Reconocimiento de Patrones

Manejo básico de imágenes

Práctica

Reconocimiento de Patrones

1. Objetivo

Al término de esta práctica, el alumno conocerá las distintas formas de desplegar una imagen en distintos formatos, así como algunas manipulaciones básicas, para el procesamiento de imágenes.

2. Introducción

2.1. La imagen

Una imagen es la representación en dos dimensiones, que contiene la información de color (intensidad) así como el diseño, esta información es guardada en los elementos más simples llamados pixeles o pel (del inglés picture element). En cuanto al diseño de la imagen contiene una disposición de matriz de *m* columnas por *n* filas (m, n), donde se encuentran ordenadamente los pixeles, el origen se ubica en la esquina superior izquierda donde se ubica el elemento (0,0). La imagen puede contener un solo canal, esto es con aquellas que son a escala de grises o blanco y negro, así mismo múltiples canales, tal es el caso de las imágenes a color RGB o YUV.

2.2. Resolución de una imagen

Una imagen puede ser juzgada por la calidad que posee al momento de ser obtenida. Para sistemas de imagenología médica, estas imágenes deben de ser diagnósticamente útiles para detectar una enfermedad. La resolución de la fuente de imágenes es definida por medio de tres cantidades:

- Resolución Espacial: Se refiere a la cantidad de pixeles en columnas (C) y renglones (R) para representar el espacio visual capturado en una imagen, se relaciona con el muestreo de la señal de una imagen. Generalmente se expresa de la forma (C x R), por ejemplo (600x 400, 1366 x 768, etc.)
- Resolución Temporal: Cuando se tiene imágenes continuas como en el caso del video, la resolución temporal determina la cantidad de cuadros por una cantidad de tiempo, en los archivos de video es común utilizar los cuadros por segundo o fps[1].
- Resolución de bit o profundidad de bit: Define el número posible de valores de intensidad/color que un píxel puede tener y se relaciona con la cuantización de la información de una imagen. En el caso de las imágenes binarias, solo disponemos de 2 colores, blanco y negro. Para una imagen de escala de grises puede tener 2^8 niveles de escala de grises. Generalmente se representa con el número de bits requeridos para almacenar el nivel de cuantización dado.

[1] Frames per second

2.3. Formato de imagen

Existen por lo menos 100 formatos de imagen. Algunos de estos formatos tales como jpeg o png, son utilizados para imágenes fotográficas. Por otro lado, los formatos DICOM o NIfTI, son utilizados en imagenología médica. Otros formatos como el TIFF, ics, ims, entre otros son usados para imágenes de microscopio. Sin embargo, no nos debemos preocupar por el formato de imagen, así como de sus metadatos. Los módulos y paquetes de Python tienen la capacidad de leerlos y extraer los datos necesarios que le serán útiles al usuario (por ejemplo, tamaño de la imagen, tipo o modo, tipo de dato). A continuación, definiremos algunos formatos más comunes de imágenes digitales.

- JPEG: es el acrónimo inglés de *Joint Photographic Experts Group*, su formato es representado como .jpg o .jpeg. Es uno de los más populares debido a su habilidad para comprimir significativamente los datos con una pérdida visual mínima. Es un formato con pérdida, que comprime los datos usando la Transformada de Coseno Discreta (TCD). Este formato no es adecuado para almacenar imágenes que contienen estructuras finas como líneas, curvas etc.
- TIFF: es el acrónimo inglés de *Tagged Image Format*. Su extensión es .tif o .tiff, fue creado en la década de los 80´s para almacenar y codificar documentos escaneados. Originalmente fue desarrollado para datos de un solo bit, pero hoy en día el estándar permite almacenar 16 bits incluso datos de punto flotante.
- DICOM: es el acrónimo inglés de *Digital Imaging and Communication in Medicine*. Es un formato estándar para codificar y transmitir Imágenes de Resonancia Magnética (MRI) e Imágenes de Tomografía Computarizada. Este formato almacena información junto con otros datos entre los que destacan; detalles del paciente, parámetros de adquisición, hora de creación, modelo del equipo, información del hospital etc. DICOM es usado en varias especialidades de medicina como Radiología, Neurología, Cirugía, Cardiología, Oncología entre otros.

2.4. Tipos de imagen

El tipo de imagen elegido para cierta aplicación involucra la información que se quiere obtener, así como su resolución de bit. Para esto se tienen diferentes tipos de imagen que se enlistan a continuación:

Las imágenes de un solo canal son aquellas en las que cada píxel es representado por un solo valor, los podemos dividir en dos tipos:

- Imágenes binarias: o también llamadas monocromáticas, el valor de un píxel 0 se representa color negro y el píxel de valor 1 representa el color blanco.
- Imágenes de escala de grises: cada píxel puede ser representado con 8 bits y tiene valores de dentro del rango de 0-255.

Las imágenes multicanal son representados por una tupla de valores. Los podemos dividir en los siguientes tipos.

- Imágenes RGB: son imágenes en arreglos de 3-D en la que cada píxel es representado por una tupla de valores (r,g,b), que representan a los colores rojo, verde y azul respectivamente. Comúnmente, se almacenan como una secuencia de enteros en orden de canal sucesivo.

- Imágenes HSV: son imágenes en las que cada píxel es representado por una tupla de valores (h,s,v) que representan al color, saturación y brillo respectivamente.
  El modelo HSV describe colores de manera similar a como el ojo humano tiende a percibir colores.
- Imágenes de RGBA: cada píxel es representado por una tupla de cuatro valores (r,g,b,a) el ultimo valor representa la transparencia.

Imágenes de punto flotante: Estas imágenes almacenan números de punto flotante que, dentro de un rango determinado definido por la precisión de punto flotante de la resolución de bits de la imagen, representa la intensidad. Las imágenes de punto flotante pueden ser almacenadas en formato TIFF o DICOM.

3. Manipulación de Imágenes con Python

Para la lectura y escritura de imágenes Python cuenta con varios paquetes que son los siguientes: Matplotlib, PIL (Python Image Library), Scikit-Image, Scipy y OpenCV. Cada uno de estos paquetes carga la imagen de una manera diferente ya que en algunos casos se ve como una clase imagen y en otros como un arreglo de Numpy. Dependiendo del tipo de manipulación que se le dé será el paquete para utilizar. De igual manera, al desplegar la imagen o conjunto de imágenes se hará con el visor del sistema operativo, el visor particular del módulo o con la ayuda de otro paquete auxiliar.

3.1. Recomendaciones para la manipulación básica de imágenes en Python

- ✓ Importar los módulos necesarios a utilizar, preferentemente solo las funciones específicas para no utilizar mucha memoria. Se puede usar las siguientes sintaxis:

  import nombre_del_módulo.funcion **as** nombre_alternativo
  from nombre_del_módulo **import** función1, función2 …
  from nombre_del_módulo **import** función **as** nombre_alternativo

- ✓ Especificar la ruta completa de la imagen, así como el nombre y la extensión de archivo. En el caso de que estén en la misma carpeta es script o cuaderno de Jupyter, solo basta con poner el nombre y la extensión del archivo.
- ✓ Para el caso de las imágenes DICOM importar el módulo correspondiente (Pydicom).

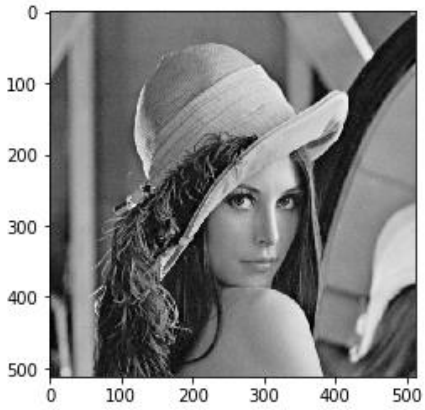
Ejemplos:

En el siguiente ejemplo se muestra como desplegar una imagen en formato jpeg.

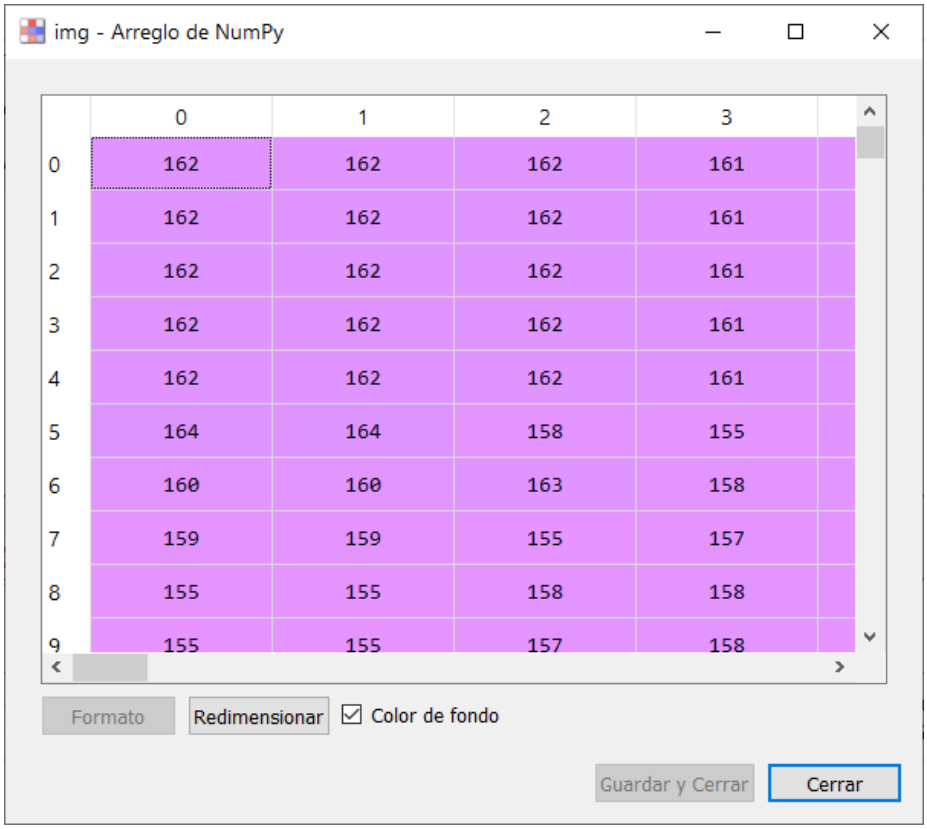
```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

img=mpimg.imread('lena_gray_512.tif')
imgplot = plt.imshow(img, cmap="gray")
```

La imagen de salida es la siguiente:



En el entorno Spyder se puede visualizar la imagen en forma de matriz con valores en la sección “Explorador de variables”, quedando como la siguiente imagen.



Matriz de valores

4. Ejercicios Imagenes

NOTA: Puede usted hacer experimentos cambiando las imágenes

De la carpeta de imágenes: realiza las siguientes actividades.

4.1. Desarrolla un script para leer y desplegar cada imagen con los paquetes de Matplotlib, OpenCV, Scikit-Image y PIL.

4.2. Imprimir el tipo de imagen, el tamaño y el tipo de dato

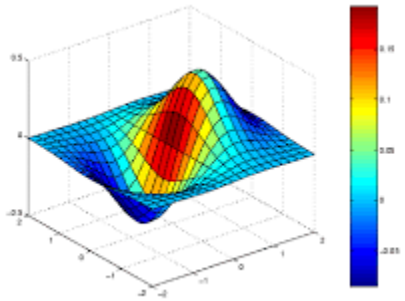
4.3. De las imágenes "lena_color_512.tif", "peppers_color.tif". Desarrolla un script con OpenCV y Scikit-Image para cambiar el espacio de color de:

4.3.1. RGB a Escala de grises

4.3.2. RGB a YUV

4.3.3. RGB a HSV

4.4. Despliega la paleta de colores de RGB por separado, ver figura siguiente, la barra de la derecha con valores es la paleta de colores.



4.5. De una imagen que usted escoja, dejarla en escala de grises y procure que sea igual en renglones y en columnas. Programe una función que realice decimación de una imagen, reduciendola a la mitad de su tamaño original. Y promediando en grupos de 4 pixeles. Pruebe con su imagen.

4.6. Convierte la imagen peppers_color.tif a escala de grises,

4.6.1. Recortela de manera que solo quede uno de los pimientos verdes en ese recorte

4.6.2. Guárdela en formato .jpg.

4.7. Un formato de imágenes sin ningún tipo de codificación se conoce como formato crudo (RAW). De la imagen "rosa800x600.raw" lea y despliegue la imagen. Tome en cuenta que esta imagen maneja la precisión de integer8 y el tamaño es de 600x800 pixeles.

4.8. Lectura de Video archivo ".avi". Probar el siguiente código y muestre el video "0X2A8498756C4D6E82_corto.avi". Si existe un error corregirlo.

```
conda install -c anaconda opencv
import numpy as np
import cv2
import matplotlib.pyplot as plt

# read video
```

```
cap = cv2.VideoCapture('/home/ast/datasets/ewap/seq_eth/seq_eth.avi')

ret, frame = cap.read()
plt.figure()
plt.imshow(frame)
```

4.9. Imagen DICOM

Mostrar la información del encabezado de los formatos, investigar como se despliega.

a. Usar imagen DICOM:

Realiza un script que lea las imágenes de la carpeta DICOM y muestra en una misma ventana las 280 imágenes de tal manera que se pueda simular el movimiento del corazón. Utiliza ciclos for y el comando pause, para leer la ruta completa de la carpeta.

5. Bibliografía

[1] A. C. a. C. Revision, «Pillow (PIL Fork),» Read the Docs, 2010. [En línea]. Available: https://pillow.readthedocs.io/en/stable/reference/Image.html. [Último acceso: 12 Agosto 2019].

[2] S. Dey, Hands-On Image Processing with Python: Expert techniques for advanced image analysis and efective interpretaction of image data, Birmingham, UK: Packt Publishing, 2018.

[3] R. Chityala y S. Pudipeddi, Image Processing and Acquisition using Python, Boca Raton, USA: CRC Press , 2014.

[4] P. Joshi, Open CV with Python By Example: Build real-world computer vision applications and develop cool demos using OpenCV for Python, Birmingham, UK: Packt Publishing, 2015.

[5] C. Solomon y T. Brekon, Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab, Oxford, UK: Wiley's global Scientific, Technical and Medical business with Blackwell Publishing., 2011.

[6] E. Moya Albor, Práctica 2: Manejo Básico de imágenes con Matlab, CDMX, 2015.