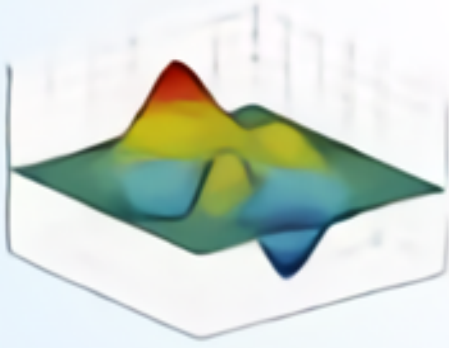
Visión Artificial y Procesamiento Digital de Imágenes usando Matlab



Ing. Oscar Sarzosa Sandoval

Visión Artificial y Procesamiento Digital de Imágenes usando Matlab

Visión Artificial y Procesamiento Digital de Imágenes usando Matlab

Iván Danilo García Santillán

Ibarra – Ecuador
2008.

Resumen Ejecutivo

La visión artificial (o visión por computador) es una disciplina compleja que involucra otras ciencias e incluye estudios de física, matemáticas, ingeniería eléctrica, computación. El continuo desarrollo de nuevos algoritmos, funciones y aplicaciones, hacen de esta disciplina y del procesamiento digital de imágenes (PDI) una tecnología perenne y en constante evolución. La visión artificial intenta emular la capacidad que tienen algunos seres vivos para ver una escena y entenderla; conjuntamente con el PDI, han experimentado una rápida evolución en las dos últimas décadas.

Los procesos del PDI pueden ser subdivididos en seis áreas principales y están agrupados de acuerdo a la complicación y delicadeza que lleva su implementación. Estas son: captura y adquisición, preprocesamiento, segmentación, descripción, reconocimiento e interpretación. No todas las aplicaciones requieren de todos los procesos y depende de la complejidad del problema que se va a resolver. Los resultados obtenidos en este tipo de aplicaciones dependen de la calidad de la imagen original, por lo que se deben tomar todas las precauciones necesarias para tener una iluminación adecuada y uniforme en el momento de su adquisición.

La industria, gobierno, educación, entretenimiento y el entorno en el cual nos desenvolvemos requieren de soluciones confiables relacionadas con este tipo de software, por lo que su introducción y aplicación será sostenible. España, a través de sus centros de educación superior, es uno de los países pioneros en el tema, por lo que podría pensarse en la transferencia de tecnología a través de proyectos de exportación que sean aplicables a nuestra región, tomando en consideración nuestras necesidades y realidades de una nación en vías de desarrollo.

Matlab es un paquete completo de software que goza de un alto nivel de implantación en centros de educación, departamentos de investigación y desarrollo de muchas compañías industriales nacionales e internacionales. Incluye algunos toolbox para diferentes áreas, tales como: finanzas, estadística, procesamiento de imágenes, procesamiento de la señal, redes neuronales, simulación, etc.

Prólogo

La presente obra denominada “Visión artificial y procesamiento digital de imágenes usando Matlab” se ha creado tanto para profesionales como estudiantes de informática, sistemas, computación, electrónica, electricidad y afines con el objetivo de proporcionar una guía teórica y práctica que sea clara y digerible y, a su vez, no pierda el rigor técnico y científico de esta compleja e importante área del conocimiento.

El texto está dividido en 8 capítulos, los cuales cubren gran parte de esta vasta disciplina. En el capítulo I se hace una introducción a la Teoría del Color. El capítulo II hace un estudio de los conceptos de la Imagen Digital. En el capítulo III se realiza un análisis de los fundamentos del Procesamiento Digital de Imágenes. En el capítulo IV se hace una Introducción a Matlab. En el capítulo V se analizan las técnicas de filtrado y realzado en imágenes digitales. En el capítulo VI se hace un estudio de las

Operaciones Morfológicas sobre imágenes digitales. En el capítulo VII se introduce a las técnicas de Segmentación; y en el capítulo VIII se examinan los métodos de Clasificación y Reconocimiento.

Los ocho capítulos son indispensables para el desarrollo e implementación de aplicaciones relacionados con la visión artificial.

El libro, además, contiene tres anexos. El anexo A, que incluye el manual de instalación del sistema de reconocimiento automático de placas; el anexo B contiene el manual de usuario del sistema y el anexo C contiene las imágenes necesarias para las prácticas, que están organizadas y clasificadas por capítulos.

Con esto se espera que estudiantes y profesionales propongan y desarrollen proyectos innovadores que solucionen problemáticas puntuales de nuestro entorno, especialmente en aquellas actividades repetitivas o que son peligrosas para el ser humano.

Algunos ejemplos de este tipo de proyectos son: control de calidad en estampados de camisetas, inspeccionar el nivel del líquido en embases de vidrio, semáforos inteligentes, análisis de imágenes médicas, reconocimiento de firmas manuscritas, detección de suelos erosionados, reconocimiento automático de placas de vehículos, inspección automática de objetos industriales, entre otras.

Finalmente, con la publicación de esta obra se pretende disminuir la brecha tecnológica que existe en este tipo de conocimientos especializados, propios de los países desarrollados.

El Autor

Agradecimientos

A toda la comunidad que conforma la Pontificia Universidad Católica del Ecuador sede Ibarra, especialmente a la Escuela de Ingeniería y al Departamento de Investigaciones, que han hecho posible la culminación del presente trabajo investigativo.

A mis alumnos, por permitirme poner de manifiesto todo lo aprendido en estos últimos años de docencia e investigación.

A todos ellos mis sentimientos de aprecio y gratitud.

Dedicatoria

Con mucho amor a mi esposa Sary y mi hijo Andy Josué, testigos y motivo de mi esfuerzo, constancia y dedicación.

Comentario

Este libro representa una investigación en mejora, por su contribución potencial al tema que fomenta, desarrollado por el Ing. Iván Danilo García Santillán, Docente y Coordinador de Investigación de la Escuela de Ingeniería de la Pontificia Universidad Católica del Ecuador – Sede Ibarra, como producto de su experiencia docente. Fundamenta sus conocimientos en el área de la Visión Artificial y Procesamiento Digital de Imágenes, usando Matlab.

El contenido desarrollado en este libro es tan vigente que será de mucha utilidad para quienes deseen introducirse en esta área del conocimiento. La Visión artificial o visión por computador es un área multidisciplinar que pretende, en cierta medida, reproducir artificialmente el sentido de la vista, mediante el procesamiento e interpretación de imágenes, captadas con distintos tipos de sensores, fundamentalmente cámaras, y utilizando la presentación en el computador. El desarrollo de este tipo de sistemas requiere una combinación de etapas de bajo nivel, para mejorar la calidad de las imágenes capturadas en otras etapas, como el reconocimiento de patrones y de la interpretación de imágenes para reconocer los elementos presentes en una escena. Las aplicaciones de esta área son, en general, trascendentales para resolver problemas concretos, con la aplicación de diversas técnicas matemáticas, estadísticas y de inteligencia artificial.

Es un libro que aporta, en gran medida, al desarrollo de la investigación y emprendimiento de alumnos y docentes. Lo recomiendo para que, en conjunto con otras áreas del conocimiento y líneas de investigación afines, contribuyan a lograr una investigación pertinente.

Ing. Perla Sevillano B.

Contenidos

Capítulo I
Fundamentos del Color

Capítulo II
La Imagen Digital

Cuantización

Capítulo III
Procesamiento Digital de Imágenes

Introducción al Procesamiento digital de imágenes
- Orígenes del procesamiento digital de imágenes.
- Aplicaciones del Procesamiento de Imágenes
- Componentes de un sistema PDI.
- Herramientas para el PDI

Fundamentos del procesamiento de imágenes digitales
- Relaciones entre píxeles
- Conectividad
- Distancia

Ruido en imágenes
- Gaussiano
- Impulsional
- Multiplicativo

Capítulo IV
Introducción a Matlab

Introducción a Matlab
- Image Processing Toolbox
- El entorno de trabajo

Manejo de Variables en Matlab

Tipos de Datos en Matlab

Lectura y escritura interactiva de variables
- Función Input
- Función Disp

Manejo de las imágenes en Matlab
- Tipos de imágenes
- Lectura de imágenes
- Acceso a un píxel de una imagen
- Visualización de imágenes
- Conversiones entre tipos de imágenes
- Comandos informativos de imágenes
- Escritura de imágenes
- Selección de una sección de una imagen
- Tamaño de una imagen
- Añadir ruido a una imagen

Manejo de ventanas en Matlab
- Subplot

Programación en Matlab
- Creación de funciones
- Creación de sub-funciones

Capítulo V
Filtrado y Realzado de Imágenes

- Operaciones básicas entre píxeles
  - Operaciones aritméticas
    - Suma
    - Resta
    - Multiplicación
    - División
  - Operaciones lógicas
    - And
    - Or
    - Not
  - Operaciones geométricas
    - Interpolación
    - Amplificación y Reducción
    - Rotación
    - Correlación
- Operaciones sobre el histograma
  - Aumento y reducción del contraste
  - Ecualizado del histograma
  - Ajuste de la intensidad
- Operaciones en el dominio de la frecuencia
- Filtrado Espacial
  - Filtros de paso bajo
    - Promedio
    - Mediana
  - Filtros de paso alto
    - Realce de bordes
    - Detección de contornos

Capítulo VI
Operaciones Morfológicas

- Definiciones básicas
- Aplicaciones de la morfología matemática
- Elementos del proceso morfológico
  - Conjuntos
  - Elementos estructurantes
  - Operadores morfológicos
    - Dilatación
    - Erosión
    - Apertura
    - Cierre
- Filtros morfológicos
  - top-hat
  - bottom-hat

Capítulo VII
Segmentación

Capítulo VIII
Clasificación y Reconocimiento

ANEXOS

Capítulo I
Fundamentos del Color

Contenidos

REPRESENTACIÓN DEL COLOR

El mundo es de colores; donde hay luz, hay color. Al descomponer la luz encontramos que está compuesta por siete colores (existen otros más, pero no son detectados por el ojo humano), tal como se muestra en la figura 1.1.

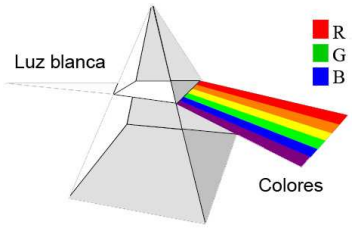


Fig. 1.1 Descomposición de la luz blanca

El color es un atributo que percibimos de los objetos cuando hay luz. La luz está constituida por ondas electromagnéticas que se propagan a unos 300.000 kilómetros por segundo.

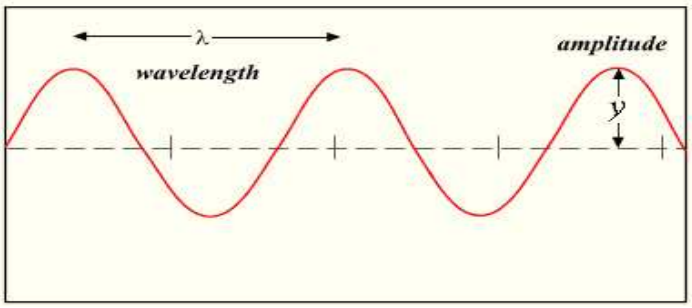


Fig. 1.2 Propiedades de las Ondas

Las ondas forman, según su longitud de onda (ver figura 1.2), distintos tipos de luz, como infrarroja (IR), visible, ultravioleta (UV) y blanca. Las ondas visibles son percibidas por el ojo humano y su longitud de onda está comprendida entre los 380 y 770 nanómetros[1], como se muestra en la figura 1.3.

[1] Nanómetro.- Medida de longitud que equivale a la milmillonésima (10^{-9}) parte del metro. Símbolo *nm*.

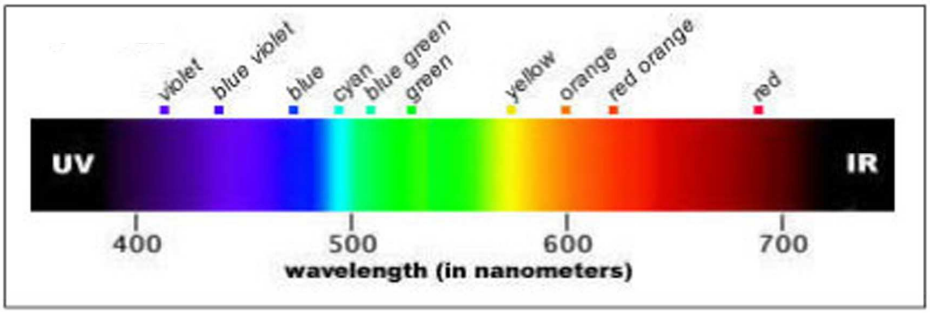


Fig. 1.3 El Espectro visible. [www04]

Todos los cuerpos están constituidos por sustancias que absorben y reflejan las ondas electromagnéticas, es decir, absorben y reflejan colores. Un cuerpo opaco (no transparente) absorbe gran parte de la luz que lo ilumina y refleja una parte pequeña. Cuando este cuerpo absorbe todos los colores contenidos en la luz blanca, el objeto aparece negro.

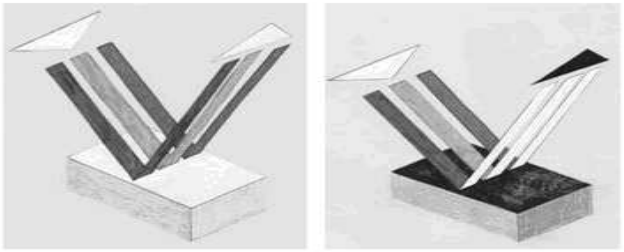


Fig. 1.4 Absorción y Reflexión de la Luz [www04]; (a) Refleja un cuerpo blanco; (b) Refleja un cuerpo negro

Cuando refleja todos los colores del espectro, el objeto aparece blanco. Los colores absorbidos desaparecen en el interior del objeto y los reflejados llegan al ojo humano. Los colores que visualizamos son, por tanto, aquellos que los propios objetos no absorben, sino que los propagan.

Por ejemplo, el tomate nos parece de color rojo, porque el ojo sólo recibe la luz roja reflejada por la hortaliza, absorbe el verde y el azul y refleja solamente el rojo. Un plátano amarillo absorbe el color azul y refleja los colores rojo y verde, los cuales sumados permiten visualizar el color amarillo.

Cuando un cuerpo se ve blanco (figura 1.4a) es porque recibe todos los colores básicos del espectro (rojo, verde y azul) los devuelve reflejados, generándose así la mezcla de los tres colores, el blanco.

Si el objeto se ve negro (figura 1.4b) es porque absorbe todas las radiaciones electromagnéticas (todos los colores) y no refleja ninguno.

COLORES PRIMARIOS

Los colores primarios son aquellos que por la mezcla producirán todos los demás colores. En realidad existen dos sistemas de colores primarios: colores luz y colores pigmento.

COLORES LUZ (Síntesis Aditiva)

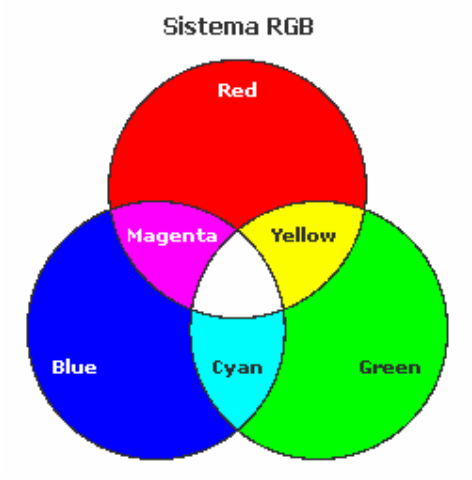


Fig. 1.5 Colores Luz

Los colores producidos por luces (en el monitor, televisión, cine, etc.) tienen como colores primarios, al rojo, el verde y el azul (RGB) cuya fusión, crean y componen la luz blanca. Por eso, a esta mezcla se la denomina, síntesis aditiva, y las mezclas parciales de estas luces dan origen a la mayoría de los colores del espectro visible.

Para representar un color en el sistema RGB se le asigna un valor entre 0 y 255 (notación decimal) o entre 00 y FF (notación hexadecimal) para cada uno de los componentes rojo, verde y azul que lo conforman. El color rojo puro, por ejemplo, se especificará como (255,0,0) en notación RGB decimal y #FF0000 en notación RGB hexadecimal.

COLORES PIGMENTO (Síntesis Sustractiva)

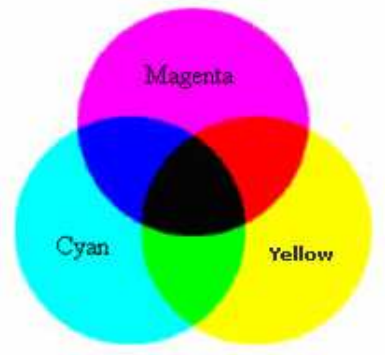


Fig. 1.6 Colores Pigmento

El color magenta, el cyan y el amarillo son los colores básicos de las tintas que se usan en la mayoría de los sistemas de impresión. La mezcla de los tres colores primarios pigmento en teoría debería producir el negro, el color más oscuro y de menor cantidad de luz, por lo cual esta mezcla es conocida como síntesis sustractiva. En la práctica el color así obtenido no es lo bastante intenso, motivo por el cual se le agrega el negro pigmento, conformándose el espacio de color CMYK.

El blanco y negro son llamados **colores acromáticos** (sin color). El negro es la ausencia de luz y de color.

Podemos distinguir dos tipos de luz en función del color, la luz acromática y la cromática.
Acromática.- Sin color, y su único atributo es la intensidad o cantidad de luz.
Cromática.- Con color, y se caracteriza mediante tres parámetros:

- ✓ *Radiancia.-* Cantidad total de energía que sale de la fuente luminosa.
- ✓ *Luminancia.-* Cantidad de energía que percibe un observador, procedente de la fuente luminosa. Por ejemplo: la fuente infrarroja tiene alta radiancia y muy baja luminancia.

- ✓ *Brillo.*- Es la luminosidad de un color. Se explicará con mayor detalle en las propiedades del color.

El uso del color en el procesamiento de imágenes y la visión artificial es fundamental principalmente por dos factores:

1. El color es un descriptor poderoso que simplifica la identificación de objetos y su extracción de una escena.
2. Los seres humanos podemos distinguir cientos de colores, sombras e intensidades, y es de gran importancia para el análisis de imágenes que hacemos en nuestro cerebro.

MODELOS DE COLOR

Su propósito es el de facilitar la especificación de los colores utilizando algún estándar. Los modelos de color están orientados, por un lado, al *hardware* (monitores e impresoras) y, por otro lado, a alguna *aplicación* para la creación de gráficas a color y animaciones. Los orientados al hardware, en la práctica utilizan el modelo RGB (rojo, verde, azul) para monitores y cámaras de video; el modelo CMY (cyan, magenta, amarillo) y el CMYK (cyan, magenta, amarillo, negro) para impresoras en color y el modelo HSI (tono, saturación, intensidad) que corresponde al modelo más cercano a la manera en que los humanos perciben el color.

1. MODELO RGB

En el modelo RGB, cada color aparece en sus componentes espectrales primarios de rojo, verde y azul. Las imágenes representadas en este modelo consisten en tres componentes de imágenes, una para cada color primario.

El número de bits que se utiliza para representar cada píxel en el espacio RGB es llamado profundidad de píxel. Así, una imagen RGB en que cada imagen roja, verde y azul es una imagen de 8-bits tiene 24 bits de profundidad, puesto que cada píxel es una tripleta de valores (R, G, B) y tiene un número total de 2^{24} = 16,777,216 colores.

Cada píxel en una imagen de 24-bit, posee uno de los 256 valores de brillo para el rojo, verde y azul.

Fig. 1.7 Píxeles de 24-bits

2. MODELO CMYK

El cyan, magenta y amarillo son los colores secundarios de la luz o bien los colores primarios de los pigmentos. Los dispositivos que depositan pigmentos coloreados sobre papel (como impresoras y fotocopiadoras en color) necesitan una entrada CMY o bien una conversión

interna de RGB a CMY. En la teoría, la mezcla de los tres colores CMY da el negro, pero en la práctica es de mala calidad, por lo que se incluye un cuarto color K que representa el negro.

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).

3. MODELO HSI

Los modelos RGB y CMY no son útiles para describir los colores en términos prácticos para la interpretación humana. Por ejemplo, una persona no se refiere al color de un auto dando porcentajes del contenido de cada uno de sus colores primarios, sino que lo describe en términos de su tono (H), saturación (S) y su brillo o intensidad (I)

Existen algunas utilidades de este modelo como el diseño de sistemas automáticos para detectar el grado de maduración de frutas y vegetales, sistemas para comparar muestras de color o inspeccionar la calidad de productos coloreados.

PROPIEDADES DEL COLOR

Todo color posee una serie de propiedades que le hacen variar de aspecto y que definen su apariencia final. Entre estas propiedades cabe distinguir:

El Tono:
Es el matiz del color, es decir, el color en sí mismo. Es simplemente un sinónimo de color. Por ejemplo, el matiz de la sangre es rojo. Según su tonalidad se puede decir que un color es rojo, amarillo, verde, etc.

Aquí podemos hacer una división entre:
- *Tonos cálidos* (rojo, amarillo y anaranjados): aquellos que asociamos con la luz solar, el fuego.
- *Tonos fríos* (azul y verde): Los colores fríos son aquellos que asociamos con el agua, la luz de la luna.

El Matiz se define como un atributo de color que nos permite distinguir el rojo del azul, y se refiere al recorrido que hace un tono hacia uno u otro lado del círculo cromático, por lo que el verde amarillento y el verde azulado serán matices diferentes del verde.

Fig. 1.8 Matices en el círculo cromático

Saturación:
Indica la concentración de color en el objeto. Puede ser definida por la cantidad de gris que contiene un color: mientras más gris o más neutro es, menos brillante o menos "saturado" es. Igualmente, cualquier cambio hecho a un color puro automáticamente baja su saturación.

Por ejemplo, decimos "un rojo muy saturado" cuando nos referimos a un rojo puro y rico. Pero cuando nos referimos a los tonos de un color que tiene algún valor de gris, los llamamos menos saturados.

Brillo o Valor:
Es la luminosidad de un color (la capacidad de reflejar el blanco). Alude a la claridad u oscuridad de un tono. La luminosidad puede variar añadiendo negro o blanco a un tono, tal como se muestra en la figura 1.9.

Fig. 1.9 Diferentes brillos del color rojo

A medida que a un color se le agrega más negro, se intensifica dicha oscuridad y se obtiene un valor más bajo. A medida que a un color se le agrega más blanco se intensifica la claridad del mismo, por lo que se obtienen valores más altos.

La descripción clásica de los valores corresponde a claro (cuando contiene cantidades de blanco), medio (cuando contiene cantidades de gris) y oscuro (cuando contiene cantidades de negro). Cuanto más brillante es el color, mayor es la impresión de que el objeto está más cerca de lo que en realidad está.

.

FALSO COLOR Y PSEUDOCOLOR.

FALSO COLOR.- Transforma una imagen de color en otra imagen a color. En el procesado de imágenes de satélite, se generan a menudo imágenes en falsos colores porque incrementan la percepción de determinados detalles de la superficie. Por ejemplo, el pasto verde le pasa a color rojo. Una imagen en falso color es una representación artificial de una imagen multiespectral[2]. En la figura 1.10 se muestra un ejemplo de color falso.

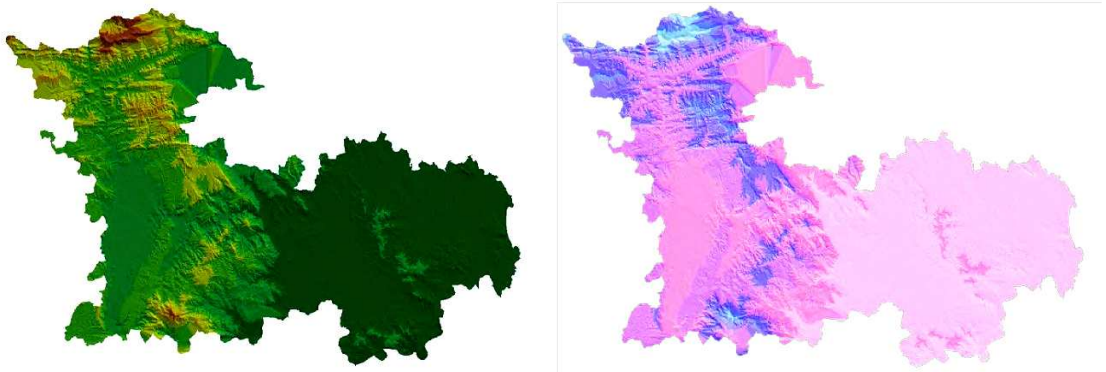


Fig. 1.10 Imagen de una superficie terrestre; (a) imagen en color verdadero; (b) imagen en falso color.

PSEUDOCOLOR.- Esta técnica utiliza imágenes con tonos de grises y realiza una transformación para generar una imagen en color, la cual es más agradable de visualizar. Para esto se ha sustituido la escala de grises por una tabla o paleta de colores, ver figura 1.11.

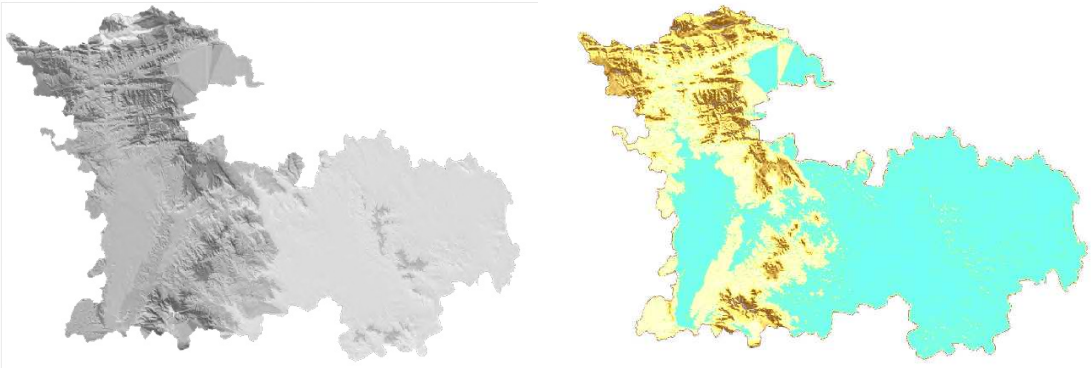


Fig. 1.11 Imagen de una superficie terrestre; (a) imagen en escala de grises; (b) imagen en pseudocolor.

[2] Conjunto de imágenes, con las mismas propiedades geométricas, y cada una de las cuales en un diferente rango de longitudes de onda del espectro electromagnético.

Capítulo II
La Imagen Digital

Definición
Clasificación de las imágenes digitales
- Imágenes Vectoriales
- Imágenes Ráster

Formatos de Imágenes Ráster
- TIFF
- BMP
- GIF
- JPEG
- PNG

Tipos de Imágenes Digitales
- RGB
- Indexadas
- Escala de grises
- Binarias

Calidad de una Imagen
- Resolución
- Tamaño

Captura de Imágenes
- Cámara digital
- Escáner

Digitalización de Imágenes
- Muestreo
- Cuantización

LA IMAGEN DIGITAL

Definición.- Una imagen puede definirse como una función bidimensional *f(x,y)* donde *x* y *y* son coordenadas en el plano y la amplitud *f* es llamada intensidad o nivel de gris en ese punto.

Cuando *(x, y)* y *f* son todos finitos (cantidades discretas) llamamos a la función como imagen digital. Es decir, una imagen digital estará compuesta por un número finito de elementos llamados *píxeles*, cada uno de los cuales con un *valor* y una *posición* particular.

El valor es relativo a alguna propiedad del punto que representa, como por ejemplo su brillo o su matiz.

El término ***píxel*** (abreviación de Picture element o elemento de imagen), se trata de la unidad mínima de información de una imagen, la cual aparece como un punto en la pantalla o en una impresora. En realidad cada píxel se compone de tres registros de color, mediante la combinación de cierta cantidad de rojo, verde y azul, el píxel adopta un color particular.

Las imágenes bidimensionales son el resultado de una proyección en perspectiva de escenas tridimensionales. Cuando se obtiene una imagen bidimensional del mundo tridimensional desaparece gran cantidad de información.

CLASIFICACIÓN DE LAS IMÁGENES DIGITALES

A grandes rasgos podríamos dividir las imágenes digitales en dos grupos:

1. **IMÁGENES VECTORIALES.-** Los gráficos vectoriales conservan la nitidez de los bordes y no pierden detalles cuando se modifica el tamaño puesto que son independientes de la resolución. La información de cada uno de los puntos se recoge en forma de ecuación matemática que lo relaciona con el resto de los puntos que forman la imagen. Ofrece la gran ***ventaja*** de que la calidad de la imagen no varía al modificar el tamaño, ya que la información de cada punto no es absoluta sino relativa al resto de la imagen. Además, debido a su definición matemática, apenas ocupa espacio, ya que una fórmula que represente su forma es suficiente para representar todos los puntos que la componen. Es el tipo adecuado para el diseño de líneas, polígonos, figuras. No es soportado de forma directa por los navegadores de Internet como: Internet Explorer, Netscape Navigator, Firefox, Mozilla, etc. Algunos formatos de este tipo de imágenes son: DWG (autoCAD), SWF y FLA (Flash).

2. **IMÁGENES RASTER O MAPA DE BITS.-** Las imágenes raster constan de un número fijo de píxeles y, por tanto, dependen de la resolución. Las imágenes raster pueden perder detalle y verse dentadas (pixeladas) cuando se amplían. Utilizan una

cuadrícula rectangular de elementos de imagen (píxeles) para representar las imágenes. A cada píxel se le asigna una ubicación y un valor de color específico. La ventaja que presenta este formato es la posibilidad de recoger una amplia gama tonal, por lo que es el tipo adecuado para representar imágenes captadas de la realidad. En cambio, la variación de tamaño supondrá modificaciones en la calidad, ya que el número de celdas que forman la imagen permanece invariable, por lo que un aumento del tamaño hace que el único recurso posible sea ampliar el tamaño de cada una de las celdas. Podemos deducir que su tamaño es muy grande por la información de cada uno de los puntos que forman la imagen.

Dentro de este tipo se encuentran muchos formatos, algunos de los cuales son soportados directamente por los navegadores, siendo el tipo de imágenes con las que vamos a trabajar. Estas imágenes son creadas por los escaners y las cámaras digitales.

FORMATOS DE IMÁGENES RASTER

Entre los formatos de imágenes más populares en nuestro medio tenemos:

- **TIFF**

Se trata de un formato de imágenes muy difundido a causa de su facilidad de lectura tanto en PC como en Macintosh, debido a la compresión de imágenes sin pérdida de calidad. Su principal desventaja es que, una vez descomprimidas, las imágenes pueden ser grandes, motivo por el cual no se les usa en la Web.

- **BMP**

Este es el formato tradicional para los usuarios de Windows. Se le puede emplear con propósitos generales, como en la edición de imágenes y tapiz del escritorio de Windows. No siempre puede ser leído por computadoras Macintosh y sus archivos tienden a ser grandes. Tampoco se le soporta en la Web. En general, este formato sólo tiene sentido en la actualidad para almacenar imágenes de uso exclusivo en Windows.

Los tres formatos de archivo comúnmente reconocidos por los navegadores Web son: GIF, JPEG y PNG.

- **GIF**

Creado por la compañía CompuServe y significa formato de intercambio de gráficos (Graphic Interchange Format). Se trata de un formato de archivo compacto de uso muy común en páginas Web. Su principal desventaja es que limita las imágenes a sólo 256 colores, lo cual puede afectar la calidad de la imagen en la pantalla.

A los diseñadores de páginas Web suele agradarles el formato GIF 89a, la que incluye tres importantes características:

- Transparencia.- Esta característica permite especificar como transparente un color de una imagen. Así, pueden ser transparentes el fondo o contorno de una imagen, para evitar desagradables uniones o marcos alrededor de ésta.

- Entrelazado.- Al descargar una imagen entrelazada en un examinador Web, primero aparece una versión general de baja resolución, cuya calidad aumenta paulatinamente a medida que se obtiene más información. Por lo general, una imagen no entrelazada se carga en grupos de líneas, de modo que primero aparece la parte de arriba y más tarde la de abajo. La descarga del entrelazado no es más rápida, pero ofrece al espectador una imagen completa mientras concluye la descarga del resto de la imagen, de manera que puede afirmarse que, subjetivamente, sí es más veloz.

- Animación.- Un GIF animado es una secuencia de imágenes reproducidas una tras otra. Son muy utilizadas en las páginas Web.

- **JPEG**

Creado por el Grupo Unido de Expertos en Fotografía (Joint Photographic Experts Group), al que debe su nombre. Es un formato de archivo comprimible con posibilidades de escalamiento para producir archivos reducidos. Sin embargo, de acuerdo con el grado de compresión de una imagen JPEG, la calidad de imagen puede variar poco o mucho. Permite la exhibición de la paleta íntegra de 16 millones de colores, a diferencia del GIF.

Al guardar un archivo en formato JPEG, usted puede especificar el nivel de compresión por aplicar. La respectiva opción predeterminada de la mayoría de los programas induce un nivel de compresión muy reducido, con el fin de preservar la calidad de imagen.

- **PNG**

Es el formato de archivo más reciente en la Web, el formato PNG (Portable Network Graphics: gráfico de red portable) es similar al JPEG en el sentido de que también permite la exhibición de imágenes de amplio colorido, pero su compresión no reduce la calidad de imagen. Como consecuencia de haber sido diseñado para Internet, este formato posee muchas otras características, pero a causa de su reciente aparición, su uso es aún restringido.

TIPOS DE IMÁGENES DIGITALES

En el procesamiento digital de imágenes (PDI) se maneja cuatro tipos de imágenes básicamente: imágenes RGB, imágenes indexadas, imágenes en escala de grises e imágenes binarias, las cuales se explicarán a continuación.

1. IMÁGENES RGB (Red-Green-Blue)

- Utilizan tres canales para reproducir los colores en la pantalla.
- Utilizan 8 bits por canal (8 bits x 3), es decir, 24 bits de color para cada píxel.

- Reproducen hasta 16,7 millones de colores.
- Soporta algunos formatos como: JPG, BMP, PNG, etc.

2. IMÁGENES INDEXADAS

- Reduce los colores de la imagen a un máximo de 256.
- Admiten los formatos GIF y PNG-8 y muchas aplicaciones multimedia.
- Reduce el tamaño de archivo porque elimina la información del color.

3. IMÁGENES EN ESCALA DE GRISES

- Utilizan distintos tonos de gris.
- En imágenes de 8 bits, puede tener hasta 256 tonos de gris.
- Cada píxel tiene un valor de brillo comprendido entre 0 (negro) y 255 (blanco).

4. IMÁGENES BINARIAS

- Tienen una profundidad de color de 1 bit.
- Utiliza uno de los dos valores de color (blanco o negro) para representar los píxeles de una imagen.

Existen además imágenes con una profundidad de píxel de 32 bits. Los 8 bits (1 byte) adicionales de profundidad sobre las imágenes de 24 bits, le permiten almacenar la transparencia de la imagen. Este byte adicional es generalmente llamado máscara o canal alfa, y almacena diferentes valores de transparencia.

En la figura 2.1 se muestra algunos ejemplos correspondientes a los tipos de imágenes.

(a) (b) (c) (d)

Fig. 2.1 Tipos de Imágenes Digitales; (a) RGB; (b) Indexada; (c) Escala de Grises; (d) Binaria

CALIDAD DE UNA IMAGEN

La imagen digital, bien sea generada por el ordenador o creada a través de algún dispositivo de captura, tal como una cámara digital o un escáner, aporta una principal ventaja que es la estabilidad, mientras que la emulsión de una imagen fotográfica clásica sufre una degradación química con el paso del tiempo, que repercute en la calidad de dicha reproducción, los ceros y unos que componen una imagen digital permanecen estables, con lo que la imagen no variará a lo largo del tiempo.

La calidad de la imagen ráster es determinada en el proceso de captura por tres factores: el tamaño del píxel (resolución espacial), la profundidad del píxel (resolución de brillo) y el ruido.

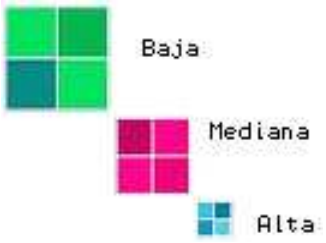


Fig. 2.2 Tamaño del píxel (Resolución espacial)

El tamaño del píxel es determinado por el rango al cual el escáner muestrea la imagen. Un intervalo de muestreo largo produce una imagen baja en **resolución espacial**. Un intervalo más corto produce una resolución espacial más alta, tal como se muestra en la figura 2.2.

Fig. 2.3 Profundidad del píxel (Resolución de brillo)

El brillo o valor de color de cada píxel es definido por un bit o un grupo de bits. Mientras más bits se usen, más alta es la resolución de brillo, (figura 2.3).

Todas las imágenes tienen cierta cantidad de **ruido**, ya sea por la cámara, escáner o el medio de transmisión de la señal. Generalmente el ruido se manifiesta como píxeles aislados que toman un nivel de gris diferente al de sus vecinos y aparece como pequeñas y aleatorias variaciones en el brillo y el color (ver figura 2.4). Los algoritmos de filtrado que se verán más adelante, permiten eliminar o disminuir este ruido.

(a) (b)

Fig. 2.4 Ruido que afecta a una imagen; (a) original; (b) imagen con ruido

La resolución de una imagen es el número de píxeles que contiene una imagen, expresada como 640 x 480, 800 x 600, por ejemplo. Es un término que debe ser considerado al utilizar imágenes en determinados trabajos.

En general, una baja resolución de imagen se utiliza para:

- Imágenes para páginas Web y correo electrónico.
- La memoria de la cámara es limitada.
- Se dispone de escaso espacio en disco duro para el almacenamiento de imágenes.

En general, se debe recurrir a una resolución más alta si:

- Las imágenes están destinadas a impresiones de alta resolución.

- Se dispone de suficiente espacio de almacenamiento, tanto en la cámara como en el disco duro.

El tamaño de una imagen se puede calcular multiplicando la cantidad de píxeles horizontales (ancho) por la cantidad de píxeles verticales (alto) y por la profundidad de brillo (en bits). En la tabla 2.1 se muestra algunos ejemplos de tamaños de imágenes.

Resolución	Profundidad del píxel	Tamaño del archivo			
		bits	bytes	Kbytes	Mbytes
640 x 480	x 1 bit	= 307.200	= 38.400	= 37.5	= 0.036
640 x 480	x 8 bits	= 2´457.600	= 307.200	= 300	= 0.292
640 x 480	x 24 bits	= 7´372.800	= 921.600	= 900	= 0.878
640 x 480	x 32 bits	= 9´830.400	= 1´228.800	= 1200	= 1.171

Tabla 2.1 Tamaño de una Imagen

CAPTURA Y DIGITALIZACIÓN DE IMÁGENES

1. CAPTURA DE IMÁGENES

El proceso de captura se refiere a la adquisición de la imagen del mundo físico. La imagen puede ser capturada o generada de varias maneras: fotográficamente, con elementos químicos; o a través de dispositivos electrónicos como el computador, escáner, cámara digital o video-cámara digital

a) LA CÁMARA DIGITAL.- Hay cámaras de toda forma, tamaño, precio y funciones. Algunas características de las cámaras sobre las cuales se deben investigar antes de su adquisición son:

Resolución.- Número de píxeles que contienen las imágenes. La mayoría de cámaras comerciales generan imágenes de 640x480 píxeles que son adecuadas para muchas aplicaciones. En la mayoría de cámaras es posible elegir entre dos o más resoluciones.

Memoria.- En general, cuanto mayor sea la memoria es mucho mejor. A usted le interesa alojar en la memoria tantas fotografías como sea posible antes de proceder a su descarga en la computadora. Las cámaras pueden incluir memoria interna fija, así como tarjetas de memoria intercambiables.

El formato de las imágenes capturadas serán del tipo JPG, por lo que podrán ser exportadas a la mayoría de los programas de tratamiento.

Compresión de Archivos.- La mayoría de cámaras permite elegir el grado de compresión de las imágenes para su almacenamiento en la memoria. A menor compresión, mayor calidad de imagen pero también será menor el número de imágenes que es posible almacenar en la memoria.

Sistema de Transferencia.- Es importante conocer la calidad de operación del sistema de transferencia de una cámara para el traspaso de imágenes a la PC. La mayoría cuentan con conexión USB.

b) EL ESCÁNER.- Es un dispositivo que le permite realizar una copia digital de una fotografía u otro tipo de documento. Se trata de un aparato semejante a las cámaras digitales y es de gran utilidad para transferir imágenes a una PC cuando solo se cuenta con su versión impresa.

Un escáner puede concebirse como una cámara digital, pero en lugar de servirse como aquella, de un CCD[3] rectangular para captar una imagen completa en un solo paso, el escáner emplea un CCD con una sola fila de píxeles fotosensibles. Este CCD lineal registra el valor lumínico de cada una de las líneas o filas de una imagen. Al terminar una línea remite la información de inmediato a la PC para poder proceder la lectura de la línea siguiente.

Los principales aspectos que debe considerar al adquirir un escáner son: resolución e intensidad de color.

La resolución del escáner depende en parte de la velocidad y precisión del motor que impulsa pausadamente al CCD a lo largo del documento. Los escáner de alta resolución ofrecen un mayor grado de detalle en las porciones más claras y más oscuras de una imagen. Algunas resoluciones de escáner son: 300, 600, 1200, 2400, 4800, 9600 dpi (dots per inch – puntos por pulgada).

En la mayoría de escáner de bajo costo la **intensidad de color** es de 24 bits, lo cual significa que puede digitalizar hasta 16.7 millones de colores, cantidad similar a la perceptible por el ojo humano. Los escáner más complejos operan a razón de 30 o 36 bit por píxel, de manera que pueden distinguir entre miles de millones de colores.

(a) (b) (c)

Fig. 2.5 Dispositivos de captura; (a) cámara digital; (b) escáner; (c) video cámara digital

Un sistema básico de captura de imagen contiene un lente y un detector. En la fotografía digital, el detector es un sensor de imagen denominado **CCD** y consta de píxeles. Cada píxel representa un punto de color en la imagen terminada. Una cámara capaz de producir imágenes de 640x480 píxeles cuenta con un CCD de 300000 píxeles.

[3] Charge Coupled Device - dispositivo de carga acoplado

Como vimos anteriormente, la calidad de la imagen escaneada es determinada por el tamaño del píxel (resolución espacial) y por la profundidad de píxel (resolución de brillo). Esto está relacionado con los pasos básicos en el proceso de la captura digital.

2. LA DIGITALIZACIÓN

Es el proceso de paso del mundo continuo (o analógico) al mundo discreto (o digital). Una vez digitalizada una imagen bidimensional, ésta queda constituida por un conjunto de elementos básicos llamados píxeles. Cada píxel ofrece cierta información sobre una región elemental de la imagen, como el color o brillo, y la posición. En imágenes en blanco y negro esta información es el brillo. En imágenes en color, la información corresponde a la intensidad de cada una de las componentes de un modelo de color como: RGB, CMYK, HSI, etc.

En la digitalización normalmente se distinguen dos procesos: el Muestreo y la Cuantización.

EL MUESTREO es el proceso de obtener la imagen. Esto implica que la imagen se muestrea en una matriz con ***m*** filas y ***n*** columnas. El muestreo determina el *tamaño del píxel y el valor del brillo*. Cuando un dispositivo de captura muestrea la imagen fotográfica, divide la imagen en píxeles. El tamaño de los píxeles depende del número de foto celdas. En la figura 2.6 y 2.7 se muestran ejemplos de muestreos con diferentes valores para *m* y *n*.

Fig. 2.6 Imagen con baja resolución

Un CCD con pocas foto celdas, muestrea a una baja resolución. A una resolución extremadamente baja, los píxeles pueden ser percibidos por un ojo normal sin ayuda. Esto se denomina **pixelización**.

Fig. 2.7 Imagen con alta resolución

Un CCD con más foto celdas, muestrea a una resolución espacial más alta. En este tipo de imagen los píxeles individuales no pueden ser vistos.

LA CUANTIZACIÓN es el proceso de asignar valores a los elementos de la matriz. Cada valor representa al valor de la variable física en ese punto. A efectos de representación visual se asume que el valor más pequeño del rango de valores corresponde a un nivel de gris negro y que el valor más grande al nivel de gris blanco. Dentro de este intervalo, cuantos más valores se puedan discriminar, mayor cantidad de matices se podrán representar.

La pregunta obvia que surge al hablar de Muestreo y Cuantización es: ¿Cuáles son los valores adecuados de número de muestras y número de niveles distinguibles? La respuesta, también obvia, lo mejor es tener el mayor número posible de muestras (para obtener la mejor aproximación a la función imagen continua) y el mayor número posible de niveles (para poder percibir todos los detalles). Sin embargo, hay que considerar que cuantas más muestras y más niveles, más datos a procesar por el ordenador y más tiempo de computación necesario para obtener los resultados.

Capítulo III
Procesamiento Digital de Imágenes

Contenido

PROCESAMIENTO DIGITAL DE IMÁGENES

INTRODUCCIÓN

El procesamiento digital de imágenes (PDI) se refiere a procesar las imágenes del mundo real de manera digital por medio de un computador. Es un tema muy amplio, en el que se incluyen estudios de física, matemáticas, ingeniería eléctrica, computación. Estudia los fundamentos conceptuales de la adquisición y despliegue de imágenes y con detalle los fundamentos teóricos y algorítmicos del procesamiento como tal. Tiene además, como objetivo mejorar el aspecto de las imágenes y hacer más evidentes en ellas ciertos detalles que se desean hacer notar.

La Visión Artificial (o visión computacional) puede ser definida como los procesos de obtención, caracterización e interpretación de información de imágenes tomadas de un mundo tridimensional. Estos procesos pueden ser subdivididos en seis áreas principales y están agrupados de acuerdo a la complicación y delicadeza que lleva su implementación. Consideramos tres niveles de procesamiento: visión de bajo, medio y alto nivel, tal como se muestra en la tabla 3.1.

Procesos del PDI	Nivel de Visión
1. Captura/adquisición	Bajo
2. Preprocesamiento	
3. Segmentación	Medio
4. Descripción	
5. Reconocimiento	
6. Interpretación	Alto

Tabla 3.1 Niveles de visión y procesos del PDI

La captura o adquisición es el proceso a través del cual se obtiene una imagen digital utilizando un dispositivo de captura como una cámara digital, video cámara, escáner, satélite, etc.
El preprocesamiento incluye técnicas tales como la reducción del ruido, realce del contraste, realce de ciertos detalles, o características de la imagen.
La segmentación es el proceso que divide una imagen en objetos que sean de nuestro interés de estudio.
La descripción es el proceso que obtiene características convenientes para diferenciar un tipo de objeto de otro, como: la forma, el tamaño, área, etc.
El reconocimiento es el proceso que identifica los objetos, como por ejemplo: una llave, un tornillo, moneda, coche, etc.
La interpretación es el proceso que asocia un significado a un conjunto de objetos reconocidos (llaves, tornillos, herramientas, etc.) y trata de emular la cognición.

Presentaremos a los procesos de captura y preprocesamiento como funciones de visión de bajo nivel; la segmentación, descripción y reconocimiento como funciones de visión de nivel intermedio; y la interpretación como función de visión de alto nivel.

Como podemos apreciar, agrupamos los métodos y/o procesos del PDI en dos categorías principales:

1. Métodos cuya entrada y salida son imágenes.
2. Métodos cuyas entradas pueden ser imágenes y las salidas son atributos extraídos de esas imágenes.

NIVEL	MÉTODOS/PROCESOS	ENTRADA	SALIDA
Bajo	Reducción de ruido Realce de contraste Realce de características	Imagen	Imagen
Medio	Segmentación (regiones, objetos) Descripción de objetos Clasificación o Reconocimiento	Imagen	Atributos de objetos: bordes, contornos, áreas identidades de objetos individuales
Alto	Interpretación Análisis de la imagen Funciones cognitivas	Objetos encontrados	Análisis de la imagen (información, sentido a los objetos.)

Tabla 3.2 Entradas y salidas de los métodos del PDI

No todas las aplicaciones de PDI requieren de todos los procesos descritos anteriormente. Por lo general, mientras la complejidad del problema a resolver crece, el número de procesos requeridos también crece.

ORÍGENES DEL PROCESAMIENTO DIGITAL DE IMÁGENES

La historia del PDI se remonta a la década de los 60 y está directamente ligada con el desarrollo y evolución de las computadoras. Su progreso ha ido de la mano con el desarrollo de las tecnologías de hardware, ya que requiere un alto poder y recursos computacionales para almacenar y procesar las imágenes. De igual manera el desarrollo de los lenguajes de programación y los sistemas operativos han hecho posible el crecimiento continuo de aplicaciones relacionadas al procesamiento de imágenes, tales como: imágenes médicas, satelitales, astronómicas, geográficas, arqueológicas, biológicas, aplicaciones industriales, entre otras.

APLICACIONES DEL PROCESAMIENTO DE IMÁGENES

Existe una amplia gama de áreas donde el PDI se utiliza de manera rutinaria para la resolución de ciertos problemas, dependiendo de la fuente de energía, sean estas: rayos gamma, rayos X, banda ultravioleta, banda infrarroja, banda visible, microondas, radio, ultrasonido [www03].

Algunos ejemplos de aplicaciones son:

Imágenes de rayos gamma.- Imágenes médicas y observaciones astronómicas.

Fig. 3.1 Imagen de rayos gamma

Imágenes de rayos X.- Aplicaciones en la medicina, astronomía e industriales.

(a) (b)

Fig. 3.2 Imágenes con rayos X; (a) Mano; (b) Arma de fuego

Imágenes de banda visible.- Inspección de objetos industriales, identificación de billetes, conteo, reconocimiento automático de placas de vehículos.

(a) (b) (c)

Fig. 3.3 Imágenes de banda visible; (a) Detección del nivel del líquido; (b) Conteo de billetes; (c) Reconocimiento automático de Placas

Imágenes de banda de radio.- Aplicaciones en medicina y astronomía.

Fig. 3.4 Imagen de banda de radio; (a) Resonancia magnética de la rodilla

COMPONENTES DE UN SISTEMA PDI.

Entre los componentes principales para un sistema de procesamiento digital de imágenes tenemos los siguientes:

- Sensores.
- Digitalizadores.
- Hardware especializado en el PDI.
- Computadora.
- Software.
- Dispositivos de almacenamiento: memoria, discos.
- Monitores: despliegue y visualización.
- Hardcopy: impresión, diapositivas, fotografías.
- Acceso a la Red: transmisión por cables ópticos, UTP, wireless, etc.

HERRAMIENTAS PARA EL PDI

En la actualidad existe una gran cantidad de herramientas de software libre y comercial destinadas al procesamiento digital de imágenes. A continuación se presenta algunas de las más utilizadas en nuestro medio.

- **Adobe Photoshop.-** Es la herramienta líder en el tratamiento de imágenes digitales por su gran popularidad, facilidad y resultados obtenidos.
  http://www.adobe.com/es/products/photoshop/photoshop/

- **Matlab (**Image Processing Toolbox y Image Acquisition Toolbox).- Paquetes específicos de Matlab sobre adquisición y procesamiento de imágenes digitales.
  http://www.mathworks.com/access/helpdesk/help/toolbox/images/images.shtml

- **Mathematica.-** Paquete específico de Matemática sobre procesamiento de imágenes digitales.
  http://www.wolfram.com/products/applications/digitalimage/

- **Micromorph .-** Software de análisis de imágenes y morfología matemática para Windows.
  http://cmm.ensmp.fr/Micromorph/

FUNDAMENTOS DEL PROCESAMIENTO DE IMÁGENES DIGITALES

RELACIONES ENTRE PÍXELES

Un píxel *p* con coordenadas *(x,y)* tiene cuatro vecinos, dos horizontales y dos verticales, cuyas coordenadas son: *(x+1,y), (x-1,y), (x,y-1), (x,y+1).* A este conjunto de píxeles se

llama *vecindad 4* de *p* y se denota por $N_4(p)$, ver la figura 3.5. Nótese que para cada uno de estos píxeles hay una distancia de 1 (uno) desde p y que en los bordes de la imagen algunos de estos píxeles quedarán fuera de la imagen.

	(x-1, y)	
(x, y-1)	(x, y)	(x, y+1)
	(x+1, y)	

Fig. 3.5 Vecindad $N_4(p)$.

(x-1, y-1)		(x-1, y+1)
	(x, y)	
(x+1, y-1)		(x+1, y+1)

Fig. 3.6 Vecindad $N_D(p)$.

Existen también 4 vecinos diagonales de *p* con coordenadas: *(x+1,y+1), (x+1,y-1), (x-1,y-1), (x-1,y+1)* y se les denota por $N_D(p)$, ver la figura 3.6. Conjuntamente, $N_4(p)$ y $N_D(p)$ forman la vecindad 8 de p denotada por $N_8(p)$.

CONECTIVIDAD

La conectividad es un concepto importante utilizado para establecer los límites de objetos en regiones dentro de una imagen digital. Para determinar si dos píxeles están conectados se determina si son adyacentes en algún sentido, sea $N_D(p)$ o $N_4(p)$ por ejemplo, y si sus niveles de gris satisfacen algún criterio de similitud (si son iguales o parecidos). Por ejemplo, en una imagen binaria con valores de 1 y 0, dos píxeles pueden ser vecinos $N_4(p)$, pero se dice que están conectados sólo cuando tienen el mismo valor.

En la figura 3.7 se observa la conectividad de píxeles en una imagen binaria. El píxel 6 está conectado con el 2 y 8. El píxel 3 está conectado con el 5.

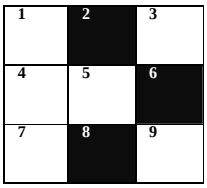


Fig. 3.7 Conectividad de píxeles

DISTANCIA

La distancia o transformada de distancia proporciona una medición de la separación existente entre dos puntos dentro de una imagen. Dados dos píxeles *p* y *q* con

coordenadas (*x*,*y*) y (*s*,*t*), respectivamente, se puede definir una función de distancia D si se cumple:
$D(p,q) \geq 0$
$D(p,q)=0$, *si* $p=q$
$D(p,q)=D(q,p)$

Las funciones de distancia comúnmente usadas son: distancia euclidiana y distancia tablero de ajedrez.

Distancia euclidiana entre p y q: $D_E(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$

En la figura 3.8 se muestra la distancia euclidiana para una imagen de 5 por 5.

$$\begin{array}{ccccc} \sqrt{8} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{8} \\ \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} \\ 2 & 1 & 0 & 1 & 2 \\ \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} \\ \sqrt{8} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{8} \end{array}$$

Fig. 3.8 Distancia euclidiana para una imagen de 5 por 5.

Distancia tablero de ajedrez: en donde se observa que los 4-vecinos están a una distancia unitaria del píxel central; si se desea que los 8-vecinos estén a la misma distancia se toma: $D(p, q)= Max(x - s, y - t)$

En la figura 3.9 se muestra la distancia tablero de ajedrez para una imagen de 5 por 5.

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}$$

Fig. 3.9 Distancia tablero de ajedrez para una imagen de 5 por 5.

RUIDO EN IMÁGENES

Todas las imágenes tienen cierta cantidad de ruido, la cual se puede deber a la cámara, escáner o al medio de transmisión de la señal. Generalmente el ruido se manifiesta como píxeles aislados que toman un nivel de gris diferente al de sus vecinos. Los algoritmos de filtrado permiten eliminar o disminuir este ruido.

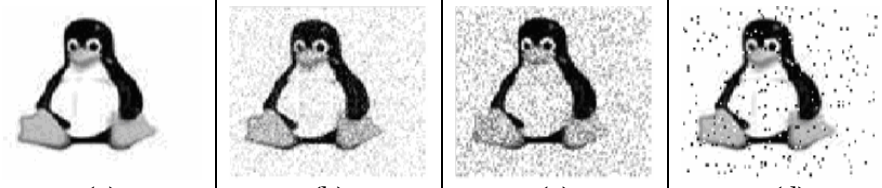
El ruido puede clasificarse en los siguientes tipos:

Gaussiano: Produce pequeñas variaciones en la imagen; generalmente se debe a diferentes ganancias en la cámara, ruido en los digitalizadores, perturbaciones en la transmisión. Se considera que el valor final del píxel sería el valor ideal más una cantidad correspondiente al error que puede describirse como una *variable aleatoria gaussiana*.

Impulsional (sal y pimienta): el valor que toma el píxel no tiene relación con el valor ideal, sino con el valor del ruido que toma valores muy altos o bajos (puntos blancos y/o negros) causados por una saturación del sensor o por un valor mínimo captado, si se ha perdido la señal en ese punto.

Multiplicativo: La imagen obtenida es el resultado de la multiplicación de dos señales.

En la figura 3.10 se muestran los diferentes ruidos afectando a una imagen.



(a) (b) (c) (d)

Fig. 3.10 Diferentes ruidos afectando a una imagen; (a) Original; (b) Gaussiano; (c) Multiplicativo; (d) Sal y Pimienta

Capítulo IV
Introducción a Matlab

Contenidos

INTRODUCCIÓN A MATLAB

El nombre de MATLAB proviene de la contracción de los términos **MATriz LAB**oratory. Es un entorno de computación y desarrollo de aplicaciones que integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo. En la actualidad goza de un alto nivel de implantación en centros de educación, así como en departamentos de investigación y desarrollo de muchas compañías industriales nacionales e internacionales.

MatLab fue originalmente desarrollado en lenguaje FORTRAN y al pasar de los años fue complementado y reimplementado en lenguaje C. Actualmente la licencia de Matlab es propiedad de MathWorks Inc. Está disponible para un amplio número de plataformas y opera bajo sistemas operativos como UNIX, Macintosh y Windows.

MATLAB dispone también de un amplio abanico de programas de apoyo especializados, denominados **Toolboxes**, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos: procesamiento de imágenes, procesamiento de señales, control robusto, estadística, análisis financiero, matemática simbólica, redes neuronales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, Simulink, etc.

Image Processing Toolbox

Este Toolbox proporciona a MATLAB un conjunto de funciones que amplían las capacidades del producto para realizar desarrollo de aplicaciones y de nuevos algoritmos en el campo del procesamiento y análisis de imágenes. Algunas de las funciones más importantes son:

- Análisis de imágenes y estadística.
- Diseño de filtros y recuperación de imágenes.
- Mejora de imágenes.
- Operaciones morfológicas.
- Definición de mapas de colores y modificación gráfica.
- Operaciones geométricas.
- Transformación de imágenes.

Además de los toolboxes, Matlab dispone de su propio *lenguaje de programación*.

EL ENTORNO DE TRABAJO DE MATLAB

Al arrancar MATLAB se abre la ventana inicial, que se muestra en la Figura 4.1.

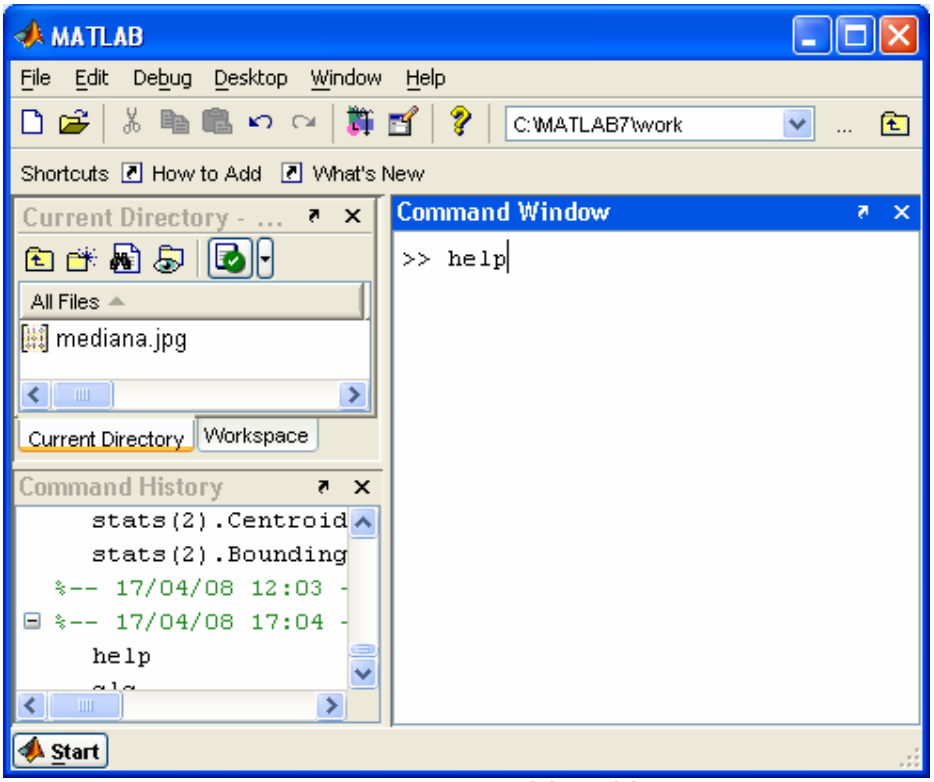


Fig. 4.1 Ventana inicial de Matlab

Aquí se puede apreciar las siguientes ventanas:

- ***Comand Window*** es la ventana principal donde se trabajará y se introducirán los comandos.
- ***Comand History*** que recoge todos los comandos introducidos anteriormente.
- ***Current Directory*** que muestra todos los ficheros de la carpeta actual.
- ***Workspace*** es el espacio de variables utilizadas.

En la ventana principal aparece el *prompt* característico de MATLAB (»), esto quiere decir que el programa está preparado para ingresar instrucciones, por ejemplo:

» clc
» help
» demo

MANEJO DE VARIABLES EN MATLAB

MATLAB puede almacenar información en variables y no se declaran con un tipo de dato específico como en otros lenguajes de programación. Las variables de Matlab deben comenzar por una letra y el resto de los caracteres pueden ser letras, dígitos o subrayados. Matlab distingue entre mayúsculas y minúsculas.

» a = 5
» A = 'visión artificial'

La función ***who*** muestra un listado de las variables que se encuentran en el espacio de trabajo.

» who
Your variables are:

A a

La función ***whos*** realiza un listado del tamaño y de la asignación de memoria de sus variables.

```
» whos
  Name      Size      Bytes  Class
  A         1x6          12  char array
  a         1x1           8  double array

Grand total is 7 elements using 20 bytes
```

El comando ***clear*** se puede utilizar para suprimir variables del espacio de trabajo.

```
» clear A
```

Si no se añade ninguna razón al comando **clear**, éste borrará todas sus variables.

```
» clear
```

TIPOS DE DATOS EN MATLAB

MATLAB trabaja con números enteros, reales, cadenas de caracteres y esencialmente con vectores y matrices numéricas rectangulares.

Vectores.- Se introducen entre corchetes y los elementos están separados por espacios o comas. Ejemplo:

```
» A = [ 1 2 3]
```

Matriz.- Las matrices se introducen por filas. Los elementos de una misma fila están separados por blancos o comas, mientras que las filas están separadas por ";" (punto y coma), ejemplo:

```
» A = [ 1 2 3; 4 5 6; 7 8 9 ]
```

MatLab emplea matrices porque con ellas se puede describir infinidad de cosas de una forma altamente flexible y matemáticamente eficiente. MATLAB permite trabajar con ***hipermatrices***, es decir, con matrices de más de dos dimensiones (ver Figura 4.2). Los elementos de una hipermatriz pueden ser números, caracteres, estructuras y vectores o matrices de celdas. El tercer subíndice representa la tercera dimensión, o sea, la *profundidad* de la hipermatriz.

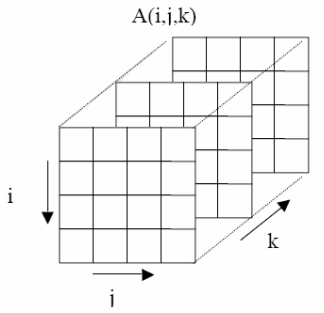


Fig. 4.2 Matriz de 3 dimensiones

LECTURA Y ESCRITURA INTERACTIVA DE VARIABLES

Se verá a continuación una forma sencilla de leer variables desde el teclado y escribir mensajes en la pantalla del PC.

FUNCIÓN INPUT

La función ***input*** permite imprimir un mensaje en la línea de comandos de MATLAB y recuperar como valor de retorno un valor numérico o el resultado de una expresión tecleada por el usuario, ejemplo:

```
» n = input('Ingrese el número de ecuaciones: \n')
» n =
      3
» n = input('Ingrese una expresión: \n')
  2*8-6
n =
   10
```

Otra posible forma de usar esta función es la siguiente (*obsérvese el parámetro* 's'):

```
» nombre = input('Ingrese su nombre: \n', 's')
nombre =
  Iván García
```

En este caso el texto tecleado *se devuelve sin evaluar* y se almacena en la variable *nombre*.

FUNCIÓN DISP

La función ***disp*** permite imprimir en pantalla un mensaje de texto o el valor de una matriz, pero sin imprimir su nombre, ejemplo:

```
» disp('Bienvenidos al curso.')
» a = 10
a =
   10
» disp(a)
   10
```

Obsérvese la diferencia entre las dos formas de imprimir la variable *a*.

MANEJO DE LAS IMÁGENES EN MATLAB

Matlab almacena la mayoría de las imágenes como arreglos bidimensionales (matrices) en los cuales cada elemento de la matriz corresponde a la intensidad de un píxel de la imagen. Algunas imágenes, como las imágenes a color (RGB), requieren de un arreglo tridimensional, donde en el primer plano en el espacio tridimensional representa la intensidad de rojo de los píxeles, el segundo plano representa la intensidad de verde de los píxeles y el tercer plano representa la intensidad de azul de los píxeles.

Para reducir el espacio en memoria requerido para almacenar imágenes, Matlab almacena los datos en arreglos de 8 o 16 bits sin signo, clases *uint8* y *uint16*, respectivamente.

Tipos de imágenes en Matlab

El toolbox de Procesamiento de Imágenes maneja cuatro tipos de imágenes básicos: imágenes indexadas, imágenes con intensidad (escala de grises), imágenes binarias e imágenes RGB, los cuales se discutieron en el capítulo II (La imagen digital).

Lectura de imágenes en Matlab

El comando ***imread*** lee una imagen desde un archivo gráfico. Si la imagen es en escala de grises, entonces devuelve una matriz bidimensional. Si la imagen es RGB, entonces devuelve un arreglo tridimensional. Su sintaxis es:

I = imread("filename")

En Matlab se soportan los siguientes formatos de imagen: JPEG, TIFF, GIF, BMP, PNG, HDF, PCX, XWD, ICO y CUR.

A continuación se muestra un ejemplo para leer y desplegar una Imagen, ver figura 4.3.

» I = imread('andy_josue.jpg');

Ahora para desplegarla en pantalla se puede usar el comando ***imshow***.

» imshow(I)

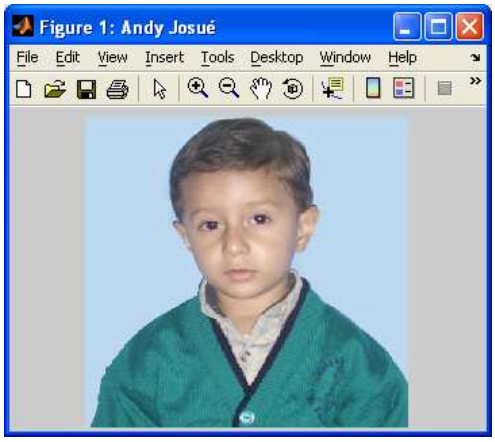


Fig. 4.3 Lectura y despliegue de una imagen

Para que se despliegue la imagen en una nueva figura, utilice el comando ***figure***:

» figure, imshow(I)

También puede usar el comando ***imview*** para desplegar una imagen en el visualizador de imágenes de Matlab, por ejemplo:

» imview(I)

El **;** *(punto y coma)* al final de una instrucción se utiliza opcionalmente. Si hace uso, el resultado no se despliega en la pantalla.

Para acceder a cada píxel de la imagen se puede usar el comando ***impixel***. Su sintaxis es: valor = impixel(imagen, columna, fila);
Ejemplo en una imagen RGB:

```
» impixel( I, 5, 12)
ans =
   252   253   248

» [col, fil, valor] = impixel( I, 5, 12)
col =
    5
fil =
   12
valor =
  252   253   248
```

Conversiones entre tipos de imágenes

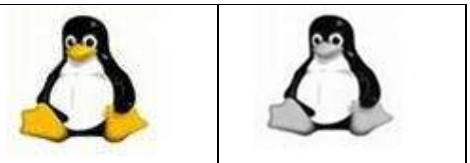
Para ciertas operaciones es necesario convertir una imagen de su tipo original a otro tipo de imagen que facilite su procesamiento. En la Tabla 4.1 se presentan algunos comandos usados para la conversión entre tipos de imágenes.

Comando	Descripción
gray2ind	Crea una imagen indexada a partir de una imagen de intensidad en escala de gris.
im2bw	Crea una imagen binaria a partir de una imagen de intensidad, imagen indexada o RGB basado en un umbral de luminancia.
ind2rgb	Crea una imagen RGB a partir de una imagen indexada
rgb2gray	Crea una imagen de intensidad en escala de gris a partir de una imagen RGB
rgb2ind	Crea una imagen indexada a partir de una imagen RGB

Tabla 4.1 Comandos de conversión de imágenes en Matlab

Ejemplo:

```
» I = imread('pinguino.jpg');
» k = rgb2gray (I);
» imshow(k)
```



(a) (b)

Fig. 4.4 Conversión de imagen; (a) imagen RGB; (b) imagen en Escala de grises

En la Tabla 4.2 se presentan algunos comandos de Matlab que pueden utilizarse para determinar el tipo de imagen con que se está trabajando.

Comando	Descripción
isbw	Regresa un valor verdadero (1) si la imagen es binaria
isgray	Regresa un valor verdadero (1) si la imagen es de intensidad

isind	Regresa un valor verdadero (1) si la imagen es indexada
isrgb	Regresa un valor verdadero (1) si la imagen es RGB
imfinfo	Regresa información sobre la imagen

Tabla 4.2 Comandos Informativos sobre imágenes

Ejemplo:

```
» bandera = isrgb(I)
ans =
    1
» info = imfinfo('pinguino.jpg')
info =
        Filename: 'pinguino.jpg'
        FileModDate: '07-Dec-2007 16:30:04'
        FileSize: 2156
        Format: 'jpg'
        FormatVersion: ''
        Width: 96
        Height: 86
        BitDepth: 24
        ColorType: 'truecolor'
        FormatSignature: ''
        NumberOfSamples: 3
        CodingMethod: 'Huffman'
        CodingProcess: 'Sequential'
        Comment: {}
```

Escritura de imágenes en Matlab

El comando ***imwrite*** permite escribir una imagen en el disco (archivo físico gráfico). Su sintaxis es: imwrite(I, filename)
donde *I* es la matriz que almacena la imagen y *filename* es el nombre de fichero donde vamos a guardar la imagen.

Ejemplo:

```
» I = imread('imagen.jpg');
» imwrite (I, 'copia_imagen.png' );
```

Selección de una sección de una imagen en Matlab

Para tomar una sección de cualquier imagen se utiliza el comando ***imcrop***.
Su sintaxis es: I2 = imcrop(I)

Ejemplo:

```
» I = imread('imagen.jpg');
» I2 = imcrop(I);
» imshow(I2)
```

Para seleccionar la región que se va a cortar, simplemente arrastre el ratón y forme un rectángulo sobre la región deseada. Cuando se suelta el botón del ratón, el comando regresa la sección seleccionada al argumento de salida especificado (*I2 en este caso*).

También es posible seleccionar la sección de interés de forma no interactiva. En este caso se debe especificar el rectángulo de la siguiente forma:

I2 = imcrop(I, [xmin ymin ancho alto])

donde *xmin* y *ymin* forman el punto de la esquina superior izquierda de la región a seleccionar.

Ejemplo:

```
» I = imread('imagen.jpg');
» I2 = imcrop(I, [25 6 35 30]);
» imshow(I2)
```

Fig. 4.5 Imagen recortada con *imcrop*

Para determinar el tamaño de una imagen podemos usar el comando ***size*** de matlab, que devuelve el número de filas, columnas y planos de colores.

```
» size(I)
ans =
   86    96     3
```

Si queremos almacenar el resultado en variables separadas, sería:

```
» [M, N] = size(I)
```

Si queremos obtener información más detallada de la imagen usamos el comando ***whos***:

```
» whos I
```

Name	Size	Bytes	Class
I	86x96x3	24768	uint8 array

Grand total is 24768 elements using 24768 bytes

Añadir ruido a una imagen en Matlab

Para añadir ruido a una imagen, hacemos uso del comando ***imnoise***. Su sintaxis es:

k = imnoise(I, tipo)

J = imnoise(I, tipo, parámetros)

El *tipo* es una cadena de caracteres que puede tener uno de los siguientes valores:

Valor	Descripción
'gaussian'	Ruido Gaussiano
'poisson'	Ruido de Poisson
'salt & pepper'	Sal y pimienta
'speckle'	Ruido Multiplicativo

Tabla 4.3 Tipos de ruidos en Matlab

El parámetro son valores que se puede asignar dependiendo del algoritmo usado.

Ejemplo:

```
» I = imread('pinguino.jpg');
```

```
» k = imnoise(I, 'salt & pepper');
» imshow(I)
» figure, imshow(k)
```

MANEJO DE VENTANAS EN MATLAB

MATLAB dispone de algunas funciones básicas para crear y manipular ventanas, entre las principales tenemos:

SUBPLOT
Divide la ventana gráfica en varias subventanas. Su sintaxis es:
subplot (m,n,p)
donde la ventana se divide en ***m*** filas y ***n*** columnas y hace que la subventana ***p*** sea la actual. Las ventanas se numeran de izquierda a derecha y de arriba hacia abajo.

En el siguiente ejemplo, se divide la ventana en una fila y dos columnas, en cada subventana se muestra una imagen (ver figura 4.6).

Ejemplo:

```
» I = imread('robot.jpg');
» subplot(1,2,1); imshow(I);
» title('1');
» I2 = rgb2gray (I);
» subplot(1,2,2); imshow(I2);
» title('2');
```

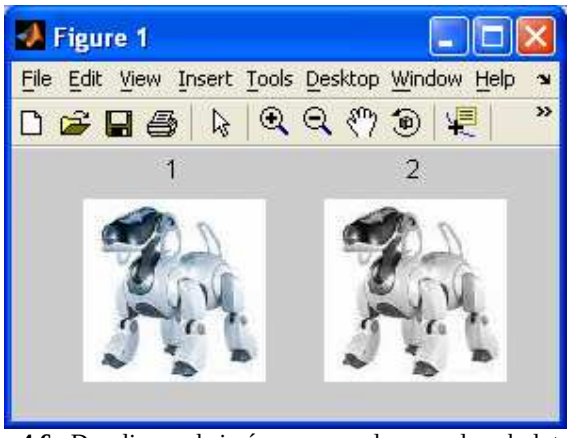


Fig. 4.6 Despliegue de imágenes con el comando subplot

PROGRAMACIÓN EN MATLAB

Matlab tiene su propio lenguaje de programación, que es parecido al ANSI C. Para estructurar mejor el código se emplea el uso y creación de archivos. Estos son archivos con la extensión ***.m*** que MATLAB utiliza para trabajar con funciones y scripts.

Un script es una secuencia de comandos que se pueden ejecutar a menudo y que se pueden guardar en un archivo de extensión ***.m*** para no tener que escribirlos de nuevo.

Las funciones son un bloque de código estructurado que se ejecutan cuando son invocadas y permiten añadir a MATLAB funciones adicionales, expandiendo así la capacidad de este programa.

Ambos ficheros, script y funciones, son archivos de texto ASCII con la extensión ***.m***, que se pueden crear con el editor incorporado de Matlab (en el menú *File/New/M-File*) o en cualquier editor de texto (wordpad, notepad, etc.).

Creación de funciones en MATLAB

Una función incluye en la primera línea del fichero una ***cabecera*** donde se especifica su ***nombre***, cuáles y cuántos ***argumentos*** de entrada tiene, y cuáles y cuántos ***valores*** devuelve. Para crear una función en Matlab se utiliza la siguiente estructura y sintaxis:

```
function [lista variables de retorno] = nombre-función (lista de argumentos)
  cuerpo de la función
end
```

El cuerpo de la función contiene las sentencias que sean necesarias para calcular los valores que la función va a devolver. Puede haber funciones sin valor de retorno y también sin argumentos. Recuérdese que los *argumentos* son los ***datos*** de la función y los *valores de retorno*, sus ***resultados***. Si no hay valores de retorno, se omiten los corchetes y el signo igual (=). Si sólo hay un valor de retorno, no hace falta poner corchetes. Tampoco hace falta poner paréntesis si no hay argumentos.

Las líneas que comienzan con "**%**" son interpretadas como comentarios.

A continuación se muestra un ejemplo de una función denominada ***info_img*** que tiene el argumento de entrada ***path_img*** (es la ruta de la imagen), despliega la imagen original y en escala de grises y devuelve el tamaño de la misma.

```
function [col, fil] = info_img(path_img)
    I = imread(path_img);
    imshow(I)
    k = rgb2gray (I);
    figure, imshow(k);
    [col, fil] = size(k);
    return;
end
```

Una vez desarrollada la función, se guarda en un fichero con el *mismo nombre de la función*, (en este caso *info_img.m*) y podremos utilizarla desde la línea de comandos de MATLAB o desde cualquier programa u otra función, ejemplo:

```
» [c, f] = info_img ('logo.jpg');
```

Sub-Funciones en Matlab

Las ***sub-funciones*** son funciones adicionales definidas en un mismo fichero ****.m***, con nombres diferentes del nombre del fichero (y del nombre de la función principal) y que

sólo pueden ser llamadas o invocadas por las funciones contenidas en ese fichero, resultando “invisibles” para otras funciones externas.

A continuación se muestra un ejemplo de un fichero llamado ***mi_funcion.m***:

```
function y = mi_funcion(a,b)
  y = subfuncion1(a,b);
  return;
end

function x = subfuncion1(y,z)
  x = y+z+2;
  return;
end
```

Probamos desde la ventana de comandos:

```
» val = mi_funcion(1,5)
val =
    8

» val = subfuncion1(1,5)
??? Undefined command/function ' subfuncion1'.
```

Capítulo V
Filtrado y Realzado de Imágenes

Contenido

OPERACIONES BÁSICAS ENTRE PÍXELES

Las operaciones que podemos realizar entre los píxeles son:
1. Operaciones aritméticas
2. Operaciones lógicas
3. Operaciones geométricas

1. OPERACIONES ARITMÉTICAS

Las operaciones aritméticas más usadas en el procesamiento de imágenes son: suma, resta, multiplicación y división. Para que se pueda llevar a cabo una operación aritmética, ambas imágenes deben ser del mismo tamaño *(ancho y alto)*. En la figura 5.1 se muestra la **suma de dos imágenes**, la cual se realiza de la forma $C(x, y) = A(x, y) + B(x, y)$ mediante el comando ***imadd*** en Matlab.
Ejemplo:

```
» I = imread('linux.jpg');
» J = imread('win.jpg');
» K = imadd(I,J);
» imview (K)
```

Fig. 5.1 Suma de Imágenes

El tamaño de las imágenes les podemos comprobar con el comando ***size***.

```
» size(I)
» size(J)
```

De igual manera podemos guardar la imagen resultante con el comando ***imwrite***.

```
» imwrite(K, 'suma.jpg');
```

También es posible **aumentar el brillo** a una imagen sumándole un valor constante a cada píxel. En la figura 5.2 se muestra el efecto de sumar el escalar 60 a una imagen, el cual se realiza de la forma $B(x, y) = A(x, y) + a$.
Ejemplo:

```
» I = imread('monedas.jpg');
» K = imadd(I, 60);
» imview (K)
```

Original Aumento del Brillo

Fig. 5.2 Aumento del brillo de una imagen

La **resta de imágenes** consiste en restar de una imagen el valor correspondiente de otra imagen. También requiere que ambas imágenes sean de igual tamaño.

En la figura 5.3 se muestra la resta de dos imágenes, la cual se realiza de la forma $C(x, y) = A(x, y) - B(x, y)$ mediante el comando ***imsubtract*** en Matlab.
Ejemplo:

» I = imread(‘linux.jpg’);
» J = imread('win.jpg');
» K = imsubtract(I, J);
» imview (K)

Fig. 5.3 Resta de Imágenes

También es posible **disminuir el brillo** a una imagen restándole un valor constante a cada píxel, la cual se lleva a cabo de la forma $B(x, y) = A(x, y) - a$.

Ejemplo:

» I = imread(‘monedas.jpg’);
» K = imsubtract(I, 60);
» imview (K)

Original Disminución del Brillo

Fig. 5.4 Disminución del brillo de una imagen

La **multiplicación de imágenes** se puede llevar a cabo entre dos imágenes del mismo tamaño, multiplicando elemento a elemento cada uno de los píxeles de la imagen, de la

forma $C(x, y) = A(x, y) \cdot B(x, y)$, mediante el comando ***immultiply*** en Matlab. En la figura 5.5 se muestra la multiplicación de dos imágenes.

Ejemplo:

```
» I = imread('linux.jpg');
» K = immultiply(I, I);
» imview (K)
```

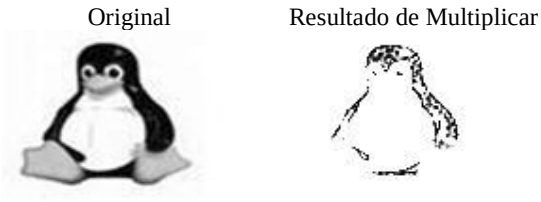


Fig. 5.5 Multiplicación de una imagen por si misma

Cuando se multiplica cada uno de los píxeles de una imagen por un escalar, se le conoce como *escalamiento*, el cual se realiza de la siguiente forma $B(x, y) = a \cdot A(x, y)$. Cuando el escalar o constante es menor a 1, se oscurece la imagen; y si es mayor, aumenta el brillo de la imagen.

La **división de imágenes** consiste en una división de elemento a elemento, como las demás operaciones vistas anteriormente. La división entre imágenes, se realiza de la forma $C(x, y) = A(x, y) \div B(x, y)$ mediante el comando ***imdivide*** en Matlab.

Ejemplo:

```
» I = imread('linux.jpg');
» J = imread('win.jpg');
» K = imdivide(I, J);
» imview(K)
```

2. OPERACIONES LÓGICAS

Las principales operaciones lógicas utilizadas en el procesamiento de imágenes son: AND, OR, NOT, las cuales se aplican sólo a imágenes binarias. En la figura 5.6 se muestran las operaciones lógicas aplicadas a dos imágenes binarias.

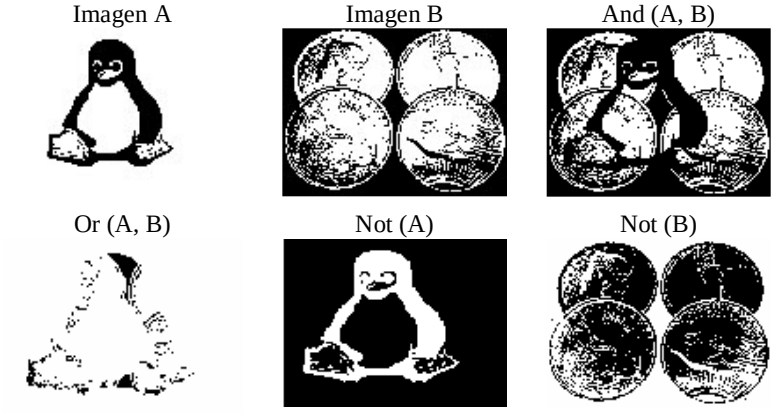


Fig. 5.6 Operación lógicas de imágenes binarias (and, or y not)

Ejemplo:

```
» I = imread('linux.bmp');
» J = imread(monedas.bmp');
» K = and(I, J);
» imview(K)
» K = or(I, J);
» imview(K)
» K = not(I);
» imview(K)
» K = not(J);
» imview(K)
```

3. TRANSFORMACIONES GEOMÉTRICAS

Las transformaciones geométricas modifican las relaciones espaciales entre píxeles. A continuación se presentan algunas de ellas.

INTERPOLACIÓN

La interpolación es el proceso en el cual se estiman los valores de una imagen en una sección específica, por ejemplo, cuando se amplía una imagen, en la nueva imagen existen más píxeles que en la imagen original.

Dentro de Matlab los comandos ***imresize*** e ***imrotate*** utilizan interpolación bidimensional como paso intermedio en sus procesos e implementa los siguientes métodos de interpolación: interpolación por el vecino más próximo, interpolación bilineal e interpolación bicúbica. Los tres métodos de interpolación funcionan de forma similar. Se asigna el valor del píxel interpolado calculando el promedio ponderado del conjunto de píxeles hallados en la vecindad de dicho punto.

Los tres métodos difieren en el conjunto de píxeles que consideran:

- **Vecino más próximo (nearest):** al píxel interpolado se le asigna el valor del píxel que corresponde. Es el método por defecto si no se especifica alguno.
- **Interpolación bilineal (bilinear):** el valor del píxel interpolado es el promedio ponderado de los píxeles en la vecindad 2x2 más cercana.
- **Interpolación bicúbica (bicubic):** el valor del píxel interpolado es el promedio ponderado de los píxeles presentes en la vecindad 4x4 más cercana.

Nótese que el número de píxeles considerado aumenta la complejidad del cálculo, es por eso que la interpolación bilineal es más lenta que el método del vecino más próximo; y el método bicúbico es más lento que el método bilineal. Por otro lado, si se considera un mayor número de píxeles en el cálculo, se tendrán mejores resultados en la imagen resultante.

Para imágenes RGB, la interpolación se ejecuta en los planos de color rojo, verde y azul de forma individual.

Amplificación y Reducción de imágenes

Para el cambio de tamaño de una imagen (sea amplificación o reducción) se utiliza el comando ***imresize***. Este comando permite especificar: el tamaño de la imagen de salida (procesada), el método de interpolación utilizado y el filtro a usar para evitar el *efecto alias*. El efecto alias se presenta al reducir el tamaño de una imagen debido a que se presenta una pérdida de información.

En las figura 5.7 se presenta un ejemplo de amplificación y reducción de imágenes usando los métodos de interpolación descritos anteriormente.

Fig. 5.7 Ampliación y reducción de imágenes

Ejemplo:

```
» I = imread('pinguino.jpg');
» K = imresize(I, 2, 'bilinear');
» imview(K)
» K = imresize(I, 0.5, 'bicubic');
» imview(K)
```

Rotación de Imágenes

Para rotar una imagen en un determinado número de grados, se utiliza el comando ***imrotate***.

Fig. 5.8 Rotación de una imagen 35 y 90 grados.

Ejemplo:

```
» I = imread('pinguino.jpg');
» K = imrotate(I, 35, 'bilinear');
» imview(K)
» K = imrotate (I, 90, 'bicubic');
» imview(K)
```

Correlación de Matrices

La correlación es una operación en la cual el valor de un píxel de salida se calcula como la suma ponderada de los píxeles vecinos. La correlación se utiliza para encontrar el parecido entre píxeles de una imagen. Si los píxeles son iguales o parecidos, se dice que están altamente correlacionados entre sí. La correlación permite hallar patrones, y se utiliza el comando ***corr2*** en Matlab, el cual calcula el coeficiente de correlación entre dos matrices del mismo tamaño.

Fig. 5.9 Dos imágenes del mismo tamaño para determinar su correlación.

Ejemplo:

```
» I=imread('R.bmp')
» J=imread('Q.bmp')
» k=corr2(I, J);
» k
    k = 0.4069              %  medianamente correlacionados
```

El coeficiente entre dos matrices es un número real comprendido entre el rango [-1 y 1] y se dice que las matrices están altamente correlacionadas si el coeficiente tiende a estos límites (-1 o 1); y una baja correlación, si tienden a cero.

Si hacemos la correlación de la misma imagen nos da el valor de uno.

```
» I=imread('R.bmp')
» k=corr2(I, I);
» k
    k = 1                   %  altamente correlacionados
```

OPERACIONES SOBRE EL HISTOGRAMA DE UNA IMAGEN

HISTOGRAMA

Un histograma ilustra en un gráfico cómo están distribuidos los píxeles de la imagen mostrando la cantidad de píxeles en cada nivel de intensidad del color. El histograma

indica si la imagen contiene suficientes detalles en las sombras (en la parte izquierda del histograma), en los medios tonos (en el centro) y las iluminaciones (en la parte derecha) para realizar una corrección correcta.

Es un diagrama de barras, en el que sobre el eje de las abscisas se representan los diferentes valores que pueden tomar los píxeles de una imagen, y en el eje de las ordenadas, el número de píxeles que se encuentran en una imagen para ese valor de cuantización.

El histograma de una imagen en niveles de gris proporciona información sobre el número de píxeles que hay para cada nivel de intensidad (*ver Figura 5.10*). En imágenes en color RGB se usan tres histogramas, uno por cada componente de color.

El histograma de una imagen se calcula en Matlab con el comando ***imhist*** y *sólo se aplica* a imágenes indexadas o con intensidad (escala de grises).

Ejemplo:

```
» I=imread('monedas.png')      % lee una imagen indexada o con intensidad
» imhist(I)                    % muestra el histograma
```

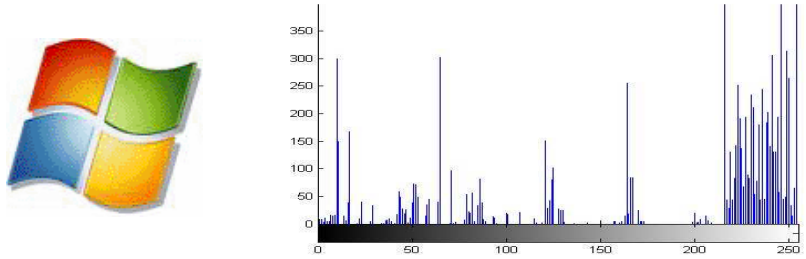
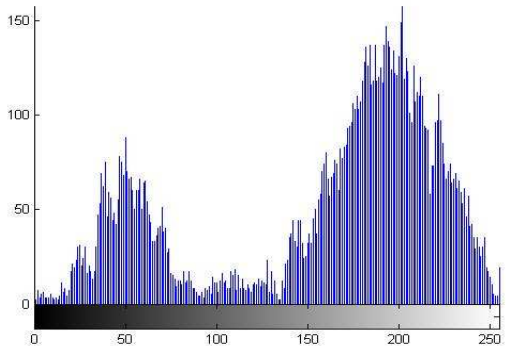


Fig. 5.10 Imagen con su respectivo histograma

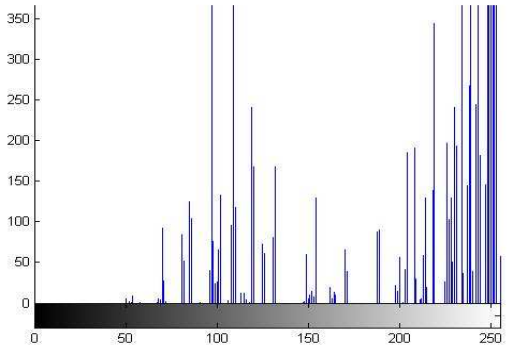
Aumento y reducción de contraste

En la figura 5.11 se muestran algunas imágenes con su respectivo histograma. Observe como al variar el contraste en la imagen, los píxeles se mueven a la izquierda, centro o derecha de su histograma.

(a) Imagen original



(b) Imagen modificada con mas brillo



(c) Imagen modificada con menos brillo

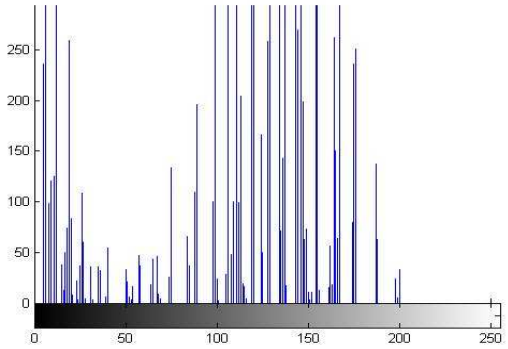


Fig. 5.11 Imágenes con distintos niveles de brillo con sus histogramas

Ecualizado del histograma

El proceso de ecualizado mejora el contraste de una imagen y tiene por objetivo obtener un nuevo histograma, a partir del histograma original, con una distribución uniforme de los diferentes niveles de intensidad. Además, mejora la calidad visual de las imágenes saturadas. El ecualizado está implementado en el comando ***histeq*** en matlab.
Ejemplo:

```
» I=imread('pinguino.jpg')      % lee una imagen indexada o con intensidad
» J=histeq(I);                  % ecualiza el histograma
» imshow(I)
» figure, imshow(J)
» figure, imhist(I)
» figure, imhist(J)
```

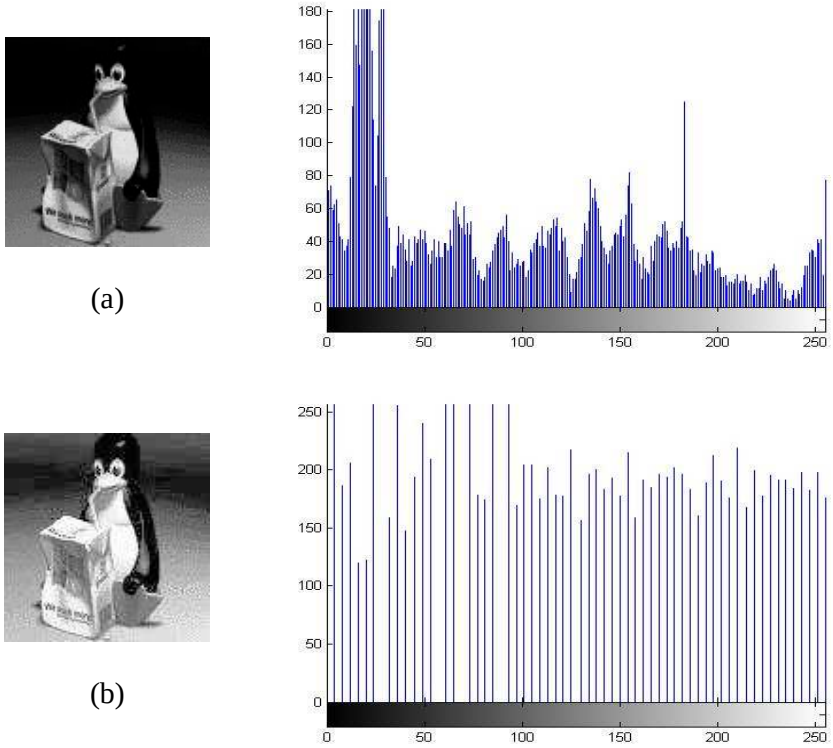


Fig. 5.12 Ecualizado del histograma de una imagen; (a) imagen original con su respectivo histograma; (b) ecualizado del histograma

Ajuste de la Intensidad

La función ***imadjust*** puede ampliar, reducir, y en general, cambiar los rangos de intensidad de la imagen de entrada a unos nuevos rangos en la imagen de salida. Se usa con imágenes en niveles de grises y de color. Aumenta el contraste de la imagen de salida. Su sintaxis es: J = imadjust(I);

Ejemplo:

```
» I=imread('flores.jpg')          % lee una imagen indexada o con intensidad
» J= imadjust (I);                % modifica el rango de intensidad (modifica el contraste)
» imshow(I)
» figure, imshow(J)
```

Fig. 5.13 Ajuste de la intensidad de una imagen; (a) imagen original; (b) imagen con ajuste de intensidad

OPERACIONES EN EL DOMINIO DE LA FRECUENCIA

Se ha visto que una imagen digital es una representación que se refiere directamente a la intensidad luminosa de puntos del espacio, por lo que se dice que una imagen digital es una representación en el dominio del espacio (o del tiempo). Existen otros tipos de representaciones, que contienen la misma información, pero que no están en el dominio del espacio. Es el caso de las representaciones en el dominio de la frecuencia.

Las representaciones en el dominio de la frecuencia, explican cómo se repiten ciertos patrones de una imagen y, con ello, consiguen representar la información de tal imagen. Este tipo de transformaciones de frecuencia se llevan a cabo para una amplia gama de procesamientos, entre los cuales se encuentran: la convolución, el mejoramiento de imágenes, la detección de características, compresión; además se pueden apreciar y alterar directamente elementos como el ruido, los bordes, las texturas, etc.

Transformada de Fourier

La transformada de Fourier se encuentra implementada en Matlab en el comando ***fft*** (para el caso unidimensional), ***fft2*** (para el caso bidimensional) y ***fftn*** (para el caso n-dimensional). Las transformadas inversas se encuentran en los comandos ***ifft*** (para el caso unidimensional), ***ifft2*** para el caso bidimensional e ***ifftn*** (para el caso n-dimensional).

La explicación de la transformada de Fourier queda fuera del alcance de este texto y deberá ser consultado en otras fuentes de información, se recomienda revisar [Vel04] y [GW96].

FILTRADO ESPACIAL DE UNA IMAGEN

El filtrado es una técnica para modificar o mejorar a una imagen. Por ejemplo, un filtro puede resaltar o atenuar algunas características. El filtrado es una operación de vecindario, en la cual el valor de un píxel dado en la imagen procesada se calcula mediante algún algoritmo que toma en cuenta los valores de los píxeles de la vecindad de la imagen original.

Dentro de los filtros espaciales tenemos los siguientes tipos: Filtros de paso bajo y filtros de paso alto.

Filtros espaciales de paso bajo (Suavizantes)

Los filtros suavizantes se emplean para hacer que la imagen aparezca algo borrosa y también para reducir el ruido.

El filtrado paso bajo espacial se basa en el promediado de los píxeles adyacentes al píxel que se evalúa (vecindad 3x3, 5x5, etc.). **El filtrado de promediado** de imágenes en Matlab es una operación lineal y está implementado en el comando ***imfilter***. La figura 5.14 (c) muestra la imagen con el filtro de promedio.

Ejemplo:

```
» I=imread('monedas.tif');
```

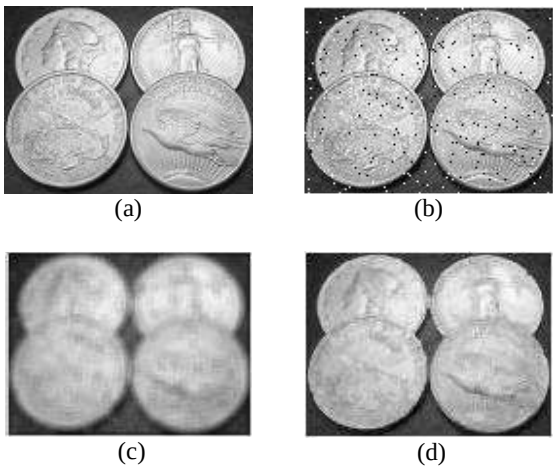
```
» J = imnoise(I,'salt & pepper',0.02);      % añade ruido a la imagen
» h=fspecial('average',5);                  % crea un tipo especial de filtro
» K=imfilter(J,h);                          % aplica el filtro de promedio
» imshow(J)
» figure, imshow(K)
```

Otro filtro de paso bajo es el **filtro de la mediana**. Éste se basa en sustituir el valor de un píxel por el de la mediana del conjunto formado por el mismo y sus ocho vecinos. Es una operación no lineal que se suele utilizar en el procesamiento de imágenes para reducir el ruido "sal y pimienta". La mediana de filtrado es más efectiva cuando el objetivo es reducir el ruido y al mismo tiempo preservar los bordes.

En Matlab este filtro se encuentra implementado en el comando ***medfilt2***. La figura 5.14 (d) muestra la imagen con el filtro de mediana.

Ejemplo:

```
» I = imread('monedas.jpg');
» J = imnoise(I,'salt & pepper',0.02);      % añade ruido a la imagen
» K = medfilt2(J);                          % aplica el filtro de la mediana
» imshow(J)
» figure, imshow(K)
```



(a) (b) (c) (d)

Fig. 5.14 Filtrado espacial de Imágenes; (a) Imagen original; (b) Imagen con ruido; (c) Imagen con filtro de Promedio; (d) Imagen con filtro de Mediana

Filtros espaciales de paso alto

Los filtros espaciales de paso alto se corresponden con las altas frecuencias, suelen corresponder a los bordes de los objetos presentes en las imágenes. Se puede utilizar para el realce de bordes o para la detección de contornos.

El realce de bordes consiste en resaltar aquellos píxeles que tienen un valor de gris diferente al de sus vecinos. Si la imagen contiene ruido, su efecto se multiplicará, por lo que primero se debe eliminar el ruido.

El realce de bordes en Matlab se consigue con el uso de algunos comandos y en la figura 5.15 se muestra un ejemplo.

```
» im=imread('flores.jpg');
» im=double(im)/255;                    % Convierte a double
» figure; imshow(im);
» h=firpm(16, [0 .1 .3 1], [0 0 1 1]);  % Cálculo de un filtro equiripple paso alto
» h=ftrans2(h);                         % Convierte en filtro 2D
» imf=filter2(h,im);                    % Filtrar la señal
» figure; imshow(im+imf);               % Muestra la imagen con bordes resaltados
```

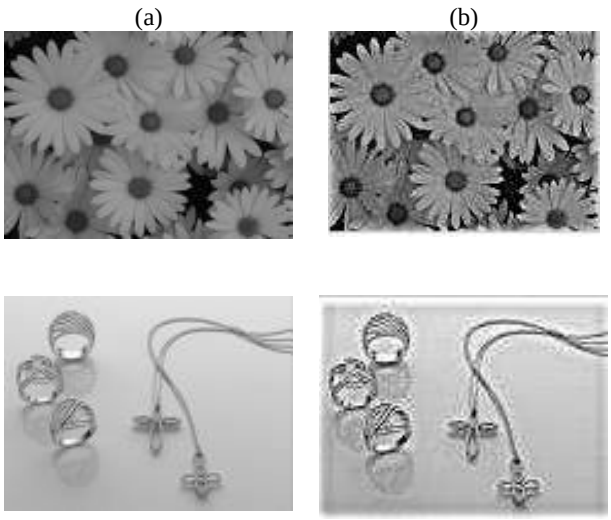


Fig. 5.15 Realce de bordes en Imágenes; (a) Imágenes originales; (b) Imagen con bordes realzados

La detección de contornos es un paso intermedio en el reconocimiento de patrones en imágenes digitales. En una imagen, los contornos corresponden a los límites de los objetos presentes en la imagen. Para hallar los contornos se buscan los lugares en la imagen en los que la intensidad del píxel cambia rápidamente.

La detección de contornos se encuentra implementada en Matlab en el comando ***edge*** y se aplica a imágenes de intensidad**.**
Su sintaxis es: *BW =edge(I, método)*

Esta función devuelve una imagen de bordes binaria, y se puede obtener con diversos métodos, como: sobel, prewitt, robert, canny.

En la figura 5.16 se muestra un ejemplo de detección de contornos.

Ejemplo:

```
» I = imread('monedas.jpg');
» imshow(I)
» BW1 = edge(I, 'sobel');
» BW2 = edge(I, 'canny');
» figure, imshow(BW1)
» figure, imshow(BW2)
```

(a) (b) (c)

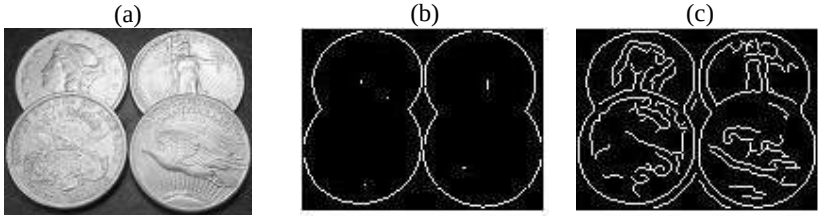


Fig. 5.16 Detección de contornos en Imágenes; (a) Imagen original; (b) Detección con filtro de Sobel; (c) Detección con filtro de Canny

Nunca un filtrado digital aumenta la información de una imagen. Sin embargo estos filtros pueden resultar útiles para destacar elementos de la imagen que se necesitan en la etapa de reconocimiento.

Capítulo VI
Operaciones Morfológicas

Contenido

DEFINICIONES BÁSICAS

La Morfología matemática es una técnica de procesado no lineal de la imagen, interesada en la geometría de los objetos. Las operaciones morfológicas proporcionan información sobre la forma o estructura de una imagen.

El Análisis morfológico permite extraer componentes de la imagen que son útiles en la representación y descripción de la forma de las regiones como: fronteras, esqueletos y permite obtener características relevantes de los objetos en la imagen como: Forma, Tamaño.

El Procesado morfológico permite transformar la forma o la estructura de los objetos en una imagen.

Existe tres tipos de morfología: Morfología binaria (es la más frecuente), Morfología de niveles de gris y Morfología de imágenes policromáticas.

En este apartado sólo se tratará detalladamente la morfología sobre imágenes binarias y en escala de grises.

Algunos ejemplos de **Aplicación de las operaciones morfológicas** se muestran en la figura 6.1, donde: (a) Eliminación del ruido, (b) Contar el número de líneas, (c) Separar llaves y monedas, (d) Contar el número de dientes de la rueda.

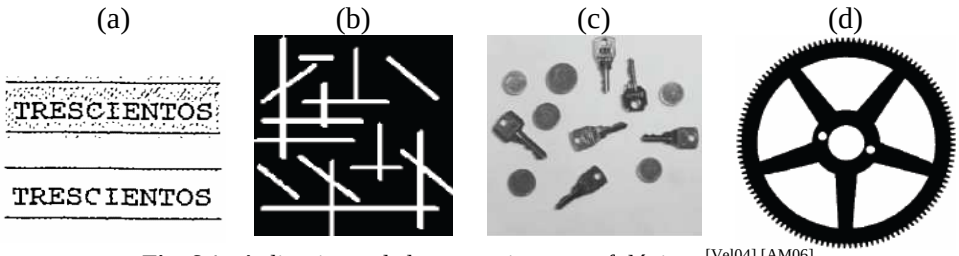


Fig. 6.1 Aplicaciones de las operaciones morfológicas [Vel04] [AM06]

Elementos del procesado morfológico

Los fundamentos del análisis y procesado morfológico se basan en el álgebra de conjuntos y en la topología. Existen tres elementos en el proceso:

a) Conjuntos (Imágenes)
b) Elementos Estructurantes
c) Operadores Morfológicos: dilatación, erosión, apertura/cierre

a) CONJUNTOS

En una **imagen binaria**, los conjuntos existentes son puntos de un espacio 2D, cada elemento es un punto de coordenadas (x,y) en el plano bidimensional de la imagen. Se definen dos conjuntos (o planos):

Primer plano: $A = \{(x,y) \mid f(x,y) = 1\}$
Fondo: $B = \{(x,y) \mid f(x,y) = 0\}$

En una **imagen de niveles de gris** puede ser representada como conjuntos cuyos componentes se encuentran en un espacio 3D. En este caso, dos componentes de cada elemento de un conjunto se refieren a las coordenadas del píxel, y el tercer componente está relacionado con la intensidad.

b) ELEMENTOS ESTRUCTURANTES

Examinar la estructura geométrica de una imagen usando como sonda un patrón de ajuste que se denomina elemento estructurante (SE.). El SE puede tener cualquier tamaño y forma (horizontal, vertical, cuadrado, circular, etc.).

En Matlab se encuentra implementado en el comando ***strel,*** el cual crea un elemento de estructura morfológica. Su sintaxis se muestra a continuación:

```
SE = strel(forma, parámetros)          % crea un elemento estructurante de una forma determinada.
```

Ejemplo:

```
» se1 = strel('square',11)      % Cuadrado de 11-por-11
» se2 = strel('line',10,45)     % Línea de longitud  10 y ángulo de 45 grados
» se3 = strel('disk',15)        % Disco de radio 15
» se4 = strel('ball',15,5)      % bola de radio 15 y alto 5
```

c) OPERADORES MORFOLÓGICOS

Ditalación
La dilatación expande los píxeles de la imagen sobre la que se aplica.

Erosión
La erosión adelgaza la imagen sobre la que se aplica siendo, en un sentido no estricto, opuesta a la dilatación.

Apertura (Opening)
Erosión seguida de una dilatación. Elimina pequeños píxeles aislados que haya en la imagen.

Cierre (Closing)
Dilatación seguida de una erosión. Rellena los pequeños agujeros que existan en la imagen.

La Apertura/cierre eliminan picos positivos/negativos más estrechos que el elemento estructurante.

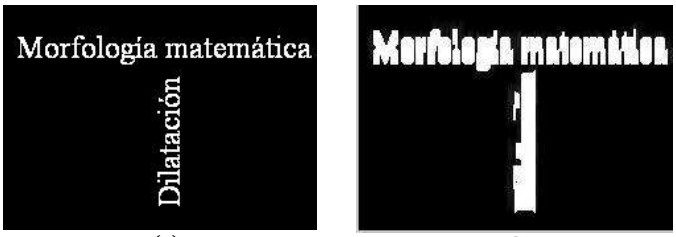
En Matlab se encuentran implementados en los siguientes comandos: *imdilate, imerode, imclose, imopen* respectivamente para crear un elemento estructurante, dilatar, erosionar, cierre y apertura.

La sintaxis de cada uno se muestra a continuación:

```
J = imdilate(I, SE)      % la imagen I puede ser binaria o en gris
J = imerode(I, SE)       % la imagen I puede ser binaria o en gris
J = imclose(I, SE)       % SE no puede ser un array de elementos estructurantes
J = imopen(I, SE)        % SE no puede ser un array de elementos estructurantes
```

En el siguiente ejemplo se **dilata** una imagen binaria con un elemento estructurante de línea vertical, ver figura 6.2.

```
» bw = imread('texto.jpg');
» se = strel('line',11,90);
» bw2 = imdilate(bw,se);
» imshow(bw), title('Original')
» figure, imshow(bw2), title('Dilatada')
```



(a) (b)

Fig. 6.2 Dilatación de una imagen binaria; (a) Imagen Original; (b) Imagen Dilatada

En el siguiente ejemplo se **erosiona** una imagen binaria con un elemento estructurante de línea horizontal, ver figura 6.3.

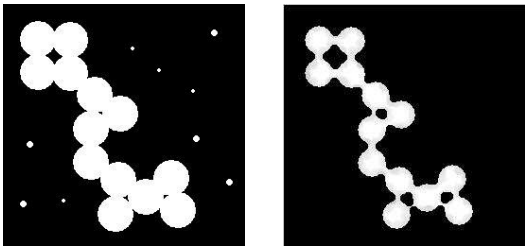
```
» bw = imread('texto.jpg');
» se = strel('line',4,0);
» bw2 = imerode(bw,se);
» imshow(bw), title('Original')
» figure, imshow(bw2), title('Erosionada')
```

(a) (b)

Fig. 6.3 Erosión de una imagen binaria; (a) Imagen Original; (b) Imagen Erosionada

En el siguiente ejemplo se **abre** una imagen binaria con un elemento estructurante de disco de radio 5, ver figura 6.4.

```
» bw = imread('circulos.jpg');
» se = strel('disk',5);
» bw2 = imopen(bw,se);
» imshow(bw), title('Original')
» figure, imshow(bw2), title('Opening')
```



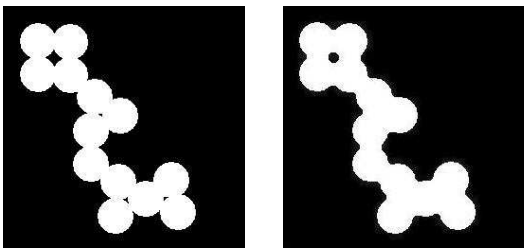
(a) (b)

Fig. 6.4 Apertura de una imagen binaria [Mat04]; (a) Imagen Original; (b) Imagen Aperturada

En este ejemplo vemos que también *elimina el ruido* de los objetos de una imagen que tienen un tamaño menor que el elemento estructurante determinado.

En el siguiente ejemplo se **cierra** una imagen binaria con un elemento estructurante de disco de radio 5, ver figura 6.5.

```
» bw = imread('circulos1.jpg');
» se = strel('disk',5);
» bw2 = imclose(bw,se);
» imshow(bw), title('Original')
» figure, imshow(bw2), title('Closing')
```



(a) (b)

Fig. 6.5 Cierre de una imagen binaria; (a) Imagen Original; (b) Imagen Cerrada

FILTROS MORFOLÓGICOS (Top-Hat):

Los filtros morfológicos se realizan sobre una imagen en escala de grises o binaria usando un elemento estructurante SE y resaltan objetos de color contrario al fondo. Tenemos dos filtros:

1. *Positivo (white top-hat):* El resultado de esta operación es útil para resaltar detalles en la presencia de sombras (pequeños detalles brillantes).

2. *Negativo (black top-hat o bottom-hat):* Su aplicación es fundamentalmente para que resalte detalles oscuros sobre un fondo local blanco.

En Matlab estos filtros están implementados en los siguientes comandos:
J= imtophat(I, SE);
J= imbothat(I, SE);

En el siguiente ejemplo se aplica el filtro bottom-hat con el comando ***imbothat*** a una imagen en escala de grises con un elemento estructurante de disco de radio 15, ver figura 6.6.

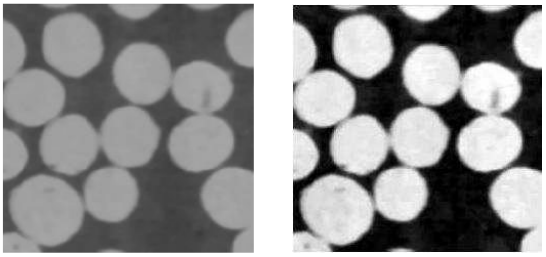
```
» bw = imread('libreta.jpg');
» se = strel('disk',15);
» bw2 = imbothat(bw,se);
» imshow(bw), title('Original')
» figure, imshow(bw2), title('bottom-hat')
```

(a) (b)

Fig. 6.6 Filtro morfológico de una imagen en escala de grises; (a) Imagen Original; (b) Imagen bottom-Hat

En el siguiente ejemplo se aplica el filtro top-hat con el comando ***imtophat*** a una imagen en escala de grises con un elemento estructurante de disco de radio 50, ver figura 6.7.

```
» bw = imread('circulos.jpg');
» se = strel('disk',50);
» bw2 = imtophat(bw,se);
» bw2 = imadjust(bw2);          % mejorar la visibilidad de los resultados
» imshow(bw), title('Original')
» figure, imshow(bw2), title('top-hat')
```



(a) (b)

Fig. 6.7 Filtro morfológico de una imagen en escala de grises; (a) Imagen Original; (b) Imagen Top-Hat

Sin embargo, debe recordarse que realizando estas u otras manipulaciones a una imagen, nunca se gana información, sólo se promociona o se descarta información ya existente en la imagen. Por ello siempre que la calidad de una imagen sea pobre,

resultando insuficiente para el uso al que se destina, debe pensarse en la posibilidad de variar las condiciones de captura de la imagen original.

Otros comandos de matlab muy importantes en el procesamiento de imágenes que utilizan la morfología matemática son los comandos ***bwmorph*** y ***imfill***.

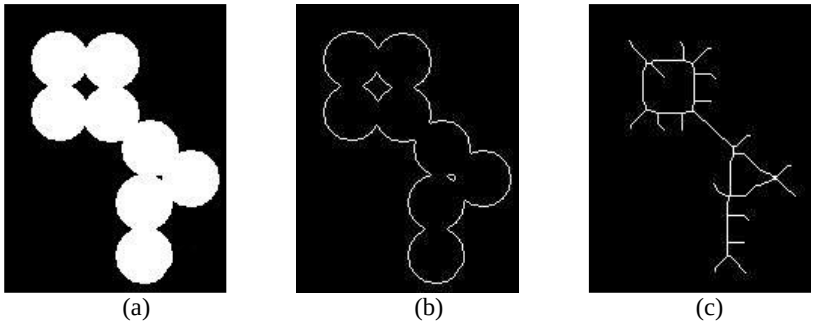
La función ***bwmorph*** realiza operaciones morfológicas en imágenes binarias. Su sintaxis es: BW2 = bwmorph(BW, operación)
La *operación* aplica una determinada operación morfológica sobre la imagen binaria BW. Es una cadena de caracteres que puede tomar algún valor de los permitidos. En la tabla 6.1 se listan algunos:

Operación	Descripción
'bothat'	Realiza la operación morfológica "bottom hat".
'clean'	Elimina píxeles aislados (1´s individuales que están rodeados por 0´s)
'close'	Realiza la operación morfológica closing.
'dilate'	Realiza la dilatación usando un elemento estructurante ones(3).
'erode'	Realiza la erosión usando un elemento estructurante ones(3).
'fill'	Llena píxeles aislados (0´s individuales que están rodeados por 1´s)
'open'	Realiza la operación morfológica opening
'remove'	Remueve píxeles interiores
'skel'	Con n = Inf, elimina los píxeles en los límites de los objetos
'tophat'	Realiza la operación morfológica "top hat"

Tabla 6.1 Operaciones disponibles en el comando bwmorph

Ejemplo:

```
» I=imread('Circulos.jpg');
» BW = im2bw(I);
» BW2 = bwmorph(BW, 'remove');
» BW3 = bwmorph(BW, 'skel', Inf);
» subplot(2,2,1), imshow(I)
» subplot(2,2,2), imshow(BW2);
» subplot(2,2,3), imshow(BW3);
```



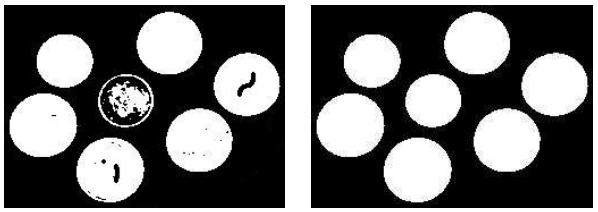
(a) (b) (c)

Fig. 6.8 Operaciones de bwmorph sobre una imagen binaria [Mat04]; (a) Imagen Original; (b) Operación remove; (c) operación skel

La función ***imfill*** nos permite rellenar regiones (los agujeros) de una imagen binaria o en escala de grises. Su sintaxis es: *bw = imfill (Imagen_binaria, 'holes')*

Ejemplo:

```
» I=imread('circulos.jpg');
» BW = im2bw(I);
» BW = imfill(BW, 'holes');
» subplot(1,2,1), imshow(I)
» subplot(1,2,2), imshow(BW);
```



(a) (b)

Fig. 6.7 Relleno de regiones en una imagen binaria; (a) Imagen Original; (b) Imagen Rellenada

Capítulo VII
Segmentación

Contenido

INTRODUCCIÓN A LA SEGMENTACIÓN

La segmentación es un proceso que consiste en dividir una imagen digital en regiones homogéneas o similares con respecto a una o más características (como por ejemplo el brillo, el color, tamaño, longitud, forma) con el fin de facilitar su posterior análisis y reconocimiento automático. Es una de las áreas más importantes y complejas de la visión artificial, la cual ha sido estudiada extensamente y continúa siendo tema de discusión.

Algunos ejemplos de problemas de segmentación son: localizar los ojos de una persona dentro de la imagen de una fotografía, separar los caracteres dentro de una imagen de un texto, localizar los vehículos en una calle dentro de una imagen, detectar ciertos tipos de células en imágenes médicas, o en general, cuando se trata de separar ciertos objetos de un fondo en una imagen cualquiera.

El proceso de segmentación de una imagen depende del problema que se desee resolver y determina el eventual éxito o fracaso del análisis y reconocimiento. En la mayor parte de los casos, una buena segmentación dará lugar a una solución correcta, por lo que se debe poner todo el esfuerzo posible en esta etapa.

Los diferentes objetos (regiones de píxeles) que aparecen en una imagen pueden localizarse atendiendo ciertas propiedades o características.

Los algoritmos de segmentación de imágenes generalmente se basan en dos propiedades básicas de los niveles de gris de la imagen: **Discontinuidad y Similitud.** *Dentro de la primera categoría* se intenta dividir la imagen basándonos en los cambios bruscos en el nivel de gris. Las áreas de interés en esta categoría son la detección de puntos, líneas y bordes en la imagen. *Las áreas dentro de la segunda categoría* están basadas en las técnicas de umbrales, crecimiento de regiones, y técnicas de división y fusión. En este apartado trataremos solamente las técnicas basadas en umbrales y regiones.

Sin embargo, cada uno de estos enfoques por separado resulta insuficiente para segmentar y describir los objetos de la mayoría de los problemas reales. Por ello, suelen usarse combinaciones de varios de los métodos propuestos y también realizarse modificaciones para ajustar estos métodos al problema particular que se trate.

No existe un método universal de segmentación. Está íntimamente ligada con la tarea que se va a resolver y termina cuando satisface los objetivos del observador, es decir, cuando se hayan detectado todos los objetos de interés para la aplicación.

1. Segmentación basada en Umbralizado

La umbralización es un proceso que permite convertir una imagen de niveles de gris o en color en una imagen binaria, de tal forma que los objetos de interés se etiqueten con un valor distinto de los píxeles del fondo.

La umbralización es una técnica de segmentación rápida, que tiene un coste computacional bajo y que incluso puede ser realizada en tiempo real durante la captura de la imagen usando un computador personal.

Si bien hay diferentes métodos para hallar un umbral, la mayoría de ellos no dan buenos resultados cuando se trabaja con imágenes del mundo real, debido a la presencia de ruido, histogramas planos o una iluminación inadecuada. Por el contrario, el **método de Otsu** fue uno de los mejores métodos de selección de umbral para imágenes del mundo real, sin embargo, necesita mucho más tiempo para seleccionar el umbral óptimo. La importancia del método de Otsu radica en que es automático, es decir, no necesita supervisión humana ni información previa de la imagen antes de su procesamiento.

Matlab posee la función ***graythresh***, que cálcula el umbral de la imagen global utilizando el método de Otsu. Su sintaxis es:

T = graythresh(I);

Donde *I* es la imagen original y *T* es el umbral devuelto y está en el rango [0 1].

Ejemplo:

```
function umbral()
  I = imread('placa.jpg');
  T = graythresh(I);                % Calcula el umbral entre [0 1] con método Otsu
  bw = im2bw(I, T);                 % binariza la imagen con el umbral calculado
  imshow(I), title('Original')
  figure, imshow(bw), title('Segmentación por Umbralizacion')
  return;
end
```

Otra manera de binarizar la imagen, es multiplicar el umbral obtenido de la función *graythresh* por 255 y aplicar el operador relacional <= ó >= a la imagen como se indica:

```
» T = T * 255;
» bw = (I <= T);          % binariza la imagen
```

En este caso se convierte a negro todo lo que sea mayor que el umbral, T, y a blanco todo lo que sea menor. Si se desea lo contrario hay que poner >= en lugar de <=.

En la figura 7.1 se muestra la segmentación basada en umbralización de la placa de un automóvil, utilizando un umbral de 128 para el cálculo.

(a) (b)

Fig. 7.1 Segmentación por umbralización de la placa de un automóvil (a) imagen original; (b) Segmentación con umbral de 128.

Ejemplo:

```
function umbral()
   I = imread('placa.jpg');
   info=imfinfo('placa.jpg');
   if (info.ColorType == 'truecolor')        % Determinar si es a color o en escala de grises
      I=rgb2gray(I);                         % para poder hacer o no la conversión.
   else
      I=I;
   end
   T = graythresh(I);
```

```
    T = T * 255;
    bw = (I <= T);
    imshow(I), title('Original')
    figure, imshow(bw), title('Umbralizacion')
    return;
end
```

2. Segmentación basadas en Regiones

La segmentación por regiones es utilizada para separar los objetos de interés. En este caso, la imagen es particionada en diferentes regiones, quedándose cada una relacionada en ciertas características y conjuntos de píxeles conectados. Así, a partir de la segmentación de la imagen en regiones, pueden ser realizadas las medidas sobre cada región y las relaciones entre las regiones adyacentes pueden ser establecidas.

Sea *R* la región correspondiente a la imagen que se va a segmentar. Vamos a ver el proceso de segmentación como un proceso en el cual dividimos la región **R** en *n* subregiones R1, R2, . . . ,Rn, tal que:

$$\bigcup_{i=1}^{n} Ri = R$$

La ecuación indica que la segmentación debe ser completa, es decir, que todo píxel debe estar en una región.

2.1 Crecimiento de Regiones

Como el nombre indica, el crecimiento de regiones es un procedimiento mediante el cual se agrupan píxeles o subregiones en regiones mayores. El procedimiento más sencillo se denomina *agregación de píxeles*, que comienza a partir de un conjunto de píxeles semilla (puntos de partida), de forma que a partir de cada semilla se crecen regiones añadiendo píxeles a dicha semilla de entre aquellos píxeles vecinos que tienen propiedades similares. El resultado de la segmentación dará lugar, como mucho, a tantas regiones como semillas haya. Sin embargo, puede darse el caso de que dos de esas semillas correspondan a píxeles de la misma región. En este caso el crecimiento desde una de las semillas absorberá a la otra.

Dos problemas fundamentales en el crecimiento de regiones son: por un lado, la selección de las semillas que representen adecuadamente a las regiones de interés; y por otro, la elección de las propiedades adecuadas que permitan ir añadiendo píxeles durante el proceso de crecimiento.

La selección de los puntos de partida en muchos casos depende de la naturaleza de la imagen que se va a segmentar. Por ejemplo, en aplicaciones militares con imágenes de infrarrojos, los blancos de interés normalmente desprenden calor, por lo que corresponden a píxeles claros frente a fondo oscuro. En este caso, los píxeles claros son una elección natural para las semillas.

La selección del criterio de similitud depende no sólo del problema considerado, sino también del tipo de imagen disponible. En cualquier caso siempre hay que tener en

cuenta la conectividad durante el proceso de crecimiento para que el resultado tenga significado dentro de su contexto.

Un ejemplo del criterio de similitud empleado para el crecimiento puede ser que el valor absoluto de la diferencia entre los niveles de gris del píxel candidato y la semilla, no exceda el diez por ciento de la diferencia entre el mayor nivel de gris de la imagen completa y el menor. Además los píxeles agregados a la región deben tener conectividad tipo ocho con respecto a los píxeles ya incluidos en la región.

Se pueden emplear criterios adicionales que incrementen la potencia del algoritmo de crecimiento de regiones, como el tamaño y la forma de la región crecida. La utilización de criterios estadísticos acerca de las regiones de interés es de suma importancia, visto que estas informaciones pueden hacer que los objetos existentes en la escena sean segmentados con mayor eficiencia.

Otro problema añadido en el crecimiento de regiones es la *regla de parada*. Básicamente, el crecimiento termina cuando no existen más píxeles en el vecindario de la región ya crecida que cumplan el criterio de similitud.

La segmentación por regiones está implementada en Matlab básicamente a través de los comandos ***bwlabel*** y ***Regionprops*.**

Bwlabel.- Etiqueta los componentes conectados en una imagen binaria. Su sintaxis es:

L = bwlabel(bw, n)

Donde retorna una matriz *L* del mismo tamaño de la imagen *bw*, que contiene etiquetas para los objetos conectados en *bw*. El valor de *n* puede ser 4 u 8, donde especifica objetos 4-conectados y 8-conectados respectivamente. Si el argumento *n* se omite, el valor es 8.

A continuación se muestra un ejemplo que utiliza la imagen de la figura 7.2.

Fig. 7.2 Imagen Binaria para el ejemplo de bwlabel

Ejemplo:

```
» I = imread('imagen.jpg');
» umbral = graythresh(I);        % Calcula el umbral entre [0 1]
» I = im2bw(I, umbral);          % binariza la imagen
» L = bwlabel(I);                % Crea regiones (etiqueta componentes conectados)
» max(max(L))                    % Imprime el número de objetos etiquetados
» ans =
        2
```

Regionprops.- Mide las propiedades de las regiones de una imagen. Su sintaxis es:

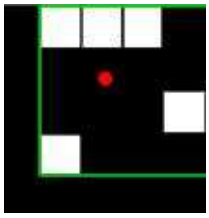
STATS = regionprops (L, propiedades)

Devuelve la medida de un grupo de propiedades para cada región etiquetada en la matriz *L*.
La tabla 7.1 muestra algunas de las propiedades válidas para el comando regionprops.

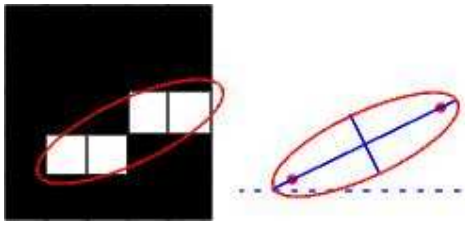
PROPIEDAD	DESCRIPCIÓN
Area	Calcula el área en píxeles cuadrados de la región
BoundingBox	Calcula la posición y dimensiones del mínimo rectángulo que envuelve a la región
Centroid	Posición del centroide (centro de masa) de la región
Eccentricity	Número escalar que da la excentricidad de la imagen.
MayorAxisLength	Longitud del eje de mayor longitud de la región (en píxeles).
MinorAxisLength	Longitud del eje de menor longitud de la región.
Orientation	El ángulo (en grados) entre el eje x y el eje de mayor longitud de la región.
Image	Imagen binaria del mismo tamaño que la frontera (bounding box) de la región.

Tabla 7.1 Algunas propiedades del comando **regionprops**

La figura 7.3 ilustra algunas propiedades. (a) La región consiste de los píxeles blancos; la caja verde es la frontera y el punto rojo es el centro de masa. (b) La orientación de la región.



Frontera y centro de masa
(a)



Orientación
(b)

Fig. 7.3 Ilustración de las propiedades: BoundingBox, Centroid, Orientation del comando regionprops [Mat04]

Si se usa la cadena '**all**' como propiedad, entonces todas las mediciones anteriores son calculadas. Si no se especifica ninguna propiedad o se usa la cadena '**basic**', se calculan solamente: Area, Centroid, y BoundingBox.

Ejemplo:

```
» I = imread('imagen.jpg');
» umbral = graythresh(I);          % Calcula el umbral entre [0 1]
» I = im2bw(I, umbral);            % binariza la imagen
» L = bwlabel(I);                  % Crea regiones
» stats = regionprops(L, 'all');   % Obtiene las estadísticas de las 2 regiones
» stats(1).BoundingBox             % imprime la frontera de la 1ra región
ans =
   10.50   15.50   61.00   62.00
» stats(1).Centroid                % imprime el centro de masa
ans =
   41.0134   46.6774
» stats(2).BoundingBox             % imprime la frontera de la 2da región
ans =
```

```
   93.50   15.50   63.00   59.00
» stats(2).Centroid
ans =
   125    45
```

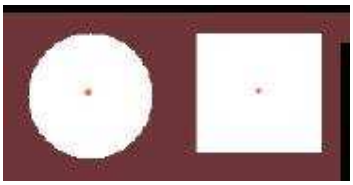


Fig. 7.4 Imagen Binaria con el centro de masa de cada región etiquetada.

En la figura 7.5 se muestra un ejemplo de la segmentación basada en Regiones de la placa de un automóvil para la extracción de los seis caracteres que la componen para su posterior reconocimiento.

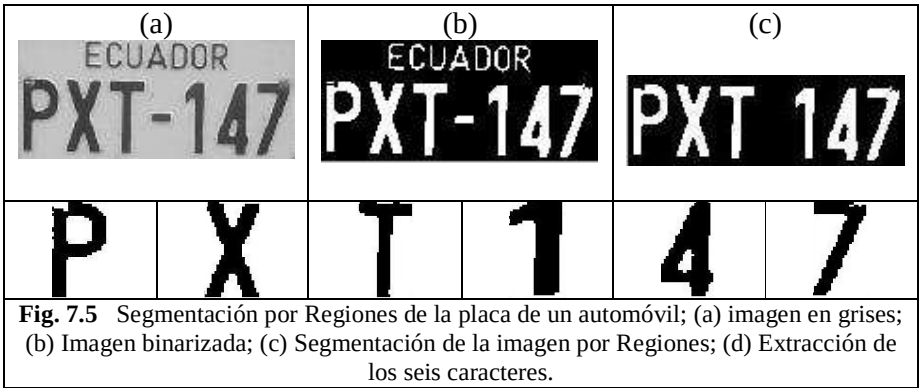


Fig. 7.5 Segmentación por Regiones de la placa de un automóvil; (a) imagen en grises; (b) Imagen binarizada; (c) Segmentación de la imagen por Regiones; (d) Extracción de los seis caracteres.

Ejemplo:

```
» I = imread('placa.jpg');
» umbral = graythresh(I);          % Calcula el umbral entre [0 1]
» I = im2bw(I, umbral);            % binariza la imagen
» L = bwlabel(I);                  % Crea regiones
» stats = regionprops(L, 'all');   % Estadísticas de las regiones
» E=stats(1).BoundingBox;          % Toma la frontera de la 1ra región
» I=imcrop(I,E);                   % Devuelve la 1ra región (carácter P)
» imshow(I)
```

Otros comandos de Matlab utilizados conjuntamente en la segmentación de imágenes son: ***find*** e ***ismember***, los cuales se describen a continuación.

Find.- Esta función se puede aplicar a vectores como a matrices. Cuando se aplica a un vector, encuentra los índices de los elementos distintos de cero. Su sintaxis es:

indices = find(X)

Si no se encuentran índices, se retorna una matriz vacía.

Ejemplo:

```
» X = [1 0 4 -3 0 0 0 8 6];
» indices = find(X)
indices =
    1     3     4     8     9
```

Se puede usar una expresión lógica, por ejemplo, find(X > 2). El resultado es un vector con los índices de los elementos que cumplen la condición. En este caso retorna los índices lineales correspondientes a las entradas de X que son mayores que 2.

```
ans =
    3     8     9
```

Cuando esta función se aplica a una matriz, la considera como un vector con una columna detrás de otra, de la primera a la última. A continuación se verán algunos ejemplos de utilización de estas funciones.

```
» A=magic(3)
A =
8 1 6
3 5 7
4 9 2
» m = find(A>4)
m =
1
5
6
7
8
```

Ismember.- Es una función lógica que devuelve 1 si $x \in A$ y 0 si $x \notin A$.
Sus sintaxis es: *f = ismember(A, S)*
Donde retorna un vector del mismo tamaño que A, que contiene el valor lógico 1 donde el elemento de A está en el grupo de S, y 0 en otro caso.

Ejemplo:

```
» A=[-5 6 2 1 4 6 1];
» ismember([-5 6 -2], A)
ans =
1 1 0
» ismember(4, A)
ans =
1
```

Capítulo VIII
Clasificación y Reconocimiento

Contenido

Luego de los procesos de segmentación, extracción de características y descripción, cada objeto queda representado por una colección (posiblemente ordenada y estructurada) de descriptores denominada **patrón**.

En los problemas de reconocimiento, cada patrón se supone que pertenece a una categoría o clase. El sistema de reconocimiento debe asignar cada objeto (de interés) a su categoría correspondiente.

En este capítulo se estudiarán diferentes algoritmos que permiten clasificar los elementos que aparecen dentro de una escena o imagen para poder entenderla.

Los métodos que permiten determinar, de manera automática, en qué clase se encuentra un objeto de un universo de trabajo se conocen como **clasificadores**.

INTRODUCCIÓN A LOS CLASIFICADORES

El resultado de la etapa de clasificación suele corresponder al último objetivo de un sistema de visión artificial. Es aquí donde un sistema de reconocimiento de caracteres clasifica una imagen como una letra determinada, o un sistema de reconocimiento biométrico facial identifica la imagen de un individuo como tal o cual persona.

Los algoritmos de clasificación tienen la misión de distinguir entre objetos diferentes de un conjunto predefinido llamado universo de trabajo (colección de clases), perteneciendo los diferentes tipos de objetos a algunas de estas clases.

Un clasificador toma un conjunto de características como entrada al proceso y produce como salida una clase etiquetada. Por ejemplo, distinguir entre tipos de rostros, llaves, monedas, etc.

Características discriminantes

Las características discriminantes o rasgos son las componentes o etiquetas que permiten discriminar a qué clases puede pertenecer un objeto del universo de trabajo, los comandos ***bwlabel*** y ***regionprops*** en Matlab, que se estudiaron en el capítulo anterior nos proporcionan estas características discriminantes.

Criterios para la selección de características discriminantes

En general, se busca el conjunto mínimo de características que permiten determinar de manera unívoca a qué clase pertenecen todos los objetos del universo de trabajo. Una mala elección puede hacer que el sistema sea caro y lento.

Algunas propiedades que deben poseer las características discriminantes para el reconocimiento automático son:

- **Economía.-** El mecanismo adecuado para el cálculo o la obtención de las características (sensores, dispositivos, etc.) debe tener un coste razonable.
- **Velocidad.-** El tiempo de cálculo debe ser adecuado.
- **Independencia.-** Las características no deben estar correlacionadas entre ellas.
- **Fiabilidad.-** Implica que objetos de la misma clase deben tener vectores de características con valores numéricos similares.

- **Capacidad discriminante.-** Los vectores de características de clases distintas tienen que tener valores numéricos distintos.

PROCESO DE CLASIFICACIÓN

Una vez determinadas las características discriminantes o rasgos para un problema concreto, la clasificación de un objeto comienza por la obtención de su patrón. El siguiente paso consiste en determinar la proximidad o grado de pertenencia de este patrón a cada una de las clases existentes, asignando el objeto a aquellas clases con las que el grado de semejanza sea mayor. A este efecto se definen las funciones discriminantes o funciones de decisión como aquellas funciones que asignan grados de semejanza de patrón a cada una de las diferentes clases.

Por ejemplo, en un sistema que controla una cinta transportadora por la que circulan monedas y llaves, se desea que cuente cuántas unidades hay de cada tipo.

El problema consiste en la determinación de aquellas características discriminantes de los objetos que van a permitir su reconocimiento. En este caso se propone usar el número de agujeros presentes en cada objeto (uno en la llave y ninguno en la moneda) y la desviación típica de las distancias del perímetro al centro del objeto (siempre cerca de cero en la llave y con un valor mayor en las monedas). En la figura 8.1 se muestra el proceso de clasificación de una imagen capturada de la cinta transportadora.

Fig. 8.1 Proceso de clasificación de objetos en una imagen [Mar06]

MÉTODOS DE CLASIFICACIÓN DE PATRONES

Existen varios métodos de los clasificadores de patrones, estos se pueden ordenar atendiendo a diferentes criterios como: la forma de construirse, el tipo de muestra, la información disponible. A continuación se muestra una posible división de los algoritmos de clasificación:

a) Adaptación (Pattern Matching)

Representan cada clase mediante un patrón prototipo. Algunos métodos de este tipo son:

- Clasificador de mínima distancia
- Adaptación por correlación

b) Clasificadores estadísticamente óptimos

Se fundamentan en la Teoría de la decisión estadística. En este método tenemos:
- Clasificador Bayesiano para clases gausianas

c) Redes neuronales

Engloba a un conjunto de técnicas que proporcionan soluciones flexibles, adaptables a cada problema. Se fundamentan en la teoría del aprendizaje estadístico.

Cada uno de estos métodos proporciona diferentes ventajas e inconvenientes. La elección de uno depende del problema que se vaya a resolver y de los resultados esperados del mismo. Describir todos los métodos queda fuera del alcance de este libro, y deberá consultarse en las fuentes de referencia citadas. En este capítulo solamente trataremos los métodos de clasificación basados en la *Adaptación por Correlación*, el cual se usará en los ejemplos posteriores y en el software de reconocimiento automático de placas de vehículos que acompaña a esta edición.

La Adaptación por Correlación se basa en la comparación de la imagen a clasificar con una o varias imágenes patrón que caracterizan a cada clase. Utilizan medidas de similitud basadas en correlaciones.

A continuación se presenta un ejemplo para la ***extracción de los caracteres de la placa de un vehículo y su reconocimiento automático***. La placa está compuesta de tres letras en mayúsculas y tres números, como se muestra en la figura 8.2.

Fig. 8.2 Extracción del primer carácter de la placa

Extracción de los caracteres

```
placa = imread('placa.jpg');        % Lee la imagen
L=bwlabel(placa);                   % Crea regiones
stats=regionprops(L, 'all');        % Obtiene las Estadísticas de las regiones
tam_stats=size(stats)               % Obtiene el número de regiones

E=stats(1).BoundingBox;             % Toma la frontera de la 1ra región (primer carácter)
X=E.*[1 0 0 0]; X=max(X);           % Determina el eje X de la esquina superior Izquierda
Y=E.*[0 1 0 0]; Y=max(Y);           % Determina el eje Y de la esquina superior Izquierda
W=E.*[0 0 1 0]; W=max(W);           % Determina el Ancho
H=E.*[0 0 0 1]; H=max(H);           % Determina la Altura
Corte=[X Y W H];                    % Determina coordenadas de corte
L1=imcrop(placa,Corte);             % Realiza el corte de la imagen

L1b=imresize(L1,[42 24]);           % Cambia el tamaño a 42x24 píxeles (alto x ancho)
L1b=(L1b==0);                       % Invierte los colores de la imagen (caracter)

subplot(1,2,1);  imshow(placa);     % Muestra la placa
title('Números y Letras de Placa'); % Pone un título a la placa
```

```
subplot(1,2,2);  imshow(L1b);          % Muestra el primer caracter
title('Primer Caracter');              % Pone un título
```

De igual manera se hace para extraer el resto de caracteres de la placa. Ahora hacemos uso de nuestro directorio que contiene los archivos de las imágenes binarias preelaboradas de 42x24 píxeles (alto x ancho) de números y letras del abecedario (*ver figura 8.3*) con el cual poder comparar y determinar el nivel de semejanza (coeficiente de correlación). Se puede construir las imágenes binarias, utilizando los programas photoshop o paint.

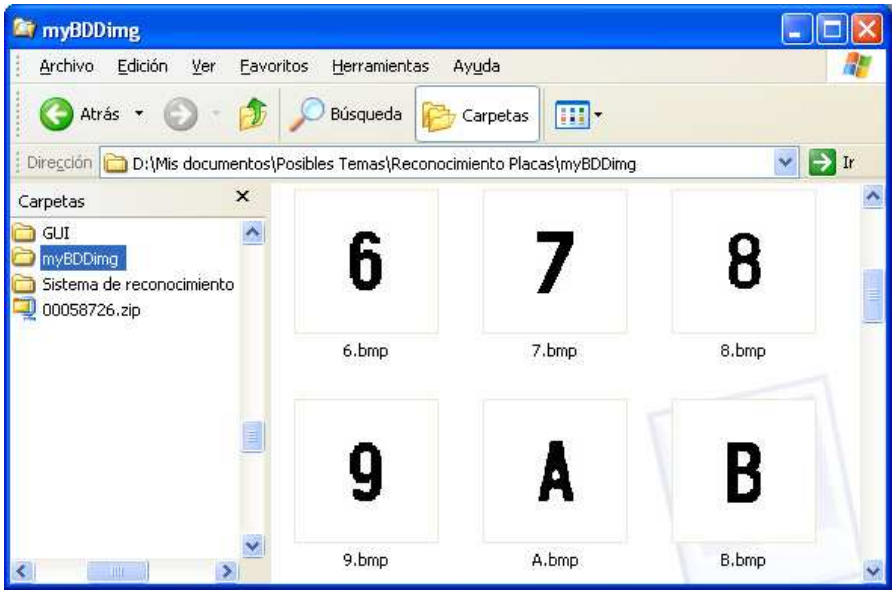


Fig. 8.3 Directorio que contiene los archivos de imágenes binarias preelaboradas de 42x24 píxeles (alto x ancho) de números y letras.

Reconocimiento o clasificación de los caracteres

```
% Cargamos las todas las imágenes binarias de letras (A-Z) preelaboradas en variables diferentes.
a=imread('A.bmp');      b=imread('B.bmp');
c=imread('C.bmp');      d=imread('D.bmp');
e=imread('E.bmp');      f=imread('F.bmp');
g=imread('G.bmp');      h=imread('H.bmp');
i=imread('I.bmp');      j=imread('J.bmp');
k=imread('K.bmp');      l=imread('L.bmp');
m=imread('M.bmp');      n=imread('N.bmp');
o=imread('O.bmp');      p=imread('P.bmp');
q=imread('Q.bmp');      r=imread('R.bmp');
s=imread('S.bmp');      t=imread('T.bmp');
u=imread('U.bmp');      v=imread('V.bmp');
w=imread('W.bmp');      x=imread('X.bmp');
y=imread('Y.bmp');      z=imread('Z.bmp');

% Cargamos las imágenes binarias de números (0-9) preelaboradas en variables diferentes.
uno=imread('1.bmp');    dos=imread('2.bmp');
tres=imread('3.bmp');   cuatro=imread('4.bmp');
cinco=imread('5.bmp');  seis=imread('6.bmp');
siete=imread('7.bmp');  ocho=imread('8.bmp');
```

```
nueve=imread('9.bmp');    cero=imread('0.bmp');


% Creamos un arreglo de matrices con las imágenes de letras y números preelaborados
alfabeto=[a b c d e f g h i j k l m n o p q r s t u v w x y z];
numeral=[uno dos tres cuatro cinco seis siete ocho nueve cero];

% Creamos un arreglo de matrices con las imágenes de los caracteres obtenidos en la segmentación de la placa
matricula=[L1b L2b L3b L4b L5b L6b];

% Dividimos la matriz en celdas que contienen submatrices (imágenes)
ab=mat2cell(alfabeto,42,[24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
24]);
numero=mat2cell(numeral,42,[24 24 24 24 24 24 24 24 24 24]);
plac=mat2cell(matricula,42,[24 24 24 24 24 24]);


% Trabajamos solo con las Letras de la placa
% Ciclo que calcula los coeficientes de correlación entre las letras de la placa y las letras del abecedario preelaborado.
fila=1;
while fila <= 3    % son 3 letras
   for posp=1:3    % las letras de la placa están en los 3 primeros caracteres
      plc=plac{1, posp};   % toma una letra de la placa
      pos=1;
      temp=0;
      while pos <= 26        % se recorre las 26 letras del abecedario
         temp=ab{1,pos};   % toma una letra del abecedario (A-Z)
         co=corr2(temp,plc);          % calcula el coeficiente de correlación
         letra(fila,pos)=co;           % guarda las correlaciones en una matriz (la fila 1 para correlaciones
                                       del 1er caracter, la fila 2 para el 2do caracter, y la fila 3 para el 3er
                                       caracter)
         pos=pos+1;
      end
      fila=fila+1;
      posp=posp+1;
   end
end

maxs=max(letra,[],2);       % calcula la correlación más alta de cada fila
for ind = 1:3                    % son 3 letras
   [posx posy]=find(letra==maxs(ind,1));   % busca la fila y la columna de la correlación más alta
   letras(ind)=posy;          % guarda la posición de la correlación más alta en el arreglo "letras"
   ind=ind+1;
end


% Asigna la letra que corresponde a cada índice del arreglo "letras."
   for lt = 1:3    % son 3 letras
      if letras(lt)== 1
         car(lt)='A';
      elseif letras(lt) == 2
         car(lt)='B';
      elseif letras(lt) == 3
         car(lt)='C';
      elseif letras(lt) == 4
         car(lt)='D';
      elseif letras(lt) == 5
         car(lt)='E';
```

```
    elseif letras(lt) == 6
       car(lt)='F';
    elseif letras(lt) == 7
       car(lt)='G';
    elseif letras(lt) == 8
       car(lt)='H';
    elseif letras(lt) == 9
       car(lt)='I';
    elseif letras(lt) == 10
       car(lt)='J';
    elseif letras(lt) == 11
       car(lt)='K';
    elseif letras(lt) == 12
       car(lt)='L';
    elseif letras(lt) == 13
       car(lt)='M';
    elseif letras(lt) == 14
       car(lt)='N';
    elseif letras(lt) == 15
       car(lt)='O';
    elseif letras(lt) == 16
       car(lt)='P';
    elseif letras(lt) == 17
       car(lt)='Q';
    elseif letras(lt) == 18
       car(lt)='R';
    elseif letras(lt) == 19
       car(lt)='S';
    elseif letras(lt) == 20
       car(lt)='T';
    elseif letras(lt) == 21
       car(lt)='U';
    elseif letras(lt) == 22
       car(lt)='V';
    elseif letras(lt) == 23
       car(lt)='W';
    elseif letras(lt) == 24
       car(lt)='X';
    elseif letras(lt) == 25
       car(lt)='Y';
    elseif letras(lt) == 26
       car(lt)='Z';
    else
       car(lt)='*';
    end
    lt=lt+1;
  end


% Trabajamos solo con los números de la placa
% Ciclo que calcula los coeficientes de correlación entre los números de la placa y los números preelaborados
fila=1;
while fila <= 3    % son 3 números
  for posp=4:6   % los números de la placa están en los 3 últimos caracteres
    plc=plac{1,posp};   % toma un número de la placa
    pos=1;
    temp=0;
    while pos <= 10        % se recorre los 10 dígitos
      temp=numero{1,pos};        % toma un número de los dígitos (0-9)
```

```
            co=corr2(temp,plc);          % calcula el coeficiente de correlación entre el número de la placa y
                                           un número de los dígitos preelaborados.
            num(fila,pos)=co;            % guarda las correlaciones en una matriz (la fila 1 para correlaciones
                                           del 1er número, la fila 2 para el 2do número, y la fila 3 para el 3er
                                           número)
            pos=pos+1;
        end
        fila=fila+1;
        posp=posp+1;
    end
end

maxs=max(num,[ ],2);      % calcula la correlación más alta de cada fila
for ind = 1:3             % son 3 números
    [posx posy]=find(num==maxs(ind,1));    % busca la fila y la columna de la correlación más alta
    nums(ind)=posy;       % guarda la posición de la correlación más alta en el arreglo "nums"
    ind=ind+1;
end

    % Asigna el número que corresponde a cada índice del arreglo "nums"
    for nm = 1:3    % son 3 números
        if nums(nm)== 1
            dig(nm)='1';
        elseif nums(nm) == 2
            dig(nm)='2';
        elseif nums(nm) == 3
            dig(nm)='3';
        elseif nums(nm) == 4
            dig(nm)='4';
        elseif nums(nm) == 5
            dig(nm)='5';
        elseif nums(nm) == 6
            dig(nm)='6';
        elseif nums(nm) == 7
            dig(nm)='7';
        elseif nums(nm) == 8
            dig(nm)='8';
        elseif nums(nm) == 9
            dig(nm)='9';
        elseif nums(nm) == 10
            dig(nm)='0';
        else
            dig(nm)='*';
        end
        nm=nm+1;
    end

plate=horzcat(car, dig);    % concatena las 3 letras y los 3 números de la placa (ASCII)
disp('La placa reconocida automáticamente es: ');
disp(plate);                % Imprime la placa (ASCII)
```

De esta manera logramos simular la visión humana, al implementar el proceso de clasificación para extraer y reconocer las letras y números de una imagen que contiene la placa de un vehículo. El proceso completo del reconocimiento automático de caracteres de la placa se muestra en la figura 8.4 y 8.5.

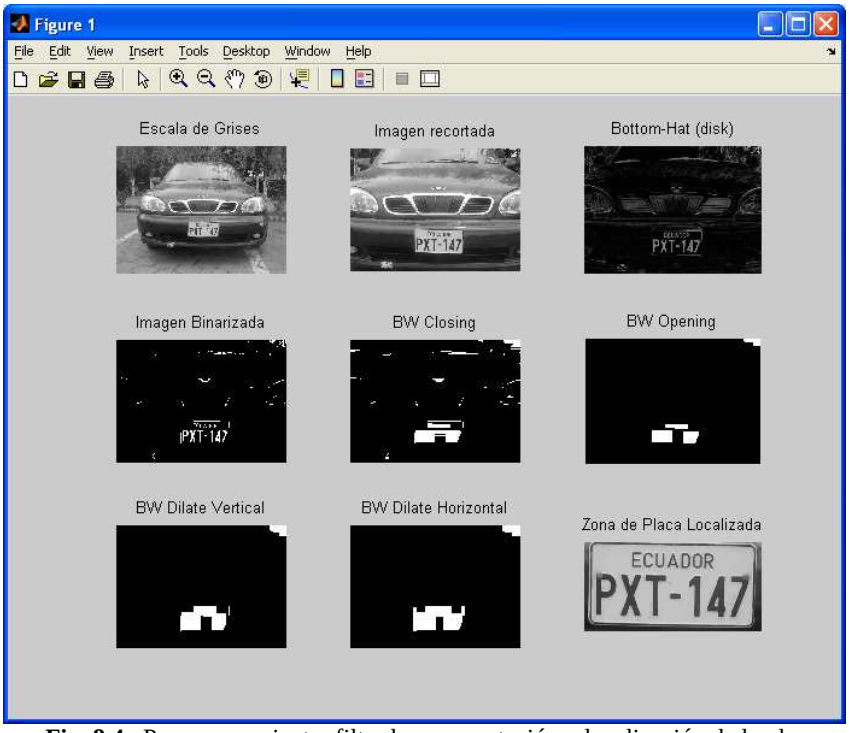


Fig. 8.4 Preprocesamiento, filtrado, segmentación y localización de la placa

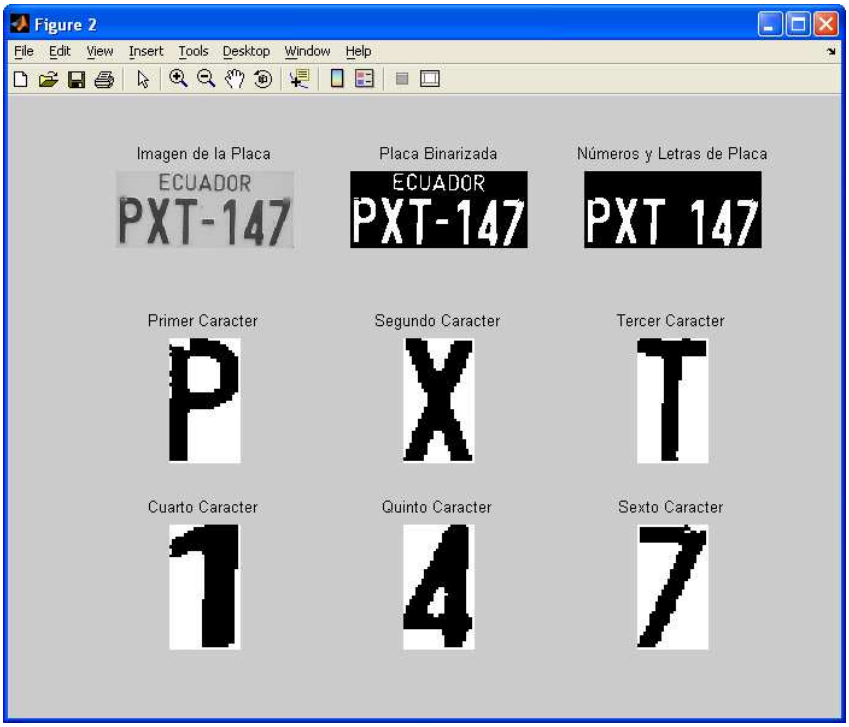


Fig. 8.5 Segmentación y Clasificación de los caracteres de la placa

Adicionalmente, una vez obtenidos los caracteres y dígitos de la placa del vehículo, se puede implementar un proceso para realizar consultas a una base de datos y extraer otra información relevante del vehículo (marca, modelo, color) y su propietario (nombres, dirección, teléfono).

Al clasificar un patrón se puede obtener un valor sobre la confianza de tal clasificación. Si la confianza en el resultado no es alta, puede volverse a etapas anteriores (como preprocesamiento, filtrado o segmentación) para aplicar otros enfoques y obtener un resultado de mayor confianza. En el peor de los casos, se puede decidir que el sistema, no ofrezca ningún resultado ante un patrón dudoso, a fin de no perder credibilidad del sistema frente a los usuarios finales.

Los resultados obtenidos en el proceso de clasificación y reconocimiento depende en gran medida de la calidad de la imagen original obtenida, por lo que, el proceso de captura de la imagen digital es crucial y se deben tomar en cuenta algunos parámetros como: la distancia entre el dispositivo de captura y el objeto de estudio; y una iluminación adecuada y uniforme.

Anexo A
Manual de instalación
Sistema de reconocimiento automático de placas

Para instalar y utilizar el sistema de reconocimiento automático de placas, inserte el CD que acompaña al libro en la unidad respectiva de su computador y copie el directorio denominado *"Reconocimiento automático de placas"* en la raíz de su disco duro, preferentemente en la unidad C:\ (ver figura A1).

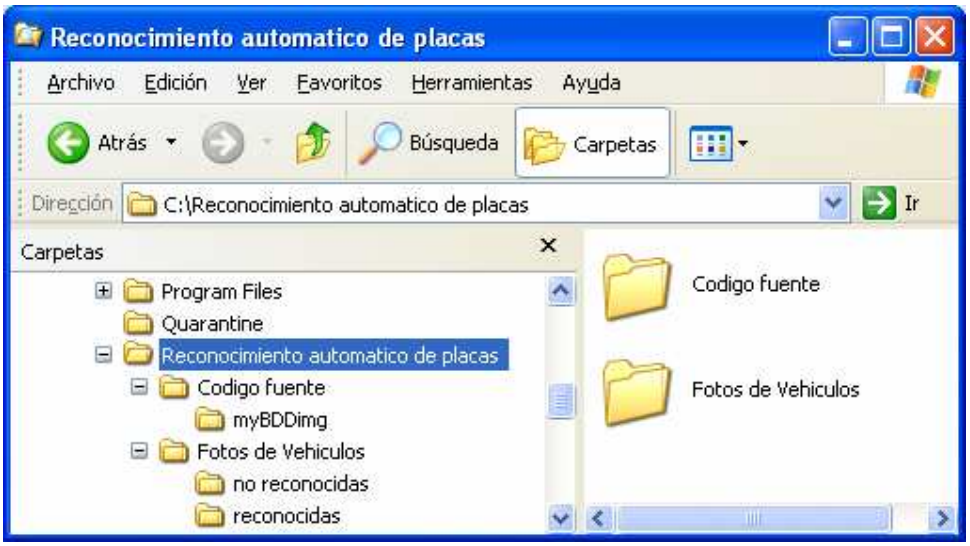


Fig. A1 Instalación del sistema de reconocimiento automático de placas

Recuerde que debe tener instalado Matlab en su computador para el correcto funcionamiento del sistema, especialmente los módulos: *Image Acquisition Toolbox* y *Image Processing Toolbox.*

Bajo el directorio denominado *"Reconocimiento automático de placas"* se dispone de dos subdirectorios, que son:

1. **Código fuente.-** contiene los archivos fuentes del programa, *.asv, *.fig, *.m (ver figura A2). También contiene el subdirectorio de las imágenes binarias de números y letras previamente elaboradas (42 de alto x 24 de ancho), ver figura A3.

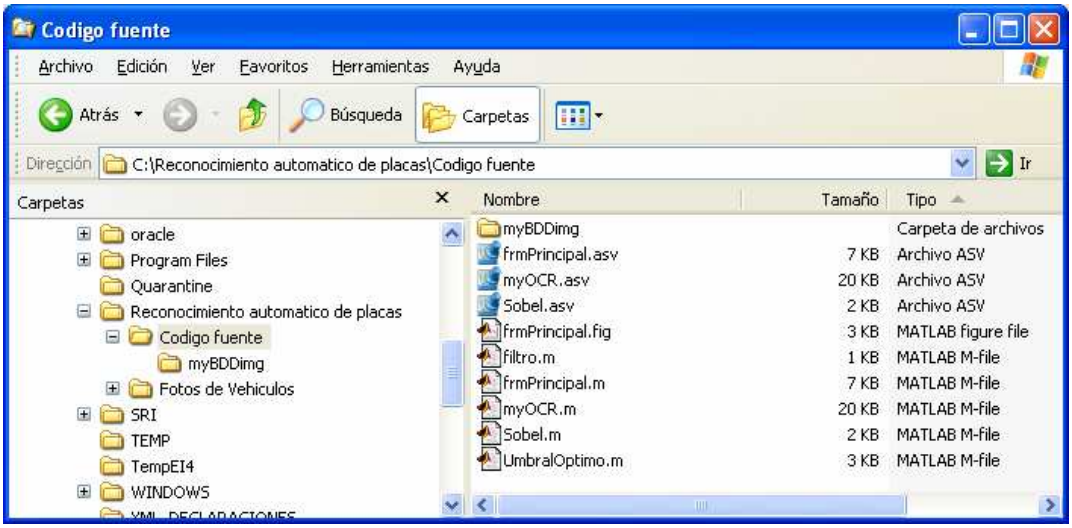


Fig. A2 Directorio del Código Fuente del sistema

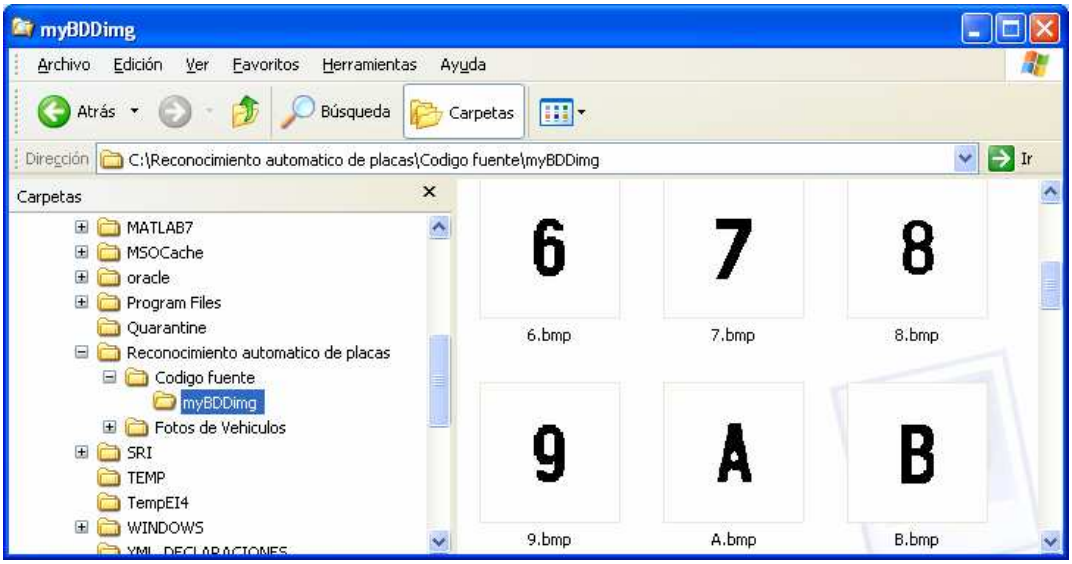


Fig. A3 Directorio de Imágenes binarias preelaboradas (42x24)

2. **Fotos de Vehículos.-** contiene los archivos gráficos (en formato jpg) de los vehículos, obtenidos con una cámara digital (de 3.2 mega píxeles) a una distancia de 3 metros entre el dispositivo de captura y el objeto de interés. Ver figura A4.

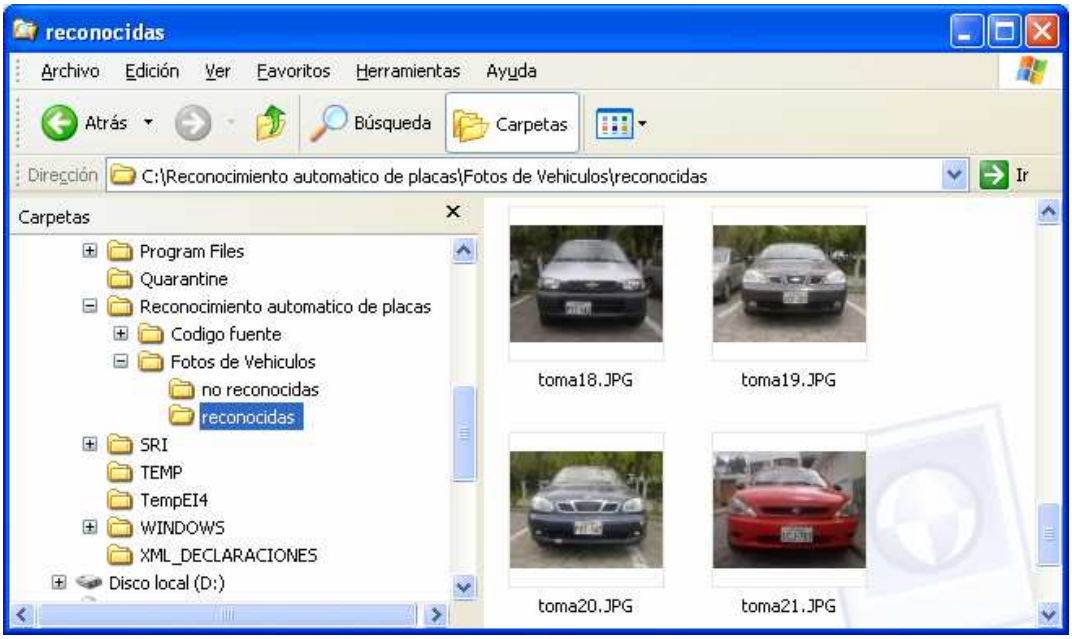


Fig. A4 Directorio de fotos de los vehículos

Las imágenes de los vehículos disponibles en el CD, fueron obtenidas en el parqueadero del campus universitario de la PUCE-SI.

Fin de la instalación, ahora puede empezar a utilizar el sistema de reconocimiento automático de placas de vehículos.

Anexo B

Manual de usuario

Sistema de reconocimiento automático de placas

Una vez instalado el sistema de reconocimiento automático de placas en su computador, vamos a ejecutar el programa. Para ello, diríjase al directorio *C:\Reconocimiento automático de placas\Codigo fuente* donde copió el código fuente del programa al disco duro, y dé doble clic en el archivo *frmPrincipal.fig*. Puede opcionalmente crear un acceso directo en el escritorio para ejecutar rápidamente.

Al iniciarse la aplicación, aparece la pantalla principal (ver figura B1).

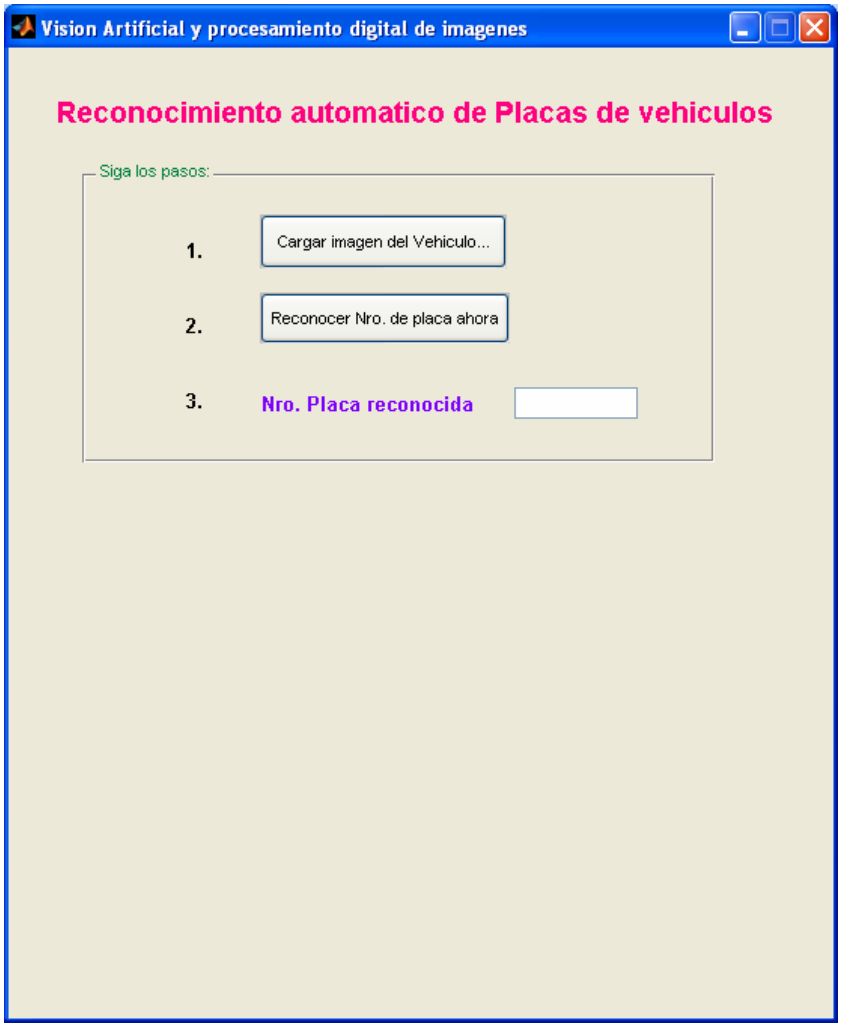


Fig. B1 Pantalla principal del sistema de reconocimiento automático de placas

Presione un clic en el botón etiquetado como *"Cargar imagen del Vehículo"*. Aparece el cuadro de diálogo "Select File to Open" que le permite seleccionar la foto de la imagen del vehículo que va a reconocer, como se muestra en la figura B2.

Fig. B2 Cuadro de diálogo Abrir

Una vez que selecciona el archivo de la foto del vehículo, presione el botón **Abrir** y se muestra la imagen seleccionada en la pantalla principal (ver figura B3).

Fig. B3 Fotografía del vehículo que va a reconocer

Ahora presione un clic en el botón etiquetado como *"Reconocer Nro. de placa ahora"*. Esperamos a que el programa procese la imagen y nos muestre el progreso de las fases del reconocimiento que se está realizando, como se muestra en la figura B4 y B5.

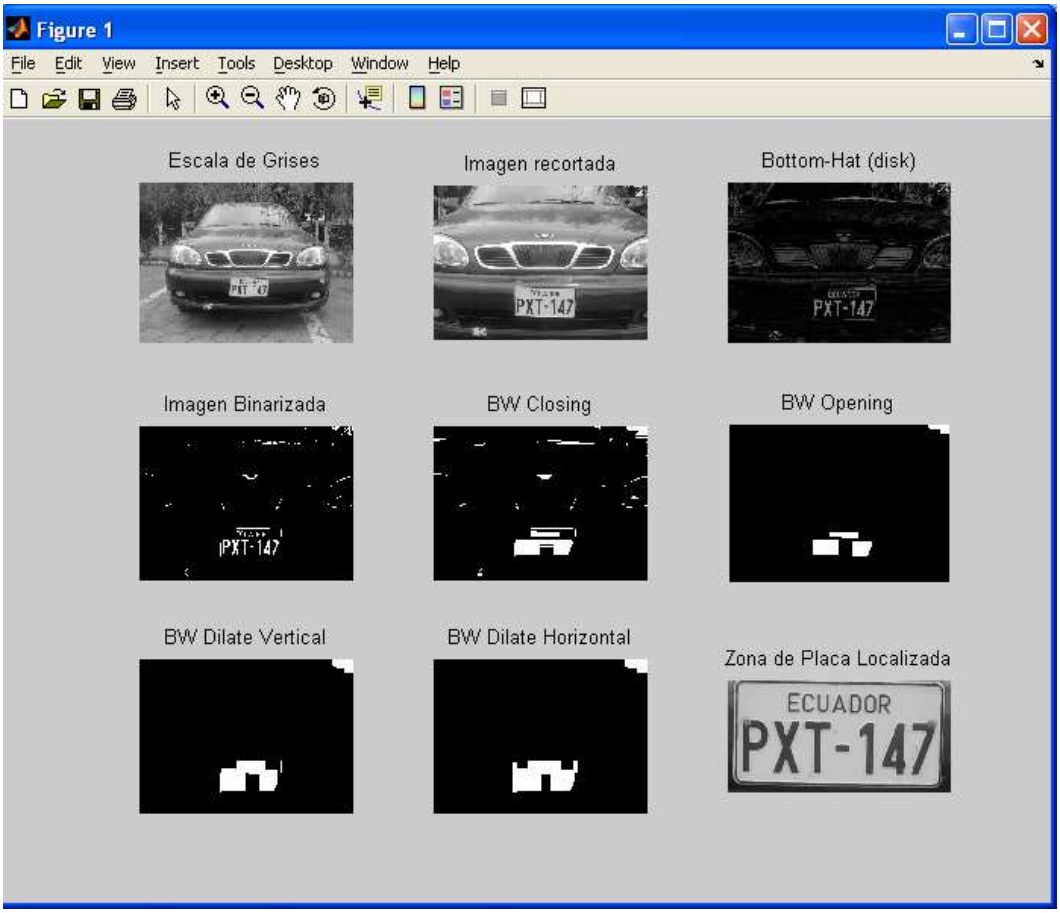


Fig. B4 Localización de la zona de la placa del vehículo

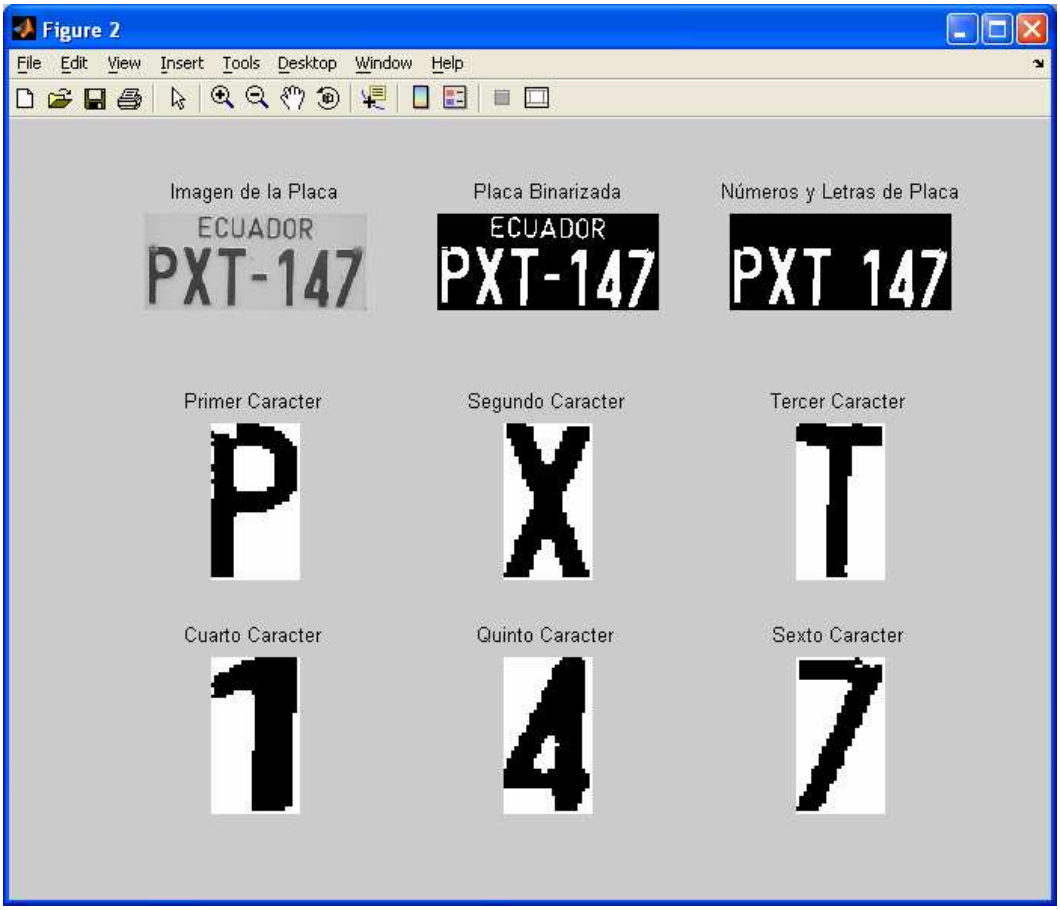


Fig. B5 Segmentación de los caracteres de la placa

Finalmente, transcurrido unos pocos segundos (*dependiendo de la velocidad de su computador*) se muestra en la pantalla principal, la placa ya reconocida en texto ASCII, ver figura B6.

Fig. B6 Reconocimiento de los caracteres de la placa

Para reconocer otras placas de vehículos, repita los pasos descritos anteriormente.

Anexo C
Imágenes por capítulos

Las imágenes que se utilizan en el libro para las prácticas y ejercicios, se encuentran organizadas y clasificadas por capítulos, y se encuentran disponibles en el CD que acompaña al libro en el directorio denominado "*Imágenes por Capítulos*".

Fig. C1 Imágenes usadas en el libro

GLOSARIO

Brillo.- Es la luminosidad de un color (la capacidad de reflejar el blanco). Alude a la claridad u oscuridad de un tono. La luminosidad puede variar añadiendo negro o blanco a un tono.
CCD.- (charge-coupled device o dispositivo de cargas interconectadas) es un sensor con diminutas células fotoeléctricas que registran la imagen. En la actualidad son más populares en las cámaras digitales. La capacidad de resolución o detalle de la imagen depende del número de células fotoeléctricas del CCD. Este número se expresa en píxeles.
Clase: conjunto de entidades que comparten alguna característica que las diferencia de otras clases.
Clase de rechazo: conjunto de entidades que no se pueden etiquetar como ninguna de las clases del problema que va a resolver.
Clasificación: proceso por el que se asigna una "etiqueta", que representa una clase, a un patrón concreto.
Clasificador: subsistema que utiliza un vector de características de la entidad cuantificable y lo asigna a una de las clases.
CMYK.- Cyan, Magenta, Yellow, blacK.
Color.- Es lo que vemos cuando llega a nuestros ojos la luz reflejada por un objeto. La luz blanca parece no tener color, pero en realidad está formada por una mezcla de colores.
Contraste.- La relación entre las zonas más claras y más oscuras de una imagen.
Filtros lineales.- Se dice que un filtro es lineal si se puede aplicar el principio de superposición.
HSI.- tono (H), saturación (S) y brillo o intensidad (I).
Imagen Digital.- Está compuesta por un número finito de elementos llamados píxeles, cada uno de los cuales con un valor de brillo y una posición particular.
Intensidad.- Ver la definición de brillo.
Imagen Multiespectral.- Es un conjunto de imágenes, con las mismas propiedades geométricas, y cada una de las cuales, en un diferente rango de longitudes de onda del espectro electromagnético.
Modelo de color.- Son las especificaciones de los colores utilizando algún estándar, por ejemplos: RGB , CMYK, HSI.
Morfología.- Estudio de la forma y la estructura.
Morfología matemática.- Es una técnica de procesado no lineal de la imagen, interesada en la geometría de los objetos.
Patrón.- Objeto que queda representado por una colección (posiblemente ordenada y estructurada) de descriptores luego de los procesos de segmentación, extracción de características y descripción.
Píxel.- Abreviatura de *picture element.* Es el elemento básico de una imagen. Es la unidad indivisible de una imagen.
Reconocimiento.- Ver la definición de clasificación.
Resolución espacial.- Es el tamaño del píxel, determinado por el rango al cual el escáner o cámara muestrea la imagen.
Resolución de Brillo.- El valor de color de cada píxel es definido por un bit o un grupo de bits, generalmente 8-bits o 24-bits por canal RGB.
Resolución de Imagen.- Es el número de píxeles que contiene las imágenes (ancho por alto), expresada como 640 x 480 por ejemplo.
RGB.- Red, Green, Blue.

Ruido.- Se manifiesta como píxeles aislados que toman un nivel de gris diferente al de sus vecinos. Aparece como pequeñas y aleatorias variaciones en el brillo y el color.
Saturación.- Se dice que un color está saturado cuando, al agregarle negro o blanco, llega a su límite tonal máximo, pero sin traspasar la frontera que lo transformaría en otro color, o simplemente en negro o blanco.
Segmentar.- Consiste en dividir una imagen digital en regiones homogéneas o similares con respecto a una o más características (como el brillo, el color, tamaño, longitud, forma).
Textura.- Disposición de las características de los elementos constituyentes de algo, especialmente, los relacionados con la apariencia superficial o la calidad al tacto.
Tono.- Es el matiz del color, es decir, el color en sí mismo. Es simplemente un sinónimo de color.

BIBLIOGRAFÍA

Bibliográfica

[Alb06] Alba J., *Métodos de análisis de Imágenes, Extracción de características*, Universidad de Vigo, marzo del 2006.

[AM06] Alba J., Martín F., *Morfología Matemática - Aplicación a procesado de imágenes binarias y monocromáticas*, Universidad de Vigo, 2006.

[Ate01] Atencia J., *Aprenda Matlab 6.0 como si estuviera en primero*, Universidad de Navarra, 2001.

[Dom94] Domingo A., *Tratamiento digital de imágenes*, Ed. Anaya Multimedia, 1994.

[Ere02] Erez J., *A Real-time vehicle License Plate Recognition (LPR) System*, Israel Institute of Technology, 2002.

[Esc01] A. de la Escalera, *Visión por computador: Fundamentos y métodos*, Pearson-Prentice Hall, 2001.

[Esq02] Esqueda J., *Fundamentos de Procesamiento de Imágenes*, Universidad Autónoma de Baja California, Noviembre del 2002.

[Ett97] Etter D. M., *Solución de Problemas de Ingeniería con Matlab*, 2ª Edición, Prentice Hall, 1997.

[Gar01] García J., *Aprenda Matlab 6.1 como si estuviera en primero*, Universidad Politécnica de Madrid, 2001.

[Gar05] García J., *Aprenda Matlab 7.0 como si estuviera en primero*, Universidad Politécnica de Madrid, 2005.

[Gon00] González J., *Visión por computador*, Ed. Paraninfo, 2000.

[GW96] González R., Woods R., *Tratamiento Digital de Imágenes*, Addison-Wesley/Díaz de Santos, 1996.

[Imb07] Imbaquingo D., *Sistema de reconocimiento espectral de imágenes digitales y reconocimiento óptico de caracteres*, Universidad Técnica del Norte, 2007.

[Mar02] Martín M., *Técnicas Clásicas de Segmentación de Imagen*, 2002.

[Mar06] Martín F., *Reconocimiento de Patrones*, Universidad de Vigo, 2006.

[Mar93] Maravall D., *Reconocimiento de Formas y Visión Artificial*, Ed. Ra-Ma, 1993.

[Mat04] The MathWorks, Inc., *MATLAB Help – Toolbox Imagging Processing*, Mayo 2004, versión 7.0

[Nar07] Narváez H., *Identificación y autenticación de personas a través del reconocimiento de patrones y el análisis biométrico utilizando procesamiento de imágenes*, Universidad Técnica del Norte, 2007.

[Par97] Parker J. R., *Algorithms for Image Processing and Computer Vision*, J. Wiley and Sons, 1997.

[Qui05] Universidad Nacional de Quilmas, *Visión Artificial*, 2005.

[SHB99] Sonka M., Hlavac V. y Boyle R., *Image Processing, Analysis and Machine Vision*, PWS Publishing, 1999.

[Vel04] Vélez S., *Visión por Computador*, Dykinson, 2004.

Sitios de Internet

[www01] http://www.pyssa.com/es/
Automatización de procesos industriales.

[www02] http://www.iti.upv.es/groups/riva
Reconocimiento de Imágenes y Visión Artificial

[www03] http://turing.iimas.unam.mx/~elena/Teaching/PDI-Lic.html
Procesamiento Digital de Imágenes (Programa de Curso), Martínez E., UNAM, Febrero de 2005

[www04] http://www.fotonostra.com/grafico/elcolor.htm
Teoría del Color

[www05] http://www.lfcia.org/~cipenedo/cursos/Ip/inicio2.html
Visión Artificial.

[www06] http://www.ele.uri.edu/~hansenj/projects/ele585/OCR/
Optical Character Recognition (OCR)

[www07] http://www.aurova.ua.es/
Grupo de automática, robótica y visión artificial.

[www08] www.vaelsys.com
Empresa especializada en soluciones de visión artificial.

Currículum del Autor

Iván Danilo García Santillán

Nacido en Ibarra – Ecuador el 07 de noviembre de 1978. Es Ingeniero en Sistemas Computacionales y sus estudios de cuarto nivel son: Diplomado superior en docencia universitaria y Especialista en Gerencia Informática. Actualmente se encuentra culminando la Maestría en Gerencia Informática.

Es Docente-Investigador de la Escuela de Ingeniería de la PUCE-SI en las asignaturas de Procesamiento de imágenes, Graficación y animación, Estructura de datos, Programación y Base de datos. También es Docente invitado en la Escuela de Ingeniería de Sistemas Computacionales de la Universidad Técnica del Norte para las asignaturas de Ingeniería de software, Administración de sistemas y Graficación.

Además, ha trabajado durante cinco años en el desarrollo de sistemas informáticos financieros-contables para instituciones gubernamentales y privadas del país.

Su labor como investigador, docente, asesor y consultor le ha permitido desarrollar y dirigir algunos proyectos e investigaciones en el área de informática, sistemas y computación; dictar conferencias y seminarios; publicar artículos científicos y de opinión en revistas y diarios de circulación regional y nacional.

Actualmente se desempeña como Docente y Coordinador de Investigación de la Escuela de Ingeniería de la PUCE-SI.

Corrección de estilo: Dr. Silverio Laquidaín
Registro de Autor: 029032