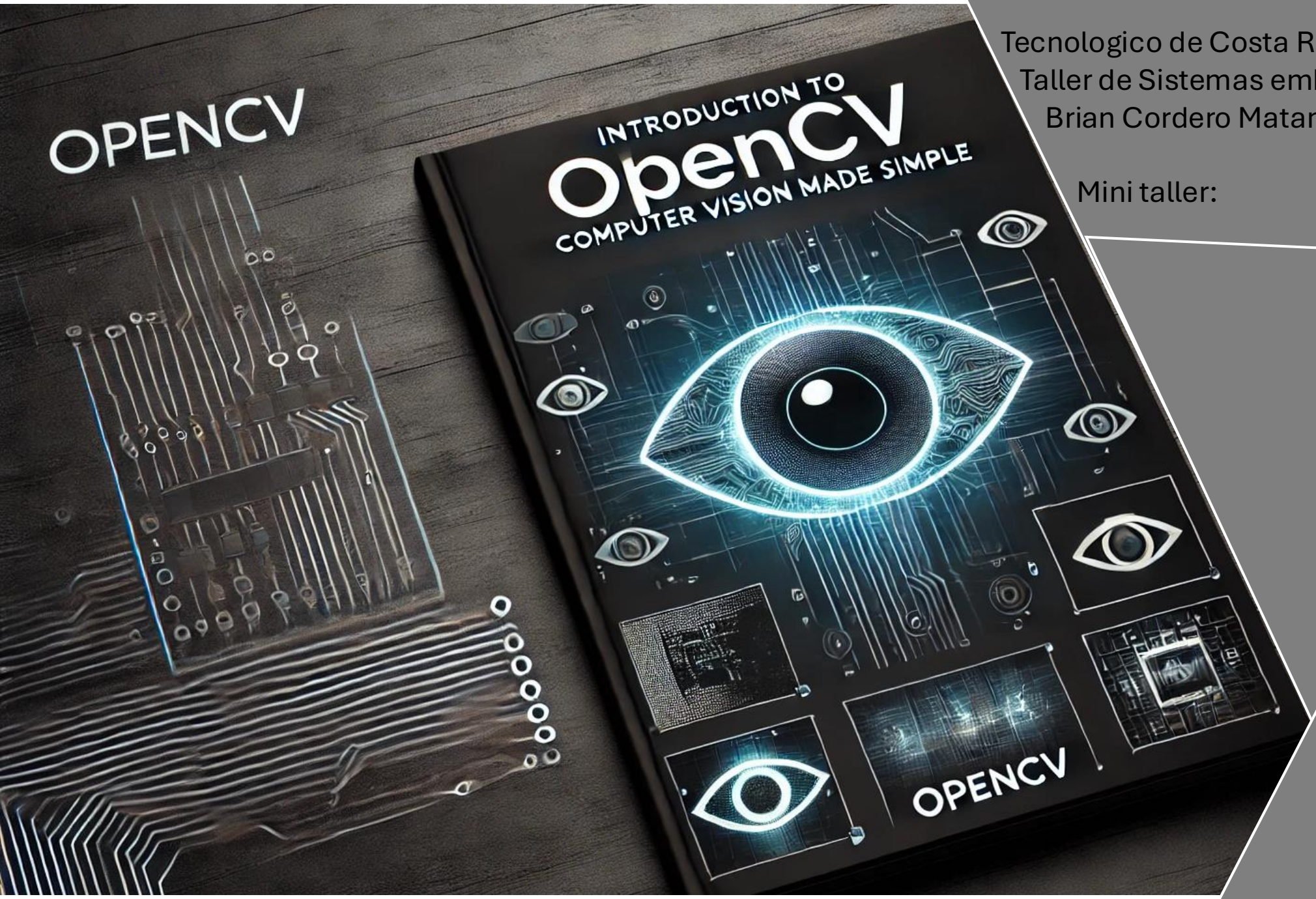


Tecnologico de Costa Rica

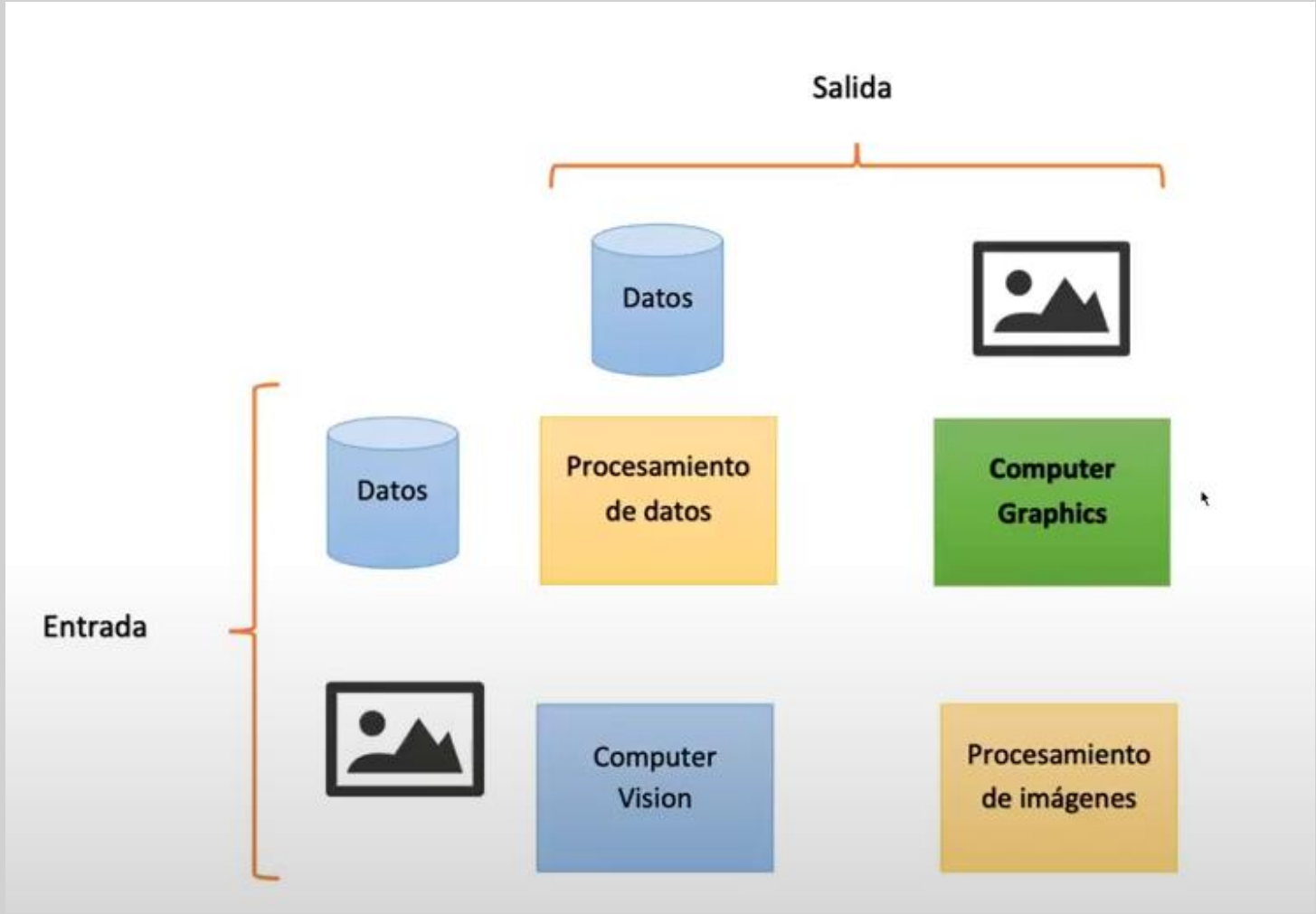
Taller de Sistemas embebidos

Brian Cordero Matamoros

Mini taller:

OPENCV

Graficos por computador



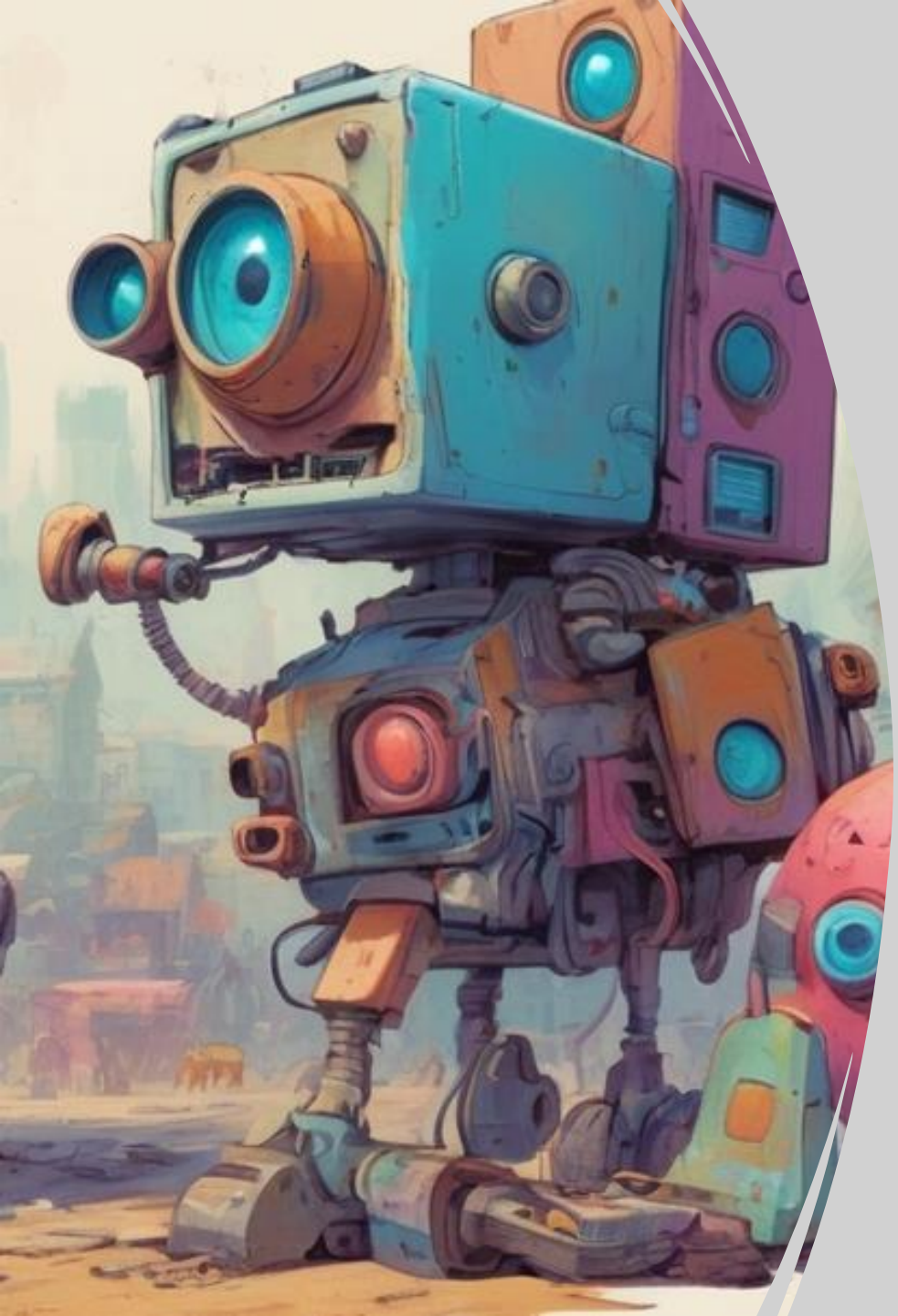
Otras interfaces de vision por computador


```
import subprocess


input_video = 'input.mp4'
output_video =
    'output_gray.mp4'


cmd = [
   'ffmpeg', '-i', input_video,
    '-vf', 'format=gray',
    output_video
]


subprocess.run(cmd)
```


```
import gi
gi.require_version('Gst', '1.0')
from gi.repository import Gst

Gst.init(None)
pipeline =
Gst.parse_launch("v4l2src !
videoconvert ! videobalance
saturation=0.0 ! autovideosink")
pipeline.set_state(Gst.State.PLA
YING)

try:
   loop = True
   while loop:
      pass
except KeyboardInterrupt:
   pipeline.set_state(Gst.State.N
ULL)
```

```
from PIL import
   Image


img =
   Image.open('input_i
   mage.jpg')
gray_img =
   img.convert('L')
gray_img.save('outpu
   t_gray_image.jpg')
```


```
import cv2
import mediapipe as mp
cap = cv2.VideoCapture(0)
while cap.isOpened():
   ret, frame = cap.read()
   if not ret:
      break
   gray_frame =
cv2.cvtColor(frame,
cv2.COLOR_BGR2GRAY)
   cv2.imshow('Grayscale
Video', gray_frame)
   if cv2.waitKey(1) & 0xFF ==
ord('q'):
      break
cap.release()
cv2.destroyAllWindows()
```



Caracteristicas

- Principios
- Lenguajes
- Librerias
- Principales Usos
- Funciones
- Demostracion

Principios

OpenCV (Open Source Computer Vision Library) se inició como un proyecto en Intel en 1999. El objetivo principal detrás de su desarrollo era proporcionar una infraestructura común para aplicaciones de visión por computadora y acelerar el uso de la percepción por parte de las computadoras.

OpenCV fue lanzado oficialmente en 2000 como un proyecto de código abierto bajo la licencia BSD. Esto permitió a los desarrolladores de todo el mundo utilizarlo gratuitamente y contribuir a su desarrollo. El enfoque principal estaba en el procesamiento en tiempo real y la optimización del rendimiento en arquitecturas de hardware x86.

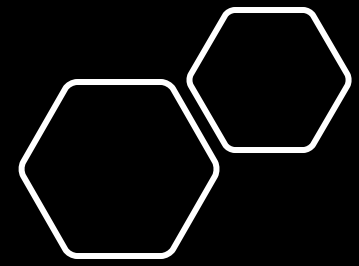
Lenguajes

Python	C	Java

import cv2 # Cargar la imagen desde #el disco
image = cv2.imread('imagen.jpg')
# Verificar que la imagen se ha #cargado correctamente 	
if image is None: print("No se pudo cargar la imagen.")
else: # Convertir la imagen a escala #de grises
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
	#include <opencv2/opencv.hpp>
	int main() {
	// Cargar la imagen desde el disco
	cv::Mat image = cv::imread("imagen.jpg");
	// Verificar si la imagen ha sido //cargada correctamente
	if(image.empty()) { printf("No se pudo cargar la imagen.\n");
	return -1; }
	// Convertir la imagen a escala de grises cv::Mat gray_image;
	cv::cvtColor(image, gray_image, cv::COLOR_BGR2GRAY);
		import org.opencv.core.Core;
		.....
		import org.opencv.highgui.HighGui;
		public class OpenCVExample { public static void main(String[] args) {
		// Cargar la biblioteca nativa de OpenCV
		System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
		// Cargar la imagen desde el disco Mat image = Imgcodecs.imread("imagen.jpg");
		// Crear un objeto Mat para almacenar //la imagen en escala de grises Mat grayImage = new Mat(image.size(), CvType.CV_8UC1);
		// Convertir la imagen a escala de grises Imgproc.cvtColor(image, grayImage, Imgproc.COLOR_BGR2GRAY);

En python por lo general se acompana de las siguientes librerias:

- NumPy
- CUDA (Compute Unified Device Architecture)
- OpenCL (Open Computing Language)
- Tesseract
- OpenVINO (Open Visual Inference and Neural Network Optimization)



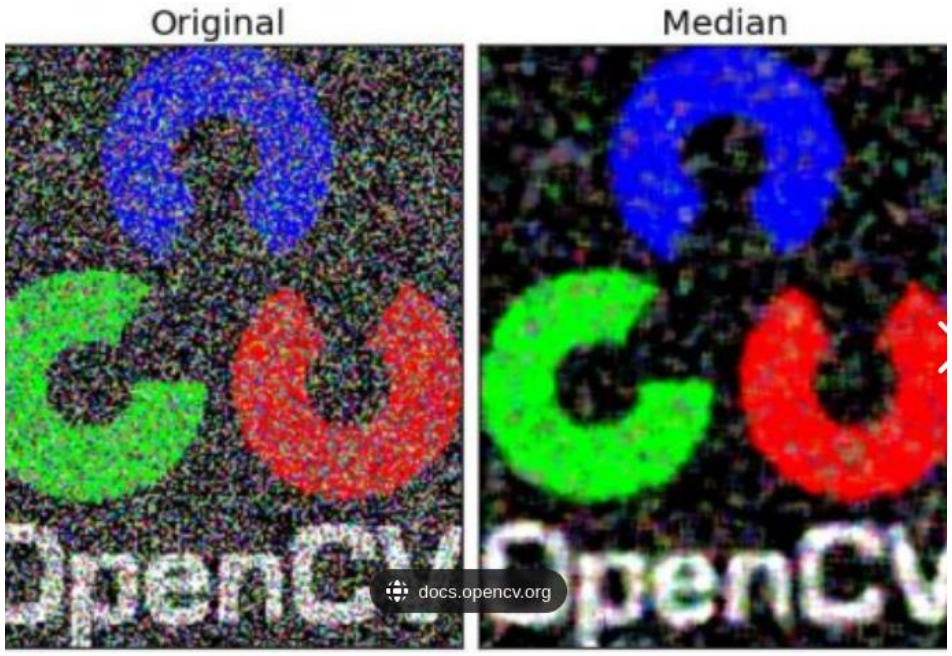
Principales Usos

Image Filtering	Color Space Conversions	Structural Analysis and Shape Descriptors
Motion Analysis and Object Tracking	Object Detection	Image Segmentation
Clustering	Eigen support	OpenCL support
Hardware Acceleration Layer	Parallel Processing	Optimization Algorithms
Fisheye camera model	Query I/O API backends registry	Flags for video I/O

cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

- **`img`**:
- Es la imagen de entrada que deseas convertir. En este caso, se espera que sea una imagen en color, generalmente en formato BGR (Blue, Green, Red).
- **`cv2.COLOR_BGR2GRAY`**:
- Especifica que deseas convertir la imagen de BGR (formato de color de OpenCV) a escala de grises. Otros códigos están disponibles para convertir a diferentes espacios de color, como RGB, HSV, .

cv2.Canny(img, threshold1, threshold2)



blurred_img = cv2.blur(src, ksize)

cv2.putText(img, text, org, font, fontScale, color, thickness, lineType)

cv2.putText(img, text, org, fontFace, fontScale, color, thickness, lineType)

- **img**: Imagen de entrada.
- **markers**: Matriz de marcadores, una imagen de enteros donde las áreas diferentes tienen diferentes etiquetas.

```
markers = cv2.watershed(img, markers)
```

- ********Falta foto

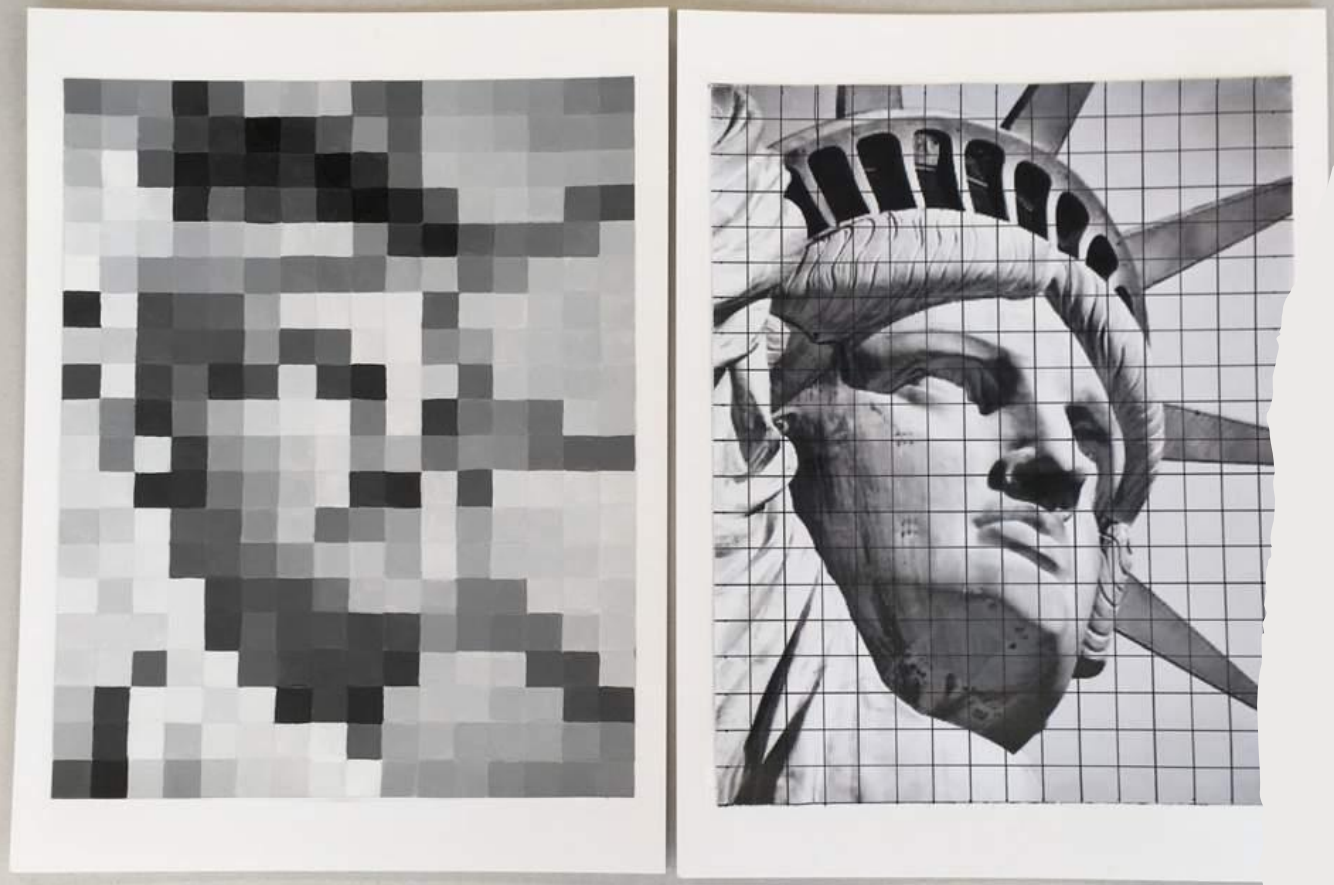
- cap = cv2.VideoCapture(0)

VideoCapture toma el video en tiempo real de la camara o objeto de video que se obtenga en ese moemento indicando con un numero si hay mas de un objeto

- cap.read()

Captura un fotograma de la camara definida

*ls /dev/video muestra las camaras detectadas por el sistema.

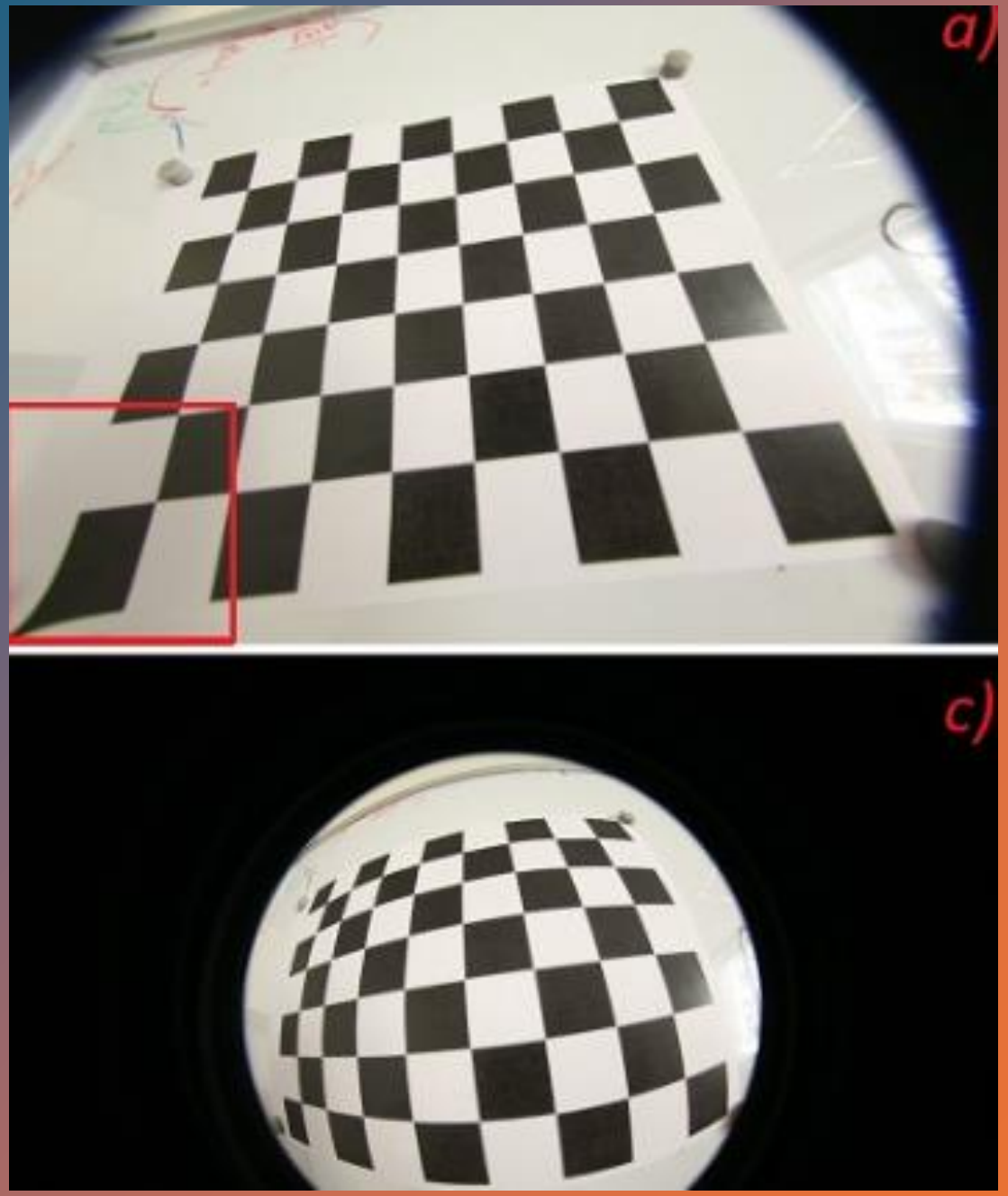


retval, eigenvalues, eigenvectors = cv2.eigen(src)

- **`retval`**: Un valor booleano que indica si la operación fue exitosa.
- **`eigenvalues`**: Matriz que contiene los valores propios de la matriz de entrada.
- **`eigenvectors`**: Matriz que contiene los vectores propios correspondientes.

undistorted_image = cv2.fisheye.undistortImage(image, K, D, Knew=None)

- `image`: La imagen de entrada que tiene distorsión de ojo de pez.
- `K`: La matriz de cámara (matriz intrínseca) que describe las propiedades ópticas de la cámara.
- `D`: Los coeficientes de distorsión de la lente



fisheye_image = cv2.fisheye.distortImage(image, K, D)

- K = np.array([[800, 0, width / 2],
- [0, 800, height / 2],
- [0, 0, 1]], dtype=np.float32)

- D = np.array([-0.4, 0.2, 0, 0], dtype=np.float32)

Uso de OpenCL

- **`cv::UMat()`**
- Clase que representa una matriz en el dispositivo OpenCL. Permite realizar operaciones en la GPU sin necesidad de gestionar la transferencia de datos manualmente.
- **`cv::UMat::copyTo(dst)`**
- **`dst`**: Objeto de destino donde se copiarán los datos.

OpenCL

- OpenCL (Open Computing Language)

Uso de OpenCL

- **`cv::ocl::runKernel()`**
- **`kernelName`**: Nombre del kernel de OpenCL que se va a ejecutar.
- **`globalSize`**: Tamaño global del trabajo (dimensiones de la grilla de trabajo).
- **`localSize`**: Tamaño local del trabajo (opcional; dimensiones del grupo de trabajo).
- **`args`**: Argumentos que se pasan al kernel.

Uso de openCL

Funciones de Procesamiento de Imágenes con OpenCL

- **`cv::ocl::HoughLines(`**
  - **`src`**: Imagen de entrada en escala de grises.
  - **`rho`**: Resolución de la distancia en píxeles.
  - **`theta`**: Resolución del ángulo en radianes.
  - **`threshold`**: Umbral para detectar líneas.

)
- **`cv::ocl::GaussianBlur(`**
  - **`src`**: Imagen de entrada.
  - **`ksize`**: Tamaño del núcleo del filtro Gaussiano (debe ser un par impar, como (5, 5)).
  - **`sigmaX`**: Desviación estándar en el eje X.
  - **`sigmaY`**: Desviación estándar en el eje Y (opcional; si se omite, se utiliza el valor de `sigmaX`).

)

Referencias:

https://opencv.org/

https://docs.opencv.org/4.x/d1/dfb/intro.html

https://www.geeksforgeeks.org/text-detection-and-extraction-using-opencv-and-ocr/

https://github.com/tesseract-ocr/tesseract