Tratamiento de Imagen Digital
Portafolio Nº2

ALEJANDRO ALBERT GRAMAJE

ÍNDICE

1. Funciones de dibujo 3
Ejemplo 1: Dibujar líneas 3
Ejemplo 2: Dibujar rectángulos 4
Ejemplo 3: Dibujar círculos 5
Ejemplo 4: Dibujar texto 6
Ejercicio Composiciones 7
2. Mejora del contraste de una imagen 9
Ejemplo 1: Ecualización de imagen en color 9
3. Detección de bordes 10
Ejemplo 1: Filtro Sobel 10
Ejemplo 2: Filtro Lapaciano 11
Ejemplo 3: Filtro Canny 12
4. Desenfoque de movimiento 13
Ejemplo 1: Desenfoque de movimiento (Motion Blur) 13
5. Mejora de una imagen (Sharpening) 14
6. Erosión y dilatación 15
7. Creación de filtro de viñeta 16
8. Creación de filtro de relieve 17

1. Funciones de dibujo

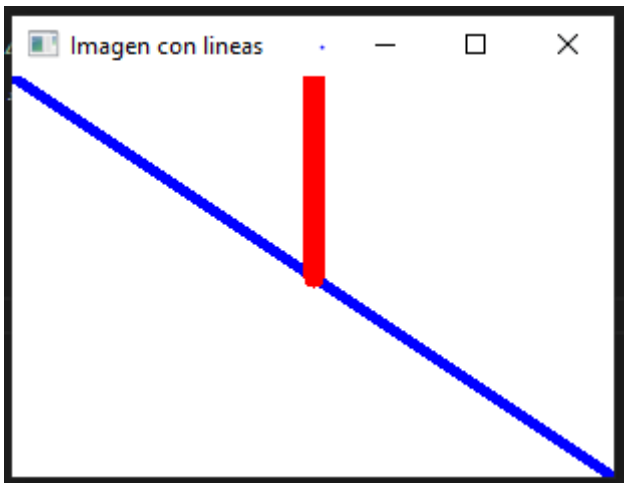
Ejemplo 1: Dibujar líneas

Se crea una imagen blanca como fondo y se dibujan dos líneas con distintos colores, posiciones y grosores sobre ella usando cv2.line(imagen, inicio, final, color, grosor).

Código:

```
import cv2
import numpy as np

# Creamos una imagen en blanco redimensionada a la mitad (200x300)
imagen = 255 * np.ones((400, 600, 3), dtype=np.uint8)
imagen_reducida = cv2.resize(imagen, (300, 200))

# Dibujamos dos líneas
cv2.line(imagen_reducida, (0, 0), (300, 200), (255, 0, 0), 4)     # Línea azul
cv2.line(imagen_reducida, (150, 0), (150, 100), (0, 0, 255), 10)  # Línea roja

# Mostramos la imagen
cv2.imshow('Imagen con lineas', imagen_reducida)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



Ejemplo 2: Dibujar rectángulos

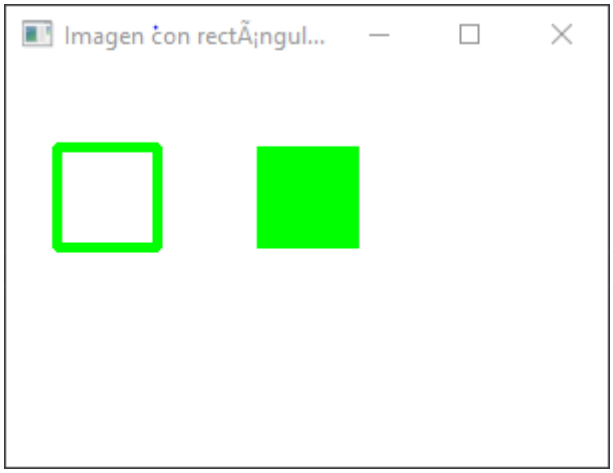
Se dibujan dos rectángulos sobre una imagen blanca: uno con solo borde y otro relleno, usando cv2.rectangle con distintos valores de grosor.

Los parámetros son: cv2.rectangle(imagen, punto_superior_izq, punto_inferior_derecho, color, grosor)

Código:

```python
import cv2
import numpy as np

# Creamos una imagen blanca de fondo y la redimensionamos a la mitad
imagen = 255 * np.ones((400, 600, 3), dtype=np.uint8)
imagen_reducida = cv2.resize(imagen, (300, 200))

# Dibujamos dos rectángulos verdes
cv2.rectangle(imagen_reducida, (25, 40), (75, 90), (0, 255, 0), 3)   # Borde verde
cv2.rectangle(imagen_reducida, (125, 40), (175, 90), (0, 255, 0), -1) # Relleno verde

# Mostramos la imagen con los rectángulos
cv2.imshow('Imagen con rectángulos', imagen_reducida)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



Ejemplo 3: Dibujar círculos

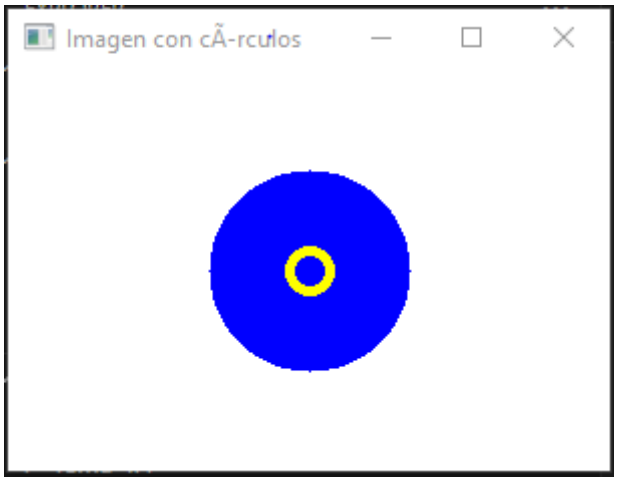
Se dibujan dos círculos centrados: uno grande relleno y otro pequeño con borde, los dos con diferentes grosores. Utilizando:

cv2.circle(imagen, centro, radio, color, grosor)

Código:

```python
import cv2
import numpy as np

# Creamos una imagen blanca de fondo y la redimensionamos a la mitad
imagen = 255 * np.ones((400, 600, 3), dtype=np.uint8)
imagen_reducida = cv2.resize(imagen, (300, 200))

# Dibujamos dos círculos
cv2.circle(imagen_reducida, (150, 100), 50, (255, 0, 0), -1)   # Círculo azul relleno
cv2.circle(imagen_reducida, (150, 100), 10, (0, 255, 255), 3)  # Círculo amarillo de borde

# Mostramos la imagen con los círculos
cv2.imshow('Imagen con círculos', imagen_reducida)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



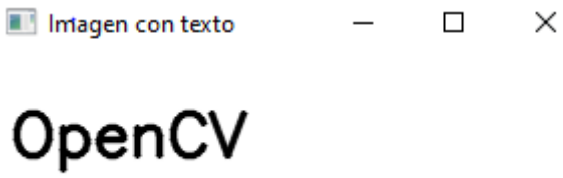
Ejemplo 4: Dibujar texto

Utilizo la función cv2.putText para escribir texto sobre la imagen fuente en blanco.
Cv2.putText(imagen, texto, posición, fuente, tamaño, color, grosor, tipo_linea)

Código:

```python
import cv2
import numpy as np

# Creamos una imagen blanca de fondo y la redimensionamos a la mitad
imagen = 255 * np.ones((400, 600, 3), dtype=np.uint8)
imagen_reducida = cv2.resize(imagen, (300, 200))

# Fuente del texto
font = cv2.FONT_HERSHEY_SIMPLEX

# Dibujamos texto en la imagen
cv2.putText(imagen_reducida, 'OpenCV', (10, 50), font, 1, (0, 0, 0), 2, cv2.LINE_AA)

# Mostramos la imagen con el texto
cv2.imshow('Imagen con texto', imagen_reducida)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



Ejercicio Composiciones

Creo quna imagen con un circulo azul como fondo. Encima de ella dibujo la figura con picos utilizando el método fillPoly, con las diferenters coordenadas calculadas.Finalmente añado el circulo rojo que representa el núcleo del sol.

Código Sol:

```
# Crear lienzo general (altura fija, ancho para 3 bloques)
alto, ancho = 400, 1200
lienzo = 255 * np.ones((alto, ancho, 3), dtype=np.uint8)

# 1. Sol con picos (izquierda)
# ============================
centro = (200, 200)
radio_azul = 120
radio_amarillo = 100
radio_rojo = 80
picos = 16

# Fondo azul
cv2.circle(lienzo, centro, radio_azul, (255, 0, 0), -1)

# Estrella amarilla (con picos)
pts = []
for i in range(picos * 2):
    angulo = i * math.pi / picos
    r = radio_amarillo if i % 2 == 0 else radio_amarillo - 20
    x = int(centro[0] + r * math.cos(angulo))
    y = int(centro[1] + r * math.sin(angulo))
    pts.append((x, y))
cv2.fillPoly(lienzo, [np.array(pts, np.int32)], (0, 255, 255))  # Amarillo

# Círculo rojo interior
cv2.circle(lienzo, centro, radio_rojo, (0, 0, 255), -1)
```

Construyo una bandera rectangular con el fondo azul, y luego voy añadiendo las diferentes cruces diagonales blancas y rojas con cv2.line. Lugo añado las dos cruces centrales en blanco, con una cruz superpuesta usando cv2.rectangle.

Código Bandera UK:

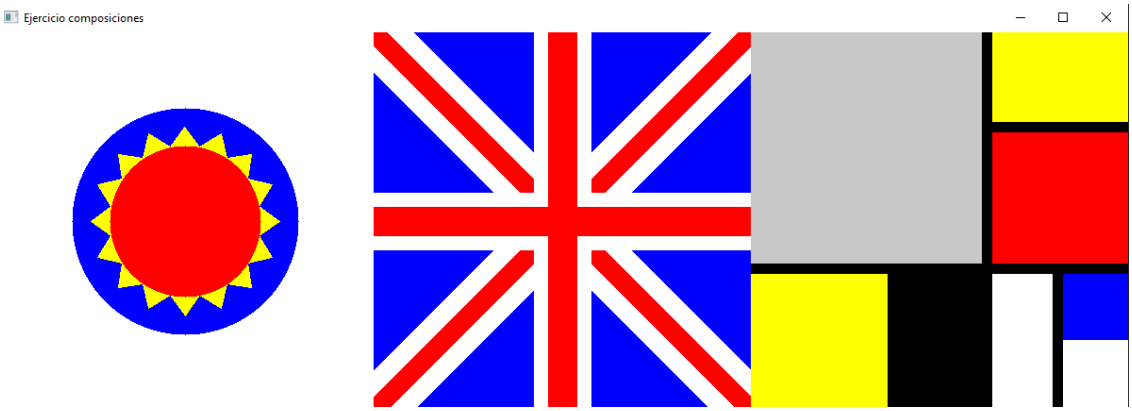
```
# 2. Bandera UK (centro)
x_offset = 400  # posición horizontal de inicio
bandera = 255 * np.ones((400, 400, 3), dtype=np.uint8)

# Fondo azul
bandera[:] = (255, 0, 0)

# Cruces diagonales blancas y rojas
cv2.line(bandera, (0, 0), (400, 400), (255, 255, 255), 60)
cv2.line(bandera, (400, 0), (0, 400), (255, 255, 255), 60)
cv2.line(bandera, (0, 0), (400, 400), (0, 0, 255), 20)
cv2.line(bandera, (400, 0), (0, 400), (0, 0, 255), 20)

# Cruces centrales
cv2.rectangle(bandera, (170, 0), (230, 400), (255, 255, 255), -1)
cv2.rectangle(bandera, (0, 170), (400, 230), (255, 255, 255), -1)
cv2.rectangle(bandera, (185, 0), (215, 400), (0, 0, 255), -1)
cv2.rectangle(bandera, (0, 185), (400, 215), (0, 0, 255), -1)

# Pegar bandera al lienzo
lienzo[:, x_offset:x_offset+400] = bandera
```

Creo una cuadricula con varios rectángulos de colores planos. Los dibujó con cv2.rectangle. Imitando el estilo de la foto del ejercicio, no es una representación exacta pero parecida.

Código Mondrian:

```python
# 3. Bloques
# =====================
x_offset = 800
mondrian = 255 * np.ones((400, 400, 3), dtype=np.uint8)

# Rectángulos de colores
cv2.rectangle(mondrian, (0, 0), (250, 250), (200, 200, 200), -1)     # gris claro
cv2.rectangle(mondrian, (250, 0), (400, 100), (0, 255, 255), -1)     # amarillo
cv2.rectangle(mondrian, (0, 250), (150, 400), (0, 255, 255), -1)     # amarillo
cv2.rectangle(mondrian, (150, 250), (250, 400), (0, 0, 0), -1)       # negro
cv2.rectangle(mondrian, (250, 100), (400, 250), (0, 0, 255), -1)     # rojo
cv2.rectangle(mondrian, (250, 250), (325, 325), (255, 255, 255), -1) # blanco
cv2.rectangle(mondrian, (325, 250), (400, 325), (255, 0, 0), -1)     # azul

# Líneas negras
cv2.line(mondrian, (250, 0), (250, 400), (0, 0, 0), 10)
cv2.line(mondrian, (0, 250), (400, 250), (0, 0, 0), 10)
cv2.line(mondrian, (150, 250), (150, 400), (0, 0, 0), 10)
cv2.line(mondrian, (325, 250), (325, 400), (0, 0, 0), 10)
cv2.line(mondrian, (250, 100), (400, 100), (0, 0, 0), 10)

# Pegar mondrian al lienzo
lienzo[:, x_offset:x_offset+400] = mondrian

# Mostrar resultado final
# =====================
cv2.imshow("Ejercicio composiciones", lienzo)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



2. Mejora del contraste de una imagen

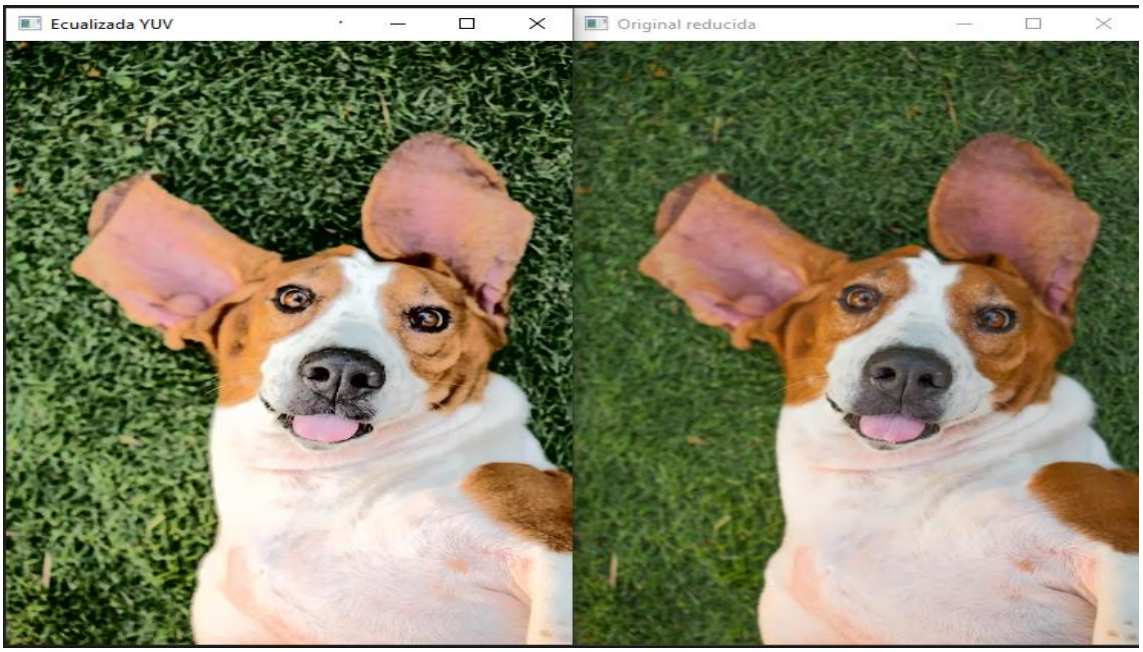
Ejemplo 1: Ecualización de imagen en color

La imagen se convierte al espacio YUV, donde el canal Y controla el brillo. Solo se ecualiza ese canal para mejorar el contraste sin cambiar los colores. Después se vuelve a pasar a BGR para poder mostrarla. Así se puede conseguir que la imagen se vea más clara o definida sin estropear los tonos originales.

Código:

```python
import cv2
import numpy as np

# Cargamos la imagen en color
img = cv2.imread("Tema_4.3/Sources/Perro.png")

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

# Convertimos al espacio de color YUV
img_yuv = cv2.cvtColor(img_reducida, cv2.COLOR_BGR2YUV)

# Ecualizamos el canal de luminancia (Y)
img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])

# Convertimos de nuevo a BGR
img_output = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)

# Mostramos las imágenes
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Ecualizada YUV', img_output)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



3. Detección de bordes

Ejemplo 1: Filtro Sobel

Se pasa la imagen a escala de grises para trabajar solo con intensidad. Luego se usan dos filtros Sobel para detectar bordes: uno detecta los que van de izquierda a derecha (horizontal) y otro de arriba a abajo (vertical). Se muestran ambos para ver por dónde cambian más los tonos.

Código:

```python
import cv2

# Cargamos la imagen en color
img = cv2.imread("Tema_4.3/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 escala de grises
img_gris = cv2.cvtColor(img_reducida, cv2.COLOR_BGR2GRAY)

# Aplicamos los filtros Sobel en dirección horizontal y vertical
sobel_horizontal = cv2.Sobel(img_gris, cv2.CV_64F, 1, 0, ksize=5)
sobel_vertical = cv2.Sobel(img_gris, cv2.CV_64F, 0, 1, ksize=5)

# Mostramos las imágenes
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Sobel Horizontal', sobel_horizontal)
cv2.imshow('Sobel Vertical', sobel_vertical)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:

Ejemplo 2: Filtro Lapaciano

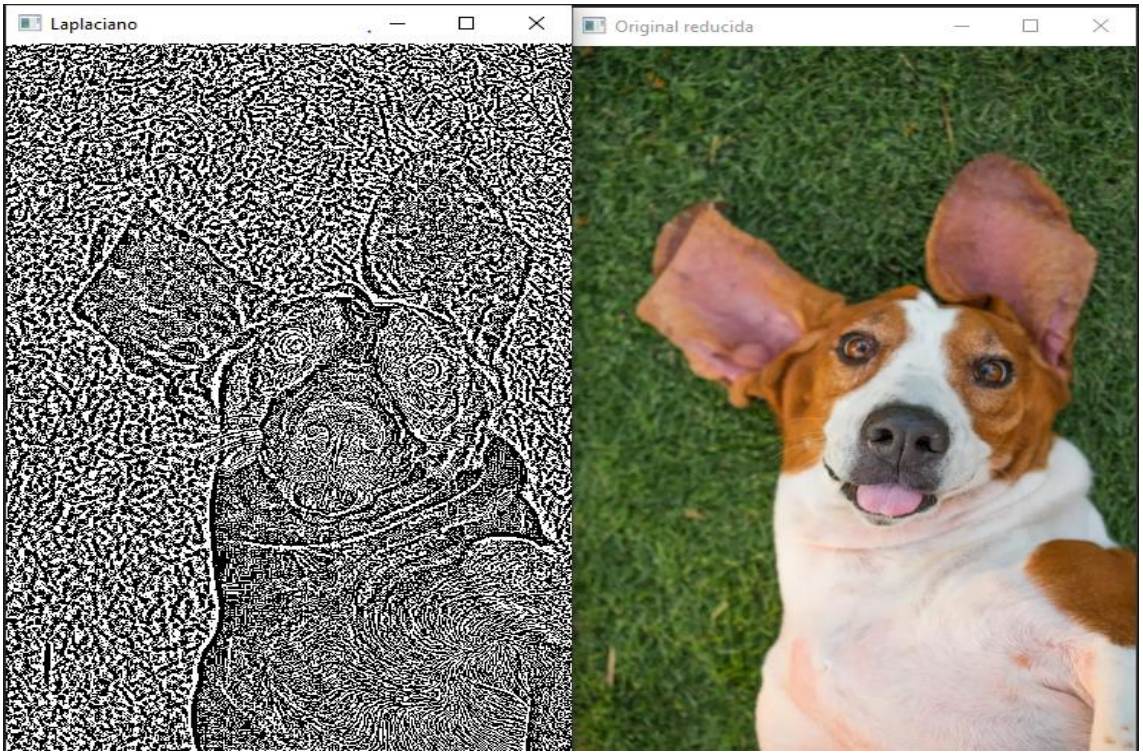
Se usa el filtro Laplaciano, que detecta bordes en todas direcciones a la vez, gracias a que usa la segunda derivada. Se aplica sobre la imagen en escala de grises y nos da una imagen con los bordes resaltados.

Capta mas detalles que el filtro Sobel.

Código:

```python
import cv2

# Cargamos la imagen en color
img = cv2.imread("Tema_4.3/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
img_gris = cv2.cvtColor(img_reducida, cv2.COLOR_BGR2GRAY)

# Aplicamos el filtro Laplaciano
laplaciano = cv2.Laplacian(img_gris, cv2.CV_64F)

# Mostramos las imágenes
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Laplaciano', laplaciano)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



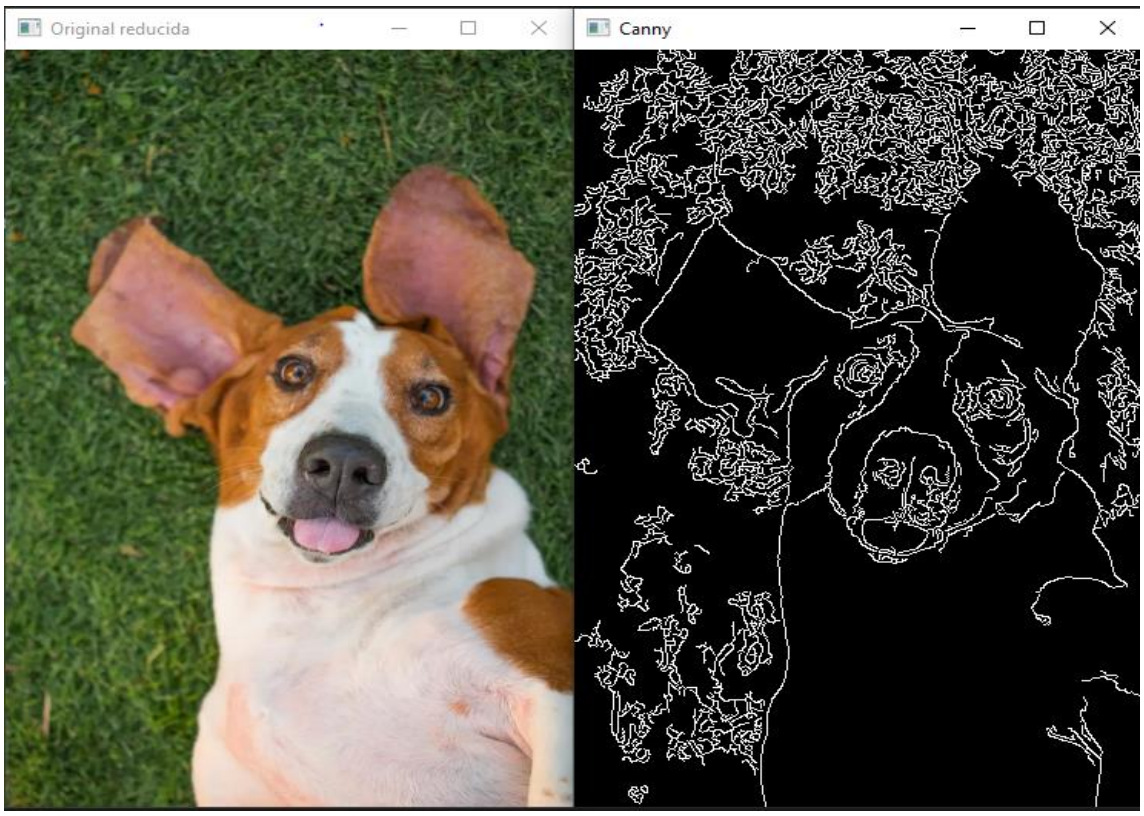
Ejemplo 3: Filtro Canny

El filtro Canny detecta bordes de forma muy precisa usando varios pasos: suaviza la imagen, calcula gradientes, y elimina bordes débiles. Es uno de los métodos mas usados, porque da resultados limpios y definidos.

Código:

```python
import cv2

# Cargamos la imagen en color
img = cv2.imread("Tema_4.3/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
img_gris = cv2.cvtColor(img_reducida, cv2.COLOR_BGR2GRAY)

# Aplicamos el detector de bordes Canny
canny = cv2.Canny(img_gris, 50, 240)

# Mostramos las imágenes
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Canny', canny)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



4. Desenfoque de movimiento

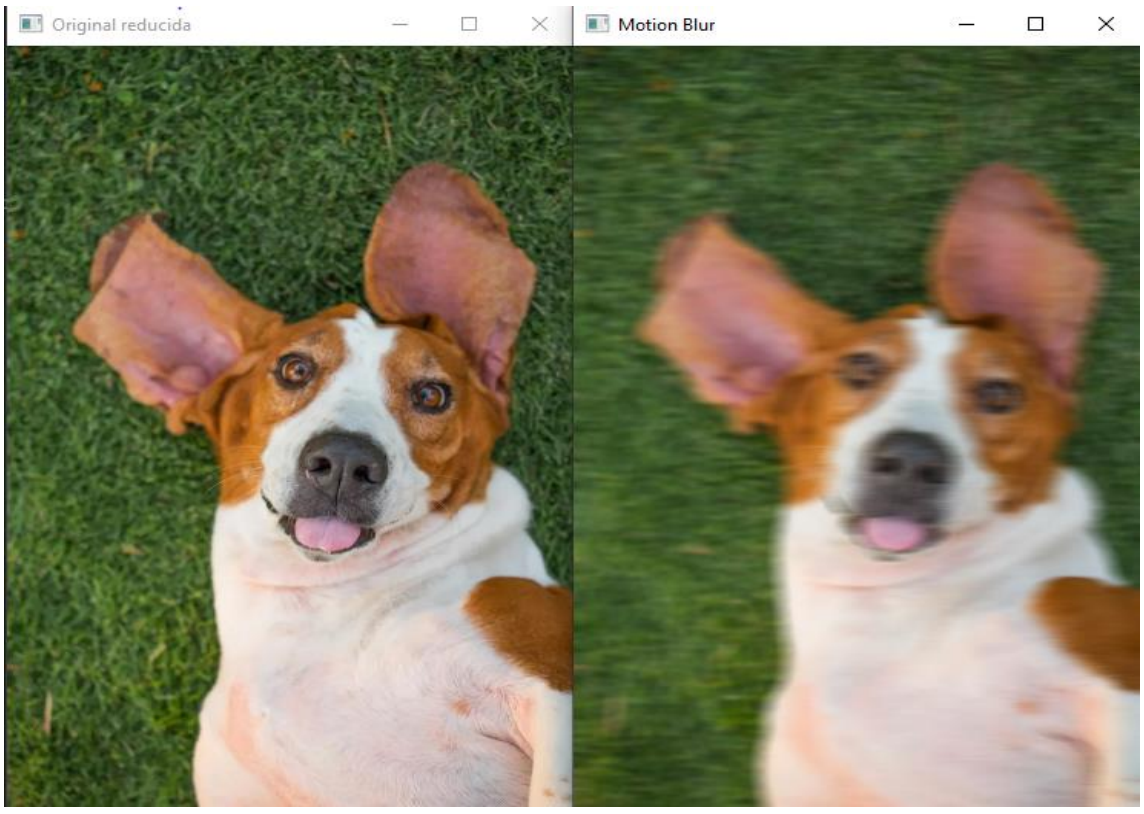
Ejemplo 1: Desenfoque de movimiento (Motion Blur)

Se genera un desenfoque que imita el movimiento en una dirección horizontal. Se crea un kernel especial que borra los bordes en esa dirección, como si la imagen se hubiera capturado en movimiento.

Código:

```python
import cv2
import numpy as np

# Cargamos la imagen en color
img = cv2.imread("Tema_4.3/Sources/Perro.png")

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

# Creamos un kernel para simular el desenfoque de movimiento horizontal
size = 15
kernel_motion_blur = np.zeros((size, size))
kernel_motion_blur[int((size - 1) / 2), :] = np.ones(size)
kernel_motion_blur = kernel_motion_blur / size

# Aplicamos el kernel a la imagen
output = cv2.filter2D(img_reducida, -1, kernel_motion_blur)

# Mostramos las imágenes
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Motion Blur', output)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



5. Mejora de una imagen (Sharpening)

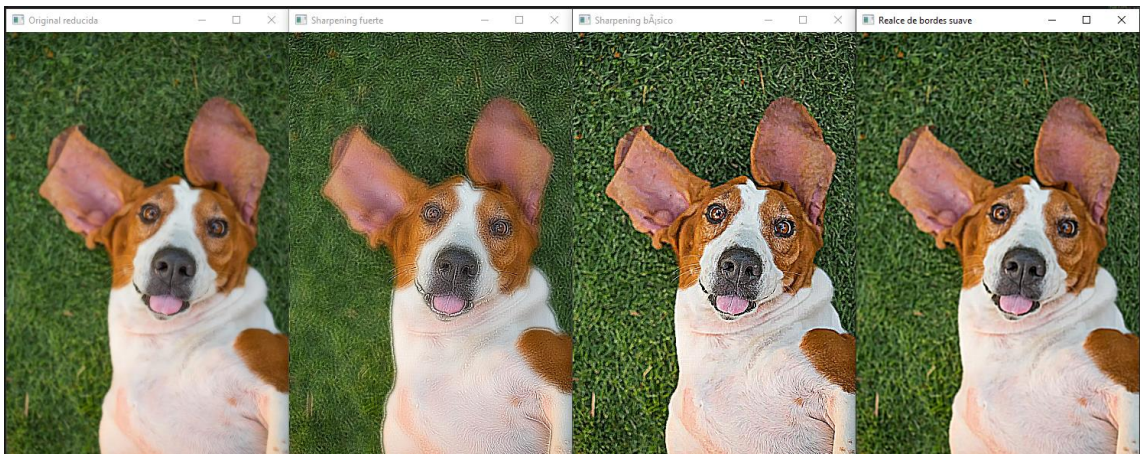
Ejemplo:

Se aplican tres filtros para que la imagen se vea más nítida. Cada kernel tiene un efecto distinto: uno realza bordes suavemente, otro exagera el contraste, y otra mejora los contornos sin deformar. Sirve para resaltar detalles en imágenes que se ven borrosas.

Código:

```python
import cv2
import numpy as np

# Cargamos la imagen en color
img = cv2.imread("Tema_4.3/Sources/Perro.png")

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

# Definimos tres kernels distintos para realzar bordes
kernel_sharp_1 = np.array([[-1, -1, -1],
                           [-1,  9, -1],
                           [-1, -1, -1]])

kernel_sharp_2 = np.array([[ 1,  1,  1],
                           [ 1, -7,  1],
                           [ 1,  1,  1]])

kernel_sharp_3 = np.array([[-1, -1, -1, -1, -1],
                           [-1,  2,  2,  2, -1],
                           [-1,  2,  8,  2, -1],
                           [-1,  2,  2,  2, -1],
                           [-1, -1, -1, -1, -1]]) / 8.0

# Aplicamos los filtros
out1 = cv2.filter2D(img_reducida, -1, kernel_sharp_1)
out2 = cv2.filter2D(img_reducida, -1, kernel_sharp_2)
out3 = cv2.filter2D(img_reducida, -1, kernel_sharp_3)

# Mostramos las imágenes
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Sharpening básico', out1)
cv2.imshow('Sharpening fuerte', out2)
cv2.imshow('Realce de bordes suave', out3)
```

Resultado:



6. Erosión y dilatación

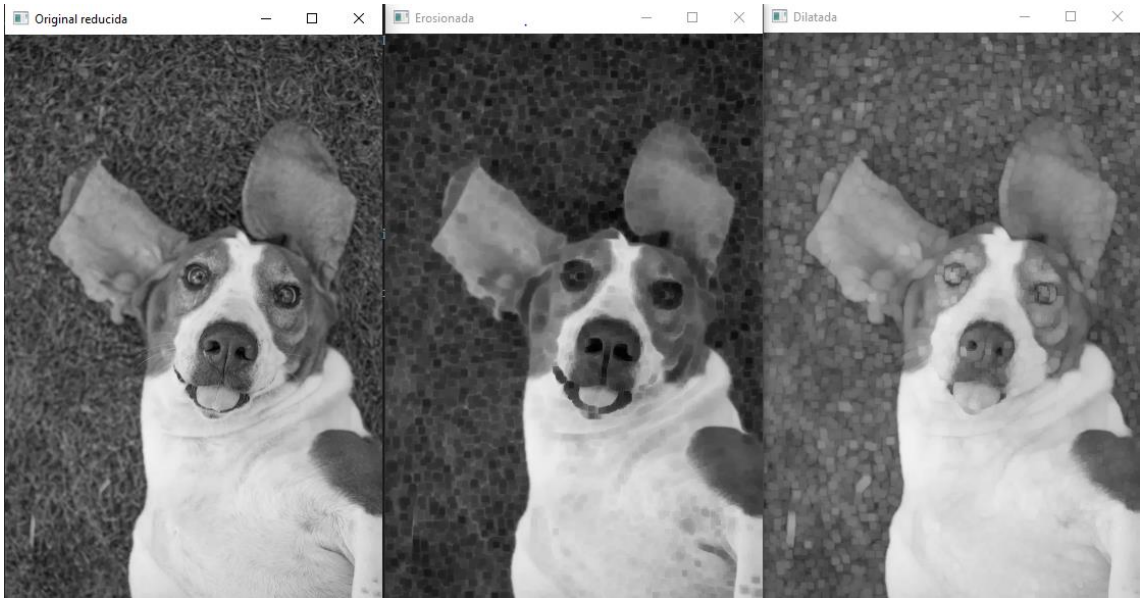
Ejemplo:

La erosión hace que las zonas claras se encojan, eliminando detalles pequeños. La dilatación hace lo contrario: agranda las partes blancas. Sirve para limpiar ruido, cerrar huecos o resaltar formas en imágenes binarias o en escala de grises.

Código:

```python
import cv2
import numpy as np

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

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

# Definimos un kernel cuadrado de 5x5
kernel = np.ones((5, 5), np.uint8)

# Aplicamos erosión (quita bordes blancos)
erosionada = cv2.erode(img_reducida, kernel, iterations=1)

# Aplicamos dilatación (agranda zonas blancas)
dilatada = cv2.dilate(img_reducida, kernel, iterations=1)

# Mostramos los resultados
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Erosionada', erosionada)
cv2.imshow('Dilatada', dilatada)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



7. Creación de filtro de viñeta

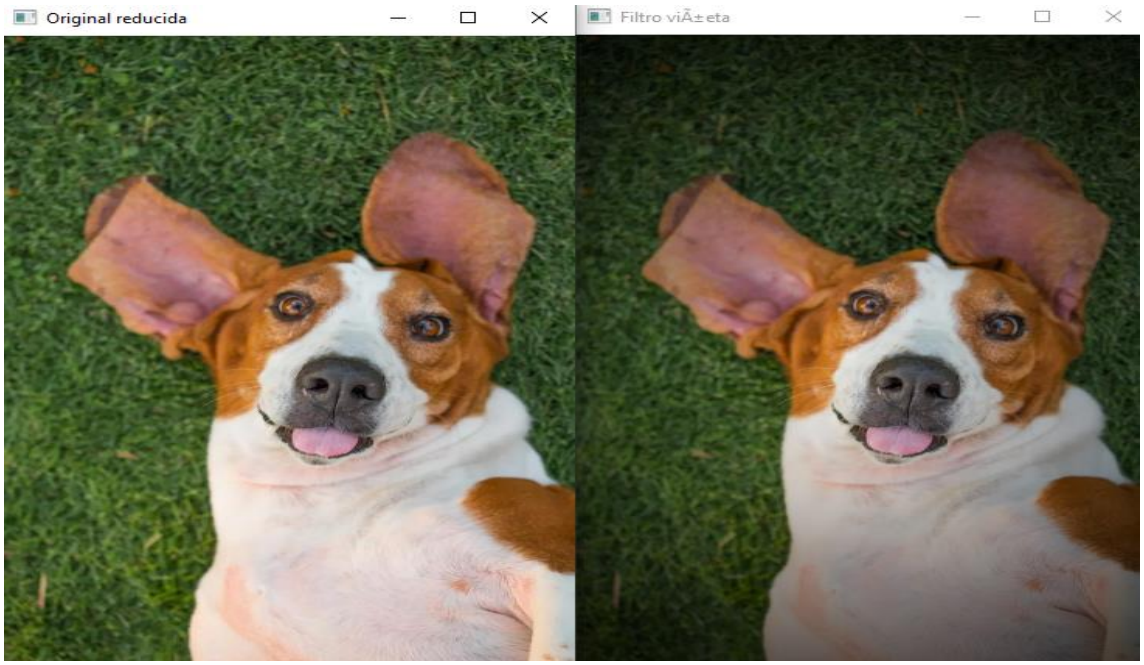
Ejemplo:

Se crea un efecto viñeta, que oscurece los bordes de la imagen para dar un toque más artístico. Se genera una máscara gaussiana que simula esa caída de luz y se multiplica por cada canal de color. El centro queda más claro que los bordes.

Código:

```python
import cv2
import numpy as np

# Cargamos la imagen en color
img = cv2.imread('Tema_4.3/Sources/Perro.png')

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

# Creamos una máscara tipo viñeta con una distribución gaussiana
kernel_x = cv2.getGaussianKernel(ancho_r, 200)
kernel_y = cv2.getGaussianKernel(alto_r, 200)
kernel = kernel_y @ kernel_x.T
mascara = kernel / kernel.max()

# Aplicamos la máscara a cada canal (BGR)
viñeta = np.empty_like(img_reducida, dtype=np.uint8)
for i in range(3):
    viñeta[:, :, i] = img_reducida[:, :, i] * mascara

# Mostramos el resultado
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Filtro viñeta', viñeta)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado:



8. Creación de filtro de relieve

Ejemplo:

Se usa un kernel especial que crea un efecto de relieve, como si la imagen estuviera tallada o en 3D. Resalta los bordes en una dirección y oscurece otros. Para que se vea mejor, se suma 128 y se ajusta el rango de grises.

Código:

```python
import cv2
import numpy as np

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

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

# Kernel para efecto relieve
kernel_relieve = np.array([[-2, -1, 0],
                           [-1,  1, 1],
                           [ 0,  1, 2]])

# Aplicamos el filtro de relieve
relieve = cv2.filter2D(img_reducida, -1, kernel_relieve)

# Ajustamos contraste sumando 128 (opcional)
relieve = cv2.convertScaleAbs(relieve + 128)

# Mostramos los resultados
cv2.imshow('Original reducida', img_reducida)
cv2.imshow('Relieve', relieve)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Resultado: