Programación con OPENCV

Desarrollo de prácticas en Python

Miguel Ángel Macías Narváez.
Facultad de Ingeniería, Ingeniería en Sistemas
Universidad de Cuenca
Cuenca, Ecuador
mangel.maciasn@ucuenca.edu.ec

Abstract. **-. The Internet is full of text and images. It is relatively simple to index and search text, but in order to index and search images, several algorithms need to know what the images contain. Computer vision es a field of study that allows computers to see what is on that images. In this article we test Computer Vision by practical exercises in Python with OpenCV.**

Keywords-component; OpenCV, computer vision, pixel, array, neural network, RGB, distribution.

I. INTRODUCCIÓN.

Las imágenes se han vuelto omnipresentes en informática. El procesamiento de imágenes es el proceso de crear una nueva imagen a partir de una imagen existe, que generalmente simplifica o mejora el contenido de alguna manera [1]. Es un tipo de procesamiento de señal digital y no se preocupa por comprender el contenido de una imagen. El objetivo de la visión por computador es comprender el contenido de las imágenes digitales. Por lo general, esto implica desarrollar métodos que intenten reproducir la capacidad de la visión humana.

En un principio se pensaba que la visión por computador era una tarea de conectar una cámara al computador. Luego de décadas de investigación, la visión por computador sigue sin resolverse en términos de satisfacer las capacidades de la visión humana.

Este artículo se enfoca en desarrollar ejercicios prácticos sobre la visión por computador usando la herramienta OpenCV en un ambiente de desarrollado basado en Python. En la segunda sección de este artículo se definen los conceptos necesarios para entender el uso de las herramientas. En la tercera sección se detalla la metodología empleada para el desarrollo del programa. En la cuarta sección se muestran los resultados y se finaliza con conclusiones en la quinta sección.

II. DEFINICIONES.

A. OpenCV-Python.

También llamada CV2 es la API desarrollada en Python para OpenCV. Enfocada en la visión por computador, brinda todas las herramientas necesarias para el desarrollo de programas orientados a esta área [2]. Trabaja muy bien con la librería Numpy, ya que todas las salidas de CV2 son arreglos de Numpy.

B. Numpy.

Numpy es el paquete funcional para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados, y una variedad de rutinas para operaciones rápida en matrices, que incluyen matemática, lógica, manipulación de formas, clasificación, selección, E/S, transformadas discretas de Fourier y mucho más [3].

III. METODOLOGÍA.

Las imágenes y videos que forman parte de las entradas principales para algoritmos de visión por computador, a menudo son representados en formato de matrices. En estas matrices se guardan detalles como el ancho, alto, profundidad, canal, y otras características más. Gracias a Numpy, se pueden manejar estas entradas con la ayuda de CV2.

A. Leer y cargar imágenes con OpenCV y Numpy.

En el proyecto adjunto se detallan dos carpetas, img y media, las cuales contendrán imágenes y video respectivamente. El objetivo de esta práctica es leer, mostrar, cambiar la escala de colores y guardar una imagen.

Para leer la imagen se usa el método *imread* de CV2. Ahora para poder cambiar la escala de colores de la imagen se usa *cvtColor*, y el parámetro *COLOR_BGR2GRAY* que establece la escala de colores que se va a usar, en este caso es gris.

```python
panda_image = cv2.imread("img/panda.jpg")
panda_gray_image = cv2.cvtColor(panda_image,
                                cv2.COLOR_BGR2GRAY)

cv2.imshow("Gray panda", panda_gray_image)
cv2.imshow("Color panda", panda_gray_image)
cv2.imwrite("img/gray_panda.jpg", panda_gray_image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig. 1. Lectura, representación, cambio de escala y escritura de una imagen

Para mostrar la imagen se usa el método *imshow* y para escribirla en nuestro directorio usamos *imwrite*.

B. Representar una imagen por medio de su Histograma RGB.

Leemos la misma imagen que ya usamos en el literal A. Usamos el método *calcHist* el cual se encarga de calcular el histograma de la imagen original. El segundo parámetro de este método especifica la escala que se va a calcular, en este caso es BLUE (0). Luego repetimos este proceso para todas las escalas, para esto nos ayudamos de un arreglo auxiliar que contendrá el valor de cada escala RGB.

```python
image = cv2.imread("img/panda.jpg")

# Plotting the histogram
histogram_image = cv2.calcHist([image], [0], None,
                               [256], [0,256])

# flaten the histogram
plt.hist(histogram_image.ravel(),256, [0,256])
plt.show()

# View color channels
color = ['b','g','r']

# Separate the color and plot the histogram
for i, col in enumerate(color):
    hist = cv2.calcHist([image], [i], None,
                        [256], [0,256])
    plt.plot(hist, color = col)
    plt.xlim([0, 256])

plt.show()
```

Fig. 2. Representación de una imagen por su histograma RGB.

C. Cargar un video a través de la webcam.

El ejercicio no es complejo, usamos el método *VideoCapture* para crear un objeto que permita leer los frames capturados por la webcam. Dentro de un bucle usamos *read* para poder capturar los frames y los mostramos en pantalla.

```python
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    cv2.imshow("frame", frame)
    key = cv2.waitKey(1)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Fig. 3. Captura de video usando la webcam.

D. Cargar un video desde un archivo.

Al igual que una imagen, primeramente, debemos leer el directorio donde se encuentra el video usando el método *VideoCapture*. Nuevamente dentro de un bucle leemos con *read* los frames uno por uno y los mostramos.

```python
ageofempires_video = cv2.VideoCapture("media/sundiata.mp4")

while True:
    ret, frame = ageofempires_video.read()
    cv2.imshow("frame", frame)
    key = cv2.waitKey(25)
    if key == 27:
        break

ageofempires_video.release()
cv2.destroyAllWindows()
```

Fig. 4. Reproducción de video desde un archivo.

E. Leer un video y escribirlo en un archivo.

El proceso es el mismo que el literal D, excepto que ahora hacemos unas pequeñas modificaciones en cada frame sobrescribiendo en uno nuevo para luego guardarlo.

```python
ageofempires_video = cv2.VideoCapture("media/sundiata.mp4")

fcc = cv2.VideoWriter_fourcc(*"XVID")
out = cv2.VideoWriter("media/new_sundiata.avi", fcc,
                      28, (640,360))

while True:
    ret, frame = ageofempires_video.read()
    frame2 = cv2.flip(frame, 1)
    cv2.imshow("frame2", frame2)
    cv2.imshow("frame", frame)
    out.write(frame2)
    key = cv2.waitKey(20)
    if key == 27:
        break

out.release()
ageofempires_video.release()
cv2.destroyAllWindows()
```

Fig. 6. Sobreescritura de video desde un archivo.

IV. EJECUCIÓN Y RESULTADOS.

El código y los resultados de la ejecución del algoritmo se adjuntan en las carpetas “src”, “img” y “media” respectivamente. También se puede acceder a este contenido a través del siguiente enlace: https://github.com/TheWorstOne/OpenCVbasics.

A. Leer y cargar imágenes con OpenCV y Numpy.

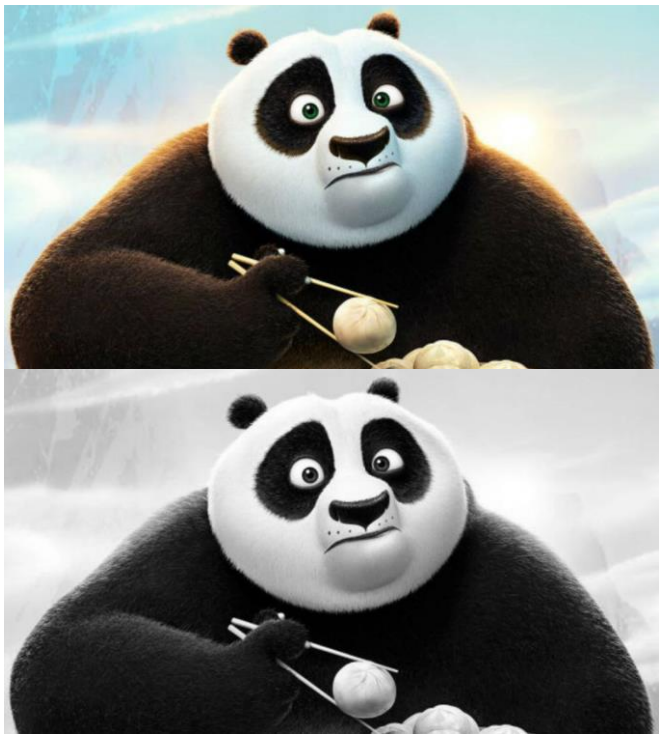

Fig. 7. Imagen original e imagen alterada a grises.

B. Representar una imagen por medio de su Histograma RGB.

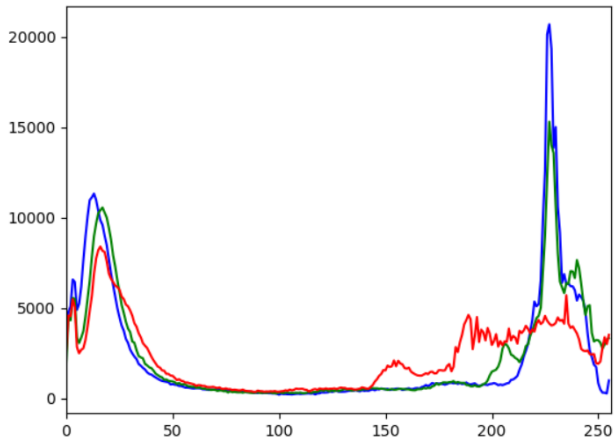

Fig. 8. Histograma RGB de la imagen original.

C. Cargar un video a través de la webcam.

Fig. 9. Captura de video a través de la webcam.

D. Cargar un video desde un archivo.

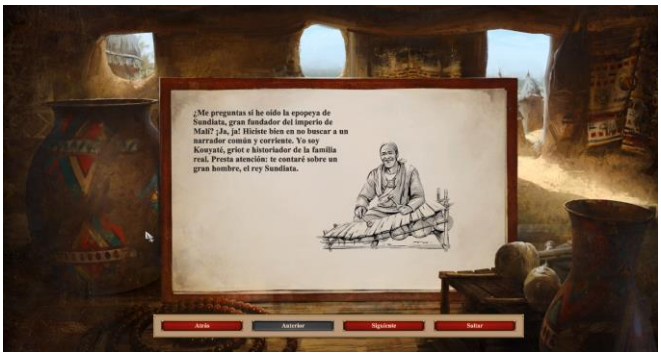

Fig. 10. Captura de video desde un archivo.

E. Leer un video y escribirlo en un archivo.

Fig. 11. Sobreescritura de video desde un archivo.

V. CONCLUSIONES.

La visión por computador es un campo realmente emocionante de la Inteligencia artificial. Conocer herramientas que permitan trabajar con imágenes y videos para obtener información de estos, o ya sea crear modelos para implementarlos en diferentes aplicaciones es realmente enriquecedor. Llevar estos conocimientos más allá del entendimiento básico nos permite crear proyectos asombrosos que puedan ser usados para resolver problemas de la era moderna actual. OpenCV sin duda es una gran herramienta, poder usar librerías como Numpy que hacen que el trabajo pesado sea más eficiente es un punto a favor para los que trabajos en Python conjuntamente con otras áreas de la ciencia, las posibilidades son muchas y esto es solo lo básico.

VI. REFERENCIAS.

[1] "Computer Vision—An Introduction", Medium, 2019. [Online]. Available: https://towardsdatascience.com/computer-vision-an-introduction-bbc81743a2f7. [Accessed: 30- Mar- 2020].

[2] "OpenCV", Opencv.org. [Online]. Available: https://opencv.org/. [Accessed: 30- Mar- 2020].

[3] "NumPy — NumPy", Numpy.org. [Online]. Available: https://numpy.org/. [Accessed: 30- Mar- 2020].