

# Visión por Computador (1782)

---

Herramientas de programación de aplicaciones  
OpenCV - Python

Luis M. Jiménez



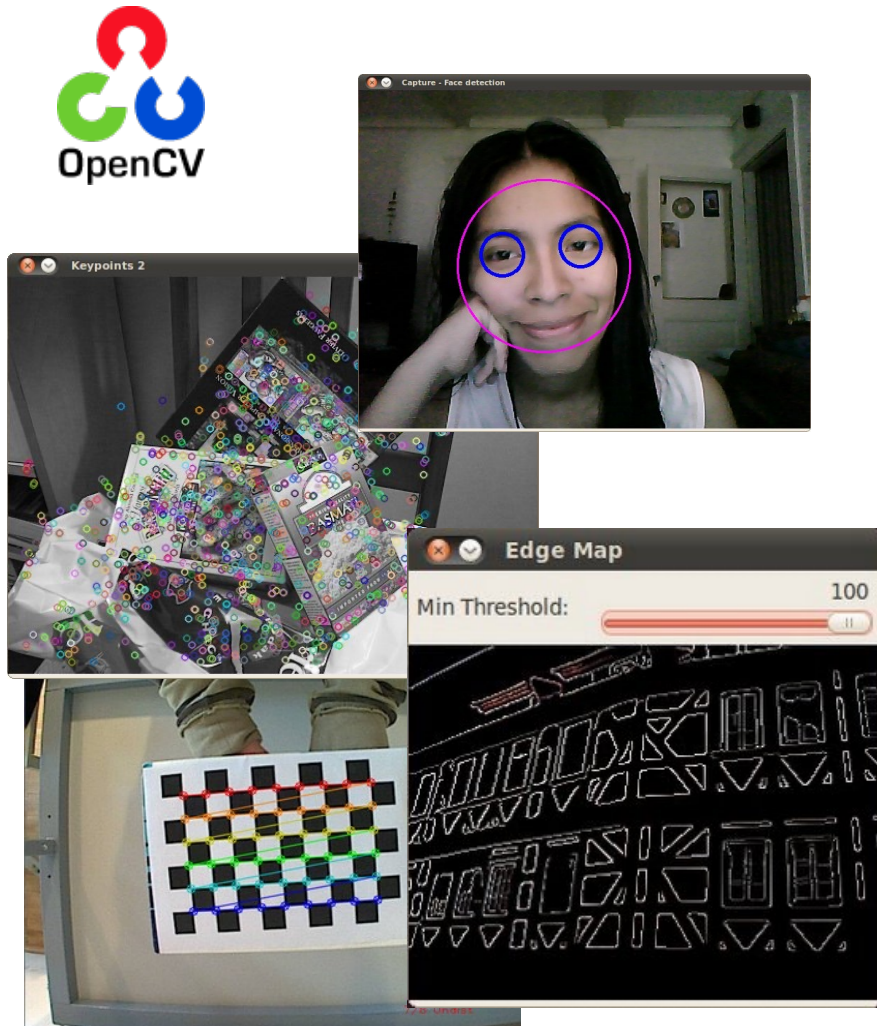
Lab. Automática, Robótica y Visión por Computador  
Universidad Miguel Hernández  
<http://arvc.umh.es>



# Prácticas

---

- OpenCV: <http://opencv.org>



- Multiplataforma:  
Windows/Linux/OSX  
Android/iOS
- Interfaz: C/C++/Java/Python
  
- Adquisición imágenes/video
- Procesamiento 2D
- Extracción características
- Machine Learning
- Reconocimiento - Clasificación
- Aceleración GPU
- Calibración 3D
- Localización - Reconstrucción 3D

# Instalación OpenCV

---

- Windows 7-11 / Mac
  - Interprete Python: (3.11) <https://www.python.org/>
  - Entorno de Desarrollo:
    - **PyCharm**: editor y entorno de desarrollo
    - **Anaconda**: interfaz para **conda** y herramientas desarrollo (*Spyder*)
  - Librería OpenCV (4.10): gestores de paquetes python
    - *pip install opencv-python*
    - *venv*
    - *conda*
- Linux
  - Utilizar un gestor de paquetes: *apt-get*
  - Instalación de módulos Python: *pip*
- Tutoriales:
  - <http://umh1782.edu.umh.es/python/>

# Programación en OpenCV (Python)

---

- Librería compilada en C++:
  - Código optimizado y rápido
  - Permite aceleración por GPU
- Paquete Opencv-python:
  - Interfaz (*Wrapper*) Python para la librería Opencv
  - No implementa todos los elementos de la librería
    - Las estructuras de datos de imagen y algunas funciones auxiliares (módulo **core**) se implementan mediante otros paquetes Python (**numpy**)
    - Las funciones y clases que implementan los algoritmos de visión, sí disponen de una función equivalente en Python.
- Importación Librería OpenCV

```
import cv2 as cv
```

```
import numpy as np
```

# Programación en OpenCV (Python)

- Estructuras de Datos/Clases básicas:

## C++

**cv::Mat:** N-dimensional array, para almacenar imágenes

**Mat.at<datatype>(row, col)[channel]** – acceso al valor de un pixel

**Mat.clone()** - copia de la matriz

**Mat.copyTo( <Mat>)** – copia de la matriz

**Mat.size()** – Devuelve el tamaño

**Mat.empty()** – indica si la matriz está vacía

**cv::Range** rangos de filas o columnas

**cv::Size** Tamaño de una imagen

**cv::Point, Point2f, Point3f** (coordenadas)

**cv::Vec (template): Vec3b, Vec3s, Vec3f,**

**cv::Vec3d** .... valores de un pixel (multicanal)

**cv::Scalar** equivalente a **cv::Vec4d**

**std::vector:** clase de la librería estándar *std*  
listas de puntos

## Python

**Numpy:** clase ndarray

Use the following import convention:

```
>>> import numpy as np
```

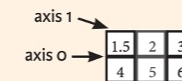


### NumPy Arrays

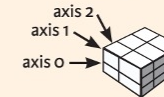
**1D array**



**2D array**



**3D array**



### Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                dtype = float)
```

### Initial Placeholders

<pre>&gt;&gt;&gt; np.zeros((3,4)) &gt;&gt;&gt; np.ones((2,3,4),dtype=np.int16) &gt;&gt;&gt; d = np.arange(10,25,5)  &gt;&gt;&gt; np.linspace(0,2,9)  &gt;&gt;&gt; e = np.full((2,2),7) &gt;&gt;&gt; f = np.eye(2) &gt;&gt;&gt; np.random.random((2,2)) &gt;&gt;&gt; np.empty((3,2))</pre>	<p>Create an array of zeros</p> <p>Create an array of ones</p> <p>Create an array of evenly spaced values (step value)</p> <p>Create an array of evenly spaced values (number of samples)</p> <p>Create a constant array</p> <p>Create a 2X2 identity matrix</p> <p>Create an array with random values</p> <p>Create an empty array</p>
---	---

**Tuplas:** (start, end) (width, height)

(x, y) (x, y, z) (b, g, r) - no mutable

**Listas:** [x, y] [x, y, z] [width, height]

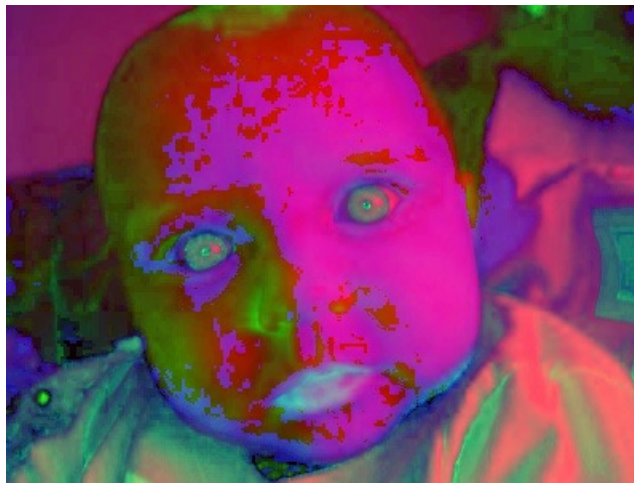
**Diccionarios:** { 'x':10, 'y':20 } → list(), tuple()

# Programación en OpenCV (C++)

---

- **Formatos de Imagen:**

- **BGR** – formato por defecto de *imread()*. 3 canales de color
- **HSV** - Hue, Saturation, Value is lightness. 3 canales
- **GRAYSCALE** – Nivel de gris. 1 canal



# EJEMPLOS

---

<http://umh1782.edu.umh.es/python/>

# EJEMPLO 1

---

Capturar y visualizar la imagen de una cámara

[https://docs.opencv.org/4.10.0/dd/de7/group\\_\\_videoio.html](https://docs.opencv.org/4.10.0/dd/de7/group__videoio.html)

[https://docs.opencv.org/4.10.0/dd/d43/tutorial\\_py\\_video\\_display.html](https://docs.opencv.org/4.10.0/dd/d43/tutorial_py_video_display.html)

# Captura de Imágenes de una Cámara: módulo video I/O

---

- Primer ejemplo (**e1.py**): capturar y visualizar la imagen de una cámara
- Clase **cv.VideoCapture**
  - Constructores:
    - **cv.VideoCapture( )** → <VideoCapture object> : crea objeto
    - **cv.VideoCapture( index )** → <VideoCapture object> : abre el dispositivo (id de cámara – *int* )
    - **cv.VideoCapture( filename )** → <VideoCapture object> : abre dispositivo (fichero video/sec - *str*)
  - Métodos:
    - **cv.VideoCapture.open( index )** → bool : abre el dispositivo (id de cámara)
    - **cv.VideoCapture.open( filename )** → bool : abre un fichero de video/sec. imágenes
    - **cv.VideoCapture.isOpened( )** → bool
    - **cv.VideoCapture.read( [, image] )** → bool, image : captura una imagen y la copia en una Matriz
    - **cv.VideoCapture.get( propId )** → retval : lee el valor de una propiedad de la cámara/fichero
    - **cv.VideoCapture.set( propId, value )** → bool : configura el valor de una propiedad de la cámara
    - **cv.VideoCapture.getExceptionMode( )** → bool : rise exceptions / error code
    - **cv.VideoCapture.setExceptionMode( enable )** : rise exceptions /error code
    - **cv.VideoCapture.release( )** : Libera el dispositivo

# Captura de Imágenes de una Cámara: módulo highgui

---

- Primer ejemplo (**e1.py**): capturar y visualizar la imagen de una cámara
- Ventana visualización:
  - `cv.namedWindow( winname [, flags=cv.WINDOW_AUTOSIZE] )` : crea una ventana
  - `cv.imshow( winname, mat )` : visualiza una imagen en una ventana
  - `cv.setWindowTitle( winname, title )` : titulo de la Ventana (por defecto: *winname*)
  - `cv.destroyAllWindows( )`
  - `cv.destroyWindow( winname )`
- Eventos del teclado:
  - `cv.waitKey( [, delay=0] )` → `retval` espera (en milisegundos) la pulsación de una tecla
  - `cv.pollKey()` → `retval` comprueba si se ha pulsado un tecla desde la última llamada

# Captura de Imágenes de una Cámara: módulo highgui /IO

- Primer ejemplo (**e1.py**): capturar y visualizar la imagen de una cámara

```
# Import libraries
import cv2 as cv
import numpy as np
```

```
WINDOW_CAMERA1 = '(W1) Camera 1' # window id
CAMERA_ID = 0 # default camera
```

```
# Open camera object
camera = cv.VideoCapture(CAMERA_ID)
if not camera.isOpened():
    print("you need to connect a camera, sorry.")
    exit()

# Getting camera resolution
cameraWidth = camera.get(cv.CAP_PROP_FRAME_WIDTH)
cameraHeight = camera.get(cv.CAP_PROP_FRAME_HEIGHT)

// Create the visualization windows
cv.namedWindow (WINDOW_CAMERA1, cv.WINDOW_AUTOSIZE);

print(f"Capturing images from camera {CAMERA_ID} ({int(cameraWidth)}, {int(cameraHeight)})")
print("...Hit q/Q/Esc to exit.")
```

# Captura de Imágenes de una Cámara: módulo highgui /IO

---

- Primer ejemplo (**e1.py**): capturar y visualizar la imagen de una cámara

```
# while there are images ...
while True:
    ret, capture = camera.read()    # Capture frame-by-frame

    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break

    cv.imshow(WINDOW_CAMERA1, capture)    # Display the resulting frame

    # check keystroke to exit (image window must be on focus)
    key = cv.pollKey()
    if key == ord('q') or key == ord('Q') or key == 27:
        break

# End while (main loop)
```

```
# free windows and camera resources
cv.destroyAllWindows()
if camera.isOpened(): camera.release()
```

# Captura de Imágenes de una Cámara: módulo highgui /IO

---

- Primer ejemplo (**e1.py**): capturar y visualizar la imagen de una cámara
  - Añadir Command-Line Parser:

```
# Import libraries
import argparse
```

```
# check command line parameters (camera id)
parser = argparse.ArgumentParser(description='OpenCV example: camera capture')

parser.add_argument('-c', dest='cameraID', type=int, default=CAMERA_ID,
                    metavar='id', help='camera id')

CAMERA_ID = parser.parse_args().cameraID
```

# EJERCICIO 1b

---

## Filtrado de Imágenes: Convolución (e1b.py)

- Incorporar al ejemplo previo el procesamiento de la imagen capturada y visualizar el resultado

[https://docs.opencv.org/4.10.0/d4/d86/group\\_imgproc\\_filter.html](https://docs.opencv.org/4.10.0/d4/d86/group_imgproc_filter.html)

[https://docs.opencv.org/4.10.0/d4/d13/tutorial\\_py\\_filtering.html](https://docs.opencv.org/4.10.0/d4/d13/tutorial_py_filtering.html)

# Filtrado de Imágenes: módulo imgproc

---

- Ejemplo (**e1b.py**): filtrado de imágenes (máscaras de convolución)
- Filtrado 2D (convolución):

- `cv.filter2D( src, ddepth, kernel [, dst, [, anchor=(-1,-1) [, delta=0.0 [, borderType=cv::BORDER_DEFAULT ]]]] ) → dst`

- **ddepth** : -1 (misma profundidad que src). `cv.CV_32F`, `cv.CV_64F`, `cv.CV_16S`

- **anchor** : centro del kernel (-1,-1) -> centro de la máscara

- **delta**: valor añadido al resultado

- **borderType**: tipo de extrapolación en los bordes

- `cv.BORDER_TRANSPARENT`, `cv.BORDER_WRAP`, `cv.BORDER_REFLECT_101`,  
`cv.BORDER_REPLICATE`, `cv.BORDER_CONSTANT`

- Conversión de Color:

- `cv.cvtColor( src, code [,dst, [, dstCn=0]] ) → dst`

- Códigos: `cv.COLOR_BGR2GRAY`, `cv.COLOR_GRAY2BGR`

- Códigos: `cv.COLOR_BGR2HSV`, `cv.COLOR_HSV2BGR` .....

# Filtrado de Imágenes: módulo imgproc

---

- Ejemplo (**e1b.py**): filtrado de imágenes (máscaras de convolución)

```
# Convolution mask
kernel = np.array( [[1.0, -2.0, 1.0],
                   [2.0, -4.0, 2.0],
                   [1.0, -2.0, 1.0] ] )
print(f"{kernel = }")
```

```
# transform to gray level
gray_image = cv.cvtColor( capture, cv.COLOR_BGR2GRAY )

# Apply filter to image
filtered_image = cv.filter2D( gray_image, -1 , kernel )
```

# EJEMPLO 2

---

Leer, Procesar y Guardar imágenes en un fichero

[https://docs.opencv.org/4.10.0/d4/da8/group\\_imgcodecs.html](https://docs.opencv.org/4.10.0/d4/da8/group_imgcodecs.html)

[https://docs.opencv.org/4.10.0/dd/d1a/group\\_imgproc\\_feature.html](https://docs.opencv.org/4.10.0/dd/d1a/group_imgproc_feature.html)

[https://docs.opencv.org/4.10.0/db/deb/tutorial\\_display\\_image.html](https://docs.opencv.org/4.10.0/db/deb/tutorial_display_image.html)

# Lectura/Escritura Imágenes: módulos highgui - imgproc

---

- Segundo ejemplo (**e2.py**): lee y procesa una imagen de un fichero
- Leer/Escribir un fichero de imagen:
  - Métodos:
    - `cv.imread( filename [, flags=cv.IMREAD_COLOR ] )` → image
      - **flags** →. `cv.IMREAD_COLOR`, `cv.IMREAD_GRAYSCALE`, ...
    - `cv.imwrite( filename, img [, params] )` → bool
- Conversión de Color:
  - `cv.cvtColor( src, code [,dst, [, dstCn=0]] )` → dst
    - Códigos: `cv.COLOR_BGR2GRAY`, `cv.COLOR_GRAY2BGR`
    - Códigos: `cv.COLOR_BGR2HSV`, `cv.COLOR_HSV2BGR` .....
- Procesamiento:
  - `cv.Canny( image, threshold1, threshold2 [, edges [, apertureSize=3 [, L2gradient=false]]] )`  
→ edges
    - Detector de Bordes

# Lectura/Escritura Imágenes: módulos highgui - imgproc

- Segundo ejemplo (**e2.py**): lee y procesa una imagen de un fichero

```
WINDOW_IMAGE = '(W1) Image'           # window id
WINDOW_BORDERS = '(W2) Canny Borders'  # window id
IMAGE_FILE = 'building.jpg'           # default image file

# Create the visualization windows
cv::namedWindow (WINDOW_IMAGE,  cv::WINDOW_AUTOSIZE)
cv::namedWindow (WINDOW_BORDERS, cv::WINDOW_AUTOSIZE)
```

```
# Open Image File
image = cv.imread( IMAGE_FILE )
if image is None:
    print(f"I can't open {IMAGE_FILE} file, sorry.")
    exit()

gray_image = cv.cvtColor( image, cv.COLOR_BGR2GRAY )    # transforms to gray level
borders_image = cv.Canny( gray_image, threshold1=80, threshold2=150 ) # Canny detector
cv.imwrite( "result.jpg", borders_image ) # store result image

cv.imshow( WINDOW_IMAGE, image ) # show image in a window
cv.imshow( WINDOW_BORDERS, borders_image ) # show image in a window
```

```
# waits for keystroke to exit (image window must be on focus)
key = cv.waitKey(0)
cv.destroyAllWindows()
```

# Captura de Imágenes de una Cámara: módulo highgui

---

- Segundo ejemplo (**e2.py**): lee y procesa una imagen de un fichero
  - Añadir Command-Line Parser:

```
# Import libraries  
import argparse
```

```
# check command line parameters (ImageFile)  
parser = argparse.ArgumentParser( description='OpenCV example: image file processing')  
  
parser.add_argument( 'imageFile', nargs='?', default=IMAGE_FILE, help='image file' )  
  
IMAGE_FILE = parser.parse_args().imageFile
```

# EJERCICIO 2b

---

- Leer imágenes de un fichero de video
  - Clase: cv.**VideoCapture**
- Detectar bordes
  - Método: cv.**Canny**
- Guardar el resultado en un fichero de video
  - Clase: cv.**VideoWriter**

[https://docs.opencv.org/4.10.0/dd/de7/group\\_\\_videoio.html](https://docs.opencv.org/4.10.0/dd/de7/group__videoio.html)

[https://docs.opencv.org/4.10.0/dd/d43/tutorial\\_py\\_video\\_display.html](https://docs.opencv.org/4.10.0/dd/d43/tutorial_py_video_display.html)

# Captura/Escritura Fichero de Video: módulo video I/O

---

- Ejercicio (**e2b.py**): Leer, procesar y almacenar desde un video
- Clase **cv.VideoCapture**
  - Constructores:
    - **cv.VideoCapture( )** → <VideoCapture object> : crea objeto
    - **cv.VideoCapture( filename )** → <VideoCapture object> : abre dispositivo (fichero video/sec - *str*)
- Clase **cv.VideoWriter**
  - Constructores:
    - **cv.VideoWriter( )** → <VideoWriter object> : crea objeto
    - **cv.VideoWriter( filename, fourcc, fps, frameSize [, isColor] )** → < VideoWriter object> :  
abre dispositivo (fichero video/sec - *str*)

**fourcc:** **cv.VideoWriter.fourcc( 'D', 'I', 'V', 'X' )**

**frameSize:** (width, height) <https://www.fourcc.org/>
  - Métodos:
    - **cv.VideoWriter.open(filename, fourcc, fps, frameSize [, isColor] )** → bool : abre fichero video
    - **cv.VideoWriter.isOpened( )** → bool
    - **cv.VideoWriter.write( image )** : guarda una imagen
    - **cv.VideoWriter.release( )** : Libera el dispositivo

# Captura/Escritura Fichero de Video: módulo video I/O

---

- Ejercicio (**e2b.py**): partiremos de e1.py

```
VIDEO_FILE = 'video.mp4'           # default video file

# Open input video object
inputVideo = cv.VideoCapture(VIDEO_FILE)

if not inputVideo.isOpened():
    print("I cannot open {VIDEO_FILE} video file, sorry.")
    exit()

# Getting video resolution / FPS
cameraWidth = int(inputVideo.get(cv.CAP_PROP_FRAME_WIDTH))
cameraHeight = int(inputVideo.get(cv.CAP_PROP_FRAME_HEIGHT))
fps = inputVideo.get(cv.CAP_PROP_FPS);

# Open output video object
outputVideo = cv.VideoWriter("result.avi", cv.VideoWriter_fourcc('D', 'I', 'V', 'X'),
                             fps, (cameraWidth, cameraHeight), False)

if not outputVideo.isOpened():
    print("I cannot open the video output file, sorry.")
    exit()
```

# Captura/Escritura Fichero de Video: módulo video I/O

---

- Ejercicio(**e2b.py**): lee y procesa una imagen de un fichero de video

```
# while there are images ...
while True:
    ret, capture = inputVideo.read() # Capture frame-by-frame
    if not ret:
        print("Stream ended. Exiting ...")
        break

    gray_image = cv.cvtColor(capture, cv.COLOR_BGR2GRAY); # transform to gray level
    borders_image = cv.Canny(gray_image, threshold1=80, threshold2=150) # Apply filter

    outputVideo.write(borders_image) # writes image to video file

    cv.imshow(WINDOW_CAMERA1, capture) # Display the resulting frame
    cv.imshow(WINDOW_BORDERS, borders_image) # Display the resulting frame

    key = cv.pollKey()
    if key == ord('q') or key == ord('Q') or key == 27: break

# End while (main loop)
```

```
cv.destroyAllWindows()
if inputVideo.isOpened(): inputVideo.release()
if outputVideo.isOpened(): outputVideo.release()
```