**SEGMENTACIÓN DE IMAGEN**

Para este ejercicio se le pide que se realicen ciertas operaciones sobre una imagen que se carga, la imagen queda a disposición del estudiante. La idea es entonces usar las funcionalidades de Numpy para realizar las siguientes operaciones sobre la imagen:

* Dividir la imagen en 16 cuadrantes de igual dimensión.
* Asignar la siguiente distribución de colores a cada cuadrante.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | No modificar | No modificar |  |
|  | No modificar | No modificar |  |
|  |  |  |  |

Tenga en cuenta que cuando se lee una imagen esta se almacena como una matriz de 3x3, donde las dimensiones *x* e *y* representan la cantidad de pixeles de esta y la *z* los colores RGB.

Horiz Px

Vert Px

**Solución**

Durante la ejecución del ejercicio se obtuvo lo siguiente:

|  |  |
| --- | --- |
| A picture containing mountain, water, sky, outdoor  Description automatically generated | A picture containing text  Description automatically generated |

La lógica empleada lo que hace es recorrer la imagen por sectores por medio de *slicing*. Este recorrido se hace de forma automática por medio de dos ciclos for anidados, que permiten recorrer la imagen en sentido horizontal y vertical. Como en total hay 16 cuadrantes, los for anidados deben generar en total 16 iteraciones. Esto último entonces establece que cada for recorrerá 4 estados del 0 al 3. Estos estados serán los que establezcan los límites del *slicing* usado.

Para la generación de los sectores en blanco y negro se debe tener en cuenta que normalmente una imagen en blanco y negro solo tiene 2 dimensiones, por lo que es necesario generar una matriz de 3D donde las capas asociadas a los colores rojo, verde y azul contenga la matriz de grises calculada.

**Módulos usados:**

****

Inicialmente se importan los paquetes pyplot, image y Numpy. El primero es usado para visualizar y guardar la imagen, el segundo para leer la imagen y el último para procesamiento de arrays.

También se implementa una función llamada rgb2gray, esta tiene el fin de convertir la imagen rgb a una imagen en blanco y negro de 3 dimensiones. Para esto inicialmente se genera la imagen en blanco y negro asociada a la imagen en rgb, esto se hace por medio de la sentencia establecida en la capa1 (En esta se aplica la operación usada para convertir de rgb a gris). Luego esta matriz se almacena en cada uno de los canales de cada color de la matriz que se retorna (aGris). Se debe tener en cuenta que el array donde se almacena la imagen es de tipo uint8 (Entero de 8bits), esto para que durante el proceso de visualización la imagen se muestre adecuadamente.

**Inicialización de variables**

Las líneas de código siguientes muestran la inicialización de las variables usadas durante el proceso de generación de la nueva imagen. La variable img contiene la matriz de 3 dimensiones asociada a la imagen leída. La variable nuevaImagen tiene asignada una matriz de ceros que tiene la misma dimensión que la imagen leída, este array a su vez es de tipo uint8, por lo dicho anteriormente. En esta última variable es que se almacenará los valores numéricos asociados a la nueva imagen que se creará en los ciclos for.



La variable color especifica qué color se asociará al cuadrante de imagen analizado. Este orden se establece según cómo es recorrida la imagen.

Los for anidados se implementaron de forma que se recorren todas las columnas asociadas a una fila primero y luego se sigue a la siguiente fila. De manera que:

Para i=0

|  |  |  |  |
| --- | --- | --- | --- |
| j=0 | j=1 | j=2 | j=3 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Para i=1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| j=0 | j=1 | j=2 | j=3 |
|  |  |  |  |
|  |  |  |  |

Para i=2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
| j=0 | j=1 | j=2 | j=3 |
|  |  |  |  |

Para i=3

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| j=0 | j=1 | j=2 | j=3 |

Bajo este orden establecido, los colores se ingresarán en cada sector de la siguiente forma: rojo, azul, azul, rojo, gris, gris, normal, normal, gris, gris, normal, normal, gris, verde, azul, azul y verde. Que codificando bajo el siguiente esquema: rojo: 1, azul:2, gris:3, verde:4 y normal:5; quedaría 🡪 [1, 2, 2, 1, 3, 5, 5, 3, 3, 5, 5, 3, 4, 2, 2, 4], la cual es la lista color.

Por último, se inicializa la variable posición, la cual se encarga de recorrer la lista color y por medio de esto definir qué color se aplicará en el cuadrante.

**Generación de nueva imagen**



En el código anterior se muestran los for anidados y la respectiva lógica aplicada para generar la nueva imagen.

* En amarillo está resaltado la imagen copia que se genera, la cual será la que se modifica en función del color que se desee establecer. Esta copia se renueva en cada iteración, ya que cada una de estas equivale a un cuadrante nuevo.
* Luego se establecen los límites que definen a un determinado cuadrante, para esto se usan los iteradores y la cantidad total de columnas y filas. Las fórmulas son sencillas de deducir si se tiene cuenta que en total las filas están dividas en 4 al igual que las columnas, y en cada iteración se aumenta una unidad.
* En azul se resalta la lógica empleada para determinar qué operación se realizará sobre la imagen copiada. En función del valor evaluado en color[posición] se decide qué elif evaluar y por lo tanto, qué realizar sobre la imagen copiada. Si se evalúa el color rojo, lo que se hace es poner en 0 las máscaras asociadas al color azul y verde. Si se evalúa el color azul, se colocan en cero las máscaras asociadas al rojo y verde. Si se evalúa el color verde, se ponen en cero las máscaras rojo y azul. Si se evalúa el color gris, se llama la función mencionada anteriormente. Se debe tener en cuenta que estas operaciones se realizan sobre los sectores evaluados.
* Por último, en verde se muestra cómo se almacena en el array que contenía ceros inicialmente, los diferentes sectores nuevos generados.

Por último, se guarda la imagen, para esto se hace lo siguiente:



La función imshow de pyplot muestra la imagen en patalla.