Python y OpenCV

José Manuel Fuertes García
Manuel J. Lucena López

Departamento de Informática
Universidad de Jaén

4 de marzo de 2015

Lenguaje de programación Python

- Lenguaje de programación interpretado, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
- Creado a finales de los ochenta por Guido van Rossum.
- Soporta orientación a objetos, programación imperativa y algo de programación funcional.
- Es multiplataforma, y usa tipado dinámico.
- Fácilmente extensible en C y C++.

Instalación de Python

- Preinstalado en prácticamente todas las distribuciones de Linux.
- Existen múltiples distribuciones tanto en código fuente como en binario para Windows, Linux y MacOS.
- La distribución *Anaconda* contiene además bibliotecas numéricas y científicas como NumPy y SciPy
  `https://store.continuum.io/cshop/anaconda/`

El modo interactivo de Python

- Python incluye un modo interactivo que permite interpretar comandos y evaluar expresiones directamente.
- Permite comprobar porciones de código, examinar la documentación, etc.

Ejemplos:

- `1+1`
- `a = range(10)`
- `import cv2`
- `import nunmpy`
- `help(cv2)`

Particularidades de Python

- Palabras reservadas *legibles*: `True`, `False`, `and`, `or`, `not`...
- La estructura básica es la *lista*.
- Los bloques de código se especifican por su indentación.
- Autodocumentado: `help(objeto)`.

Listas en Python

- Lista de enteros de 0 a 9: `A=range(10)`
- Lista de enteros en saltos de 2: `range(0,11,2)`
- Longitud de `A`: `len(A)`
- Primer elemento de `A`: `A[0]`
- Último elemento: `A[len(A)-1]` o `A[-1]`
- Sublista: `A[1:5]`
- Concatenar listas: `A+B`
- ¿Pertenece el 3 a `A`?: `3 in A`.
- Filtros: `[x for x in A if x% 2 == 0]`

Estructuras básicas de control en Python

```
## Condicional
if (A and B):
    print("A y B son ciertos")
else:
    print("A o B son falsos")

## Ciclos
for i in range(10):
    print(i)

while (x):
    print("La condicion se cumple")
```

Funciones en Python

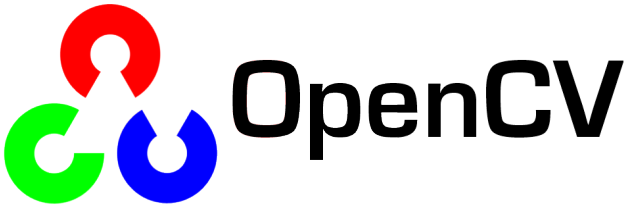
- Puede tener valores por defecto para sus parámetros.
- Puede devolver más de un valor.

```
def funcion(par1, par2, par3 = defecto):
    """ Descripcion de la funcion
    """

    result1 = par1 + par2 ^ par3
    result2 = par1 - par2

    return (result1, result2)
```

Módulos en Python

- Cargar un módulo:
  `import cv2`
- Acceder a un objeto del módulo:
  `cv2.imread('imagen.png')`
- Cargar un módulo con un *alias*:
  `import numpy as np`
- Cargar elementos concretos que no necesitarán prefijo:
  `from sys import exit`

Biblioteca OpenCV



- Orientada a procesamiento de imágenes.
- Licencia BSD.
- Más de 2500 algoritmos optimizados.
- Interfaces para C, C++, Python, Java y Matlab.
- Funciona en Windows, Linux, MacOS y Android.

Instalación de Python-OpenCV

- Instrucciones (página oficial)
  - Es necesario configurar las variables de entorno `OPENCV_DIR` y `PATH` de la siguiente forma:
    ```
    set OPENCV_DIR=c:\opencv\build\x64\vc10
    set PATH=%PATH%;%OPENCV_DIR%\bin
    ```
  - Puede hacerse mediante un fichero `.bat` que ha de ejecutarse en la misma consola en la que se vayan a ejecutar los scripts.
- Suele venir empaquetado para Linux.
- Es necesario tener el paquete de Python *NumPy*.

Ejemplo de programa sencillo en OpenCV

```
import numpy as np
import cv2, sys

img = cv2.imread(sys.argv[1])

if (img is None):
    print("Error al cargar imagen")
    sys.exit()

r = cv2.imshow('image',img)

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

Operaciones con imágenes en OpenCV

- Lectura: `im = cv2.imread(nombre, modo)`
  - El parámetro `modo` es opcional.
    - `cv2.IMREAD_COLOR` lee la imagen en color.
    - `cv2.IMREAD_GRAYSCALE` lee la imagen en tonos de gris.
    - `cv2.IMREAD_UNCHANGED` lee también el canal alfa.
  - Devuelve un array NumPy de 2 o 3 dimensiones.
  - Las imágenes en color se representan en B-G-R.
- Dimensiones de la imagen: `im.shape`
  - Devuelve `(filas, columnas, bandas)`
- Espacio que ocupa y tipo base: `im.size`, `im.dtype`
- Separación y mezcla de canales:

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

Acceso a las imágenes en OpenCV

- Leer el valor de un píxel:
  - Tonos de gris: `im.item(y, x)`
  - Componente azul: `im.item(y, x, 0)`
  - Componente verde: `im.item(y, x, 1)`
  - Componente roja: `im.item(y, x, 2)`
- Modificar el valor de un píxel:

```
im.itemset((y, x), valor)
im.itemset((y, x, banda), valor)
```

- Extracción de subregiones:

```
region = im[100:200, 120:180]
```

- Copia de bloques de la imagen:

```
im2[70:170, 30:90] = region
```

- Poner a 5 el valor rojo de todos los píxeles de una imagen:

```
im2[:, :, 2] = 5
```

- El acceso a través de NumPy se hace indicando primero la coordenada `y`, mientras que a través de `cv2` se hace indicando primero la `x`.

Funciones básicas de dibujo en OpenCV

- Crear una imagen:

```
img = np.zeros((alto, ancho, bandas), np.uint8)
```

- Grabar una imagen:

```
cv2.imwrite('archivo.png',img)
```

- Parámetros comunes:
  - `color`: El color en el que se quiere dibujar. Puede ser una terna BGR (0, 255, 0) o un escalar.
  - `grueso`: El grosor de la línea. Si es -1, rellena las formas cerradas.
  - `tipo`: Línea 4 u 8-conectada. Con `cv2.CV_AA` dibuja la línea con *anti-aliasing*.
  - `coordenadas`: Se representan con un par `(x, y)`.

Funciones básicas de dibujo en OpenCV (2)

- Dibujar una línea:

  ```
  cv2.line(imagen, (x1, y1), (x2, y2), color, grueso)
  ```

- Dibujar un rectángulo:

  ```
  cv2.rectangle(imagen, (x1, y1), (x2, y2), color, grueso)
  ```

- Dibujar una circunferencia:

  ```
  cv2.circle(imagen, (cx, cy), radio, color, grueso)
  ```

- Dibujar una elipse:

  ```
  cv2.ellipse(imagen, (cx, cy), (radioX, radioY),
  rotacion, anguloInicio, anguloFin, color, grueso)
  ```

Funciones básicas de dibujo en OpenCV (3)

- Dibujar una polilínea:

```
pts = np.array([[10,5], [20,30], [70,20], [50,10]],
np.int32)
pts = pts.reshape((-1,1,2))
cv2.polylines(img, [pts], True, color, tipo)
```

  - El tercer parámetro indica si la polilínea es cerrada (`True`) o abierta (`False`).

- Escribir texto:

```
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'OpenCV',(x, y), font, escala, color,
grueso ,cv2.LINE_AA)
```

- Saber el tamaño de una caja de texto:

```
size, baseline = cv2.getTextSize('openCV', font, escala,
grueso)
```

Mostrar un vídeo

```
import cv2, numpy as np

cap = cv2.VideoCapture('video.avi')

while(cap.isOpened()):
    ret, frame = cap.read()

    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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