OpenCV: Introducción en Python

Cátedra de Procesamiento Digital de Imágenes, FICH-UNL
Leandro Bugnon (lbugnon@sinc.unl.edu.ar)

9 de marzo de 2018

Este documento es una breve guía para utilizar OpenCV sobre Python. Este lenguaje tiene un ecosistema con una gran variedad de herramientas para el procesamiento de imágenes y datos en general, además de la facilidad para resolver cuestiones de bajo nivel que no son el contenido central del curso. Para utilizar C++, revisar la guía *OpenCV-C++*.

1. Instalación

Se recomienda utilizar la última versión estable de OpenCV (3.4). Para instalar, se puede utilizar el repositorio oficial, con el comando:

`pip install opencv-python`

o compilando directamente el código fuente. Revisar en la documentación los requisitos mínimos. Algunas funcionalidades, como el procesamiento de video o el uso de CUDA, requerirán librerías específicas. Consultar la documentación oficial en cada caso.

Guías y documentación:

- Documentación de OpenCV 3.4
- Tutoriales OpenCV-Python
- Instalación OpenCV.

Problemas comunes:

- Versiones no reconocidas de GPUs o sus drivers: desactivar CUDA en la configuración de compilación (`cmake -D WITH_CUDA=OFF ../`)
- Las ventanas no se grafican: Revisar dependencias de GTK2.
- No funciona el video: Revisar dependencias de video como ffmpeg, gstreamer y libav-dev.

2. Ejemplo Inicial

El siguiente ejemplo muestra el uso de las funciones básicas de manipulación de imágenes. Copiar y ejecutar el código para verificar la instalación.

```
import numpy as np
import cv2 as cv

print(cv.__version__)

img1=cv.imread("camino.tif") # RGB

# Las funciones reciben banderas (numeros enteros)
img2=cv.imread("camino.tif",cv.IMREAD_GRAYSCALE) # escala de grises

cv.imshow("Titulo 1",img1)
cv.imshow("Titulo 2",img2)
cv.waitKey(0)
cv.destroyAllWindows()

# Usando la misma ventana
cv.namedWindow("imagen")
cv.imshow("imagen",img1)
cv.waitKey(0)
cv.imshow("imagen",img2)
cv.waitKey(0)
cv.destroyAllWindows()

# Crear una imagen vacía (en este caso una máscara). Las imágenes se representan
# en matrices de Numpy de forma [alto,ancho,Ncanales], con un tipo de dato determinado
# (flotante,entero,booleano, etc.), por tanto se pueden utilizar todas las propiedades
# de Numpy.

H,W,C=img1.shape
imtype=img1.dtype
print("Imágen de tipo %s, alto=%d, ancho=%d, ncanales=%d" %(imtype,H,W,C))


mask=np.zeros((H,W,C),dtype=imtype) # dtype: 'int8', bool, float, ...

# O bien:
# mask=img1.copy()
# mask[:]=0
# Las imágenes son objetos y para hacer copias se debe tener en cuenta que el operador
# de asignación ("=") copia también el puntero al objeto.

mask[int(H/3):int(2*H/3),int(W/3):int(2*W/3),:]=255  # Máscara rectangular, con valores [0,255]

# operación AND para cada pixel
img3=cv.bitwise_and(img1,mask)
```

```
# Guardar la nueva imagen
cv.imwrite("nueva.jpg",img3)

# Capturar imágenes desde un stream de video.

#origen='archivo'
origen='webcam'

if origen=="archivo":
    cam=cv.VideoCapture("video.mp4")

# Video: a partir de càmara habilitada por defecto
if origen=='webcam':
    cam=cv.VideoCapture(0)

cv.namedWindow("Video")

# capturar 10 frames
for k in range(10):

    isok,frame=cam.read()

    if isok:
        cv.imshow("Video",frame)
        cv.waitKey(0)

cam.release() # liberar stream
```