Blog

OPERACIONES BÁSICAS EN IMÁGENES

Objetivo del 6º tutorial de Curso de Procesamiento de Imágenes y Visión Artificial

Acceder a los valores de píxeles y modificarlos

Acceder a las propiedades de imagen

Fijar la Región de Imagen (ROI, siglas en inglés de *Region of Image*)

Dividir y Combinar imágenes

Operaciones Básicas en Imágenes en OpenCV con

Python

Casi todas las operaciones en esta sección están principalmente relacionadas con Numpy (https://unipython.com/numpy-algebra/) más que a OpenCV. Conocer bien Numpy (https://unipython.com/numpy-algebra/) es un requerimiento para escribir un código mejor optimizado con OpenCV. Si no conoces Numpy te recomendamos esta introducción a Numpy. (https://unipython.com/numpy-algebra/)

(Mostraremos los ejemplos en la terminal Python dado que la mayor parte de ellos corresponden a líneas únicas de código)

Accediendo y Modificando los valores de píxeles

Primero carguemos una imagen a color:

```
1 import cv2
2 import numpy as np
3 img = cv2.imread('trump.jpg')
```

Puedes acceder al valor de un pixel por medio de las coordenadas de su fila y su columna. Para imágenes RGB, regresan una gama de valores entre Azul, Verde y Rojo. Para las imágenes en escala de grises, sólo la intensidad correspondiente es regresada.

```
1 px = img[100,100]
2 print px
3 # accessing only blue pixel
4 blue = img[100,100,0]
5 print blue
```

Puedes modificar los valores de pixel de la misma forma.

```
1 img[100,100] = [255,255,255]
2 print&amp;amp;nbsp; img[100,100]
```

⚠️ Advertencia

Numpy es una biblioteca optimizada para cálculos rápidos. Así que simplemente acceder a cada valor de pixel y modificarlo podría resultar muy lento y no es recomendado.

aaaa

📝 Nota

El método mencionado anteriormente es usado normalmente para seleccionar una región, por ejemplo, las primeras 5 filas y las últimas 3 columnas de este modo. Para acceder a los píxeles de forma individual, los métodos de Numpy array.item() y array.itemset() son considerados mejores. Pero siempre regresa un escalar. Así que si deseas acceder a los valores R,G,B necesitas llamar array.item() de forma separada para todos.

aaaa

Un mejor método para acceder al pixel y editarlo:

```
1 # accessing RED value
2 img.item(10,10,2)
3 # modifying RED value
4 img.itemset((10,10,2),100)
5 img.item(10,10,2)
```

Accediendo a las Propiedades de Imagen

Las propiedades de imagen incluyen número de filas, columnas y canales, tipo de data de imagen, número de píxeles, etc.

Se accede a la forma de la imagen por medio de img.shape. Este regresa una tupla de números de filas, columnas y canales (si la imagen es a color):

```
1 print img.shape
```

📝 **Nota**

Si la imagen es a escala de grises, la tupla regresada contiene únicamente el número de filas y columnas. Así que es un buen método para verificar si la imagen cargada está a escala de grises o es una imagen a color.

aaaa

Se accede al número total de píxeles por medio de img.size:

```
print img.size
```

El tipo de datos (*datatype*) de la imagen se obtiene por medio de img.dtype:

```
print img.dtype
```

📝 **Nota**

img.dtype es muy importante al momento de la depuración porque un gran número de errores en el código OpenCV-Python son causados por tipos de datos inválidos.

aaaa

Dividiendo y Combinando Canales de Imagen

Los canales R,G,B de una imagen pueden dividirse en sus planos individuales cuando sea necesario. Luego, puede combinarse nuevamente dichos canales para nuevamente formar una imagen:

```
b,g,r = cv2.split(img)
img = cv2.merge((b,g,r))
```

O

```
b = img[:,:,0]
```

Supongamos que quieres hacer que todos los píxeles rojos valgan cero, no necesitas dividirlos todos de esta forma y colocarlos igual a cero. Puedes simplemente usar indexación Numpy la cual es más rápida.

```
1 img[:,:,2] = 0
```

⚠️ **Advertencia**

cv2.split() es una operación costosa (en términos de tiempo), así que sólo utilízala si es necesario. La indexación Numpy es mucho más eficiente y debería usarse siempre que sea posible.

aaaa

Haciendo Bordes para la Imagen (*Padding*)

Si deseas crear un borde alrededor de una imagen, algo como un marco, puedes usar la función **cv2.copyMakeBorder()**. Pero tiene más aplicaciones para la operación de circonvolución, *zero padding*, etc. Esta función toma los siguientes argumentos:

src – introducir imagen

top, **bottom**, **left**, **right** – ancho de borde en número de píxeles en correspondencia con las direcciones.

borderType – Marca que define el tipo de borde a añadir. Puede ser de los siguientes tipos:

BORDER_CONSTANT – Añade un borde de color constante. El valor debería ser provisto como el siguiente argumento.

BORDER_REFLECT – El border será el reflejo de sus elementos, como este: fedcba|abcdefgh|hgfedcb

BORDER_REFLECT_101 or **cv2.BORDER_DEFAULT** – Igual que arriba, pero con un pequeño cambio, como este: gfedcb|abcdefgh|gfedcba

BORDER_REPLICATE – El último elemento es replicado alrededor, así: aaaaaa|abcdefgh|hhhhhhh

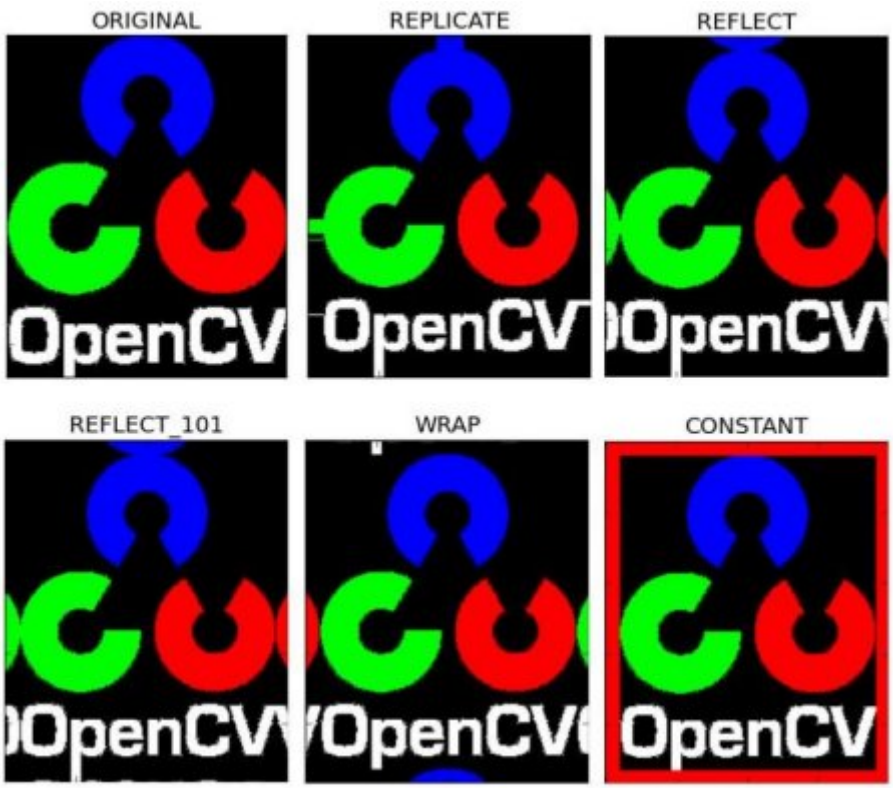
BORDER_WRAP – No puedo explicarlo, luciría así: cdefgh|abcdefgh|abcdefg

valor – El color del borde si el tipo de borde es cv2.BORDER_CONSTANT

Más abajo hay una muestra del código demostrando todos estos tipos de borde para su mejor entendimiento:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
BLUE = [255,0,0]
img1 = cv2.imread('opencv_logo.png')
replicate = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_REPLICAT
reflect = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_REFLECT)
reflect101 = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_REFLECT
wrap = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_WRAP)
constant= cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER_CONSTANT,v
plt.subplot(231),plt.imshow(img1,'gray'),plt.title('ORIGINAL')
plt.subplot(232),plt.imshow(replicate,'gray'),plt.title('REPLICATE'
plt.subplot(233),plt.imshow(reflect,'gray'),plt.title('REFLECT')
plt.subplot(234),plt.imshow(reflect101,'gray'),plt.title('REFLECT_1
plt.subplot(235),plt.imshow(wrap,'gray'),plt.title('WRAP')
plt.subplot(236),plt.imshow(constant,'gray'),plt.title('CONSTANT')
plt.show()
```

Fíjate en el resultado debajo. (La imagen se expone usando *matplotlib*. Así que los planos del ROJO y el AZUL están intercambiados):



➡️ Cada vez aumentamos más nuestros conocimientos en el manejo de imágenes. En nuestro **curso Python de OpenCV** se estudian operaciones más avanzadas con imágenes:

Curso de Procesamiento de Imágenes y Visión Artificial

(https://unipython.com/curso-procesamiento-imagenes-opencv-python/)

Share:

2 COMENTARIOS

Funciones para dibujar - ® Cursos Python desde 0 a Experto ? garantizados (https://www.aprenderpython.net/funciones-dibujar-opencv/)

abril 5, 2018 a 9:26 am (https://unipython.com/operaciones-basicas-imagenes/#comment-108) Responder

[...] ➡️ Operaciones Básicas en Imágenes [...]

julian

abril 17, 2019 a 5:34 am (https://unipython.com/operaciones-basicas-imagenes/#comment-537)

Responder

como puedo desarrollar un programa que divida una imagen en 16 partes. (4×4). El usuario puede escoger en los primeros 4 cuadros que color desea, entre: rojo, azul, verde, normal o gris

Deja una respuesta

Sobre

Unipython es una plataforma de aprendizaje online dirigida a personas que quieran mejorar su carrera profesional. El objetivo de Unipython es proporcionar cursos online de calidad en los campos de la Programación, Internet de las cosas, Analisis de Datos, Inteligencia Artificial, Desarrollo Web/Apps, Testeo, Videojuegos y Tecnología Creativa.

Nombre *

Email *

Mensaje *

+Info

- Que dicen de nosotros (https://unipython.com/unipython-en-los-medios/)
- Contacto (https://unipython.com/contacto-5/)
- FAQ (https://unipython.com/faq/)
- Política de Privacidad (https://unipython.com/politica-de-privacidad/)
- Términos y condiciones (https://unipython.com/terminos-y-condiciones/)

PUBLICAR EL COMENTARIO

Para empresas

- Contrata a nuestros graduados (https://unipython.com/contrata-a-nuestros-graduados/)
- Servicios para empresas (https://unipython.com/contrata-nuestros-servicios/)

Síguenos

- Linkedin (https://www.linkedin.com/company/unipython/)
- Youtube (https://www.youtube.com/channel/UCe0GlySXvnv1OrLXLOoq6yA)