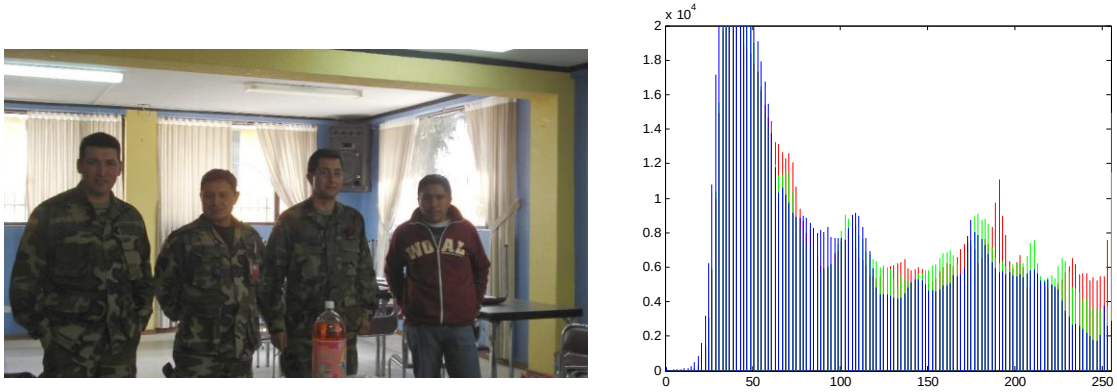
La figura 4.21 (a) muestra la imagen realzada de la figura 4.18 (a) mediante la ecualización uniforme del histograma; su histograma en (b).



(a) **(b)**

Figura 4.21 (a) Imagen realzada de la figura 4.18 (a) mediante la ecualización del histograma; (b) su histograma.

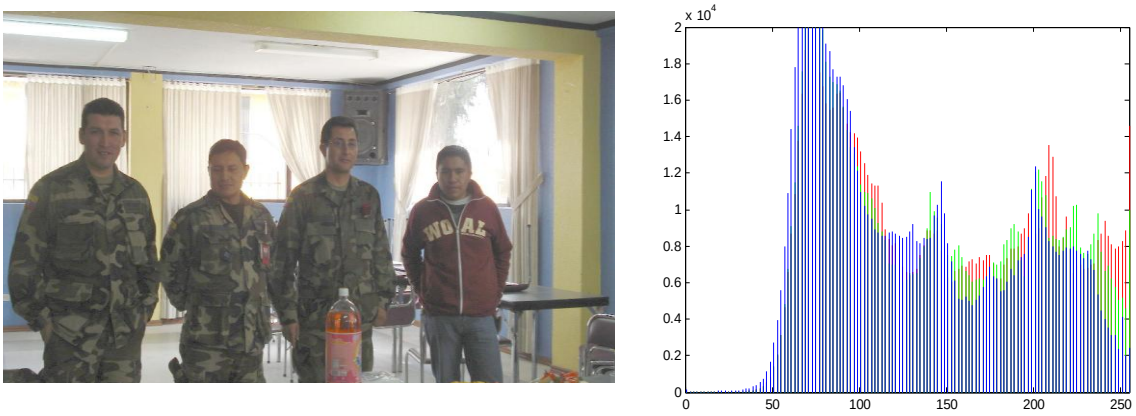
Figura 4.22 (a) muestra la imagen realzada de la figura 4.18 (a) para un factor de corrección gamma de 0.5; el histograma resultante (b).



(a) **(b)**

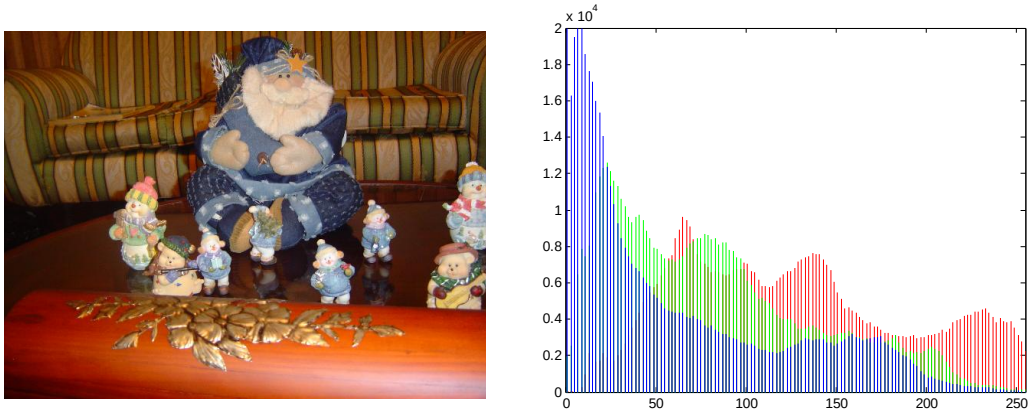
Figura 4.22 (a) Muestra la imagen realzada de la figura 4.18 (a) con un factor de 0.5 de corrección gamma; (b) el histograma resultante.

En la figura 4.23 (a) se muestra la imagen resultante tras obtener la raíz cúbica de la figura 4.18 (a); su histograma en (b).



(a) **(b)**

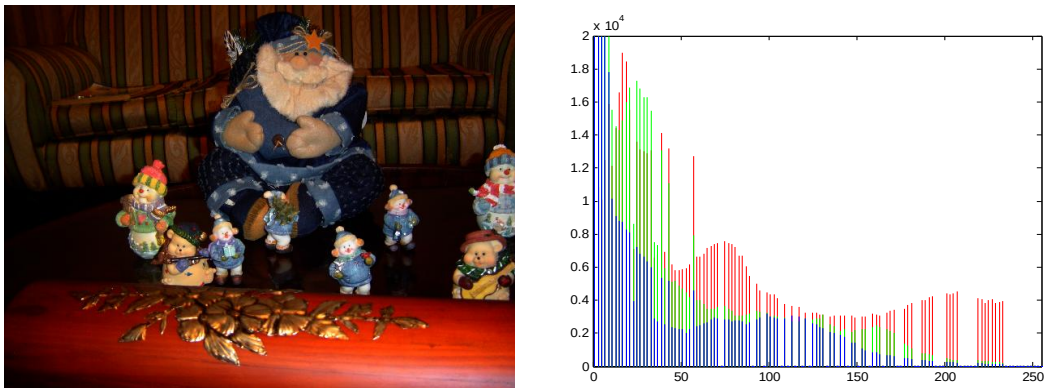
Figura 4.23 (a) Imagen resultante tras obtener la raíz cúbica de la figura 4.18 (a); (b) su histograma.



(a) **(b)**

Figura 4.24 (a) Imagen Original, (b) Su Histograma

En la figura 4.25 (a) se indica la imagen realzada de la figura 4.25 (a) mediante la función cuadrada; en (b) su histograma.

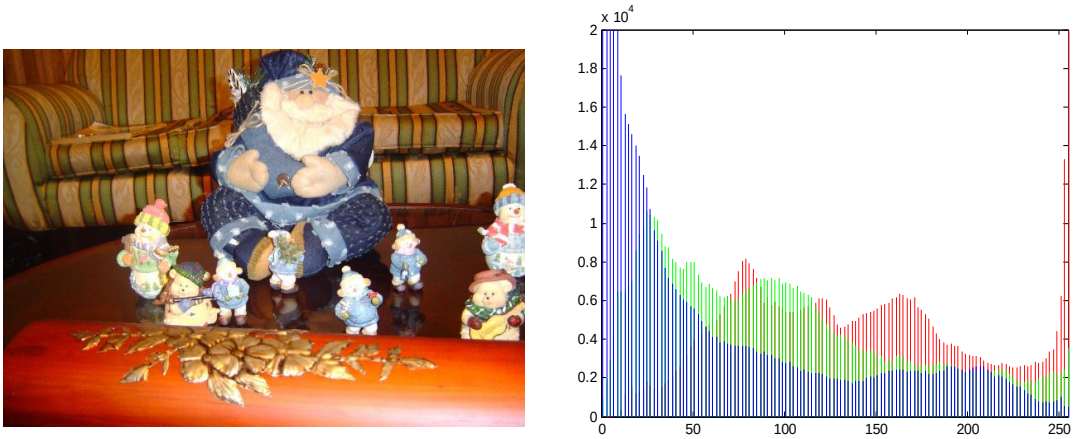


(a) **(b)**

Figura 4.25 (a) Imagen realzada de la figura 4.24 (a) mediante la función cuadrada; (b) su histograma.

En la figura 4.26 (a) se muestra la imagen resultante después de multiplicar por un factor de 1.2 a la imagen de la figura 4.24(a), en (b) el histograma

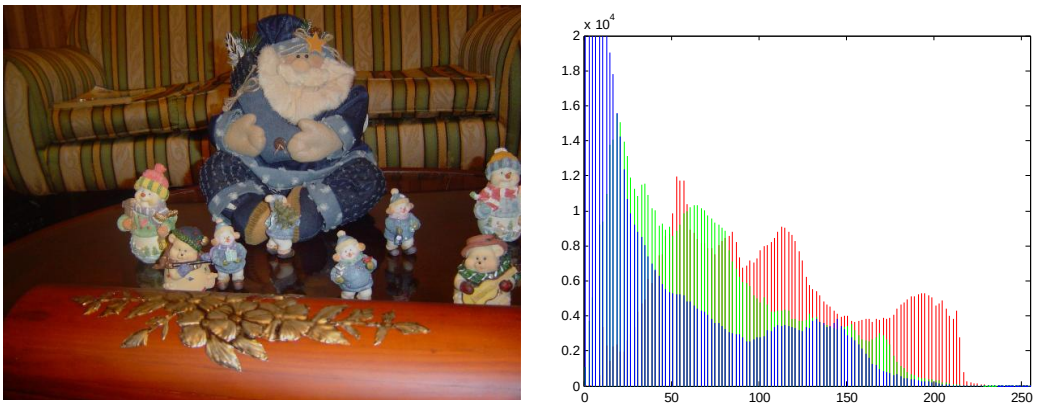
.



(a) **(b)**

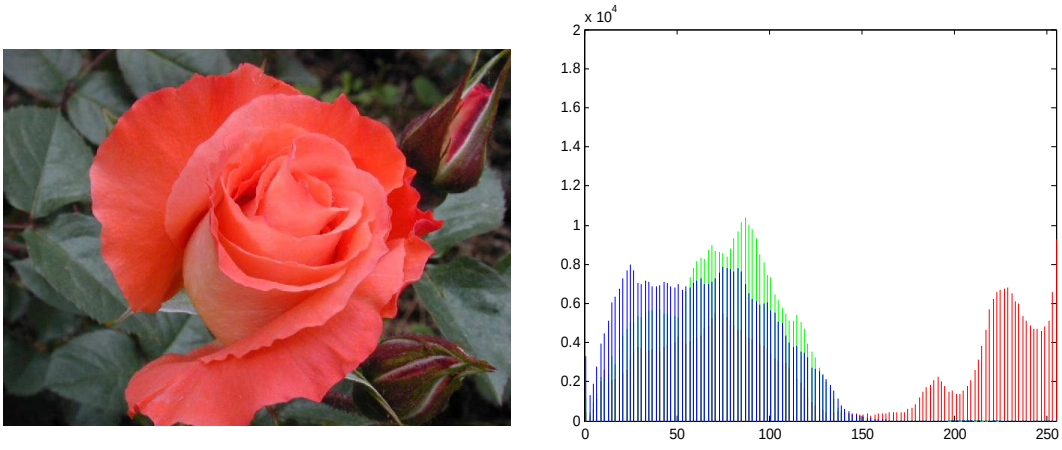
Figura 4.26 (a) Imagen realzada de la figura 4.24 (a) después de multiplicar por un factor de 1.2; (b) su histograma resultante.

La figura 4.27 (a) muestra la imagen realzada de la figura 4.24 (a), después de dividir por un factor de 1.2; en (b) el histograma resultante.



(a) **(b)**

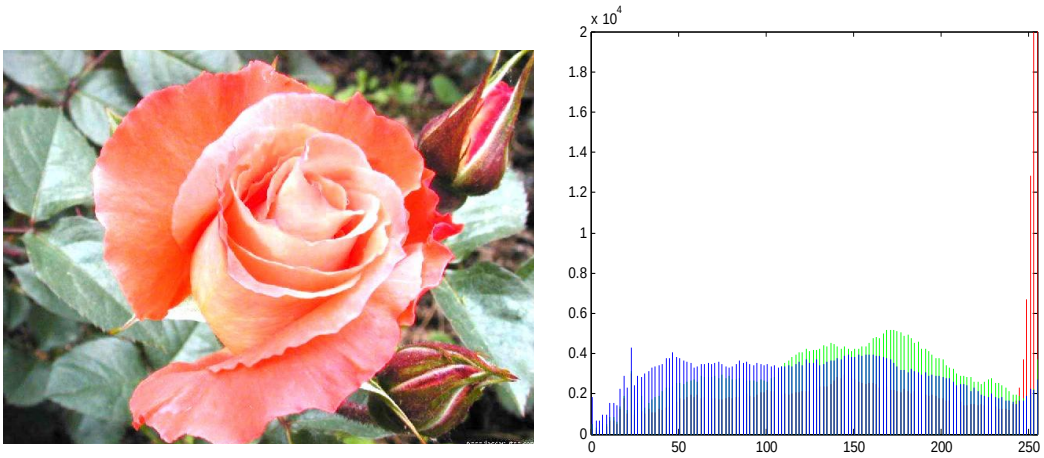
Figura 4.27 (a) Imagen realzada de la figura 4.24 (a), después de dividir por un factor de 1.2; (b) su histograma resultante.



(a) **(b)**

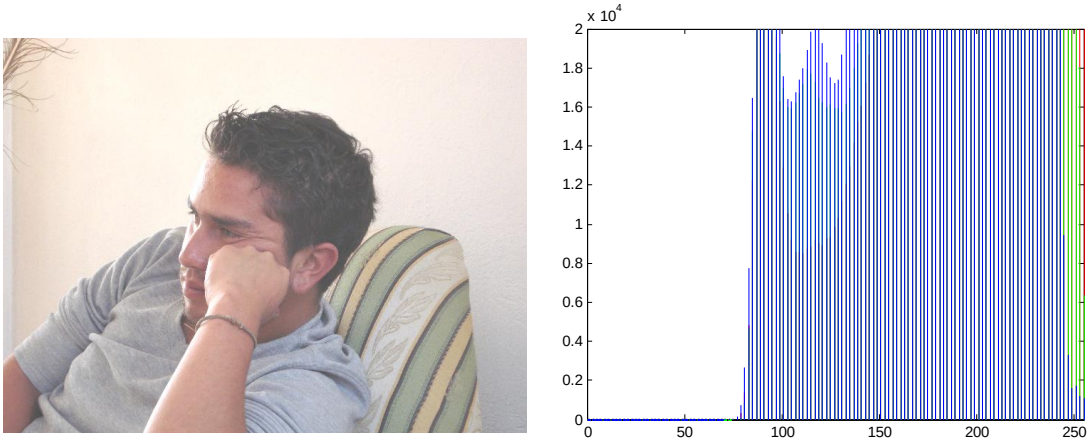
Figura 4.28 (a) Imagen Original (b) Su histograma

En la figura 4.29(a), se muestra la imagen resultante tras aplicar la suma aritmética (se suma la misma imagen), en (b) el histograma resultante de dicha operación.



(a) **(b)**

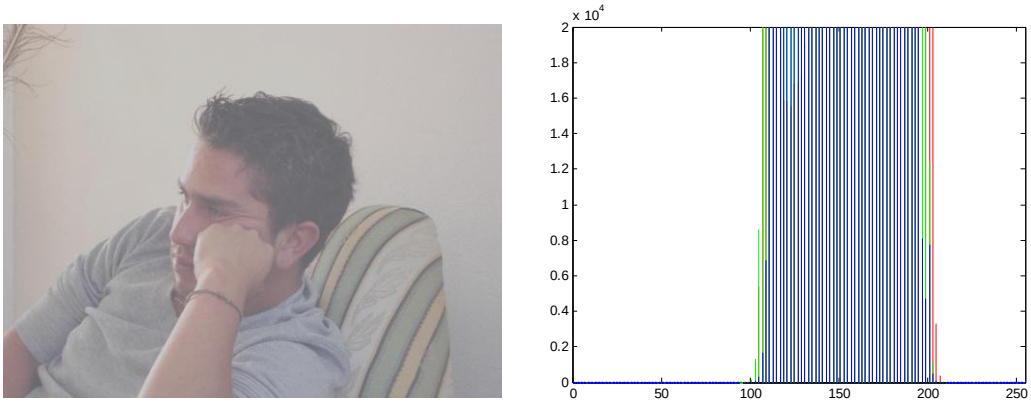
Figura 4.29 (a) Imagen resultante tras aplicar la suma aritmética, (b) su histograma



(a) **(b)**

Figura4.30 (a) Imagen Original, (b) su histograma

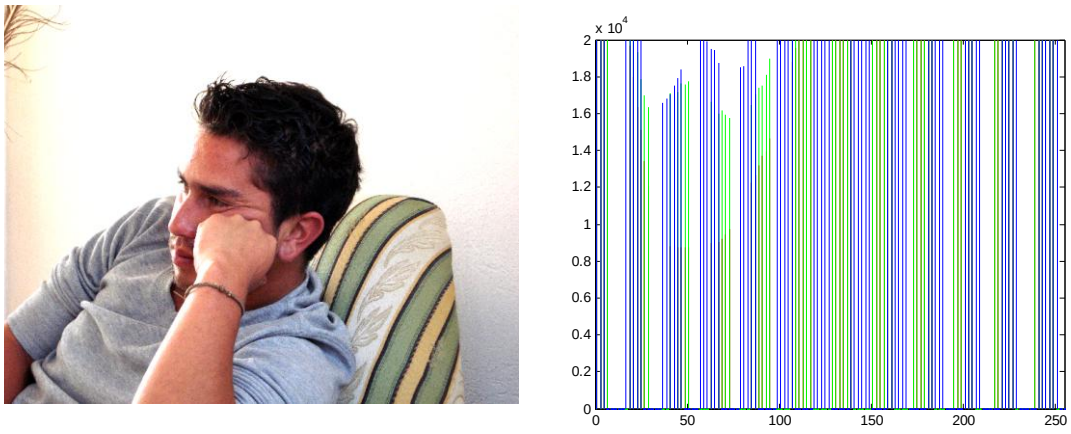
La figura 4.31 (a) muestra la imagen resultante tras aplicar una contracción del histograma a la imagen de la figura 4.30 (a); en (b) el histograma resultante.



(a) **(b)**

Figura 4.31 (a) Imagen resultante tras contraer el histograma a la imagen de la figura 4.30(a) (b) el histograma resultante.

La figura 4.32(a) muestra la imagen tras aplicar la expansión del histograma a la imagen de la figura 4.30(a); y su histograma en (b).



(a) **(b)**

Figura 4.32 (a) Imagen resultante después de aplicar la expansión del histograma a la imagen de la figura 4.30(a); (b) el histograma resultante de la expansión.

Después de emplear los diferentes métodos para el mejoramiento de imágenes a color con técnicas del histograma se obtienen los siguientes resultados:

- La contracción del histograma produce una disminución de contraste en la imagen, esta técnica no produce un realzado en la imagen.

- Al desplazar el histograma hacia la derecha se tiene una mejora del brillo, y al desplazar a la izquierda se obtiene un mejor contraste.

- La ecualización del histograma permite una distribución uniforme de los niveles de gris, esta técnica realza la imagen original. Al aplicar esta técnica se tiene una mejora del contraste en la imagen.

- El resultado de aplicar la ecualización del histograma a la imagen, es tener una mejora en la calidad de la imagen realzada.

- Con un factor de 0.5 de corrección gamma o con la raíz cúbica, las áreas obscuras de la imagen son realzadas.

- Al realizar una multiplicación o división de la imagen por un factor o por una constante, se corrige los niveles de gris, los principales usos de la multiplicación de la imagen o de la división, es corregir el nivel de gris.

- Con la función cuadrado, se esta realzando las partes claras de la imagen original.

- Al sumar imágenes se reduce el ruido promedio en una imagen.

- Al realizar la expansión del histograma, se incrementa el contraste de una imagen de bajo contraste.

- Cuando se incrementa el brillo, el valor de cada píxel se acerca más a 255, y cuando se disminuye, el valor de cada píxel se reduce más cerca del 0.

- Para modificar el contraste se puede hacer: con una compresión del histograma en caso de una reducción, o una expansión en caso de aumento de contraste.

- De acuerdo con los resultados obtenidos, se valida la interfaz, ya que esta cumple con todos los parámetros necesarios para el funcionamiento de la misma.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se diseñó e implementó la interfaz gráfica para el mejoramiento de imágenes a color con técnicas de manejo de histogramas.

- Se analizó las imágenes a color con sus características básicas, las mismas que manejan los tres colores primarios; rojo, verde, azul, y a partir de la suma de estos, se obtienen los complementarios cyan, magenta y amarillo.

- Se desarrolló los algoritmos matemáticos para el análisis y resultados de las imágenes a tratarse.

- Mejoró la visualización y las características del brillo y contraste de las imágenes a color con técnicas del manejo de histogramas, como son: contracción, expansión, ecualización, desplazamiento, corrección gamma.

- Con las técnicas de mejoramiento de imágenes se optimizó la calidad de la imagen eliminando degradaciones.

- El objetivo principal de las técnicas de mejoramiento de imagen es procesar una imagen con el fin de hacerla más adecuada para una determinada aplicación o procesamiento posterior.

- En la ecualización del histograma, el objetivo es obtener en la imagen de salida un histograma uniforme.

- En la expansión del histograma, se aumenta el rango dinámico de zonas de interés a costa de perderlo en otras.

- Los histogramas miden e ilustran en forma gráfica el contraste y brillo característicos de una imagen.

- A la imagen resultante no se añade nueva información, simplemente se resalta la existente, para una mejor valoración por el ojo humano.

- Un histograma con una amplia distribución de los niveles de gris concentrados en la parte baja del rango corresponde a una imagen oscura, un histograma con los valores concentrados en la parte alta del mismo corresponde a una imagen brillante.

- Mediante el Procesamiento Digital de Imágenes es posible manipular imágenes digitales en un computador con el fin de obtener información objetiva de la escena captada por una cámara.

5.2 RECOMENDACIONES

- Los límites de contracción y expansión deben estar dentro del rango de 0 a 255.

- Si se desea que la expansión no cubra el rango total posible de niveles de gris, se pueden especificar diferentes valores para máximo y mínimo.

- Para realizar cualquier operación aritmética o lógica, las imágenes deben tener la misma dimensión, caso contrario se debe realizar una interpolación.

- Para mejorar el brillo (aclarar) de una imagen basta con sumar un número fijo a todos los niveles de gris.

- Para mejorar el contraste (oscurecer) se realiza una simple sustracción de un número fijo a todos los niveles de gris.

- Una aplicación del histograma, es como una herramienta de análisis de imágenes, útil para la obtención de soluciones.

- Para realizar el diseño e implementación de la interfaz en la guía de matlab se debe tener un conocimiento previo de cómo funciona todas las opciones dadas, para un mejor manejo de este.

Anexo A

Programación Pantalla Principal

```
function varargout = ESPE(varargin)
% ESPE M-file for ESPE.fig
%      ESPE, by itself, creates a new ESPE or raises the existing
%      singleton*.
%      H = ESPE returns the handle to a new ESPE or the handle to
%      the existing singleton*.
%      ESPE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ESPE.M with the given input arguments.
%      ESPE('Property','Value',...) creates a new ESPE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before ESPE_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ESPE_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help ESPE
% Last Modified by GUIDE v2.5 14-May-2007 16:16:51
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @ESPE_OpeningFcn, ...
   'gui_OutputFcn',  @ESPE_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before ESPE is made visible.
function ESPE_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ESPE (see VARARGIN)
 % Choose default command line output for ESPE
handles.output = hObject;
 % Update handles structure
% set(0,'Units','pixels')
sc = get(0,'ScreenSize');
fty=get(handles.figure1,'Position');
set(handles.figure1,'Units','pixels','Position',[5 5 sc(3)-5 sc(4)-25])
in = imread('rosa162.jpg');
rga = imread('logo_electronica.jpg');
axes(handles.sello)
imshow(in)
[d,e,g]=size(in);
for m = 1:4
    for k = 1:8
        for i = 1:d
            for j=1:e
                if in(i,j,1)>120
                    in(i,j,1)= in(i,j,1)+12;
                    in(i,j,2)= in(i,j,2)-4.5;
                    in(i,j,3) = in(i,j,3)-3;
                elseif in(i,j,2)>120
                    in(i,j,1)= in(i,j,1)+12;
                    in(i,j,2)= in(i,j,2)-4.5;
                    in(i,j,3) = in(i,j,3)-3;
```

```
            elseif in(i,j,3)>120
                in(i,j,1)= in(i,j,1)+12;
                in(i,j,2)= in(i,j,2)-4.5;
                in(i,j,3) = in(i,j,3)-3;
            else
            end
        end
    end
    axes(handles.sello)
    imshow(in)
end
 for k = 1:8
    for i = 1:d
        for j=1:e
            if in(i,j,1)>120
                in(i,j,1)= in(i,j,1)-12;
                in(i,j,2)= in(i,j,2)+4.5;
                in(i,j,3) = in(i,j,3)+3;
            elseif in(i,j,2)>130
                in(i,j,1)= in(i,j,1)-12;
                in(i,j,2)= in(i,j,2)+4.5;
                in(i,j,3) = in(i,j,3)+3;
            elseif in(i,j,3)>130
                in(i,j,1)= in(i,j,1)-12;
                in(i,j,2)= in(i,j,2)+4.5;
                in(i,j,3) = in(i,j,3)+3;
            else
            end
        end
    end
    axes(handles.sello)
    imshow(in)
    % figure, imshow(in)
end
```

```
end
imga = in;
axes(handles.sello)
imshow(rga)
uiwait
% pause(100)
% UIWAIT makes ESPE wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% pantalla=('espe.fig');
% set(pantalla, 'FullScreen', 0);

% --- Outputs from this function are returned to the command line.
function varargout = ESPE_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Get default command line output from handles structure
% varargout{1} = handles.output;

% --- Executes on button press in ingreso.
function ingreso_Callback(hObject, eventdata, handles)
% hObject    handle to ingreso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.figure1,'visible','off')
fig = figure(openfig('MENU_PRINCIPAL','reuse'));
% Generate a structure of handles to pass to callbacks, and store it.
handles = guihandles(fig);
guidata(fig, handles);
close ESPE
% --- Executes on button press in salida.
function salida_Callback(hObject, eventdata, handles)
% hObject    handle to salida (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
decision = questdlg('¿Está seguro que desea SALIR?','ESPE','SI','NO','default');%cuadro de
dialogo
switch decision
   case 'SI',
      quit
   case 'NO',
      quit cancel;
end
```

Programación Pantalla Menú Principal

```
function varargout = MENU_PRINCIPAL(varargin)
% MENU_PRINCIPAL M-file for MENU_PRINCIPAL.fig

% --- Executes just before MENU_PRINCIPAL is made visible.
function MENU_PRINCIPAL_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MENU_PRINCIPAL (see VARARGIN)
% Choose default command line output for MENU_PRINCIPAL
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
sc = get(0,'ScreenSize');
fty=get(handles.figure1,'Position');
set(handles.figure1,'Units','pixels','Position',[5 5 sc(3)-5 sc(4)-45])
% UIWAIT makes MENU_PRINCIPAL wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = MENU_PRINCIPAL_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in aritmeticas.
function aritmeticas_Callback(hObject, eventdata, handles)
% hObject    handle to aritmeticas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fig = figure(openfig('Aritmeticas','reuse'));
handles = guihandles(fig);
      guidata(fig, handles);
      close MENU_PRINCIPAL

% --- Executes on button press in logicas.
function logicas_Callback(hObject, eventdata, handles)
% hObject    handle to logicas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fig = figure(openfig('Logicas','reuse'));
handles = guihandles(fig);
guidata(fig, handles);
 close MENU_PRINCIPAL

% --- Executes on button press in histograma.
function histogra_Callback(hObject, eventdata, handles)
% hObject    handle to histogra (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
fig = figure(openfig('Histogramas','reuse'));
handles = guihandles(fig);
guidata(fig, handles);
 close MENU_PRINCIPAL

% --- Executes on button press in funciones.
function funciones_Callback(hObject, eventdata, handles)
% hObject    handle to funciones (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fig = figure(openfig('Funciones','reuse'));
 % Generate a structure of handles to pass to callbacks, and store it.
handles = guihandles(fig);
       guidata(fig, handles);
       close MENU_PRINCIPAL

 % --------------------------------------------------------------------
function archivo_Callback(hObject, eventdata, handles)
% hObject    handle to archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
  % --------------------------------------------------------------------
function histogramas_Callback(hObject, eventdata, handles)
% hObject    handle to histogramas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --------------------------------------------------------------------
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 decision = questdlg('¿Está seguro que desea regresar a la Pantalla
Pricipal?','ESPE','SI','NO','default');%cuadro de dialogo
switch decision
```

```
        case 'SI',
fig = figure(openfig('ESPE','reuse'));
close MENU_PRINCIPAL
        case 'NO',
          quit cancel;
end
```

Programación Pantalla Mejoramiento de Imágenes con Operaciones Aritméticas

```
function varargout = Aritmeticas(varargin)
% ARITMETICAS M-file for Aritmeticas.fig
% --- Executes just before Aritmeticas is made visible.
function Aritmeticas_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Aritmeticas (see VARARGIN)
 % Choose default command line output for Aritmeticas
```

```
handles.output = hObject;
 % Update handles structure
guidata(hObject, handles);
 % UIWAIT makes Aritmeticas wait for user response (see UIRESUME)
 global contador
contador=0;
mkdir('c:\imagenes temporales');
g=dir(fullfile('c:','imagenes temporales/*.jpg'))
t=size(g)
sc = get(0,'ScreenSize');
fty=get(handles.figure1,'Position');
set(handles.figure1,'Units','pixels','Position',[5 5 sc(3)-5 sc(4)-45])
set([handles.suma handles.resta handles.multiplicacion handles.division ],'enable','off')
set([handles.colores handles.rojo handles.verde handles.azul],'enable','off')
% --- Outputs from this function are returned to the command line.
function varargout = Aritmeticas_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in suma.
function suma_Callback(hObject, eventdata, handles)
% hObject    handle to suma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB im1 im2 color im11 r1 r2 r
axes(handles.histograma3)
cla
[m1,n1,i1]=size(im1)
t1=im1;
[m2,n2,i2]=size(im11)
```

```
t2=im11;
r1=t1(:,:,1);
r2=t2(:,:,1);
g1=t1(:,:,2);
g2=t2(:,:,2);
b1=t1(:,:,3);
b2=t2(:,:,3);
[s1,p1,h1]=size(r1)
[s2,p2,h2]=size(r2)
if s1>s2
   r2 = imresize(r2,[s1 p1]);
else
   r1 = imresize(r1,[s2 p2]);
end
[q1,o1,j1]=size(g1)
[q2,o2,j2]=size(g2)
if q1>q2
   g2 = imresize(g2,[q1 o1]);
else
   g1 = imresize(g1,[q2 o2]);
end
[f1,c1,k1]=size(b1)
[f2,c2,k2]=size(b2)
if f1>f2
   b2 = imresize(b2,[f1 c1]);
else
   b1 = imresize(b1,[f2 c2]);
end
 switch color
   case 1
      r=imadd(r1,r2);
      g=imadd(g1,g2);
      b=imadd(b1,b2);
   case 2
```

```
        if s1>s2
            g=ones(s1, p1, 1);
            b=ones(s1, p1, 1);
        else
            g=ones(s2, p2, 1);
            b=ones(s2, p2, 1);
        end
        r=imadd(r1,r2);
        g=g;
        b=b;
    case 3
        if q1>q2
            r=ones(q1, o1, 1);
            b=ones(q1, o1, 1);
        else
            r=ones(q2, o2, 1);
            b=ones(q2, o2, 1);
        end
        r=r;
        g=imadd(g1,g2);
        b=b;
    case 4
        if f1>f2
            r=ones(f1, c1, 1);
            g=ones(f1, c1, 1);
        else
            r=ones(f2, c2, 1);
            g=ones(f2, c2, 1);
        end
        r=r;
        g=g;
        b=imadd(b1,b2);
end
rgb=cat(3,r,g,b);
```

```
im2=rgb;
axes(handles.imagen3)
imshow(im2)
axes(handles.histograma3)
histogr(im2)
temporal(im2)
uimenu(handles.almac,'Label',r)

% --- Executes on button press in resta.
function resta_Callback(hObject, eventdata, handles)
% hObject    handle to resta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB im1 im2 a b im11 r color
axes(handles.histograma3)
cla
[m1,n1,i1]=size(im1)
[m2,n2,i2]=size(im11)
t1=im1;
t2=im11;
r1=t1(:,:,1);
r2=t2(:,:,1);
g1=t1(:,:,2);
g2=t2(:,:,2);
b1=t1(:,:,3);
b2=t2(:,:,3);
[s1,p1,h1]=size(r1)
[s2,p2,h2]=size(r2)
if s1>s2
   r2 = imresize(r2,[s1 p1]);
else
   r1 = imresize(r1,[s2 p2]);
end
[q1,o1,j1]=size(g1)
```

```
[q2,o2,j2]=size(g2)
if q1>q2
   g2 = imresize(g2,[q1 o1]);
else
   g1 = imresize(g1,[q2 o2]);
end
[f1,c1,k1]=size(b1)
[f2,c2,k2]=size(b2)
if f1>f2
   b2 = imresize(b2,[f1 c1]);
else
   b1 = imresize(b1,[f2 c2]);
end
switch color
   case 1
      r=imsubtract(r1,r2);
      g=imsubtract(g1,g2);
      b=imsubtract(b1,b2);
   case 2
      if s1>s2
         g=ones(s1, p1, 1);
         b=ones(s1, p1, 1);
      else
         g=ones(s2, p2, 1);
         b=ones(s2, p2, 1);
      end
      r=imsubtract(r1,r2);
      g=g;
      b=b;
   case 3
      if q1>q2
         r=ones(q1, o1, 1);
         b=ones(q1, o1, 1);
      else
```

```
            r=ones(q2, o2, 1);
            b=ones(q2, o2, 1);
        end
        r=r;
        g=imsubtract(g1,g2);
        b=b;
    case 4
        if f1>f2
            r=ones(f1, c1, 1);
            g=ones(f1, c1, 1);
        else
            r=ones(f2, c2, 1);
            g=ones(f2, c2, 1);
        end
        r=r;
        g=g;
        b=imsubtract(b1,b2);
end
rgb=cat(3,r,g,b);
im2=rgb;
axes(handles.imagen3)
imshow(im2)
axes(handles.histograma3)
histogr(im2)
temporal(im2)
uimenu(handles.almac,'Label',r)

% --- Executes on button press in multiplicacion.
function multiplicacion_Callback(hObject, eventdata, handles)
% hObject    handle to multiplicacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB im1 im2 color im11 r
cte=str2num(get(handles.factor,'string'))
```

```
multiplicar= questdlg('Qué imagen desea multiplicar a ó b?','Imagen a
multipicar','a','b','Cancelar','default');%cuadro de dialogo
switch multiplicar
   case 'a',
      flag1=isempty(cte)
      if flag1==1
         button = questdlg('Ingrese el factor de multiplicación','Atención','Aceptar','default');
      end
      axes(handles.histograma3)
      cla
      [m1,n1,i1]=size(im1)
      t1=im1;
      [m2,n2,i2]=size(im11)
      t2=im11;
      r1=t1(:,:,1);
      g1=t1(:,:,2);
      b1=t1(:,:,3);
      switch color
         case 1
            r=r1*cte;
            g=g1*cte;
            b=b1*cte;
         case 2
            r=r1*cte;
            g=g1;
            b=b1;
         case 3
            g=g1*cte;
            r=r1;
            b=b1;
         case 4
            b=b1*cte;
            g=g1;
            r=r1;
```

```
    end
    im2=cat(3,r,g,b);
    axes(handles.imagen3)
    imshow(im2)
    axes(handles.histograma3)
    histogr(im2)
    temporal (im2)
    uimenu(handles.almac,'Label',r)
case 'b'
    flag1=isempty(cte)
    if flag1==1
       button = questdlg('Ingrese el factor de multiplicación','Atención','Aceptar','default');
    end
    axes(handles.histograma3)
    cla
    [m1,n1,i1]=size(im1)
    t1=im1;
    [m2,n2,i2]=size(im11)
    t2=im11;
    r2=RGB(:,:,1);
    g2=RGB(:,:,2);
    b2=RGB(:,:,3);
    switch color
       case 1
          r=r2*cte;
          g=g2*cte;
          b=b2*cte;
       case 2
          r=r2*cte;
          g=g2;
          b=b2;
       case 3
          g=g2*cte;
          r=r2;
```

```
            b=b2;
        case 4
            b=b2*cte;
            g=g2;
            r=r2;
    end
    im2=cat(3,r,g,b);
    axes(handles.imagen3)
    imshow(im2)
    axes(handles.histograma3)
    histogr(im2)
    temporal (im2)
    uimenu(handles.almac,'Label',r)
  case 'cancelar'
    quit cancel;
end

% --- Executes on button press in division.
function division_Callback(hObject, eventdata, handles)
% hObject    handle to division (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB im1 im2 im11 color r
cte=str2num(get(handles.factor,'string'))
dividir= questdlg('Qué imagen desea Dividir a ó b?','Imagen a
Dividir','a','b','Cancelar','default');%cuadro de dialogo
switch dividir
  case 'a',
    flag1=isempty(cte)
    if flag1==1
      button = questdlg('Ingrese el factor de división','Atención','Aceptar','default')%
(message,title,'icon')
    end
    axes(handles.histograma3)
```

```
cla
[m1,n1,i1]=size(im1)
t1=im1;
[m2,n2,i2]=size(im11)
t2=im11;
 r1=t1(:,:,1);
g1=t1(:,:,2);
b1=t1(:,:,3);
switch color
   case 1
      r=r1/cte;
      g=g1/cte;
      b=b1/cte;
   case 2
      r=r1/cte;
      g=g1;
      b=b1;
   case 3
      g=g1/cte;
      r=r1;
      b=b1;
   case 4
      b=b1/cte;
      g=g1;
      r=r1;
end
im2=cat(3,r,g,b);
axes(handles.imagen3)
imshow(im2)
axes(handles.histograma3)
histogr(im2)
temporal (im2)
uimenu(handles.almac,'Label',r)
case 'b'
```

```
flag1=isempty(cte)
if flag1==1
   button = questdlg('Ingrese el factor de división','Atención','Aceptar','default')
end
axes(handles.histograma3)
cla
[m1,n1,i1]=size(im1)
t1=im1;
[m2,n2,i2]=size(RGB)
t2=im11;
r2=RGB(:,:,1);
g2=RGB(:,:,2);
b2=RGB(:,:,3);
switch color
   case 1
      r=r2/cte;
      g=g2/cte;
      b=b2/cte;
   case 2
      r=r2/cte;
      g=g2;
      b=b2;
   case 3
      g=g2/cte;
      r=r2;
      b=b2;
   case 4
      b=b2/cte;
      g=g2;
      r=r2;
end
im2=cat(3,r,g,b);
axes(handles.imagen3)
imshow(im2)
```

```
        axes(handles.histograma3)
        histogr(im2)
        temporal (im2)
        uimenu(handles.almac,'Label',r)
    case 'cancelar'
        quit cancel;
end

% --- Executes on button press in regresar.
function regresar_Callback(hObject, eventdata, handles)
% hObject    handle to regresar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 salida = questdlg('¿Está seguro que desea regresar al Menu Pricipal?','Menu
Principal','SI','NO','default');
switch salida
    case 'SI',
        fig = figure(openfig('MENU_PRINCIPAL','reuse'));
        close Aritmeticas
    case 'NO',
        quit cancel;
end
% --------------------------------------------------------------------
function importar_Callback(hObject, eventdata, handles)
% hObject    handle to importar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 set([handles.suma handles.resta handles.multiplicacion handles.division],'enable','on')
set([handles.colores handles.rojo handles.verde handles.azul],'enable','on')
global RGB im1 im2 color a b im11 r
abrir = questdlg('¿En qué cuadro desea cargar la imagen?','Abrir','a','b','Cancelar','default');
color=1;
switch abrir
    case 'a',
```

```
    for i=1:1;
       [filename, pathname] = uigetfile( ...
          {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
          '*.bmp','BMP(*.bmp)'; ...
          '*.jpg','JPG(*.jpg)'; ...
          '*.tiff','TIFF(*.tiff)'; ...
          '*.gif','GIF(*.gif)'; ...
          '*.*',  'Todos los archivos (*.*)'}, ...
          'Importar');
       if filename==0
          return
       end
       e=fullfile(pathname,filename);
       RGB=imread(e);
        if i==1
           axes(handles.imagen1)
          imshow(RGB)
          im1=RGB;
          axes(handles.histograma1)
           histogr(RGB);
       end
    end
case 'b'
    for i=1:1;
       [filename, pathname] = uigetfile( ...
          {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
          '*.bmp','BMP(*.bmp)'; ...
          '*.jpg','JPG(*.jpg)'; ...
          '*.tiff','TIFF(*.tiff)'; ...
          '*.gif','GIF(*.gif)'; ...
          '*.*',  'Todos los archivos (*.*)'}, ...
          'Importar');
       if filename==0
          return
```

```
            end
            e=fullfile(pathname,filename);
            RGB=imread(e);
            if i==1
               RGB=imread(e);
               im11=RGB;
               axes(handles.imagen2)
               imshow(im11)
               axes(handles.histograma2)
               cla
               histogr(im11);
            end
         end
      case 'Cancelar'
         quit cancel
end
im1=im2double(im1);
im11=im2double(im11);
% --------------------------------------------------------------------
function guardar_Callback(hObject, eventdata, handles)
% hObject    handle to guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 global  rgb RGB t im2 r
[filename, pathname, filterindex] = uiputfile( ...
   {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
   '*.bmp','BMP(*.bmp)'; ...
   '*.jpg','JPG(*.jpg)'; ...
   '*.tiff','TIFF(*.tiff)'; ...
   '*.gif','GIF(*.gif)'; ...
   '*.*',  'Todos los archivos (*.*)'}, ...
   'Guardar como');
if isequal(filename,0) | isequal(pathname,0)
   disp('User selected Cancel')
```

```
else
   t=fullfile(pathname,filename)
   imwrite(im2,t)
end
 % --------------------------------------------------------------------
function imprimir_Callback(hObject, eventdata, handles)
% hObject    handle to imprimir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im2
g=figure, imshow(im2)
set(gcf,'visible','off')
printpreview(g)
% --------------------------------------------------------------------
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fig = figure(openfig('MENU_PRINCIPAL','reuse'));
% Generate a structure of handles to pass to callbacks, and store it.
handles = guihandles(fig);
guidata(fig, handles);
close Aritmeticas
% --------------------------------------------------------------------
function archivo_Callback(hObject, eventdata, handles)
% hObject    handle to archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --------------------------------------------------------------------
function histogr(RGB)
R=RGB(:,:,1);
G=RGB(:,:,2);
B=RGB(:,:,3);
h=imhist(R);
```

```
h1=h(1:2:256);
horz=1:2:256;
stem(horz,h1,'marker','none','color','r');
hold on
h2=imhist(G);
h12=h2(1:2:256);
horz2=1:2:256;
stem(horz2,h12,'marker','none','color','g');
hold on
h3=imhist(B);
h13=h3(1:2:256);
horz3=1:2:256;
stem(horz3,h13,'marker','none','color','b');
axis([0 255 0 20000])
% --------------------------------------------------------------------
function almac_Callback(hObject, eventdata, handles)
% hObject    handle to almac (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 h=gco
function temporal(im2)
global im2 contador r
d=num2str(contador);
r=strcat('imagen',d,'.jpg')
 w=fullfile('c:\imagenes temporales',r);
imwrite(im2,w);
contador=contador+1;
% --------------------------------------------------------------------
function ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 winopen('help.hlp')
% --------------------------------------------------------------------
```

```
function acer_Callback(hObject, eventdata, handles)
% hObject    handle to acer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 msgbox( ...
    {'Mejoramiento de Imágenes a Color';
    '------------------------------------------------------------------------';
    'Carrera de Ing. Electrónica e Instrumentación';
     'Escuela Politécnica del Ejército Sede Latacunga';
    '------------------------------------------------------------------------';
    'Implementado por Paulina A. Vinueza G.'}, ...
    'Acerca de')
% --------------------------------------------------------------------
function infoimagen_Callback(hObject, eventdata, handles)
% hObject    handle to infoimagen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im2
s=im2;
axes(handles.histograma3)
histogr(s)
axes(handles.imagen3)
imshow(s)
imageinfo

% --- Executes on button press in colores.
function colores_Callback(hObject, eventdata, handles)
% hObject    handle to colores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of colores
global color
val=get(handles.colores,'value')
if val==1
```

```
   color=1;
end

% --- Executes on button press in rojo.
function rojo_Callback(hObject, eventdata, handles)
% hObject    handle to rojo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of rojo
global color
val=get(handles.rojo,'value')
if val==1
   color=2;
end

% --- Executes on button press in verde.
function verde_Callback(hObject, eventdata, handles)
% hObject    handle to verde (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of verde
global color
val=get(handles.verde,'value')
if val==1
   color=3;
end

% --- Executes on button press in azul.
function azul_Callback(hObject, eventdata, handles)
% hObject    handle to azul (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of azul
global color
```

```
val=get(handles.azul,'value')
if val==1
   color=4;
end
```

Programación Pantalla Mejoramiento de Imágenes con Operaciones Lógicas

```
function varargout = Logicas(varargin)
% LOGICAS M-file for Logicas.fig
% --- Executes just before Logicas is made visible.
function Logicas_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Logicas (see VARARGIN)
% Choose default command line output for Logicas
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes Logicas wait for user response (see UIRESUME)
% uiwait(handles.figure1);
global contador
contador=0;
mkdir('c:\imagenes temporales');
g=dir(fullfile('c:','imagenes temporales/*.jpg'))
t=size(g)
sc = get(0,'ScreenSize');
fty=get(handles.figure1,'Position');
set(handles.figure1,'Units','pixels','Position',[5 5 sc(3)-5 sc(4)-45])
h = gcbo;
set([handles.and handles.or handles.not ],'enable','off')
h = gcbo;
% --- Outputs from this function are returned to the command line.
```

```
function varargout = Logicas_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

 % --- Executes on button press in and.
function and_Callback(hObject, eventdata, handles)
% hObject    handle to and (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB im1 z t im11 r
axes(handles.histograma3)
cla
[m1,n1,i1]=size(im1);
t1=im1;
[m2,n2,i2]=size(im11);
t2=im11;
if m1>m2
   t2 = imresize(t2,[m1 n1]);
else
    t1 = imresize(t1,[m2 n2]);
end
z=bitand(t1,t2);
axes(handles.resultado)
imshow(z)
axes(handles.histograma3)
histogr(z)
 temporal(z)
uimenu(handles.almac,'Label',r)

 % --- Executes on button press in or.
```

```
function or_Callback(hObject, eventdata, handles)
% hObject    handle to or (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB im1 z t im11 r
axes(handles.histograma3)
cla
[m1,n1,i1]=size(im1);
t1=im1;
[m2,n2,i2]=size(im11);
t2=im11;
if m1>m2
  t2 = imresize(t2,[m1 n1]);
else
   t1 = imresize(t1,[m2 n2]);
end
x=bitor(t1,t2);
axes(handles.resultado)
imshow(x)
axes(handles.histograma3)
histogr(x)
z=x;
temporal(z)
uimenu(handles.almac,'Label',r)

% --- Executes on button press in not.
function not_Callback(hObject, eventdata, handles)
% hObject    handle to not (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 global RGB z t im1 im11 r
 negativo = questdlg('¿De qué imagen desea el NOT','NOT','a','b','Cancelar','default')
 switch negativo
          case 'a',
```

```
axes(handles.histograma3)
cla
y=imcomplement(im1);
z=y;
axes(handles.resultado)
imshow(z)
axes(handles.histograma3)
histogr(z)
temporal(z)
uimenu(handles.almac,'Label',r)
   case 'b',
      axes(handles.histograma3)
cla
y=imcomplement(im11);
z=y;
axes(handles.resultado)
imshow(z)
axes(handles.histograma3)
histogr(z)
temporal(z)
uimenu(handles.almac,'Label',r)
   case 'Cancelar'
      quit cancel
end
% --------------------------------------------------------------------
function archivo_Callback(hObject, eventdata, handles)
% hObject    handle to archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function importar_Callback(hObject, eventdata, handles)
% hObject    handle to importar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
 set([handles.and handles.or handles.not ],'enable','on')
global RGB im1 z t im11 r
 abrir = questdlg('¿En qué cuadro desea cargar la
imagen','Abrir','a','b','Cancelar','default');%cuadro de dialogo
 switch abrir
           case 'a',
          for i=1:1;
 [filename, pathname] = uigetfile( ...
       {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
       '*.bmp','BMP(*.bmp)'; ...
       '*.jpg','JPG(*.jpg)'; ...
       '*.tiff','TIFF(*.tiff)'; ...
       '*.gif','GIF(*.gif)'; ...
       '*.*',  'Todos los archivos (*.*)'}, ...
       'Importar');
    if filename==0
     return
    end
  e=fullfile(pathname,filename);
   RGB=imread(e);
    if i==1
       RGB=imread(e);
       axes(handles.imagen1)
       imshow(RGB)
       im1=RGB;
       axes(handles.histograma1)
       cla
       histogr(RGB);
     end
     end
          case 'b',
         for i=1:1;
 [filename, pathname] = uigetfile( ...
```

```
        {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
        '*.bmp','BMP(*.bmp)'; ...
        '*.jpg','JPG(*.jpg)'; ...
        '*.tiff','TIFF(*.tiff)'; ...
        '*.*',  'Todos los archivos (*.*)'}, ...
        'Importar');
     if filename==0
      return
     end
    e=fullfile(pathname,filename);
    RGB=imread(e);
     if i==1
        RGB=imread(e);
        im11=RGB;
        axes(handles.imagen2)
        imshow(im11)
        axes(handles.histograma2)
        cla
        histogr(im11);
end
              end
    case 'Cancelar'
        quit cancel
end

% --------------------------------------------------------------------
function guardar_Callback(hObject, eventdata, handles)
% hObject    handle to guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global  z RGB q r
[filename, pathname, filterindex] = uiputfile( ...
        {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
        '*.bmp','BMP(*.bmp)'; ...
```

```
        '*.jpg','JPG(*.jpg)'; ...
        '*.tiff','TIFF(*.tiff)'; ...
        '*.*',  'Todos los archivos (*.*)'}, ...
        'Guardar como');
if isequal(filename,0) | isequal(pathname,0)
   disp('User selected Cancel')
else
q=fullfile(pathname,filename)
imwrite(z,q)
    end
% --------------------------------------------------------------------
function imprimir_Callback(hObject, eventdata, handles)
% hObject    handle to imprimir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global z
g=figure, imshow(z)
set(gcf,'visible','off')
printpreview(g)
% --------------------------------------------------------------------
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
salida = questdlg('¿Está seguro que desea regresar al Menu Pricipal?','Menu
Pricipal','SI','NO','default');%cuadro de dialogo
switch salida
              case 'SI',
fig = figure(openfig('MENU_PRINCIPAL','reuse'));
close logicas
            case 'NO',
              quit cancel;
end
%--------------------------------------------------------------
```

```
% --- Executes on button press in regresar.
function regresar_Callback(hObject, eventdata, handles)
% hObject    handle to regresar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
sal = questdlg('¿Está seguro que desea regresar al Menu Pricipal?','Menu
Principal','SI','NO','default');
switch sal
          case 'SI',
fig = figure(openfig('MENU_PRINCIPAL','reuse'));
close Logicas
          case 'NO',
            quit cancel;
end
% --------------------------------------------------------------------
function ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('help.hlp')
% --------------------------------------------------------------------
function acer_Callback(hObject, eventdata, handles)
% hObject    handle to acer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 msgbox( ...
      {'Mejoramiento de Imágenes a Color';
       '------------------------------------------------------------------------';
       'Carrera de Ing. Electrónica e Instrumentación';
       'Escuela Politécnica del Ejército Sede Latacunga';
       '------------------------------------------------------------------------';
       'Implementado por Paulina A. Vinueza G.'}, ...
       'Acerca de')
```

```
% --------------------------------------------------------------------
function infoimagen_Callback(hObject, eventdata, handles)
% hObject    handle to infoimagen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global z
s=z;
axes(handles.histograma3)
histogr(s)
axes(handles.resultado)
imshow(s)
imageinfo
% --------------------------------------------------------------------
function almac_Callback(hObject, eventdata, handles)
% hObject    handle to almac (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
get(handles.almac)
function temporal(z)
global z contador r
d=num2str(contador);
r=strcat('imagen',d,'.jpg')
w=fullfile('c:\imagenes temporales',r);
imwrite(z,w);
contador=contador+1;
```

Programación Pantalla Mejoramiento de Imágenes con Técnicas de Histograma

```
function varargout = Histogramas(varargin)
% HISTOGRAMAS M-file for Histogramas.fig
% --- Executes just before Histogramas is made visible.
```

```
function Histogramas_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Histogramas (see VARARGIN)
 % Choose default command line output for Histogramas
handles.output = hObject;
 % Update handles structure
guidata(hObject, handles);
 % UIWAIT makes Histogramas wait for user response (see UIRESUME)
% uiwait(handles.figure1);
 global contador
contador=0;
mkdir('c:\imagenes temporales');
g=dir(fullfile('c:','imagenes temporales/*.jpg'))
t=size(g)
sc = get(0,'ScreenSize');
fty=get(handles.figure1,'Position');
set(handles.figure1,'Units','pixels','Position',[5 5 sc(3)-5 sc(4)-45])
set([handles.contraccion handles.expansion handles.equalizar handles.desplazamiento
],'enable','off')
set([handles.contraccion handles.maximo handles.max handles.minimo handles.min
handles.emax handles.emin handles.des ],'enable','off')
set([handles.colores handles.rojo handles.verde handles.azul ],'enable','off')

% --- Outputs from this function are returned to the command line.
function varargout = Histogramas_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in contraccion.
function contraccion_Callback(hObject, eventdata, handles)
% hObject    handle to contraccion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB R G B mxr mxg mxb mnr mng mnb maxrgb minrgb rgb color a
sas=str2num(get(handles.max,'string'));
ses=str2num(get(handles.min,'string'));
flag1=isempty(sas);
flag2=isempty(ses);
if flag1==1 | flag2==1
button = questdlg('Ingrese los factores de contracción entre 250 y
0','Precaución','Aceptar','default')
end
if (sas>255)|(ses<0)
   button = questdlg('Ingrese los factores de contracción entre 255 y
0','Precaución','Aceptar','default')
end
if sas<ses
   button = questdlg('Ingrese el Valor max mayor al Valor min
','Precaución','Aceptar','default')
end
switch color
case 1
axes(handles.histograma2);
cla
[h,l]=size(R);
for i=1:h
   for j=1:l
      r(i,j)=(((sas-ses)/(mxr-mnr))*(R(i,j)-mnr))+ses;
   end
end
for i=1:h
```

```
    for j=1:l
        g(i,j)=(((sas-ses)/(mxg-mng))*(G(i,j)-mng))+ses;
    end
end
for i=1:h
    for j=1:l
        b(i,j)=(((sas-ses)/(mxb-mnb))*(B(i,j)-mnb))+ses;
    end
end
    case 2
        axes(handles.histograma2);
cla
[h,l]=size(R);
for i=1:h
    for j=1:l
        r(i,j)=(((sas-ses)/(mxr-mnr))*(R(i,j)-mnr))+ses;
    end
end
g=G;
b=B;
case 3
        axes(handles.histograma2);
cla
[h,l]=size(R);
for i=1:h
    for j=1:l
        g(i,j)=(((sas-ses)/(mxg-mng))*(G(i,j)-mng))+ses;
    end
end
r=R;
b=B;
case 4
        axes(handles.histograma2);
cla
```

```
[h,l]=size(R);
for i=1:h
   for j=1:l
      b(i,j)=(((sas-ses)/(mxb-mnb))*(B(i,j)-mnb))+ses;
   end
end
r=R;
g=G;
end
rgb=cat(3,r,g,b);
axes(handles.modificada)
imshow(rgb)
axes(handles.histograma2)
histogr(rgb)
temporal(rgb)
uimenu(handles.almac,'Label',a)

% --- Executes on button press in desplazamiento.
function desplazamiento_Callback(hObject, eventdata, handles)
% hObject    handle to desplazamiento (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB R G B mxr mxg mxb mnr mng mnb maxrgb minrgb rgb color a
dez=str2num(get(handles.des,'string'));
flag=isempty(dez);
if flag==1
button = questdlg('Ingrese el valor de desplazamiento 255','Precaución','Aceptar','default')
end
if (dez>255)
   button = questdlg('El valor de desplazamiento debe ser menor a
25','Precaución','Aceptar','default')
end
switch color
case 1
```

```
axes(handles.histograma2);
cla
[h,l]=size(R);
for i=1:h
   for j=1:l
      r(i,j)=R(i,j)+dez;
   end
end
for i=1:h
   for j=1:l
      g(i,j)=G(i,j)+dez;
   end
end
for i=1:h
   for j=1:l
      b(i,j)=B(i,j)+dez;
   end
end
   case 2
   axes(handles.histograma2);
   cla
   [h,l]=size(R);
 for i=1:h
   for j=1:l
      r(i,j)=R(i,j)+dez;
   end
 end
g=G;
b=B;
   case 3
      axes(handles.histograma2);
   cla
   [h,l]=size(G);
      for i=1:h
```

```
    for j=1:l
        g(i,j)=G(i,j)+dez;
    end
end
 r=R;
 b=B;
    case 4
        axes(handles.histograma2);
    cla
    [h,l]=size(B) ;
        for i=1:h
    for j=1:l
        b(i,j)=B(i,j)+dez;
    end
        end
r=R;
g=G;
end
rgb=cat(3,r,g,b);
axes(handles.modificada)
imshow(rgb)
axes(handles.histograma2)
histogr(rgb)
temporal(rgb)
uimenu(handles.almac,'Label',a)

% --- Executes on button press in expansion.
function expansion_Callback(hObject, eventdata, handles)
% hObject    handle to expansion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB R G B mx1 mn1 a mxr mxg mxb mnr mng mnb maxrgb minrgb rgb color
mx=str2num(get(handles.emax,'string'))
mn=str2num(get(handles.emin,'string'))
```

```
 flag1=isempty(mx);
flag2=isempty(mn);
if flag1==1 | flag2==1
button = questdlg('Los límites de espansión por defoult son max 255 y min
0','Precaución','Aceptar','default')
end
if (mx>255)|(mn<0)
   button = questdlg('Ingrese los factores de expansión entre 255 y
0','Precaución','Aceptar','default')
end
if mx<mn
   button = questdlg('Ingrese el Valor max mayor al Valor min
','Precaución','Aceptar','default')
end
mx1=mx/255
mn1=mn/255
axes(handles.histograma2);
cla
[h,l]=size(RGB);
 switch color
   case 1
      r=imadjust(R,stretchlim(R),[mn1; mx1]);
      g=imadjust(G,stretchlim(G),[mn1; mx1]);
      b=imadjust(B,stretchlim(B),[mn1; mx1]);
   case 2
      r=imadjust(R,stretchlim(R),[mn1; mx1]);
      g=G;
      b=B;
   case 3
      g=imadjust(G,stretchlim(G),[mn1; mx1]);
      r=R;
      b=B;
   case 4
      b=imadjust(B,stretchlim(B),[mn1; mx1]);
```

```
        r=R;
        g=G;
end
rgb=cat(3,r,g,b);
axes(handles.modificada)
imshow(rgb)
axes(handles.histograma2)
histogr(rgb)
temporal(rgb)
uimenu(handles.almac,'Label',a)

% --- Executes on button press in equalizar.
function equalizar_Callback(hObject, eventdata, handles)
% hObject    handle to equalizar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 global RGB R a G B  rgb color
% color=1;
switch color
case 1
 axes(handles.histograma2);
cla
[h,l]=size(R);
r=histeq(R);
g=histeq(G);
b=histeq(B);
    case 2
    axes(handles.histograma2);
    cla
    [h,l]=size(R);
r=histeq(R);
g=G;
b=B;
    case 3
```

```
        axes(handles.histograma2);
    cla
    [h,l]=size(R);
g=histeq(G);
r=R;
 b=B;
    case 4
        axes(handles.histograma2);
    cla
    [h,l]=size(R) ;
r=R;
g=G;
end
rgb=cat(3,r,g,b);
axes(handles.modificada)
imshow(rgb)
axes(handles.histograma2)
histogr(rgb)
temporal(rgb)
uimenu(handles.almac,'Label',a)
% --------------------------------------------------------------------
function importar_Callback(hObject, eventdata, handles)
% hObject    handle to importar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 set([handles.contraccion handles.expansion handles.equalizar handles.desplazamiento
],'enable','off')
 global r RGB R G B ts lk mxr mxg mxb mnr mng mnb maxrgb minrgb color
for i=1:1;
    color=1;
 [filename, pathname] = uigetfile( ...
        {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
        '*.bmp','BMP(*.bmp)'; ...
        '*.jpg','JPG(*.jpg)'; ...
```

```
        '*.tiff','TIFF(*.tiff)'; ...
        '*.gif','GIF(*.gif)'; ...
        '*.*',  'Todos los archivos (*.*)'}, ...
        'Importar');
       if filename==0
      return
     end
      e=fullfile(pathname,filename);
    RGB=imread(e);
     R=RGB(:,:,1);
     G=RGB(:,:,2);
     B=RGB(:,:,3);
     if i==1
        RGB=imread(e);
        axes(handles.original)
        imshow(RGB)
        axes(handles.histograma1)
        cla
        histogr(RGB);
        set([handles.contraccion handles.expansion handles.equalizar handles.desplazamiento
],'enable','on')
         set([handles.colores handles.rojo handles.verde handles.azul ],'enable','on')
         set([handles.min handles.max handles.minimo handles.maximo],'enable','on')
         set([handles.des handles.minimo handles.maximo],'enable','on')
         set([handles.emin handles.emax handles.minimo handles.maximo],'enable','on')
     end
end
[x,y]=imhist(R);
 ind=find(x);
ts=max(ind)-1
lk=min(ind)-1
mxr=ts
mnr=lk
[x,y]=imhist(G);
```

```
ind=find(x);
ts=max(ind)-1
lk=min(ind)-1
mxg=ts
mng=lk
[x,y]=imhist(B);
ind=find(x);
ts=max(ind)-1
lk=min(ind)-1
mxb=ts
mnb=lk
maxrgb=max([mxr mxg mxb])
minrgb=min([mnr mng mnb])
 % --------------------------------------------------------------------
function guardar_Callback(hObject, eventdata, handles)
% hObject    handle to guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rgb RGB t
[filename, pathname, filterindex] = uiputfile( ...
       {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
       '*.bmp','BMP(*.bmp)'; ...
       '*.jpg','JPG(*.jpg)'; ...
       '*.tiff','TIFF(*.tiff)'; ...
       '*.gif','GIF(*.gif)'; ...
       '*.*',  'Todos los archivos (*.*)'}, ...
       'Guardar como');
if isequal(filename,0) | isequal(pathname,0)
   disp('User selected Cancel')
else
t=fullfile(pathname,filename)
imwrite(rgb,t)
    end
% --------------------------------------------------------------------
```

```
function imprimir_Callback(hObject, eventdata, handles)
% hObject    handle to imprimir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rgb
g=figure, imshow(rgb)
set(gcf,'visible','off')
printpreview(g)
 % --------------------------------------------------------------------
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 decision = questdlg('¿Está seguro que desea regresar al Menu Pricipal?','Menu
Principal','SI','NO','default');
switch decision
          case 'SI',
fig = figure(openfig('MENU_PRINCIPAL','reuse'));
close Histogramas
          case 'NO',
            quit cancel;
end
% --------------------------------------------------------------------
function archivo_Callback(hObject, eventdata, handles)
% hObject    handle to archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % --- Executes on button press in regresar.
function regresar_Callback(hObject, eventdata, handles)
% hObject    handle to regresar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 salida = questdlg('¿Está seguro que desea regresar al Menu Pricipal?','Menu
Pricipal','SI','NO','default');
```

```
switch salida
          case 'SI',
fig = figure(openfig('MENU_PRINCIPAL','reuse'));
close Histogramas
          case 'NO',
           quit cancel;
end
 %        Generate a structure of handles to pass to callbacks, and store it.
%         handles = guihandles(fig);
%         guidata(fig, handles);

  function maximo_Callback(hObject, eventdata, handles)
% hObject    handle to maximo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'String') returns contents of maximo as text
%        str2double(get(hObject,'String')) returns contents of maximo as a double
mf=round(get(handles.maximo,'value'));
set(handles.max,'string',mf);
set(handles.emax,'string',mf);
set(handles.des,'string',mf);

% --- Executes during object creation, after setting all properties.
function maximo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to maximo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
 % Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
 % --------------------------------------------------------------------
```

```
 function minimo_Callback(hObject, eventdata, handles)
% hObject    handle to minimo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'String') returns contents of minimo as text
%        str2double(get(hObject,'String')) returns contents of minimo as a double
mg=round(get(handles.minimo,'value'));
set(handles.min,'string',mg);
set(handles.emin,'string',mg);

 % --- Executes during object creation, after setting all properties.
function minimo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to minimo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
 % Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
 function [ts, lk]=valores(x)
ind=find(x);
ts=max(ind)-1
lk=min(ind)-1
function valor_Callback(hObject, eventdata, handles)
% hObject    handle to valor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'String') returns contents of valor as text
%        str2double(get(hObject,'String')) returns contents of valor as a double

% --- Executes during object creation, after setting all properties.
function valor_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to valor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
 % Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

 % --- Executes on button press in colores.
function colores_Callback(hObject, eventdata, handles)
% hObject    handle to colores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hint: get(hObject,'Value') returns toggle state of colo
 global color
val=get(handles.colores,'value')
if val==1
   color=1;
end

  % --- Executes on button press in rojo.
function rojo_Callback(hObject, eventdata, handles)
% hObject    handle to rojo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hint: get(hObject,'Value') returns toggle state of rojo
global color
val=get(handles.rojo,'value')
if val==1
   color=2;
end
```

```
 % --- Executes on button press in verde.
function verde_Callback(hObject, eventdata, handles)
% hObject    handle to verde (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hint: get(hObject,'Value') returns toggle state of verde
global color
val=get(handles.verde,'value')
if val==1
    color=3;
end

 % --- Executes on button press in azul.
function azul_Callback(hObject, eventdata, handles)
% hObject    handle to azul (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of azul
global color
val=get(handles.azul,'value')
if val==1
  color=4;
end
 % --------------------------------------------------------------------
function infoimagen_Callback(hObject, eventdata, handles)
% hObject    handle to infoimagen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global rgb
s=rgb;
axes(handles.histograma2)
histogr(s)
axes(handles.modificada)
imshow(s)
```

```
imageinfo
% --------------------------------------------------------------------
function acer_Callback(hObject, eventdata, handles)
% hObject    handle to acer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
msgbox( ...
      {'Mejoramiento de Imágenes a Color';
       '-------------------------------------------------------------------------';
       'Carrera de Ing. Electrónica e Instrumentación';
        'Escuela Politécnica del Ejército Sede Latacunga';
       '-------------------------------------------------------------------------';
       'Implementado por Paulina A. Vinueza G.'}, ...
       'Acerca de')
 % --------------------------------------------------------------------
function ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('help.hlp')
 % --------------------------------------------------------------------
function almac_Callback(hObject, eventdata, handles)
% hObject    handle to almac (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
get(handles.almac)
function temporal(rgb)
global rgb contador a
d=num2str(contador);
a=strcat('imagen',d,'.jpg')
w=fullfile('c:\imagenes temporales',a);
imwrite(rgb,w);
contador=contador+1;
```

Programación Pantalla Mejoramiento de Imágenes con Corrección Gamma

```
function varargout = Funciones(varargin)
% FUNCIONES M-file for Funciones.fig
% --- Executes just before Funciones is made visible.
function Funciones_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Funciones (see VARARGIN)
% Choose default command line output for Funciones
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes Funciones wait for user response (see UIRESUME)
% uiwait(handles.figure1);
global contador
contador=0;
```

```
mkdir('c:\imagenes temporales');
g=dir(fullfile('c:','imagenes temporales/*.jpg'))
t=size(g)
sc = get(0,'ScreenSize');
fty=get(handles.figure1,'Position');
set(handles.figure1,'Units','pixels','Position',[5 5 sc(3)-5 sc(4)-45])
set([handles.cuadrada handles.cubica handles.rcuadrada handles.rcubica handles.negativo
handles.log],'enable','off')
set(handles.cgama,'value',1)
h = gcbo;
% --- Outputs from this function are returned to the command line.
function varargout = Funciones_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in cuadrada.
function cuadrada_Callback(hObject, eventdata, handles)
% hObject    handle to cuadrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im1 RGB c t r
whos
y=(im1.^2);
c=im2uint8(y);
axes(handles.modificado)
imshow(c)
axes(handles.histograma2)
cla
histogr(c)
temporal(c)
uimenu(handles.almac,'Label',r)
```

```
 % --- Executes on button press in cubica.
function cubica_Callback(hObject, eventdata, handles)
% hObject    handle to cubica (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im1 RGB c t r
whos
y=(im1.^3);
c=im2uint8(y);
axes(handles.modificado)
imshow(c)
axes(handles.histograma2)
cla
histogr(c)
temporal(c)
uimenu(handles.almac,'Label',r)
% --- Executes on button press in rcuadrada.
function rcuadrada_Callback(hObject, eventdata, handles)
% hObject    handle to rcuadrada (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im1 RGB c t r
whos
y=(im1.^(1/2));
c=im2uint8(y);
axes(handles.modificado)
imshow(c)
axes(handles.histograma2)
clc
histogr(c)
temporal(c)
uimenu(handles.almac,'Label',r)
 % --- Executes on button press in rcubica.
function rcubica_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to rcubica (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im1 RGB c t r
whos
y=(im1.^(1/3));
c=im2uint8(y);
axes(handles.modificado)
imshow(c)
axes(handles.histograma2)
cla
histogr(c)
temporal(c)
uimenu(handles.almac,'Label',r)
 % --- Executes on button press in log.
function log_Callback(hObject, eventdata, handles)
% hObject    handle to log (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im1 RGB c t r
whos
y=(log(im1+1)/log(2));
c=im2uint8(y);
axes(handles.modificado)
imshow(c)
axes(handles.histograma2)
cla
histogr(c)
temporal(c)
uimenu(handles.almac,'Label',r)
 % --- Executes on button press in negativo.
function negativo_Callback(hObject, eventdata, handles)
% hObject    handle to negativo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global RGB c t r
whos
y=imcomplement(RGB);
c=y;
axes(handles.modificado)
imshow(y)
axes(handles.histograma2)
cla
histogr(y)
temporal(c)
uimenu(handles.almac,'Label',r)
 % --------------------------------------------------------------------
function importar_Callback(hObject, eventdata, handles)
% hObject    handle to importar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global RGB im1 c r t
for i=1:1;
 [filename, pathname] = uigetfile( ...
       {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
       '*.bmp','BMP(*.bmp)'; ...
       '*.jpg','JPG(*.jpg)'; ...
       '*.tiff','TIFF(*.tiff)'; ...
       '*.gif','GIF(*.gif)'; ...
       '*.*',  'Todos los archivos (*.*)'}, ...
       'Importar');
    if filename==0
     return
    end
     e=fullfile(pathname,filename);
   RGB=imread(e);
    if i==1
      RGB=imread(e);
```

```
        axes(handles.original)
        imshow(RGB)
        axes(handles.histograma1)
        cla
        histogr(RGB);
        set([handles.cuadrada handles.cubica handles.rcuadrada handles.rcubica
handles.negativo handles.log],'enable','on')
     end
end
im1 =im2double(RGB);
% --------------------------------------------------------------------
function guardar_Callback(hObject, eventdata, handles)
% hObject    handle to guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global   RGB t c
[filename, pathname, filterindex] = uiputfile( ...
        {'*.bmp;*.jpg;*.tiff;*.gif','Todos los archivos de imagen';
        '*.bmp','BMP(*.bmp)'; ...
        '*.jpg','JPG(*.jpg)'; ...
        '*.tiff','TIFF(*.tiff)'; ...
        '*.gif','GIF(*.gif)'; ...
        '*.*',  'Todos los archivos (*.*)'}, ...
        'Guardar como');
if isequal(filename,0) | isequal(pathname,0)
   disp('User selected Cancel')
else
t=fullfile(pathname,filename)
imwrite(c,t)
    end
% --------------------------------------------------------------------
function imprimir_Callback(hObject, eventdata, handles)
% hObject    handle to imprimir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global c
g=figure, imshow(c)
set(gcf,'visible','off')
printpreview(g)
% --------------------------------------------------------------------
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fig = figure(openfig('MENU_PRINCIPAL','reuse'));
      % Generate a structure of handles to pass to callbacks, and store it.
      handles = guihandles(fig);
      guidata(fig, handles);
close Funciones
 % --------------------------------------------------------------------
function archivo_Callback(hObject, eventdata, handles)
% hObject    handle to archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % --------------------------------------------------------------------
function help_Callback(hObject, eventdata, handles)
% hObject    handle to help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('help.hlp')
 %--------------------------------------------------------------------
% --- Executes on button press in regresar.
function regresar_Callback(hObject, eventdata, handles)
% hObject    handle to regresar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
salida = questdlg('¿Está seguro que desea regresar al Menu Pricipal?','Menu
Pricipal','SI','NO','default');
```

```
switch salida
          case 'SI',
fig = figure(openfig('MENU_PRINCIPAL','reuse'));
close Funciones
          case 'NO',
            quit cancel;
end
 % --- Executes during object creation, after setting all properties.
function histo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to histo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
 % Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
 % --- Executes on slider movement.
function cgama_Callback(hObject, eventdata, handles)
% hObject    handle to cgama (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'Value') returns position of slider
%       get(hObject,'Min') and get(hObject,'Max') to determine range of slider
global RGB r im1 c t
valor=get(handles.cgama,'value');
set(handles.gam,'string',valor)
c=imadjust(RGB,[],[],valor);
axes(handles.modificado)
imshow(c)
axes(handles.histograma2)
cla
histogr(c)
```

```
temporal(c)
uimenu(handles.almac,'Label',r)
 % --- Executes during object creation, after setting all properties.
function cgama_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cgama (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
 % Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function gam_Callback(hObject, eventdata, handles)
% hObject    handle to gam (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hints: get(hObject,'String') returns contents of gam as text
%        str2double(get(hObject,'String')) returns contents of gam as a double


% --- Executes during object creation, after setting all properties.
function gam_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gam (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --------------------------------------------------------------------
function ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('help.hlp')
```

```
 % --------------------------------------------------------------------
function acer_Callback(hObject, eventdata, handles)
% hObject    handle to acer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
msgbox( ...
      {'Mejoramiento de Imágenes a Color';
       '-------------------------------------------------------------------------';
       'Carrera de Ing. Electrónica e Instrumentación';
       'Escuela Politécnica del Ejército Sede Latacunga';
       '-------------------------------------------------------------------------';
       'Implementado por Paulina A. Vinueza G.'}, ...
      'Acerca de')
 % --------------------------------------------------------------------
function infoimagen_Callback(hObject, eventdata, handles)
% hObject    handle to infoimagen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global c
s=c;
axes(handles.histograma2)
histogr(s)
axes(handles.modificado)
imshow(s)
imageinfo
% --------------------------------------------------------------------
function almac_Callback(hObject, eventdata, handles)
% hObject    handle to almac (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
get(handles.almac)
function temporal(rgb)
global c contador r
d=num2str(contador);
```

```
r=strcat('imagen',d,'.jpg')
w=fullfile('c:\imagenes temporales',r);
imwrite(c,w);
contador=contador+1;
```

Anexo B

Manual de Usuario

1. Instalación del software

- Inserte el cd de software de Mejoramiento de Imágenes a Color.
- Grabe la carpeta Mejoramiento de imágenes a color en la carpeta work de matlab.

2. Instrucciones de Uso

- Instalado el SW de Mejoramiento de Imágenes a Color:
- En la carpeta work, mejoramiento de imágenes a color haga doble clic sobre el icono ESPE.
- Una vez visualizada la ventana PRINCIPAL.

1. Seleccione ingresar para iniciar el proceso de mejoramiento de una imagen a color.

Figura 1 Ventana Principal.

2. Una vez visualizada la pantalla de Menú Principal.

a) Operaciones Aritméticas.

1. Seleccione el tipo de método para el mejoramiento, el método operaciones aritméticas.

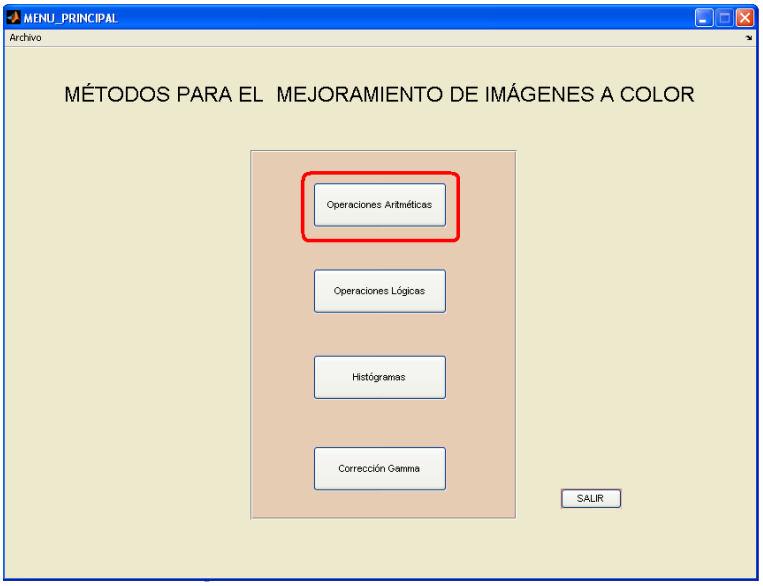


Figura 2 Pantalla Menú Principal

Se visualiza la pantalla siguiente.

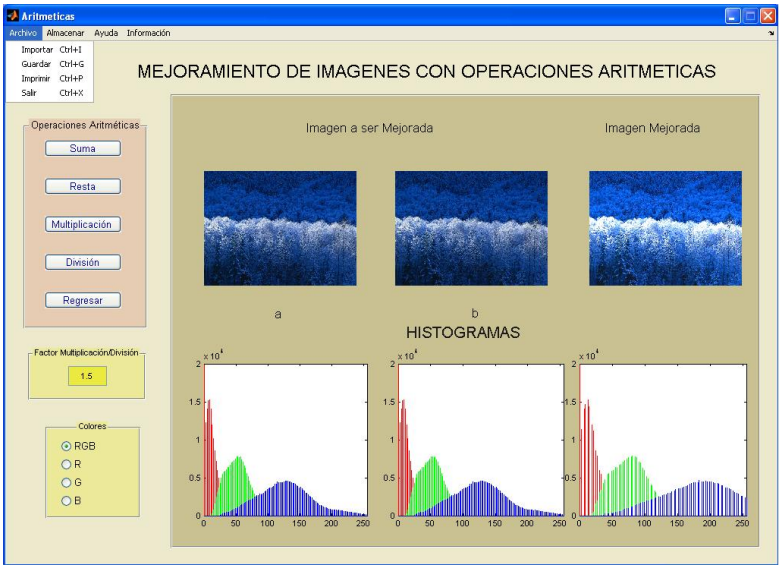


Figura 3 Pantalla Mejoramiento de imágenes con operaciones Aritméticas.

2. Importar las imágenes a procesar (para la suma o resta debe cargar la imagen original y una copia de la misma o a su vez una ya procesada anteriormente por el método de histogramas o el de Funciones matemáticas.), en la caja a o b.

3. Ingrese el factor de multiplicación y/o el de división.

4. Presione cada uno de los botones de Operaciones Aritméticas para observar cual de ellas da el mejoramiento adecuado a la imagen tratada o a su vez escoja la operación que desea realizar, adicional puede realizar las operaciones solo entre tonos ya sea rojo, verde o azul, y en las tonalidades RGB.

5. Si la imagen mejorada cumple con lo requerido con su percepción visual almacénela o imprímala, caso contrario regrese al paso 3.

6. Para retornar al Menú principal presione Regresar.

b) Operaciones Lógicas.

1. Seleccione el tipo de método para el mejoramiento, el método operaciones lógicas.

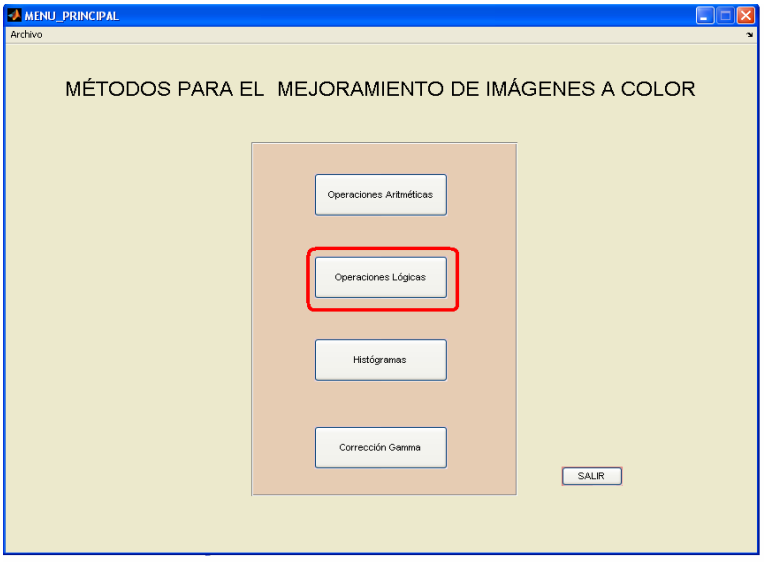


Figura 4 Pantalla Menú Principal

Se visualiza la siguiente pantalla.

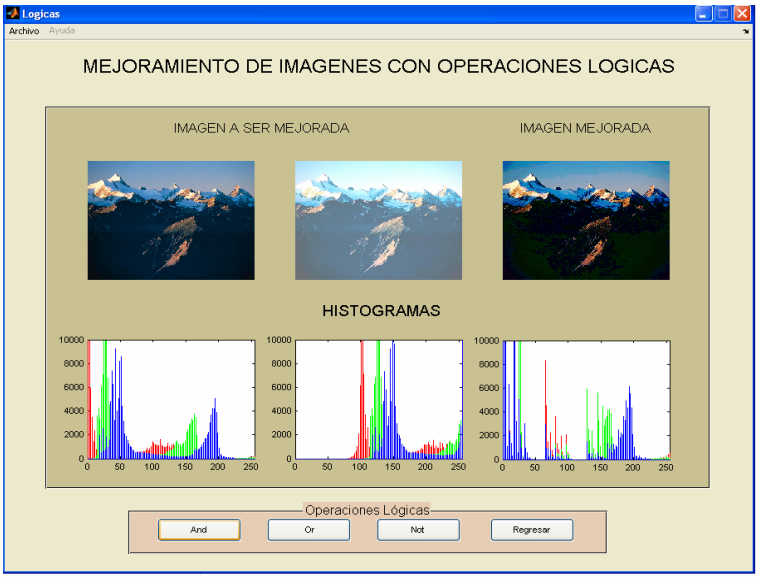


Figura 5 Pantalla Mejoramiento de imágenes con Operaciones Lógicas

2. Importar las imágenes a procesar, en este caso se debe cargar la imagen original y una ya procesada anteriormente por el método de histogramas o el de Funciones matemáticas para realizar las operaciones AND y OR.

3. Presione cada uno de los botones de Operaciones lógicas para observar cual de ellas da el mejoramiento adecuado a la imagen tratada.

4. Si la imagen mejorada cumple con lo requerido con su percepción visual almacénela o imprímala, caso contrario regrese al paso 3.

5. Presione NOT para observar el negativo de la imagen original.
6. Para retornar al Menú principal presione Regresar.

c) Técnicas del Histogramas

1. Seleccione el tipo de método para el mejoramiento, el método de histogramas.

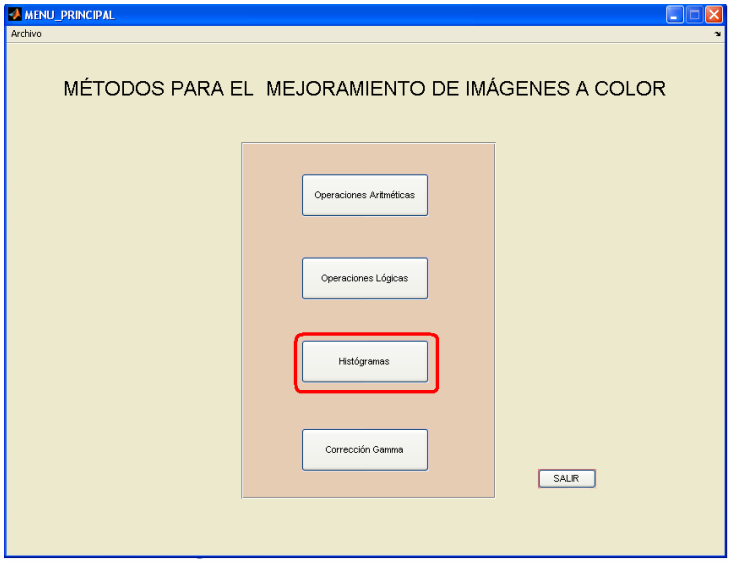


Figura 6 Pantalla Menú Principal

Se visualiza la pantalla siguiente.

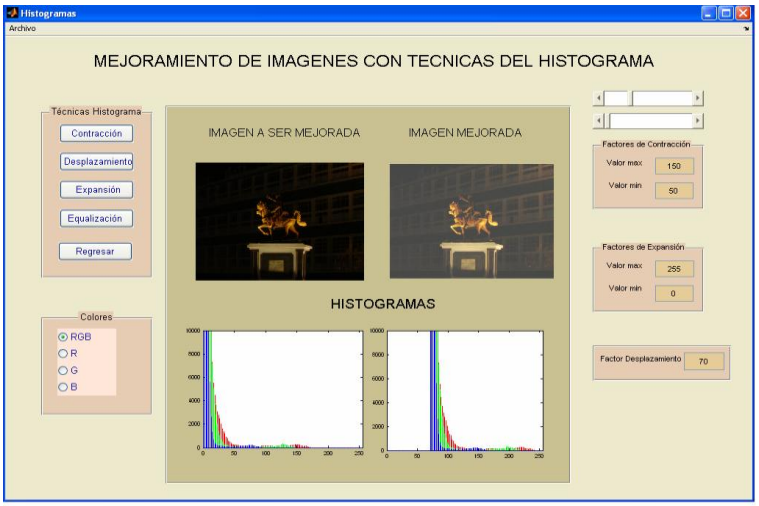


Figura 7 Pantalla Mejoramiento de imágenes con Técnicas del Histograma

2. Importar la imagen a procesar.

3. Ingrese los factores de contracción, expansión entre 0 y255, y desplazamiento.

4. Presione cada uno de los botones de Técnicas Histogramas para observar cual de ellas da el mejoramiento adecuado a la imagen tratada.

5. Para observar el cambio de tonalidades escoja R para rojo G para verde y B para azul o la combinación de los tres colores RGB.

6. Si la imagen mejorada cumple con lo requerido con su percepción visual almacénela o imprímala, caso contrario regrese al paso 3.
7. Para retornar al Menú principal presione Regresar.

d) Corrección gamma

1. Seleccione el tipo de método para el mejoramiento, el método Corrección Gamma

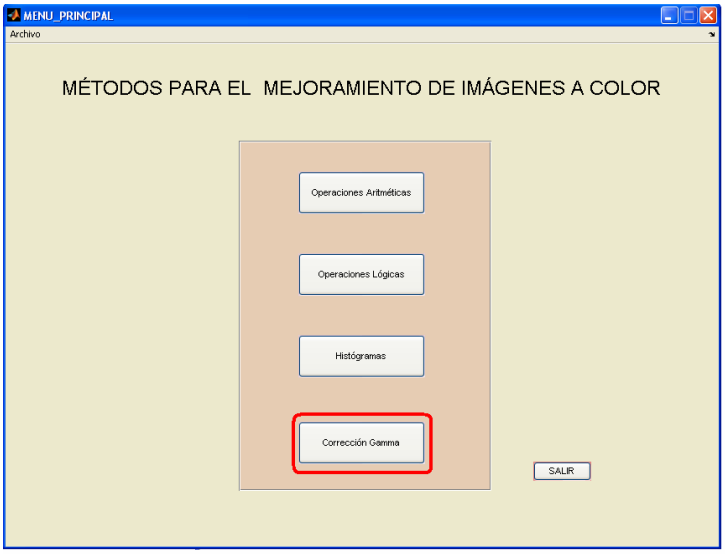


Figura 8 Pantalla Menú Principal

Se visualiza la pantalla siguiente.

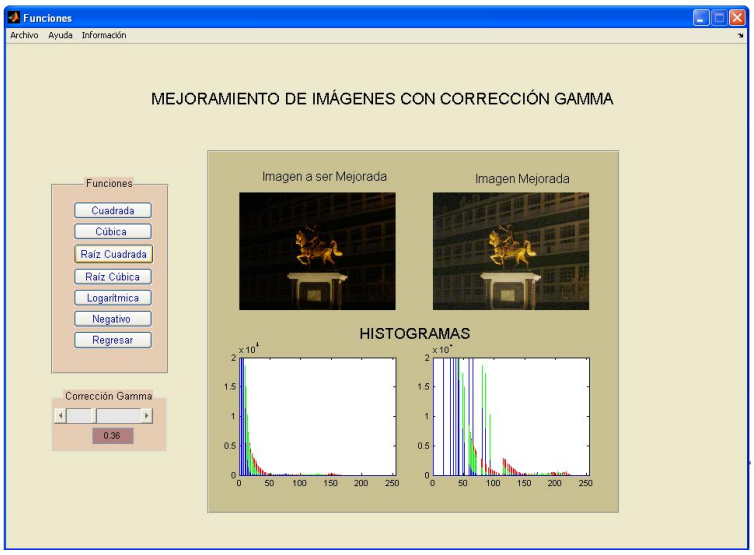


Figura 9 Pantalla Mejoramiento de imágenes con corrección gamma

2. Importar la imagen a procesar.

3. Presione cada uno de los botones de Funciones para observar cual de ellas da el mejoramiento adecuado a la imagen tratada.

4. Para ingresar el valor de corrección gamma puede hacerlo mediante el slider o en la caja de texto, observara que la imagen va cambiando automáticamente.

5. Si la imagen mejorada cumple con lo requerido con su percepción visual almacénela o imprímala, caso contrario regrese al paso 3.

6. Para retornar al Menú principal presione Regresar.

- **Menús**

  Archivo.- aquí se encuentra como importar o abrir guardar, imprimir una imagen, y salir de la ventana.

  Almacenar.- se almacena la imagen resultante en una carpeta temporal llamada Imágenes temporales en el disco C:

  Ayuda.- esta la ayuda del programa, además esta acerca del programa.

  Información.- se encuentra la información de la imagen mejorada.

Anexo C

GLOSARIO

Brillo.- Es añadir una cantidad constante al valor del tono de cada uno de los píxeles de una imagen.

Contracción del Histograma.- Contraer el histograma es disminuir el rango dinámico de la distribución de niveles de gris de la imagen.

Contraste.- Es la diferencia de tonos que hay entre las distintas zonas de la imagen. Una imagen resulta visible gracias a su diferencia de contraste respecto a los valores de los tonos que la rodean.

Corrección Gamma.- La corrección gamma es una forma especial de aumento de contraste diseñada para mejorar el contraste en áreas muy claras o muy oscuras. Esto se logra modificando los valores medios, particularmente los medios-bajos, sin afectar el blanco (255) ni el negro (0).

Frames.- Son cajas que contienen o delimitan regiones de una figura (ventana), estas cajas suelen ser introducidas para separar por grupos varios controles para mayor facilidad.

Histograma.-Es una representación gráfica de una variable en forma de barras, donde la altura o eje vertical es proporcional a la frecuencia de los valores producidos, y la anchura o eje horizontal corresponde a los intervalos o valores de la clasificación.

Histograma de una imagen.- es la función discreta de la frecuencia relativa de ocurrencia de los píxeles de una imagen en función de los niveles de intensidad.

Imagen digital.- Representación numérica de una imagen mediante píxeles (Picture elements), cuyo valor puede representar, entre otros, un nivel de gris o un valor dentro de una banda espectral perteneciente a un modelo de color.

Intensidad Luminosa.- Es el cociente entre el flujo luminoso que emite una fuente de luz en una dirección dada, contenido en un ángulo sólido, y dicho ángulo sólido. Su unidad de medida en el Sistema Internacional de Unidades es la candela (cd).
Su expresión matemática es $I = \Psi / \Omega$

Donde:

I = Intensidad

Ψ = Flujo luminoso

Ω = Ángulo sólido

Interpolación.- es el proceso en el cual se estiman los valores de una imagen en una sección específica, cuando por ejemplo, se cambia el tamaño de una imagen y en la nueva imagen existen más píxeles que en la imagen original. Se trata del proceso de calcular valores numéricos desconocidos a partir de otros ya conocidos mediante la aplicación de algoritmos concretos.

Lumen.- La unidad internacional (SI) para medir el flujo luminoso o cantidad de luz. Por ejemplo, la vela de un candelabro proporciona cerca de 12 lúmenes.

Luminancia.- Se define como la sensación luminosa, que por efecto de la luz, se produce en la retina del ojo. Es la densidad superficial de la intensidad luminosa y se expresa como la relación entre la intensidad luminosa y la superficie desde la cual se emite:

$$L = \frac{I}{S}$$

Matiz.- atributo asociado a la longitud de onda predominante en un color real (mezcla de longitudes de onda)

Nivel de gris.- Valor l de una imagen monocromática f en el punto dado por las coordenadas (x,y), dentro del rango $L_{mín} \leq l \leq L_{máx}$, que representa una variable física dada.

Píxel.- El píxel ("elemento de la imagen") es la menor unidad en la que se descompone una imagen digital, es un único punto en una imagen gráfica, ya sea una fotografía, un fotograma de vídeo o un gráfico.

Procesamiento digital de imágenes.- Se entiende a la manipulación de una imagen a través de una computadora, de modo que la entrada y salida del proceso sean imágenes

Resolución.- Nos indica la cantidad de píxeles que hay en una determinada medida de longitud (una pulgada o un centímetro);

RGB.- ***R****ed, **G**reen, **B**lue*. Modelo de color dependiente del dispositivo compuesto por los 3 colores básicos Rojo, Verde y Azul. Al combinarlos aditivamente entregan una gama dada de colores y que al unirlos en proporciones aproximadamente iguales, generan los tonos de gris.

Saturación.- Se refiere a la pureza relativa o cantidad de luz blanca mezclada con un matiz. La saturación es inversamente proporcional a la cantidad de luz blanca. Ej. El punto "blanco" del espacio CIE tiene saturación 0.

Tono.- Se entiende por **TONO** a la brillantez visual de una zona de una imagen que puede distinguirse de otras partes más claras o más oscuras.

Umbral.- Valor que sirve para binarizar las imágenes de niveles de gris. Se utiliza para dividir un conjunto de valores en dos grupos: objetos y fondo.

YIQ - ***(Y)****Illumination, **(I)** Phase, **Q**uadrature*. Es el modelo de color que se utiliza como estándar en las emisiones de la televisión en el formato NTSC. Sirve para convertir imágenes de color a imágenes de intensidad.