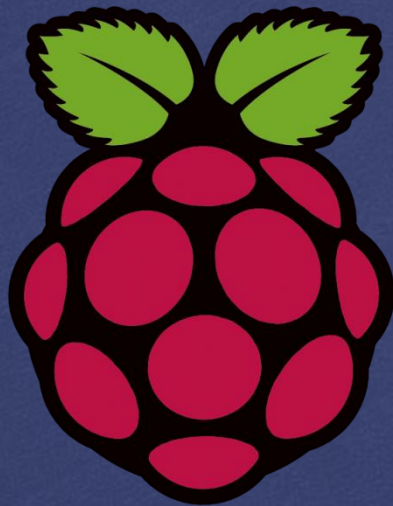


Introducción a openCV

Transformaciones geométricas

POR. Christian Quispe Canchari



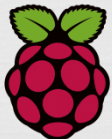
OpenCV

CURSO DE RASPBERRY PI3 -

¿Qué es?



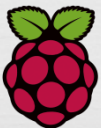
OpenCV (Open Source Computer Vision) es una librería software open-source de visión artificial y machine learning.



Temas a tratar

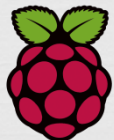
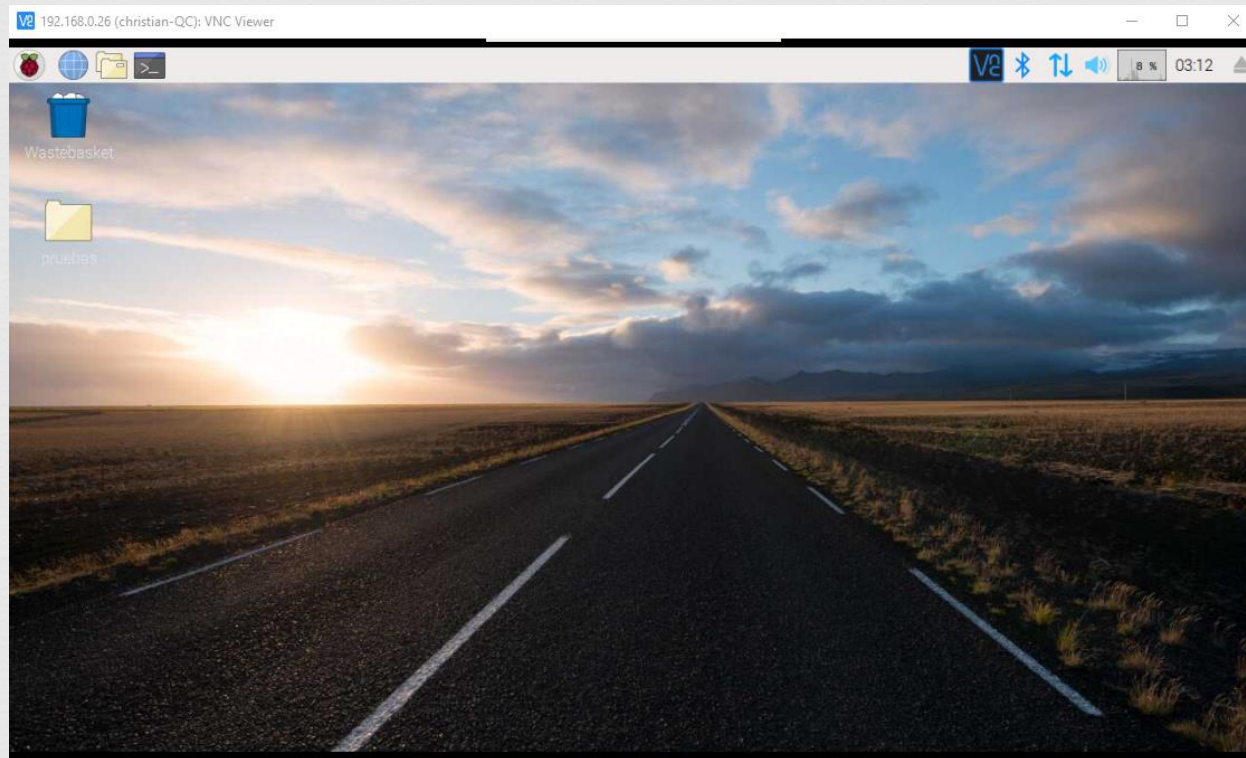
Aplicando Transformaciones Geométricas a Imágenes

1. Lectura, visualización y guardado de imágenes
2. Cargando y guardando una imagen
 - Cambios de formato de la imagen
3. Espacios de color en una imagen
 - Conversión de espacio de color
 - División de canales de la imagen
 - Fusionando canales de la imagen
4. Traslación de imágenes
5. Rotación de imágenes
6. Escala de imagen
7. Transformaciones afines
8. Transformaciones proyectivas
9. Deformación de una imagen



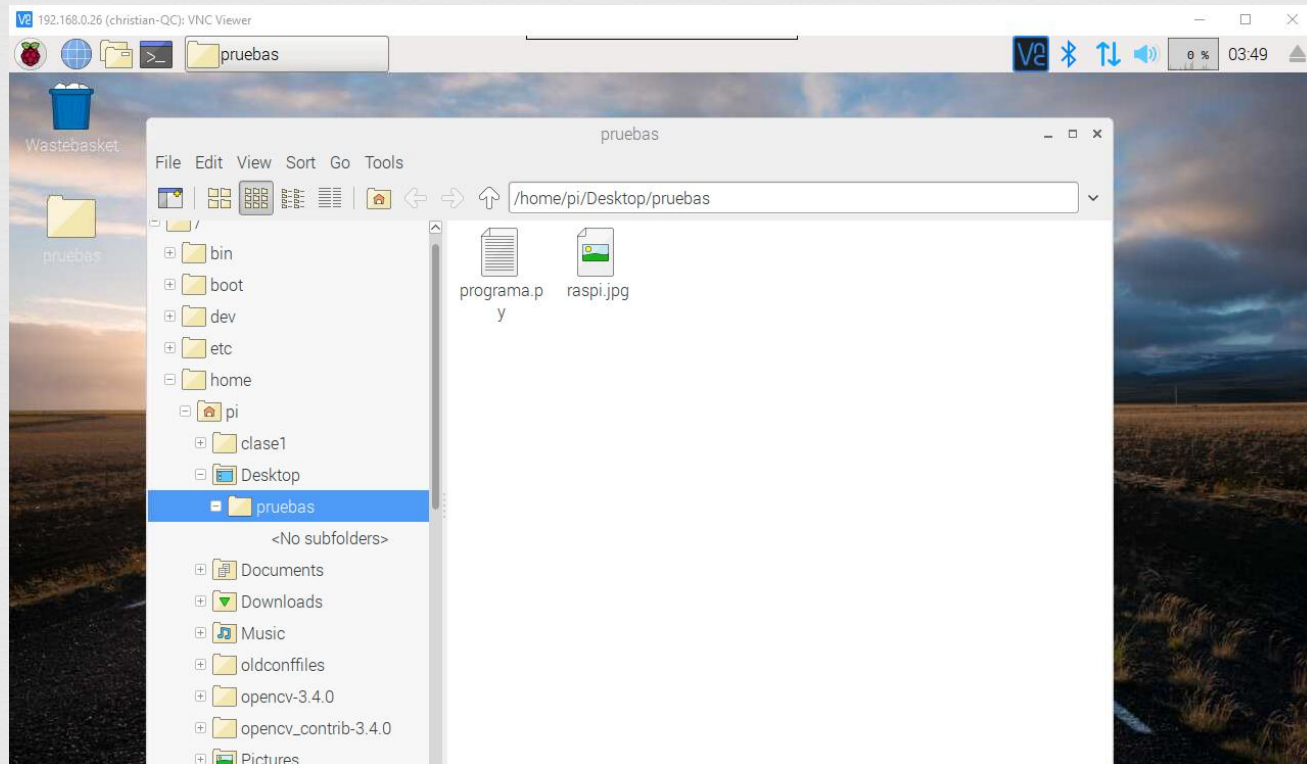
Antes de empezar ...

Entramos por VNC a la raspberry y crearemos un directorio llamado prueba para nuestros proyectos en Desktop



Lectura, visualización y guardado de imagenes

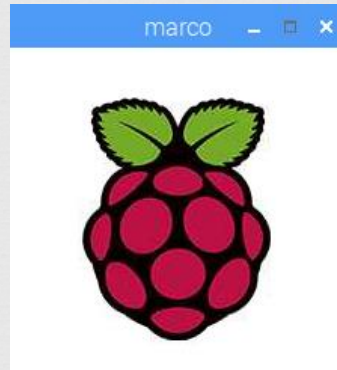
Veamos cómo podemos cargar una imagen. Crea un archivo llamado *ejemplo1.py* dentro del directorio *pruebas* y ábrelo con tu editor de código favorito y asegúrese de tener una imagen llamada *raspi.png* en esa carpeta.



programa.py ✕

```
import cv2
img = cv2.imread("raspi.jpg")
cv2.imshow("marco",img)
cv2.waitKey()
```

Si ejecuta el programa anterior, verá una imagen que se muestra en una nueva ventana



Comprendamos el código anterior

```
import cv2
img = cv2.imread("raspi.jpg")
cv2.imshow("marco",img)
cv2.waitKey()
```

```
# importación de la biblioteca openCV
# lee la imagen y la almacena aen la variable img
#mostramos la imagen en una ventana llamada "marco"
#espera indefinidamente a que se presione una tecla
```


2. Cargando y guardando una imagen

OpenCV proporciona múltiples formas de cargar una imagen. Digamos que queremos cargar una imagen en color en el modo de escala de grises, podemos hacerlo utilizando el siguiente fragmento de código:

```
programa.py *  
import cv2  
img_gray=cv2.imread("raspi.jpg",cv2.IMREAD_GRAYSCALE)  
cv2.imshow("marco",img_gray)  
cv2.waitKey()
```



Para guardar:

```
cv2.imwrite("raspi.jpg",img) # cv2.imwrite("nuevo nombre", imagen)
```

2.1 Cambiando el formato de la imagen

```
programa.py *  
import cv2  
img = cv2.imread("raspi.jpg")  
cv2.imwrite("raspi.png",img,[cv2.IMWRITE_PNG_COMPRESSION])  
cv2.waitKey()
```



3. Espacios de color de una imagen

En la visión por computadora y el procesamiento de imágenes, el espacio de color se refiere a una forma específica de organizar los colores. Un espacio de color es en realidad una combinación de dos cosas, un modelo de color y una función de mapeo. La razón por la que queremos los modelos de color es porque nos ayuda a representar los valores de píxel usando tuplas y la función de mapeo asigna el modelo de color al conjunto de todos los colores posibles que se pueden representar.

Tomemos un par de espacios de color y veamos qué información proporcionan:

- RGB
- YUV
- HSV

Teniendo en cuenta todos los espacios de color, hay alrededor de 190 opciones de conversión disponibles en OpenCV. Si desea ver una lista de todos los indicadores disponibles, vaya al shell de Python y escriba lo siguiente:

Verá una lista de opciones disponibles en OpenCV para convertir de un espacio de color a otro. Podemos convertir prácticamente cualquier espacio de color en otro. Veamos cómo podemos convertir una imagen a una imagen en escala de grises:

programa.py ✕

```
import cv2

img = cv2.imread('raspberrypi.jpg')
gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
cv2.imshow('marco en gris', gray_img)
cv2.waitKey()
```



Conversión a YUV

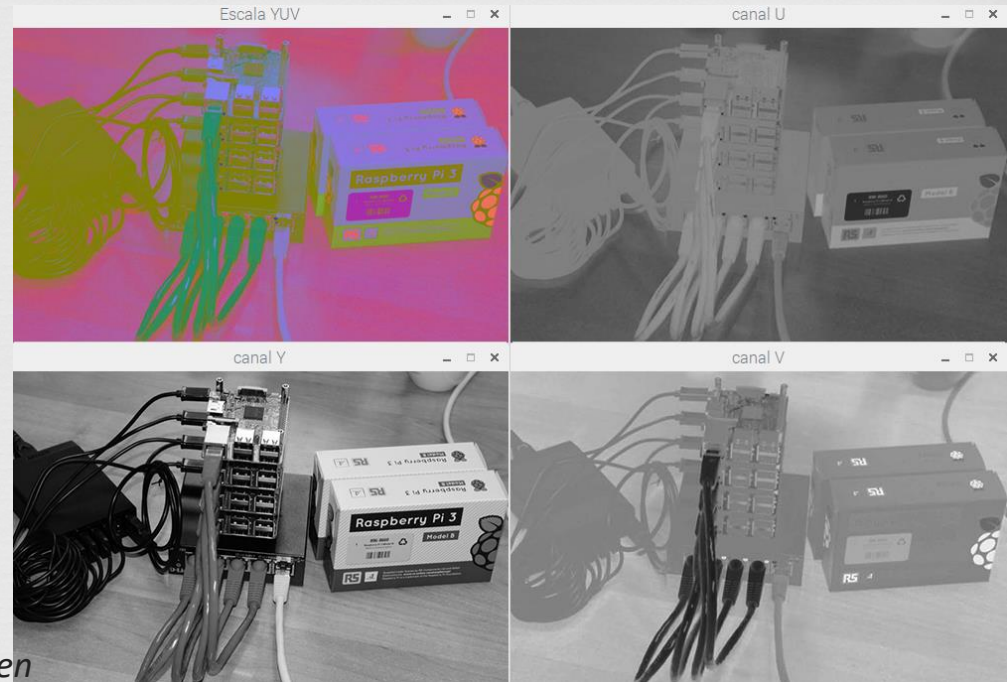
programa.py ✕

```
import cv2

img = cv2.imread('raspberrypi.jpg')
yuv_img = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
y,u,v = cv2.split(yuv_img)

cv2.imshow('Escala YUV', yuv_img)
cv2.imshow('canal Y', y)
cv2.imshow('canal U', u)
cv2.imshow('canal V', v)

cv2.waitKey()
```



Esto puede parecer una versión deteriorada de la imagen original, pero no lo es.


3.3 Fusionando canales de imagen

```
programa.py *%  
  
import cv2  
  
img = cv2.imread('raspberry.jpg')  
  
g,b,r = cv2.split(img)  
ggr_img = cv2.merge((g,g,r))  
rbr_img = cv2.merge((r,b,r))  
  
cv2.imshow('Original', img)  
cv2.imshow('GGR', ggr_img)  
cv2.imshow('RBR', rbr_img)  
  
cv2.waitKey()
```



4. Traslación de imágenes

En esta sección, discutiremos el cambio de una imagen. Digamos que queremos mover la imagen dentro de nuestro marco de referencia. En terminología de visión por computadora, esto se conoce como traslación. Avancemos y veamos cómo podemos hacer eso:

programa.py *

```
import cv2
import numpy as np
img = cv2.imread('raspberry.jpg')
num_rows, num_cols = img.shape[:2]
translation_matrix = np.float32([ [1,0,70], [0,1,110] ])
img_translation = cv2.warpAffine(img, translation_matrix, (num_cols,num_rows),cv2.INTER_LINEAR)
cv2.imshow('Traslación', img_translation)
cv2.waitKey()
```

