Tratamiento de Imagen Digital
Portafolio Nº1

ALEJANDRO ALBERT GRAMAJE

ÍNDICE

1. Cambio de espacios de Color

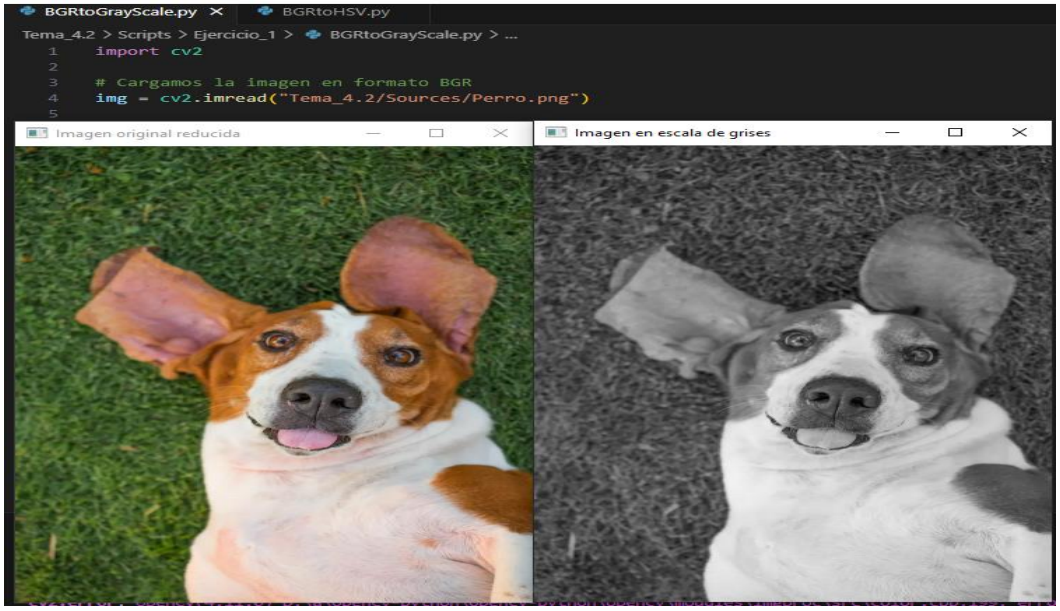
Ejemplo 1: Conversión de una imagen BGR a escala de grises

Se carga una imagen en color (BGR), se convierte a escala de grises usando cv2.cvtColor, eliminando la información de color y dejando solo la intensidad luminosa. Después se guarda y se muestra la imagen resultante.

Código:

```python
import cv2

# Cargamos la imagen en formato BGR
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Convertimos la imagen a escala de grises
img_gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Guardamos la imagen en escala de grises en un archivo nuevo
cv2.imwrite("imagen_gris.jpg", img_gris)

# Mostramos la imagen original y la imagen en escala de grises
cv2.imshow("Imagen original", img)
cv2.imshow("Imagen en escala de grises", img_gris)

# Esperamos a que el usuario presione una tecla para cerrar las ventanas
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



Ejemplo 2: Conversión de una imagen BGR a HSV

Se convierte la imagen cargada de BGR a HSV, por el cual se separa los canales de tono, saturación y brillo. Esto permite trabajar mejor con colores en distintas condiciones. Finalmente, se guarda y muestra la imagen HSV.

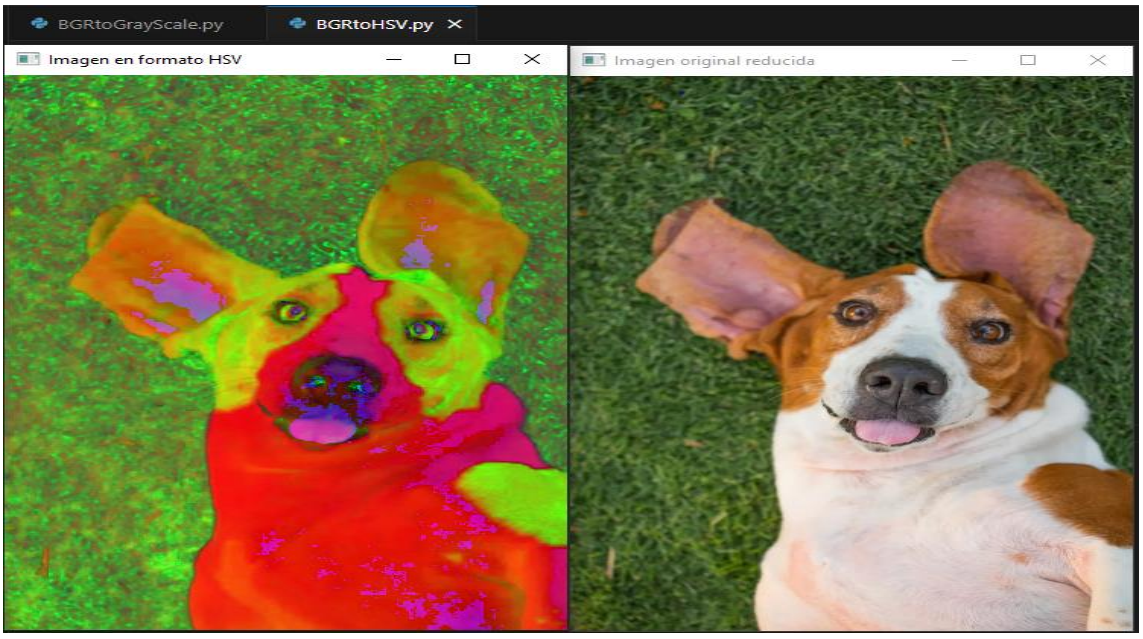
Código:

BGRtoGrayScale.py | BGRtoHSV.py

Tema_4.2 > Scripts > Ejercicio_1 > BGRtoHSV.py > ...

```python
import cv2

# Cargamos la imagen en formato BGR
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Convertimos la imagen a formato HSV
img_hsv = cv2.cvtColor(img_reducida, cv2.COLOR_BGR2HSV)

# Guardamos la imagen en formato HSV en un archivo nuevo
cv2.imwrite("imagen_hsv.jpg", img_hsv)

# Mostramos la imagen original reducida y la imagen en formato HSV
cv2.imshow("Imagen original reducida", img_reducida)
cv2.imshow("Imagen en formato HSV", img_hsv)

# Esperamos a que el usuario presione una tecla para cerrar las ventanas
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado;



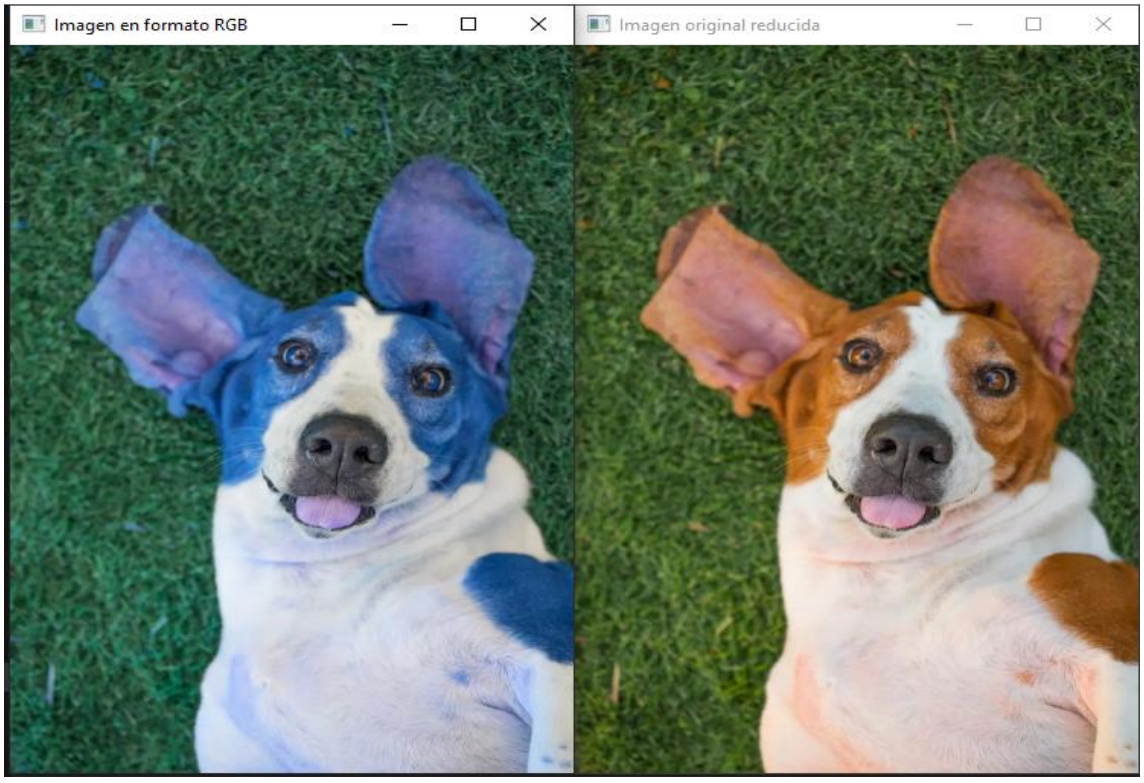
Ejemplo 3: conversión de una imagen BGR a RGB

Se carga una imagen en formato BGR y se convierte al espacio de color RGB usando cv2.cvtColor. Esto reordena los canales de color para que coincidan con el formato estándar RGB. Luego se guarda y se muestran ambas imágenes.

Código:

```python
import cv2

# Cargamos la imagen en formato BGR
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Convertimos la imagen a formato RGB
img_rgb = cv2.cvtColor(img_reducida, cv2.COLOR_BGR2RGB)

# Guardamos la imagen en formato RGB en un archivo nuevo
cv2.imwrite("imagen_rgb.jpg", img_rgb)

# Mostramos la imagen original reducida y la imagen en formato RGB
cv2.imshow("Imagen original reducida", img_reducida)
cv2.imshow("Imagen en formato RGB", img_rgb)

# Esperamos a que el usuario presione una tecla para cerrar las ventanas
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



2. Transformaciones geométricas en imágenes

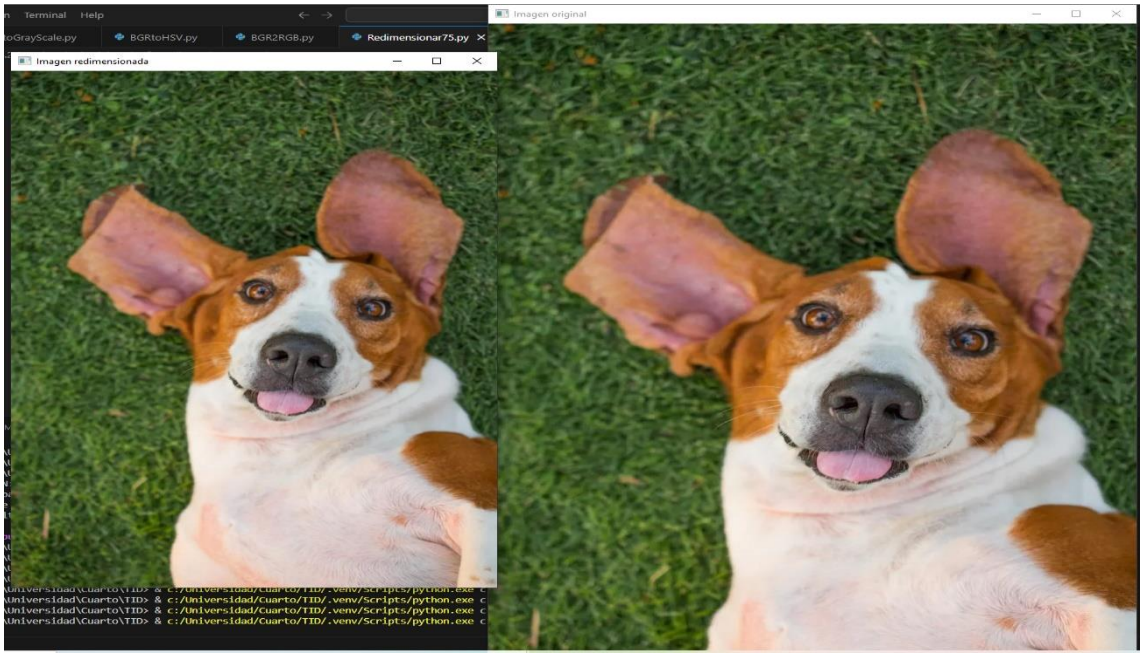
Ejemplo 1: Redimensionar una imagen al 75%

Se carga una imagen y se reduce su tamaño al 75% usando cv2.resize. Esta operación ajusta el ancho y el alto proporcionalmente. Finalmente, se guarda y se muestra la imagen escalada.

Código:

```python
import cv2

# Cargamos la imagen
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Obtenemos las dimensiones de la imagen
alto, ancho = img.shape[:2]

# Redimensionamos la imagen al 75%
img_redimensionada = cv2.resize(img, (int(ancho * 0.75), int(alto * 0.75)))

# Guardamos la imagen redimensionada en un archivo nuevo
cv2.imwrite("imagen_redimensionada.jpg", img_redimensionada)

# Mostramos la imagen original y la redimensionada
cv2.imshow("Imagen original", img)
cv2.imshow("Imagen redimensionada", img_redimensionada)

# Esperamos a que el usuario presione una tecla para cerrar las ventanas
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



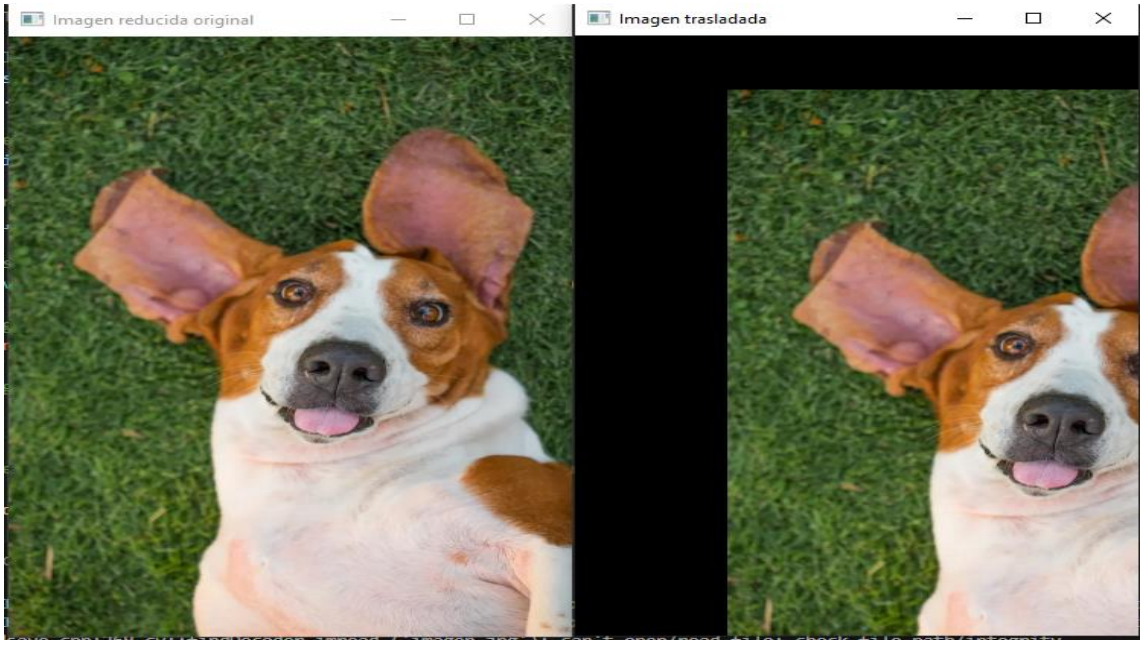
Ejemplo 2: Trasladamos una imagen (100,50)

Se aplica una traslación a la imagen, desplazándola 100 píxeles a la derecha y 50 hacia abajo. Esto se logra usando una matriz de transformación con cv2.warpAffine. Luego se guarda y se muestra el resultado.

Código:

```python
import cv2
import numpy as np

# Cargamos la imagen
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Nuevas dimensiones
alto_r, ancho_r = img_reducida.shape[:2]

# Definimos la matriz de transformación para la traslación
M = np.float32([[1, 0, 100], [0, 1, 50]])

# Aplicamos la traslación a la imagen
img_trasladada = cv2.warpAffine(img_reducida, M, (ancho_r, alto_r))

# Guardamos la imagen trasladada en un archivo nuevo
cv2.imwrite("imagen_trasladada.jpg", img_trasladada)

# Mostramos la imagen reducida original y la trasladada
cv2.imshow("Imagen reducida original", img_reducida)
cv2.imshow("Imagen trasladada", img_trasladada)
```

Resultado:



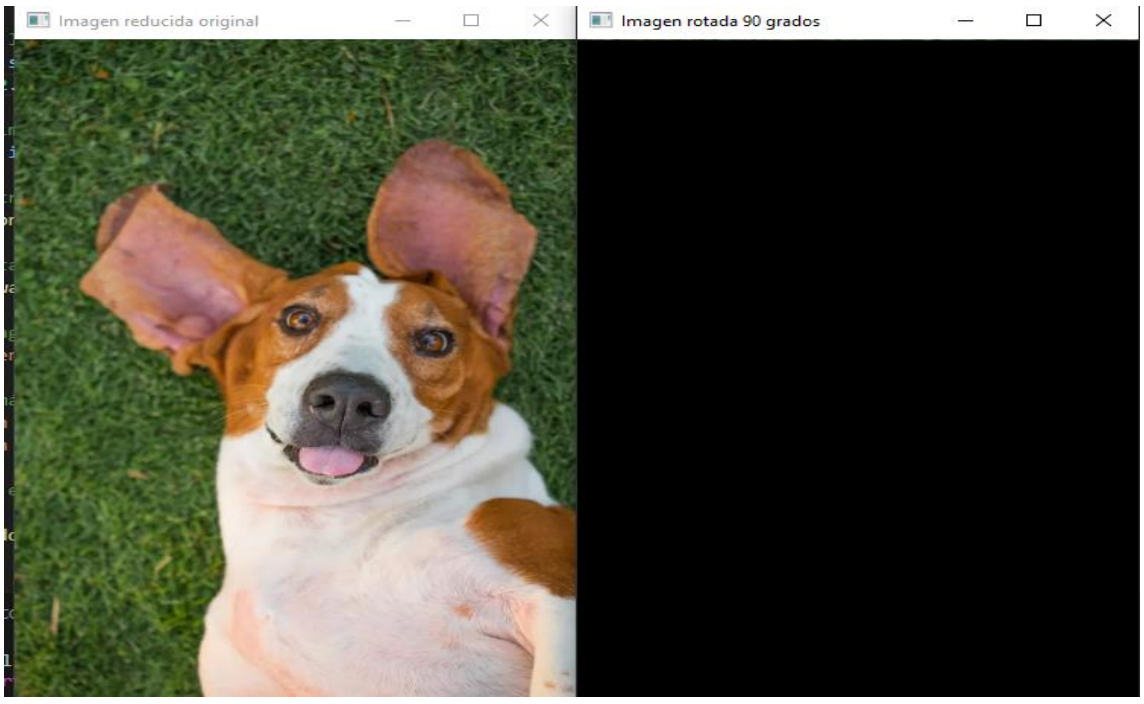
Ejemplo 3: Rotamos una imagen 90 grados desde el ángulo superior izquierdo

Se rota una imagen 90 grados en sentido antihorario tomando como centro el vértice superior izquierdo. Por tanto, la imagen queda fuera del "recuadro" de la imagen, ya que esta justo encima.

Código:

```python
# Cargamos la imagen
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Obtenemos las dimensiones de la imagen reducida
alto_r, ancho_r = img_reducida.shape[:2]

# Definimos la matriz de rotación (90° antihorario desde la esquina superior izquierda)
M = cv2.getRotationMatrix2D((0, 0), 90, 1)

# Aplicamos la rotación
img_rotada = cv2.warpAffine(img_reducida, M, (ancho_r, alto_r))

# Guardamos la imagen rotada
cv2.imwrite("imagen_rotada.jpg", img_rotada)

# Mostramos las imágenes
cv2.imshow("Imagen reducida original", img_reducida)
cv2.imshow("Imagen rotada 90 grados", img_rotada)
```

Resultado:



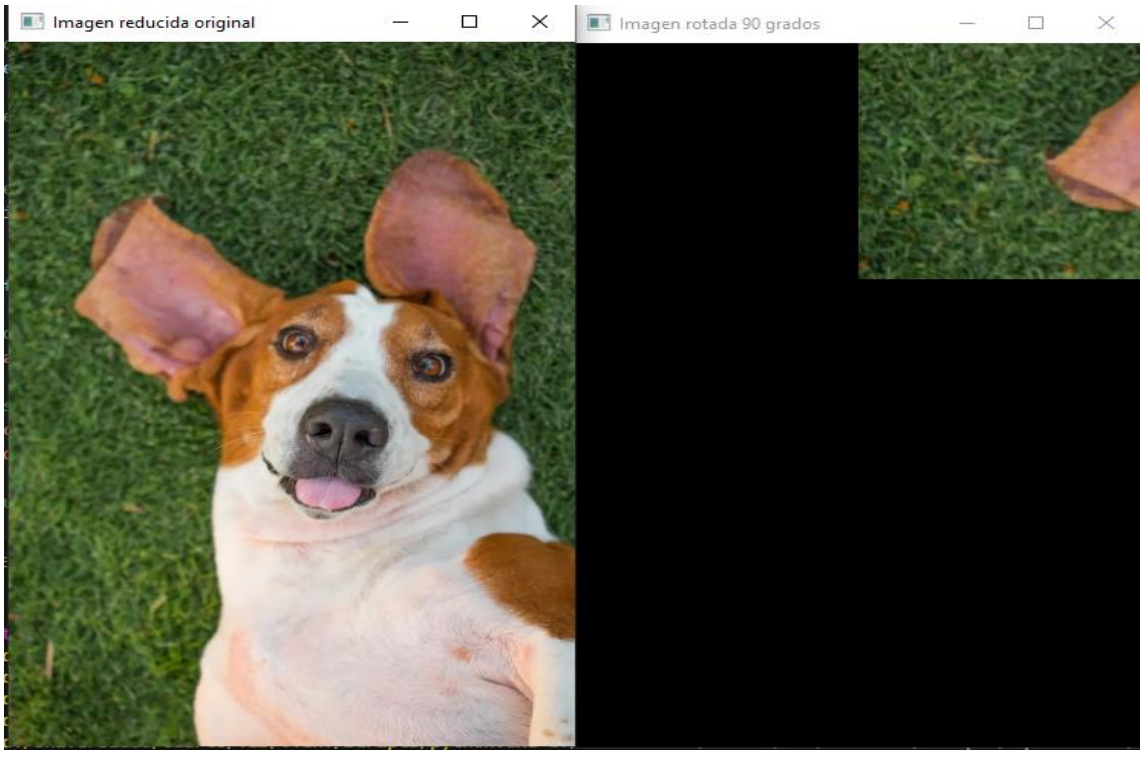
Ejemplo 4: Rotamos una imagen 90 grados desde el centro de la parte superior

Se rota la imagen 90 grados en sentido antihorario tomando como centro el punto medio de la parte superior. Se divide el ancho de la imagen de la imagen y el alto se pone a 0 para encontrar el centro de la parte superior de la imagen.

Código:

```python
# Cargamos la imagen
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Obtenemos las dimensiones de la imagen reducida
alto_r, ancho_r = img_reducida.shape[:2]

# Definimos el centro de la parte superior
centro = (ancho_r // 2, 0)

# Definimos la matriz de rotación (90° antihorario desde el centro superior)
M = cv2.getRotationMatrix2D(centro, 90, 1)

# Aplicamos la rotación
img_rotada = cv2.warpAffine(img_reducida, M, (ancho_r, alto_r))

# Guardamos la imagen rotada
cv2.imwrite("imagen_rotada.jpg", img_rotada)

# Mostramos las imágenes
cv2.imshow("Imagen reducida original", img_reducida)
cv2.imshow("Imagen rotada 90 grados", img_rotada)
```

Resultado:



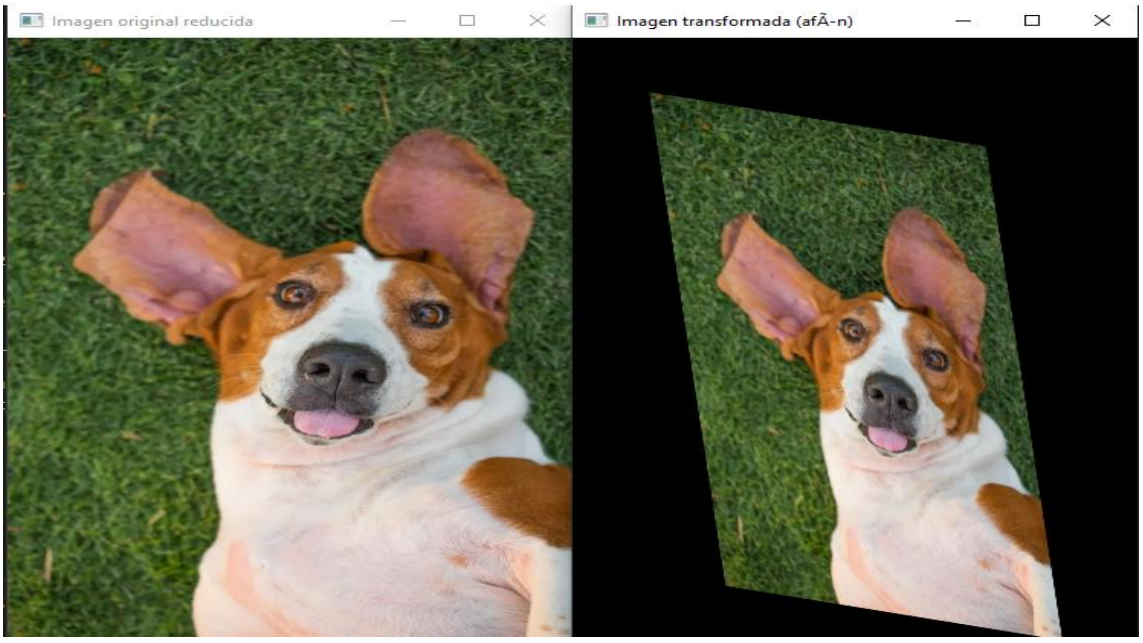
Ejemplo 5: Transformación afín de una imagen

Se aplica una transformación afin, donde se combina tanto la operación de traslacion, como la de rotacion y escalado. Se define usando 3 puntos de referencia para la transformacion. En este caso se han utilizado 3 puntos de origen y otros 3 de destino.

Código:

```python
# Cargamos la imagen
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Nuevas dimensiones
alto_r, ancho_r = img_reducida.shape[:2]

# Definimos tres puntos de origen y tres puntos de destino
src_pts = np.float32([[0, 0], [ancho_r - 1, 0], [0, alto_r - 1]])
dst_pts = np.float32([[50, 50], [ancho_r - 100, 100], [100, alto_r - 50]])

# Obtenemos la matriz de transformación afín
M = cv2.getAffineTransform(src_pts, dst_pts)

# Aplicamos la transformación afín
img_afín = cv2.warpAffine(img_reducida, M, (ancho_r, alto_r))

# Guardamos la imagen transformada
cv2.imwrite("imagen_transformada.jpg", img_afín)

# Mostramos las imágenes
cv2.imshow("Imagen original reducida", img_reducida)
cv2.imshow("Imagen transformada (afín)", img_afín)
```

Resultado:



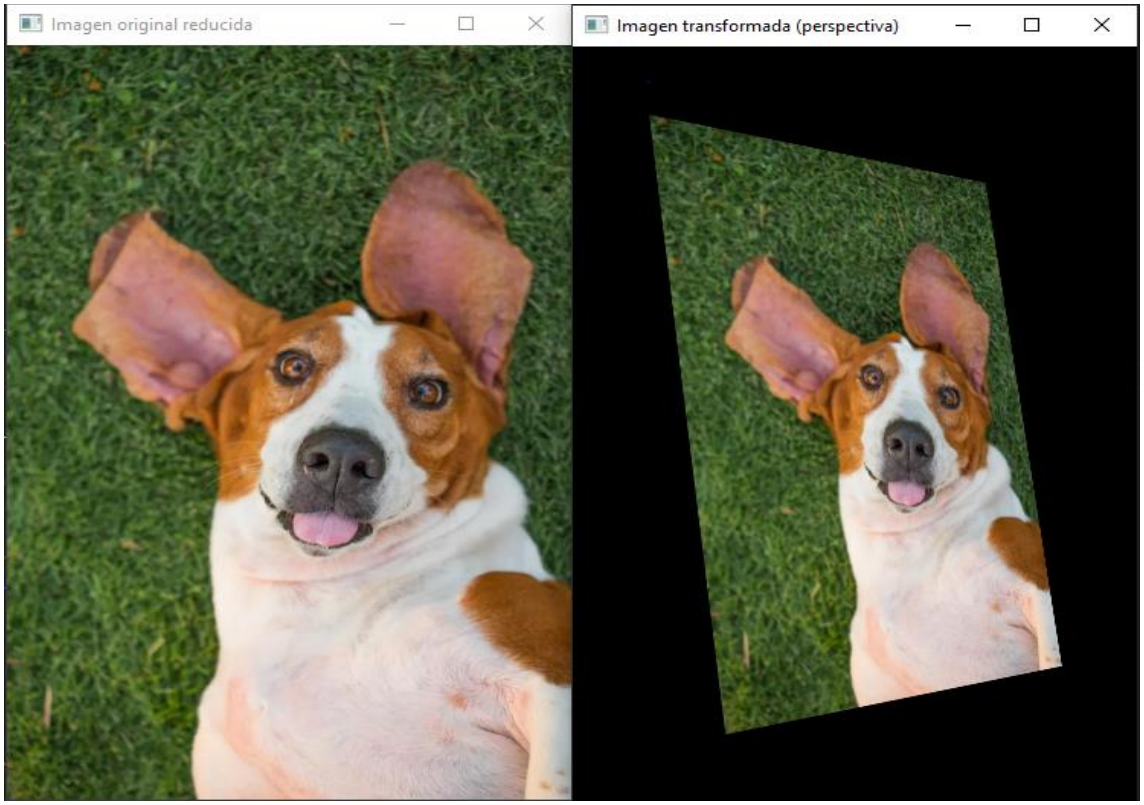
Ejemplo 6: Transformación en perspectiva

Se aplica una transformación en perspectiva que simula un cambio en el ángulo de visión de la imagen. Se usan 4 puntos de referencia para definir cómo se deformará la imagen, permitiendo representar profundidad y convergencia de líneas.

Código:

```python
# Cargamos la imagen
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Nuevas dimensiones
alto_r, ancho_r = img_reducida.shape[:2]

# Definimos cuatro puntos de origen y destino para la transformación
src_pts = np.float32([[0, 0], [ancho_r - 1, 0], [0, alto_r - 1], [ancho_r - 1, alto_r - 1]])
dst_pts = np.float32([[50, 50], [ancho_r - 100, 100], [100, alto_r - 50], [ancho_r - 50, alto_r - 100]])

# Obtenemos la matriz de transformación en perspectiva
M = cv2.getPerspectiveTransform(src_pts, dst_pts)

# Aplicamos la transformación en perspectiva
img_perspectiva = cv2.warpPerspective(img_reducida, M, (ancho_r, alto_r))

# Guardamos la imagen transformada
cv2.imwrite("imagen_transformada.jpg", img_perspectiva)

# Mostramos las imágenes
cv2.imshow("Imagen original reducida", img_reducida)
cv2.imshow("Imagen transformada (perspectiva)", img_perspectiva)
```

Resultado:



3. Umbralización de imágenes

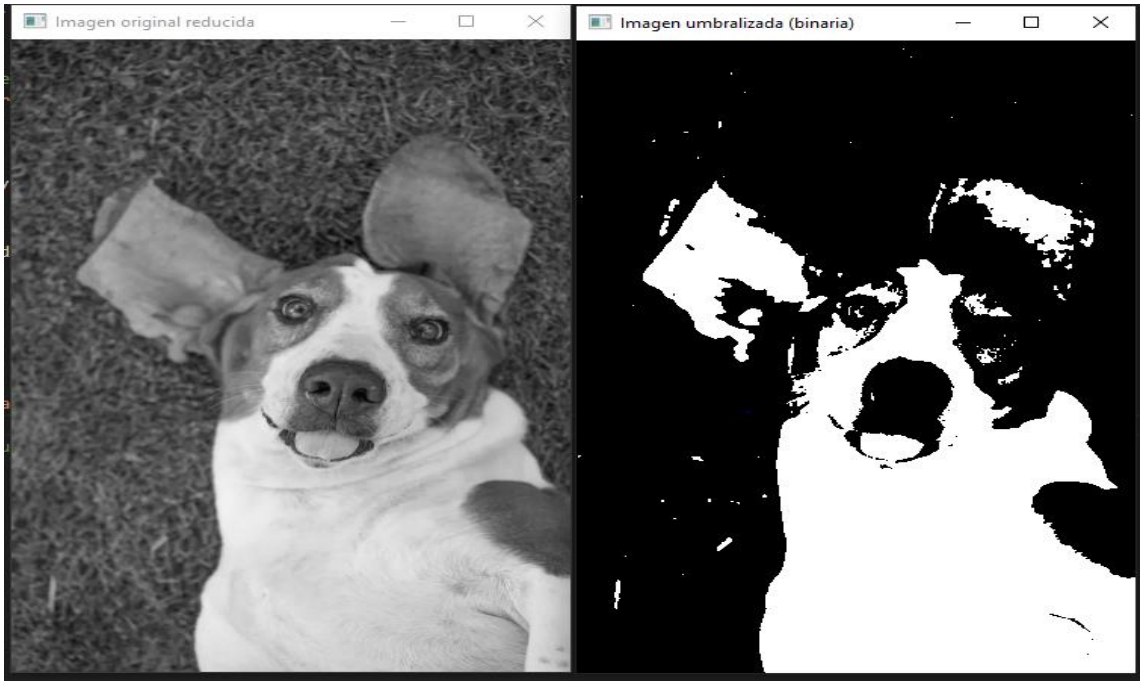
Ejemplo 1: Umbralización simple BINARY

Se aplica umbralización binaria sobre la imagen reducida. Los píxeles mayores al umbral se vuelven blancos y los menores, negros.

Código:

```python
import cv2

# Cargamos la imagen en escala de grises
img = cv2.imread("Tema_4.2/Sources/Perro.png", cv2.IMREAD_GRAYSCALE)

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Aplicamos la umbralización binaria
thresh, img_umbralizada = cv2.threshold(img_reducida, 128, 255, cv2.THRESH_BINARY)

# Guardamos la imagen umbralizada
cv2.imwrite("ejemplo_umbralizado.jpg", img_umbralizada)

# Mostramos las imágenes
cv2.imshow("Imagen original reducida", img_reducida)
cv2.imshow("Imagen umbralizada (binaria)", img_umbralizada)
```

Resultado:



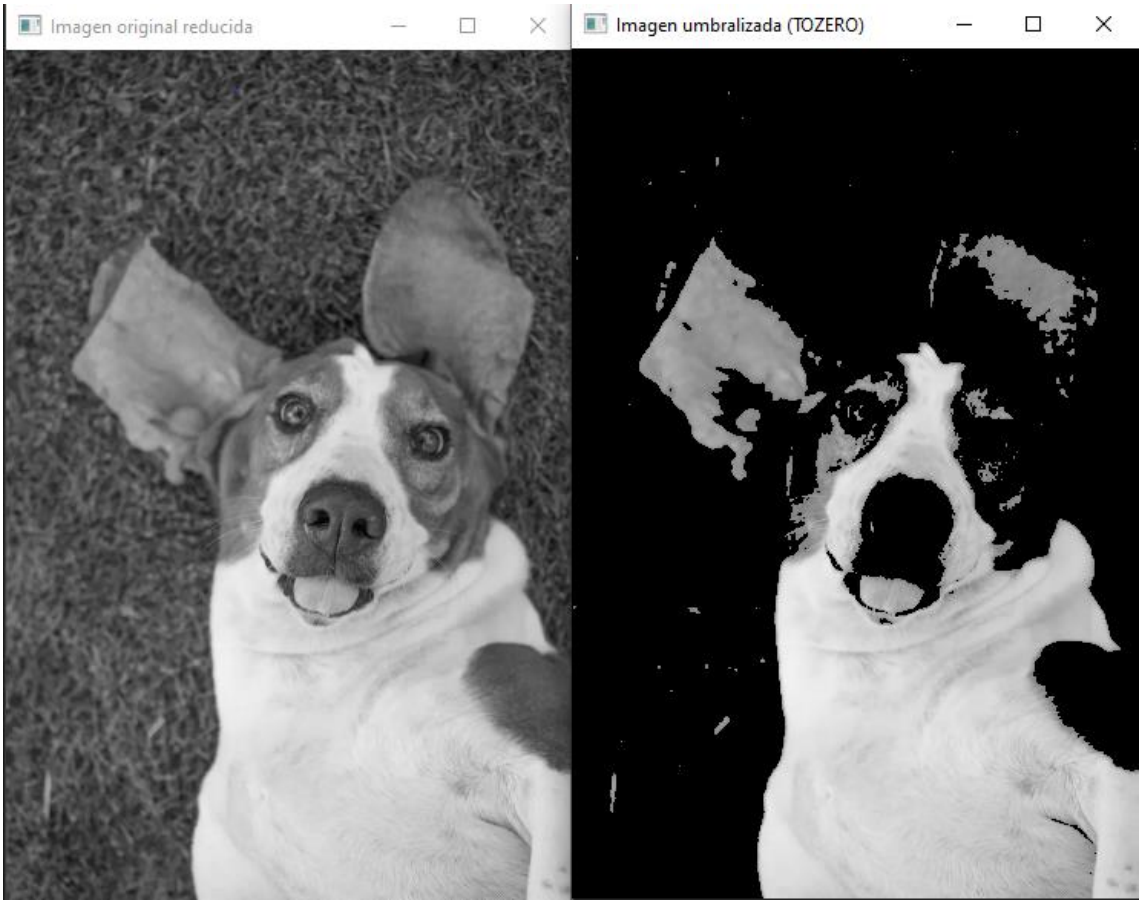
Ejemplo 2: Umbralización simple TOZERO

Se aplica umbralización TOZERO, la cual pone a 0 los pixeles que estén por debajo del umbral y deja intactos los pixeles que estén por encima de este.

Código:

```python
import cv2

# Cargamos la imagen en escala de grises
img = cv2.imread("Tema_4.2/Sources/Perro.png", cv2.IMREAD_GRAYSCALE)

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Aplicamos la umbralización TOZERO
thresh, img_umbralizada = cv2.threshold(img_reducida, 128, 255, cv2.THRESH_TOZERO)

# Guardamos la imagen umbralizada
cv2.imwrite("ejemplo_umbralizado.jpg", img_umbralizada)

# Mostramos las imágenes
cv2.imshow("Imagen original reducida", img_reducida)
cv2.imshow("Imagen umbralizada (TOZERO)", img_umbralizada)
```

Resultado:



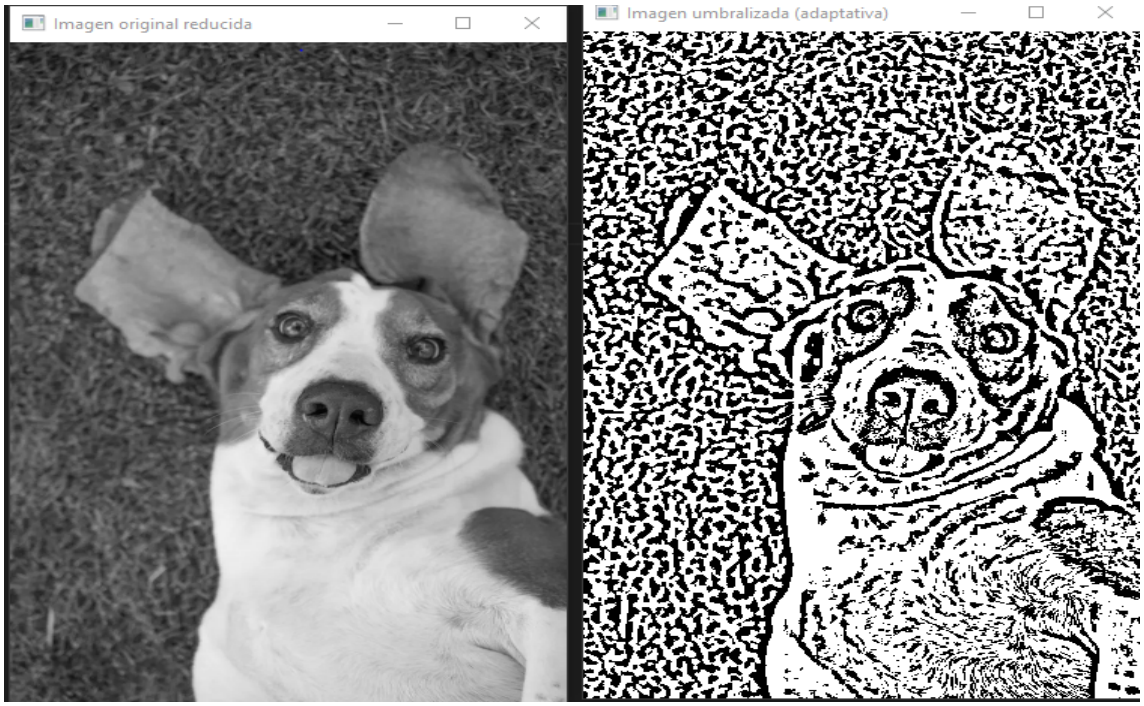
Ejemplo 3: Umbralización adaptativa Mean Thresholding

Se aplica umbralización adaptativa que calcula el umbral localmente para cada región. Esto permite segmentar mejor las imágenes con iluminación no uniforme, usando como referencia la media local. ¿¿?¿?¿?¿?¿

Código:

```python
import cv2

# Definimos los nombres de los archivos
nombre_archivo = 'Tema_4.2/Sources/Perro.png'
nombre_salida = 'ejemplo_umbralizado.jpg'

# Cargamos la imagen en escala de grises
img = cv2.imread(nombre_archivo, cv2.IMREAD_GRAYSCALE)

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Aplicamos la umbralización adaptativa con método de la media
img_umbralizada = cv2.adaptiveThreshold(
    img_reducida, 255,
    cv2.ADAPTIVE_THRESH_MEAN_C,
    cv2.THRESH_BINARY,
    11, 2
)

# Guardamos la imagen umbralizada
cv2.imwrite(nombre_salida, img_umbralizada)

# Mostramos las imágenes
cv2.imshow('Imagen original reducida', img_reducida)
cv2.imshow('Imagen umbralizada (adaptativa)', img_umbralizada)
```

Resultado:



4. Histogramas

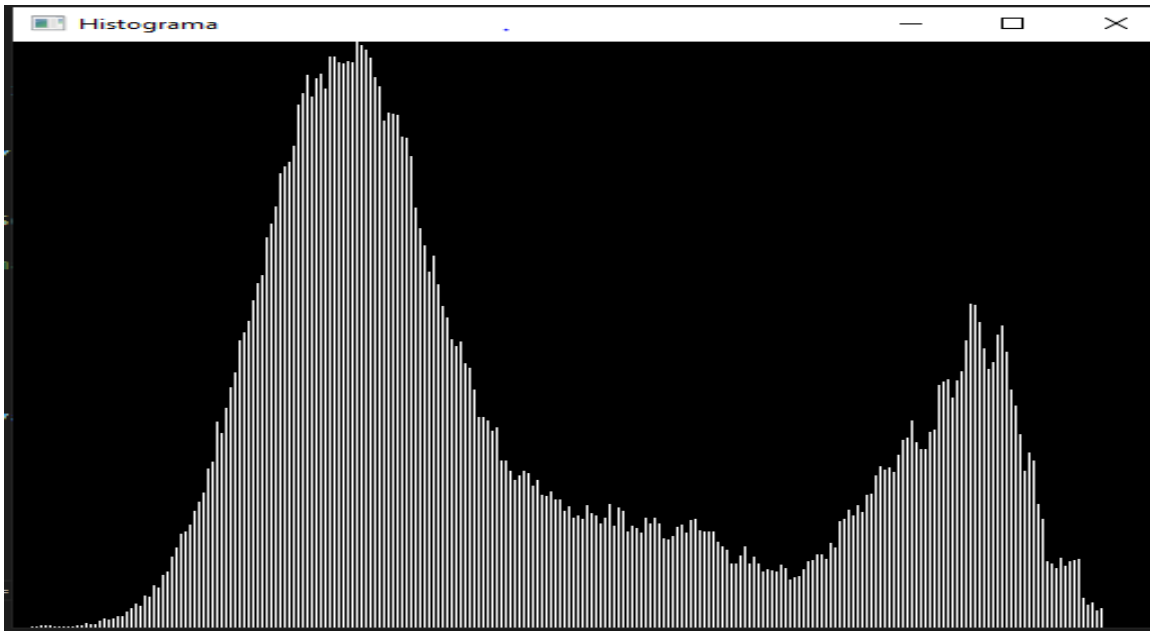
Ejemplo 1: Crear un histograma de una imagen en escala de grises

En este ejercicio se convierte una imagen a escala de grises y se calcula su histograma, que representa la distribución de intensidades de píxel (de 0 a 255). Se utiliza cv2.calcHist para contar cuántos píxeles hay de cada intensidad. Luego, se dibuja ese histograma sobre una imagen negra, representando visualmente la frecuencia de cada nivel de gris. Finalmente, se muestra el histograma en una ventana y se guarda en un archivo CSV para su posible análisis posterior.

Código:

```python
import cv2
import numpy as np

# Cargamos la imagen en color
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Convertimos a escala de grises
gray = cv2.cvtColor(img_reducida, cv2.COLOR_BGR2GRAY)

# Calculamos el histograma
hist = cv2.calcHist([gray], [0], None, [256], [0, 256])

# Creamos una imagen negra para dibujar el histograma
hist_img = np.zeros((512, 512, 3), np.uint8)
hist_max = np.max(hist)

# Dibujamos el histograma
for i in range(256):
    valor = int(hist[i] * 512 / hist_max)
    cv2.line(hist_img, (i * 2, 512), (i * 2, 512 - valor), (255, 255, 255))

# Mostramos el histograma
cv2.imshow('Histograma', hist_img)
```

Resultado:



Ejemplo 2: Crear un histograma de una imagen en color

Se separan los tres canales de color (azul, verde y rojo) de una imagen y se calcula su histograma individual. Cada histograma representa cómo se distribuyen los valores de cada componente de color en la imagen. Luego, se dibujan los tres histogramas juntos en una única imagen para comparar visualmente los colores predominantes.

Código:

```python
import cv2
import numpy as np

# Cargamos la imagen en color
img = cv2.imread("Tema_4.2/Sources/Perro.png")

# Redimensionamos la imagen a la mitad
alto, ancho = img.shape[:2]
img_reducida = cv2.resize(img, (ancho // 2, alto // 2))

# Dividimos la imagen en canales B, G y R
b, g, r = cv2.split(img_reducida)

# Calculamos el histograma de cada canal
hist_b = cv2.calcHist([b], [0], None, [256], [0, 256])
hist_g = cv2.calcHist([g], [0], None, [256], [0, 256])
hist_r = cv2.calcHist([r], [0], None, [256], [0, 256])

# Creamos una imagen negra para dibujar los histogramas
hist_img = np.zeros((512, 512, 3), np.uint8)
hist_max = max(np.max(hist_b), np.max(hist_g), np.max(hist_r))

# Dibujamos los histogramas de cada canal
for i in range(256):
    h_b = int(hist_b[i] * 512 / hist_max)
    h_g = int(hist_g[i] * 512 / hist_max)
    h_r = int(hist_r[i] * 512 / hist_max)
    cv2.line(hist_img, (i * 2, 512), (i * 2, 512 - h_b), (255, 0, 0))  # Azul
    cv2.line(hist_img, (i * 2, 512), (i * 2, 512 - h_g), (0, 255, 0))  # Verde
    cv2.line(hist_img, (i * 2, 512), (i * 2, 512 - h_r), (0, 0, 255))  # Rojo

# Mostramos el histograma combinado
cv2.imshow('Histograma de color (RGB)', hist_img)
```

Resultado: