

Práctica 1: Manejo Básico de Imágenes

Reconocimiento de Patrones - 0757

*Facultad de Ingeniería,
Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas*

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

1st Barrero Olgun A. P

Lic. en Ciencia de Datos

I.I.M.A.S - U.N.A.M.

Ciudad de México, México

patobarrero@gmail.com

2nd Bustamante Piza K.M.

#314135308.

Ingeniería en computación

Facultad de Ingeniería - U.N.A.M.

Ciudad de México, México

karla.bustamante@ingenieria.unam.mx

Martínez Ostoa N.I.

#315618648

Lic. Ciencia de Datos

I.I.M.A.S - U.N.A.M.

Ciudad de México, México

nestor.martinez@iimas.unam.mx

4th Ramírez Bondi J. A.

#314634825

Lic. Ciencia de Datos — Ing. en Computación

I.I.M.A.S. — Facultad de Ingeniería, UNAM

Ciudad de México, México

bondi@hey.com

5th Salas Mora M.

#314061481

Ingeniería en computación

Facultad de Ingeniería - U.N.A.M.

Ciudad de México, México

monica.salas@comunidad.unam.mx

Profesores:

Dr. Boris Escalante Ramírez

Dra. Olveres Montiel Jimena

Rosario Cruz

Mauricio Gómez

I.I.M.A.S. - U.N.A.M.

Resumen—En esta práctica se revisan conceptos básicos de procesamiento de imágenes; desde leer imágenes utilizando bibliotecas de Python como OpenCV, Scikit-Image, Pillow hasta manipulaciones de imágenes como decimación, rotación y transformaciones entre diferentes espacios de colores.

I. INTRODUCCIÓN

Para poder llevar un buen desarrollo en esta práctica es necesario tener claro ciertos conceptos. Recordemos que un script es una serie de comandos que se ejecutan de forma secuencial y se utilizan para automatizar el trabajo, este puede contener cualquier comando. De igual forma sirven para procesar estructuras de archivos complejas y hacer cálculos complejos en los campos de datos. A veces, los análisis más complejos solo se pueden hacer con un script. [1]

En la práctica se realizará un manejo básico de imágenes como lo es cambiar de color, rotar, escalar entre otras en donde se utilizará Python para su manejo pues este contiene bibliotecas que permiten realizar las trasformaciones en imágenes. Es necesario que la imagen se analice correctamente y se obtenga toda la información que pueda ser útil.

Según el sitio de RevistaDigital, una imagen es una matriz estándar de Numpy que contiene píxeles de puntos de datos. Cuanto mayor sea el número de píxeles en una imagen, mejor es su resolución. Para ser procesado por una computadora, una imagen debe convertirse en una forma binaria. [4]

Dentro de los paquetes que contiene Python para el manejo de imágenes están:

■ Sci-Py

Módulo científico de Python, es utilizado para las tareas básicas de modificación y arreglo de imágenes. Scipy suministra funciones que trabajan en matrices NumPy de ‘n’ dimensiones. Algunas de sus funciones es que sirve para el filtrado lineal y no lineal, mediciones de objetos, interpolación B-spline y morfología binaria.

■ Seikit-Image

Este es un paquete de código abierto de Python, usa algoritmos y se emplea en investigaciones, educación y algunas aplicaciones industriales. De igual forma, está dedicado al procesamiento de imágenes y usa matrices Numpy de forma nativa como objetos de imagen.

■ Matplotlib

Es de los paquetes más utilizados para gráficos 2D. Permite visualizar datos de forma muy rápida y figuras con calidad de publicación en varios formatos

■ OpenCV

Es una biblioteca de visión por computadora, actualmente la más grande en cuestión de funciones poseídas. Uno de sus usos más importantes es la detección de rostros y objetos, principalmente, en ámbitos como la fotografía, el marketing o la seguridad. Tiene interfaces para múltiples lenguajes, incluidos Python, Java y C++.

- **Python Image Library (PIL)**

Es una biblioteca para la manipulación de imágenes que se su desarrollo se ha estancado pues su último lanzamiento fue en el año 2009. Soporta una variedad de formatos, incluidos los más utilizados como GIF, JPEG y PNG. Una gran parte del código está escrito en C, por cuestiones de rendimiento. Actualmente existe una bifurcación llamada Pillow

II. DESARROLLO

El desarrollo de esta práctica consiste en 5 ejercicios relacionados con el procesamiento de imágenes y manipulación básica de estas. Las imágenes provienen de diversas fuentes y tienen los siguientes formatos:

- png
- tif
- dcm
- raw
- jpg
- bmp

II-A. 4.1

De la carpeta de imágenes: realiza las siguientes actividades.

II-A1. Desarrolla un script para leer y desplegar cada imagen con los paquetes de : Matplotlib, OpenCV, Scikit-Image, PIL y Sci-Py

Se anexan las imágenes leidas con los modulos, el script puede verse el cuaderno de python. No se incluye codigo porque es un poco extenso.



Figura 1: cameraman.tif



Figura 2: house.tif

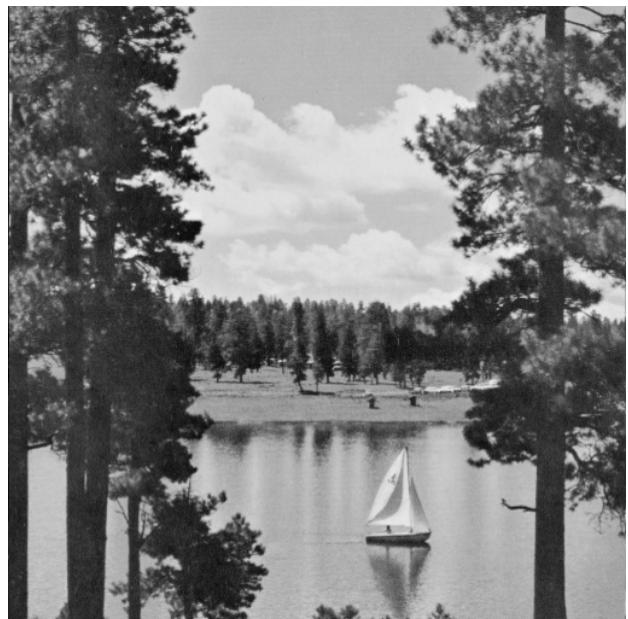


Figura 3: lake.tif



Figura 4: lena_color_512.tif



Figura 6: Anonymized20200210.dcm

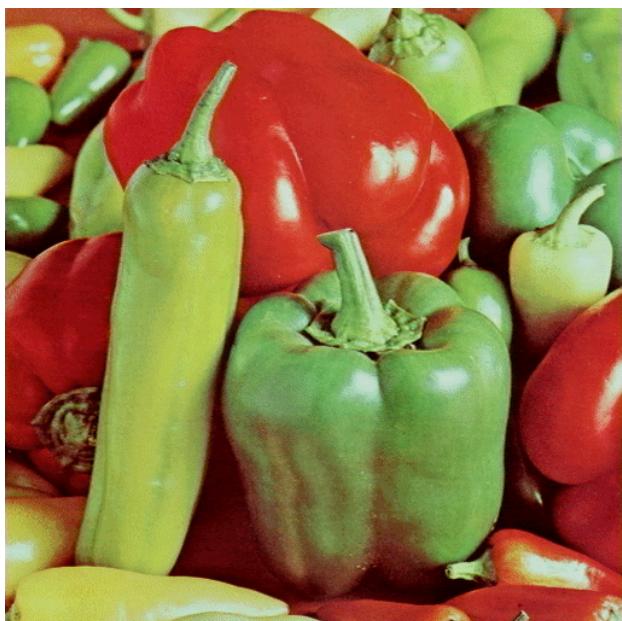


Figura 5: peppers_color.tif

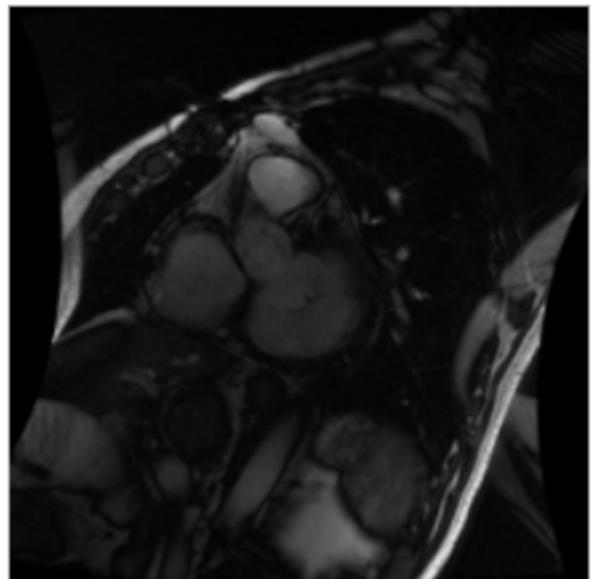


Figura 7: IM-0001-0007.dcm

Las siguientes imágenes dcm fueron leídas mediante un modulo de python llamado **pydicom** creado específicamente para la lectura de estas imágenes.

Las siguientes imágenes raw fueron leídas mediante **numpy**.



Figura 8: rosa800x600.dcm

II-A2. Imprimir el tipo de imagen, el tamaño y el tipo de dato: Para la imagen "lena_color_512.tif." obtuvimos los siguientes metadatos:

```
Type of Image = tiff
Size File = 262480 b
ImageWidth = (512,)
ImageLength = (512,)
BitsPerSample = (8,)
Compression = (1,)
PhotometricInterpretation = (1,)
ResolutionUnit = (1,)
ImageDescription = ("shape": [512, 512],)
StripOffsets = (336,)
Software = ('tifffile.py',)
SamplesPerPixel = (1,)
RowsPerStrip = (512,)
StripByteCounts = (262144,)
XResolution = ((1, 1),)
YResolution = ((1, 1),)
```

Utilizamos el modulo **os** para obtener el tamaño del archivo, el modulo **imghdr** para obtener el tipo de imagen y resto los obtuvimos con el modulo **PIL**.

II-A3. RGB a Escala de grises: Se convirtieron las imágenes de "lena_color_512.tif" "peppers_color.tif." a blanco y negro.



Figura 9: lena_color_512.tif convertida a blanco y negro desde cero

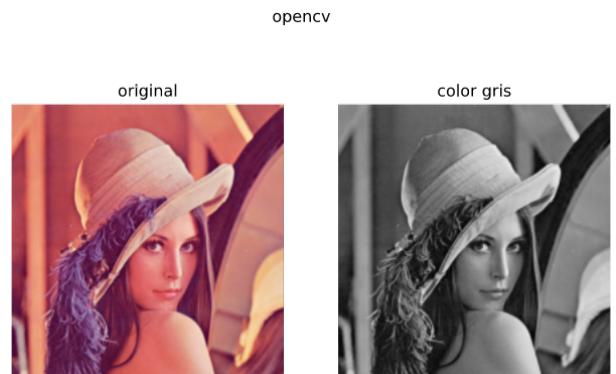


Figura 10: lena_color_512.tif convertida a blanco y negro usando opencv

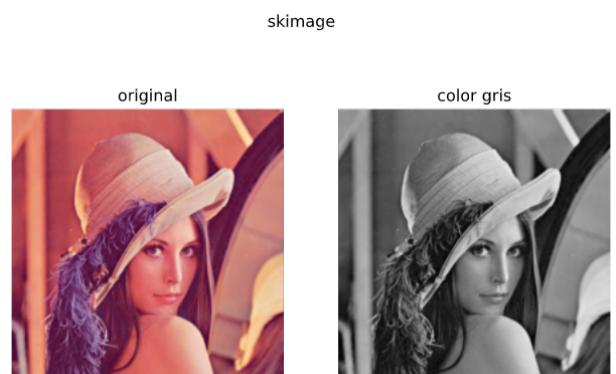


Figura 11: lena_color_512.tif convertida a blanco y negro usando skimage

Scratch

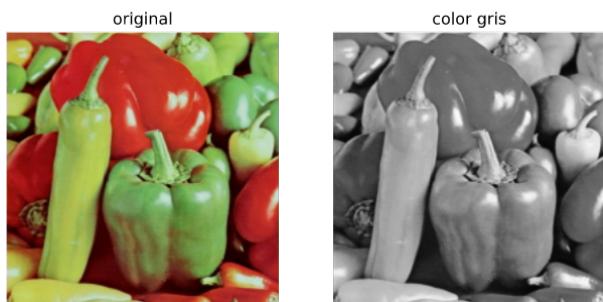


Figura 12: peppers_color.tif convertida a blanco y negro desde cero

opencv

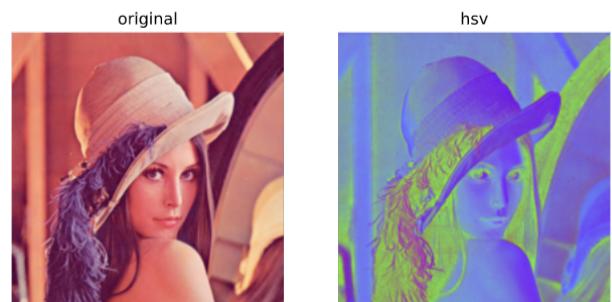


Figura 15: lena_color_512.tif convertida al modelo de color hsv usando opencv

opencv

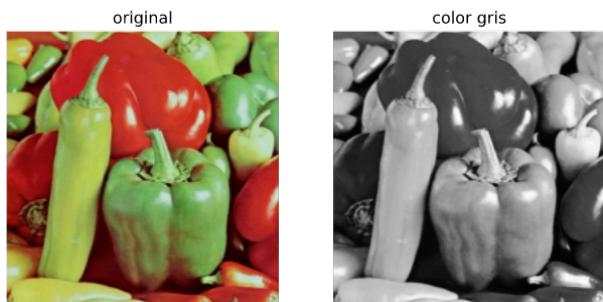


Figura 13: peppers_color.tif convertida a blanco y negro usando opencv

skimage

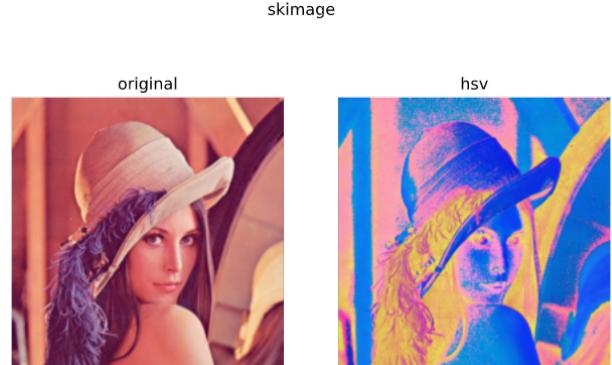


Figura 16: lena_color_512.tif convertida al modelo de color hsv usando skimage

skimage

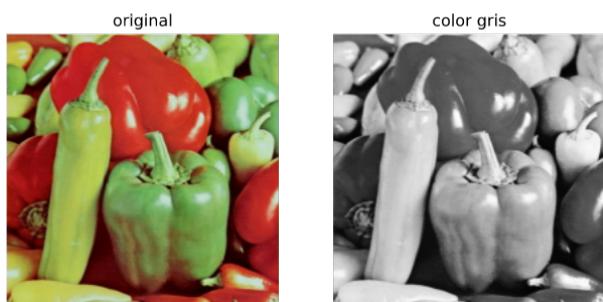


Figura 14: peppers_color.tif convertida a blanco y negro usando skimage

opencv



II-A4. *RGB a HSV*: Se convirtieron las imágenes de "lena_color_512.tif" "peppers_color.tif" al modelo de color HSV.

Figura 17: peppers_color.tif convertida al modelo de color hsv usando opencv

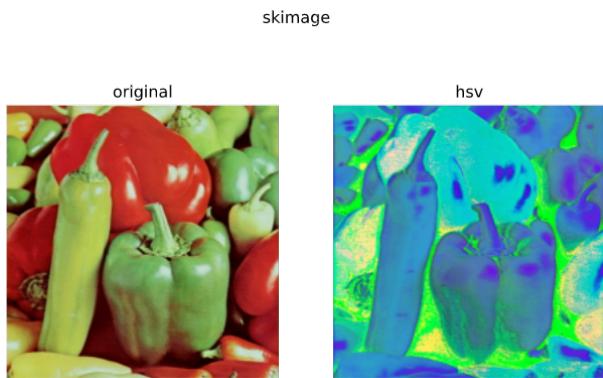


Figura 18: peppers_color.tif convertida al modelo de color hsv usando skimage

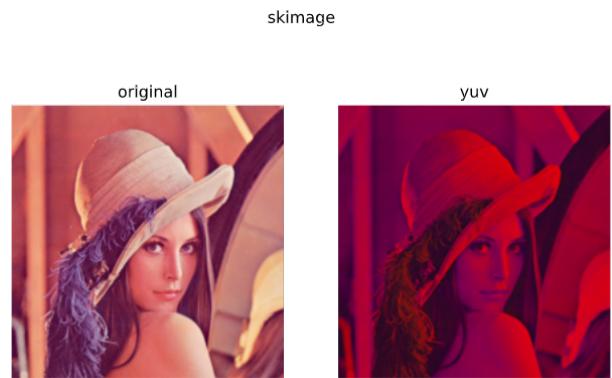


Figura 21: lena_color_512.tif convertida al modelo de color yuv usando skimage

II-A5. RGB a YUV: Se convirtieron las imágenes de "lena_color_512.tif" "peppers_color.tif". al modelo de color YUV.

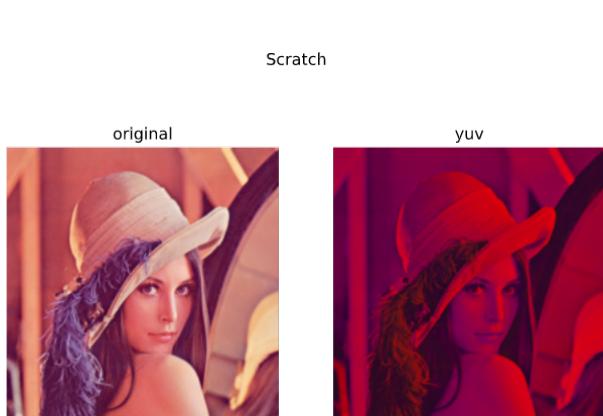


Figura 19: lena_color_512.tif convertida al modelo de color yuv con una implementación desde cero

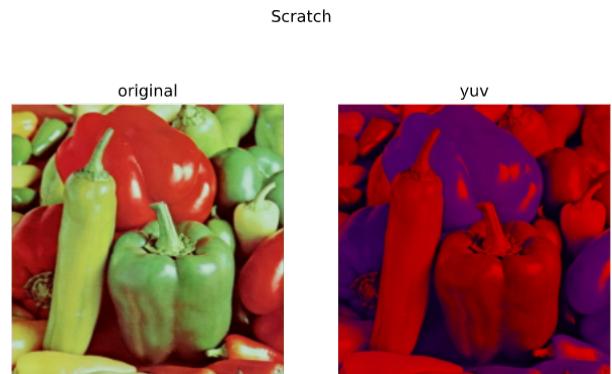


Figura 22: peppers_color.tif convertida al modelo de color yuv con una implementación desde cero



Figura 20: lena_color_512.tif convertida al modelo de color yuv usando opencv

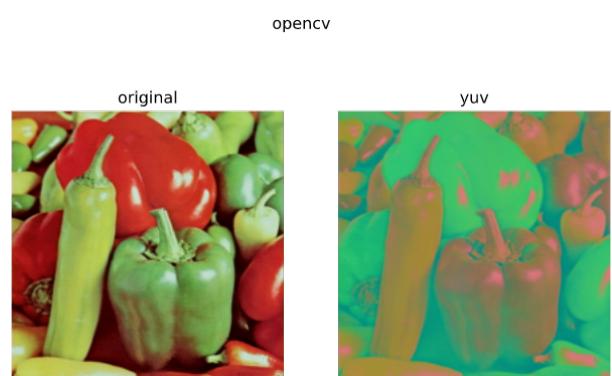


Figura 23: peppers_color.tif convertida al modelo de color yuv usando opencv

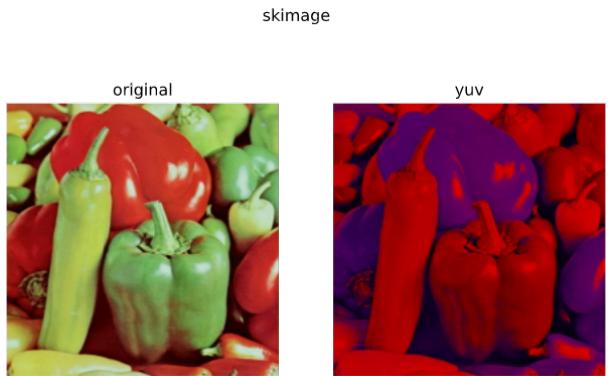


Figura 24: peppers_color.tif convertida al modelo de color yuv usando skimage



Figura 26: Paleta de color de peppers_color.tif en el modelo de color RGB

II-A6. Paleta de colores RGB: Se obtuvieron las paletas de colores de las imágenes "lena_color_512.tif" "peppers_color.tif" con el modelo de color RGB.

II-A7. Paleta de colores HSV: Se obtuvieron las paletas de colores de las imágenes "lena_color_512.tif" "peppers_color.tif" con el modelo de color HSV.

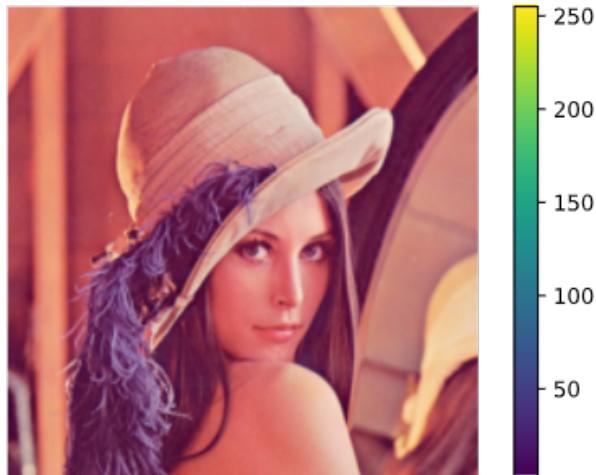


Figura 25: Paleta de color de lena_color_512.tif en el modelo de color RGB



Figura 27: Paleta de color de lena_color_512.tif en el modelo de color HSV

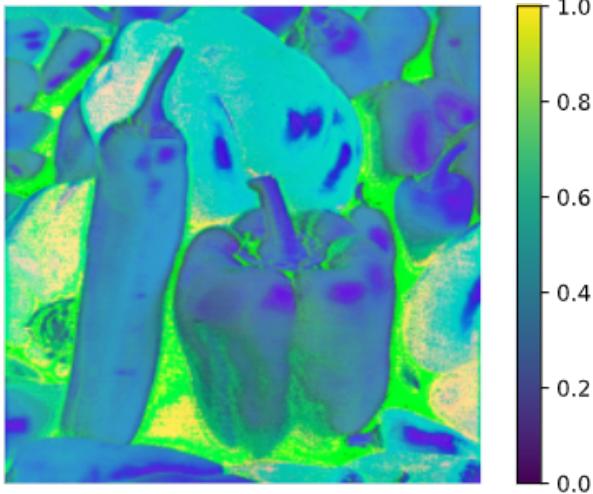


Figura 28: Paleta de color de peppers_color.tif en el modelo de color HSV

II-B. 4.2 Escala de grises y decimación

De una imagen elegida realice las siguientes operaciones:

II-B1. Transformación a escala de grises: Imagen original (RGB):

```
path = './resources'
image = Image.open(path + '/landscape.png')
show_image_plt(np.array(image))
```

Código para transformar de RGB a escala de grises:

```
gs_image = image.convert('L')
show_image_plt(np.array(gs_image), grayscale=True)
```

Los resultados de esta transformación se muestran en la figura 29



Figura 29: Misma imagen en RGB y escala de grises

Después de analizar los dos métodos, la diferencia sutil que pudimos observar en este caso es que para la imagen en RGB se tiene una matriz (concretamente un arreglo de numpy <ndarray>) de dimensiones (240, 240, 4). Mientras que para la imagen en escala de grises, ya no es una matriz en 3 dimensiones, sino una matriz de 2 dimensiones: (240, 240). En cuanto a los tamaños, la imagen en RGB contiene 230,400 bytes mientras que la imagen en escala de grises contiene



(a) Decimación de 2 pixeles

(b) Decimación de 4 pixeles

Figura 30: Decimación de la figura 29

57600 bytes. Esto se debe a que las imágenes en escala de grises solo tienen un valor entre 0 y 255, concretamente, el valor más pequeño encontrado fue 6 y el más alto 241; es decir, las imágenes en escala de grises ya no necesitan de un espacio de color en RGB, simplemente un valor numérico (numpy UInt8) para indicar la intensidad de los grises.

II-B2. Decimación: Programe una función que realice la decimación de una imagen, reduciéndola a la mitad de su tamaño original y promediando en grupos de 4 pixeles.

Tomando como referencia la misma imagen de la sección pasada (Figura 29) aplicamos una función de decimación encargada de promediar en grupos de 2 y 4 pixeles respectivamente. Lo que encontramos se presenta en la figura 30.

Lo que pudimos observar después de realizar la decimación es que, en efecto, las imágenes reducen en dimensiones. La imagen original tenía dimensiones de (240, 240) y al aplicar la decimación de 2 y 4 pixeles obtuvimos dimensiones de (120, 120) y (60, 60) respectivamente. Cuando hacemos la decimación de 2 (figura 30a) no es muy visible a reducción de pixeles, sin embargo, para la decimación de 4 (figura 30b) ya se puede notar que la imagen está modificada y no se aprecian bien los contrastes de los elementos de la imagen original (figura 29a).

II-C. 4.3 Reajuste de imágenes y rotaciones

II-C1. Reajuste de la carpeta 'Imagenes' 10x: Para reajustar el tamaño de las imágenes pertenecientes a esta carpeta utilizamos la función `resize(image, size)` del paquete PIL. El código general que usamos para realizar todo el proceso es el siguiente¹:

```
images = read_images(dir_path, verbose=True)
resized_images = resize_images(images, scale=10,
                               verbose=True)
save_images(resized_images, save_dir_path, verbose=True)
```

¹Revisar anexo IV-A

II-C2. Reajuste de la carpeta 'Imagenes' 3x: De igual manera que en la sección anterior, para hacer el reajuste de imágenes a 3 veces su tamaño original, empleamos la misma secuencia de código pero con una `scale=3` en la función `read_images` como se muestra a continuación:

```
images = read_images(dir_path, True)
resized_imgs = resize_images(images, scale=3,
                             verbose=True)
save_images(resized_imgs, '/home/nestor/Desktop/
rp_imgs/' + '/Imagenes_3x', verbose=True)
```

Los resultados tanto para la sección II-C1 como II-C2 se presentan en la siguiente tabla:

Nombre del archivo	Original [KB]	10× [MB]	3× [MB]
Anonymized20200210.dcm	263	21	6.3
cameraman.tif	263	2.6	0.785
house.tif	525	5.2	1.6
IM-0001-0007.dcm	140	5.2	1.6
lake.tif	525	5.2	1.6
lena_color_512.tif	787	7.9	2.4
peppers_color.tif	264	2.6	0.787
rosa800x600.raw	480	4.8	1.4

Cuadro I: Tamaño de archivos después del reajuste a 10× y 3× de su tamaño original

Como podemos observar en el cuadro I, los únicos valores que no coinciden de manera adecuada al escalarlos en 10 y 3 veces respectivamente son los archivos con extensión .dcm y esto se debe a que el proceso para escalarlos fue diferente a los demás archivos. Para poder leer los archivos .dcm tuvimos que emplear del paquete de Python `dycom`² la función `dcmread`, la cual lee archivos `dycom`. Esta función al leer archivos hace un escalamiento negativo. Para el archivo `IM-0001-0007.dcm` con tamaño 140,000 bytes originalmente, disminuyó a 65,000 bytes al leerlo con `dcmread`. Aunado a esto, para guardar los archivos `dcm` escalados (a 3 o 10 veces su tamaño original) ocupamos la función `tofile()` del paquete `numpy`. Esta función adicionalmente disminuye ligeramente la matriz que se le pasa de entrada. Los demás archivos no sufrieron de estos fenómenos pues pudieron ser leídos, escalados y guardados con el módulo `PIL` de Python.

II-C3. Rotación a 45°: Para las rotaciones definimos una función llamada `rotate`³ y tomamos como base la figura 29a para hacer las rotaciones.

Código:

```
image = Image.open(path + '/landscape.png')
print(type(image))
rotate(image, 45)
```

Imagen rotada: Ver figura 31

II-C4. Rotación a 90°: Código:

```
rotate(image, 90)
```

Imagen rotada: Ver figura 32



Figura 31: Figura rotada 45°



Figura 32: Figura rotada 90°

II-C5. Rotación a 180°: Código:

```
rotate(image, 90)
```

Imagen rotada: Ver figura 33

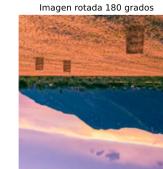


Figura 33: Figura rotada 180°

II-D. 4.4

II-D1. 4.4.1: Convierta la imagen `peppers_color.tif` a escala de grises y recórtela para que solo quede el pimiento verde de la imagen original.

```
img = Image.open('Imagenes/peppers_color.tif')
convert('LA')
print(img.size)
img.show()
```

Imagen original: Ver figura 34



Figura 34: Figura en color y formato original.

Imagen en escala de grises: Ver figura 35

A continuación, se muestra la imagen recortada con el siguiente código:

```
img_crop = img.crop((175, 175, 430, 500))
img_crop.show()
```

Imagen en escala de grises y recortada: Ver figura 36

²pip install dycom

³Ver Anexo IV-A



Figura 35: Figura en escala de grises.



Figura 36: Figura en escala de grises.

II-D2. 4.4.2: Guarde la imagen en formato png.

Es un comando sencillo el utilizado para hacer el recorte de la imagen.

```
img_crop.save('Imagenes/peppers_color_crop.png',  
'png')
```

II-E. 4.5

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.

Con el siguiente código en Python nos es posible convertir la imagen—de un formato raw el cual ni siquiera nos permite abrirlo en una aplicación de manejo de imágenes ya que el códec utilizado no es compatible con ningún estándar—a un arreglo de tipo *numpy array*. A partir de ahí, es posible visualizar la imagen gracias a las bibliotecas con las que contamos en Python.

```
image_raw = np.fromfile('Imagenes/rosa800x600.  
raw', np.uint8).reshape(800, 600)  
image = Image.fromarray(image_raw)  
image.show()
```

El resultado es el siguiente:

Se muestra imagen en formato RAW: Ver figura 37

III. CONCLUSIONES

III-A. Barrero Olgún Adolfo Patricio

Cuando realice esta practica no tenia mucha experiencia trabajando con imágenes, esta practica me ayudo a mejorar mis habilidades leyendo, mostrando y manipulando imágenes en distintos formatos.

Me encontré con ciertos conflictos como que la biblioteca *scipy* ya no da soporte a la lectura de imágenes, en su documentación recomienda usar el paquete *imageio*.



Figura 37: Figura en formato RAW después de la lectura de la misma.

III-B. Bustamante Piza Karla Mireli

Realizar esta práctica me sirvió para poder comprender o conocer las formas existentes para desplegar y manipular las imágenes en sus distintos formatos, incluso saber de lo qué es el formato RAW que es un arreglo de bits. Me permitió conocer muchas librerías que no tenía idea que existían en Python para poder lograr el procesamiento de estas imágenes, como lo fue el pasar a formato RGB, escalar o redimensionar, obtener la escala de grises, por mencionar algunas de las actividades obtenidas.

En general, elaborar la práctica me resultó un tanto compleja pues realmente nunca había realizado algo de este estilo por lo que fue completamente nuevo para mí la mayoría las funciones empleadas, dificultando un poco un correcto desarrollo en esta.

III-C. Martínez Ostoa Néstor Iván

Lo que pude observar de los ejercicios en las secciones II-B y II-C lo más complicado fue realizar la función de decimación pues computacionalmente hablando si no la programábamos bien, la complejidad pudiera ser muy mala. Concretamente, y como se mencionaba en la sección II-B, hasta realizar la decimación de 4 pixeles pudimos observar una reducción y difuminado en los pixeles, con 2 pixeles, todavía se aprecian con detalle los elementos de la imagen. Finalmente, para la sección II-C, el método que se empleo para ampliar las imágenes una cantidad n de veces dio muy buenos resultados (ver cuadro I) para todos los formatos de archivos excepto para los archivos *dycom*. Esto se debe a que estos archivos contienen información extra a la pura imagen y esto provoca que al incrementar el tamaño del archivo, no se haga de la manera esperada.

III-D. Ramírez Bondi, J. Alejandro

Como tal, las imágenes son un instrumento complejo que hemos logrado pasar al plano de la computación mediante diversas técnicas que nos permiten almacenarlas, procesarlas y representarlas a través de una pantalla o, incluso, en un formato físico gracias a las impresoras. Lo anterior, es algo

que ya consideramos como elementos ubicuos de la vida cotidiana. La explosión de aplicaciones o plataformas de socialización como Instagram o Twitter nos han permitido y facilitado la creación y envío de este tipo de objetos visuales. Por lo anterior, es aún más importante poder contar con el conocimiento del uso de herramientas que nos permitan transformarlas lo suficiente para poder analizarlas, sobre todo como ingenieros o científicos de datos. Durante el desarrollo de esta práctica, podemos concluir que hemos desarrollado las suficientes habilidades, tanto técnicas como de análisis, que nos permitirán transformar cualquier imagen—sea cual sea el formato en el que se encuentren—para utilizarlas dentro de modelos o herramientas que nos permitan reconocer patrones dentro de ellas. Por lo tanto, hemos cumplido con los alcances propuestos previo a la realización de la misma y contamos con las herramientas para las siguientes actividades.

En particular, durante el trabajo nos encontramos con que la gran mayoría de los ejercicios estaban acotados—en cuanto al dominio de las imágenes refiere—por el formato o estándar utilizado en su almacenamiento y representación. En cambio, al procesar la imagen en formato RAW, nos encontramos con que esta está simplemente bajo un formato de arreglo y que los bits de representación visual están acomodados de la misma manera. Es así que nuestra tarea fue la de encontrar una manera de poder pasar de dicho dominio original a uno que fuera utilizado por la biblioteca de Python para mostrar la visualización de la imagen. Lo anterior, considero que fue uno de los mayores retos y aprendizajes encontrados.

III-E. Salas Mora Mónica

En esta práctica logramos conocer distintos métodos para poder desplegar una o varias imágenes en Python, con lo cual aprendimos que no solo es necesario utilizar software como Matlab, sino que en lenguajes de programación esto también es posible, haciéndolo un recurso de gran utilidad en distintas áreas de la computación. Cada uno de los paquetes utiliza instrucciones y se despliegan las imágenes de forma distinta, de esta manera podemos elegir la forma que más nos convenga o nos facilite el proceso. Aprendí las distintas formas de manipular o modificar una imagen como, pasar una imagen de RGB(Rojo, Verde y Azul) a escala de grises, o a otros formatos como YUV o HSV, a escalar y redimensionar una imagen, a rotarla en distintos ángulos, y a almacenar estas nuevas imágenes manipuladas y modificadas, entre otras cosas.

IV. ANEXO

IV-A. Funciones

- `show_image_plt(image, grayscale=False)`
 - Descripción: Función encargada de mostrar una imagen usando matplotlib
 - Parámetros:
 - `image : ndarray`: Numpy array que representa los pixeles de la imagen

- `grayscale : Boolean`: Bandera que indica si la imagen de desplegará en una escala de grises
- Regresa:
 - `None`
- `decimation(image, points)`
 - Descripción: Función encargada de realizar la decimación de imagen una cantidad n (points) de veces
 - Parámetros:
 - `image : ndarray`: Numpy array que representa los pixeles de la imagen
 - `points : int`: Entero indicando el factor de reescalamiento hacia abajo de una imagen
 - Regresa:
 - `dec_image : ndarray`: Numpy array que representa la imagen original decimada
- `read_images(dir_path, verbose=False)`
 - Descripción: asf
 - Parámetros:
 - Regresa:
- `rotate(image, degrees, gs=False)`
 - Descripción: Función encargada de rotar y desplegar una imagen una cierta cantidad de grados
 - Parámetros:
 - `image : PngImageFile`: Imagen en el formato del paquete PIL
 - `degrees : double`: Real entre 0 y 360
 - `gs : Boolean (default = False)`: Bandera que controla si la imagen se despliega en escalas de grises
 - Regresa:
 - `None`

REFERENCIAS

- [1] ¿Qué es un script? Sitio web. Última vez consultado el 09 de marzo del 2021 de: https://help.hightbond.com/helpdocs/analytics/141/scripting-guide/es/Content/how_to_script/complete_beginners/what_is_a_script.htm
- [2] Instalar PIL / Pillow y aplicar efectos visuales. Sitio web (2014). Última vez consultado el 09 de marzo del 2021 de, <https://recursospthon.com/guias-y-manuales/instalar-pil-pillow-efectos/>
- [3] Manipulación de imágenes Sitio web (). Última vez consultado el 09 de marzo del 2021 de, <https://python-guide-es.readthedocs.io/es/latest/scenarios/imaging.html>
- [4] ¿Qué es OpenCV? Instalación en Python y ejemplos básicos. Revista Digital. Sitio web (2020). Última vez consultado el 09 de marzo del 2021 de, <https://revistadigital.inesem.es/informatica-y-tics/opencv/>
- [5] 1.4. Matplotlib: Gráficas usando pylab. Sitio web. Última vez consultado el 09 de marzo del 2021 de, <https://claudioz.github.io/scipy-lecture-notes-ES/intro/matplotlib/matplotlib.html>
- [6] Gouillart, Emmanuelle. 3.3. Scikit-image: procesamiento de imágenes. Sitio web. Última vez consultado el 09 de marzo del 2021 de, <http://scipy-lectures.org/packages/scikit-image/index.html>
- [7] 10 librerías para manipular imágenes en Python. Sitio web. Última vez consultado el 09 de marzo del 2021 de, <https://unipython.com/10-librerias-para-manipular-imagenes-en-python/>