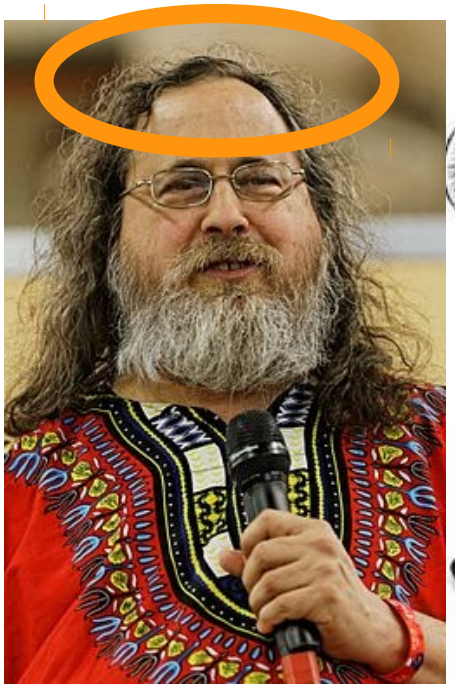
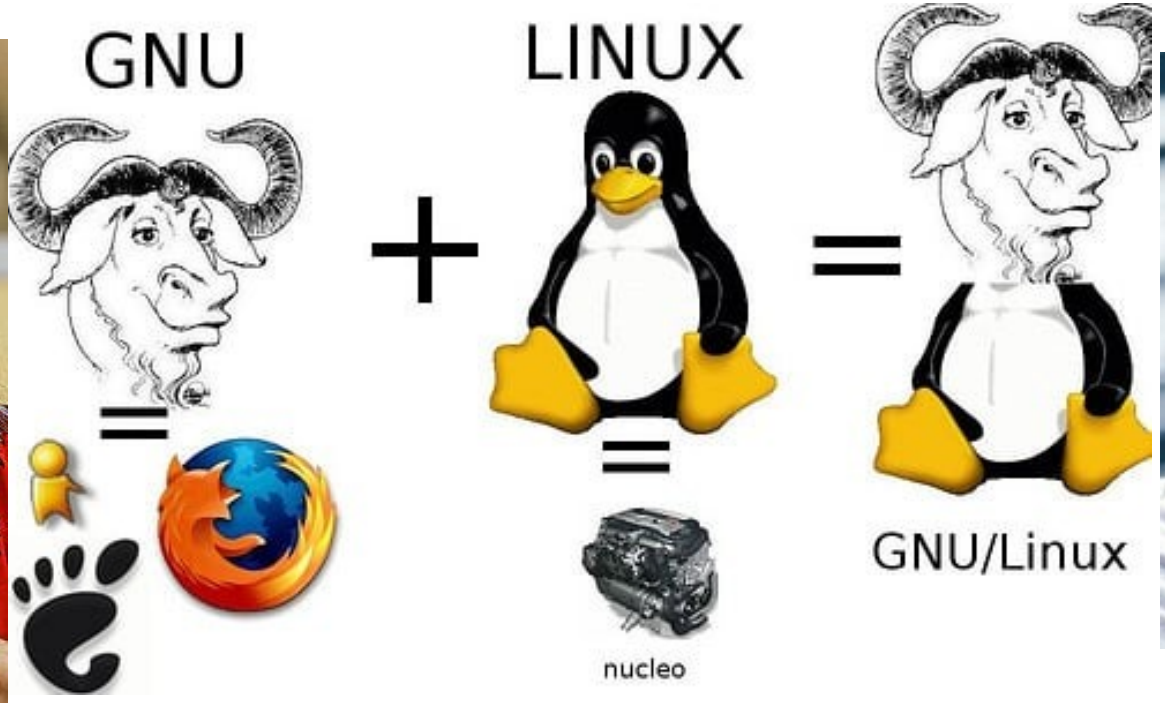
Robochallenge Semana 3

Open Computer Vision Library

Linux: KERNEL del Sistema Operativo GNU/Linux.

GNU: Free Software





Human-Machine-Interface

Hardware

remote
(SSH, HTTP, ...)

Supercomputer
Computer Cluster
Mainframe computer
Distributed computing

Keyboard & Mouse
also Braille, Touch-Display, Speech recognition, Graphics tablet, 3D-Mouse, Wii nunchak, etc.

Touch-Display
Attitude sensor, Motion sensor, Speech recognition

Speech recognition
Attitude sensor
Motion sensor

Display, Sound
Vibration

Desktop Computer
Workstation
Home Computer
Desktop replacement laptop
Thin client

Mobile computer
Note-/ Net-/ Smartbook
Tablet
Smartphone
PDA / Handheld game console

Wearable Computer
Wristwatch
Virtual Retina Display
Head-mounted display

remote
(SSH, HTTP, Serial, I²C, ...)

Embedded Computer
Customer-premises equipment
Measurement Equipment
Laboratory Equipment
Layer3-Switches
other embedded systems

Linux kernel

High-performance computing (HPC)

Real-time computing (RTC)

Linux Process Scheduler
Linux Security Modules
Linux Network scheduler
Network stack
Netfilter
Linux device drivers
Linux file system drivers

Pool of free and open-source and proprietary software

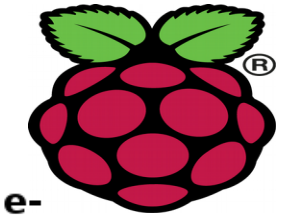
Web server solution stacks (LAMP)
Distributed Computing
Routing daemons
Software Development
Package management systems
CAD, CAM & CAE Software
Office
Image Processing
Desktop Publishing (DTP)

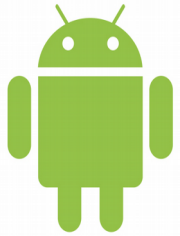
Graphical User Interfaces (Shells)
Windowing Systems

Desktop UI
Touch UI
Wearable UI

Video processing software
3D computer graphics
Computer animation
Motion graphics
Digital Audio Workstation
DJ Mixing Software
Video games
Home cinema solutions

Debian software archives: 37,000 software packages





Human-Machine-Interface

Hardware

remote
(SSH, HTTP, ...)

Keyboard & Mouse
also Braille, Touch-Display, Speech recognition, Graphics tablet, 3D-Mouse, Wii nunchak, etc.

Touch-Display
Attitude sensor, Motion sensor, Speech recognition

Speech recognition
Attitude sensor
Motion sensor

Display, Sound Vibration

remote
(SSH, HTTP, Serial, I²C, ...)

Supercomputer
Computer Cluster
Mainframe computer
Distributed computing

Desktop Computer
Workstation
Home Computer
Desktop replacement laptop
Thin client

Mobile computer
Note-/ Net-/ Smartbook
Tablet
Smartphone
PDA / Handheld game console

Wearable Computer
Wristwatch
Virtual Retina Display
Head-mounted display

Embedded Computer
Customer-premises equipment
Measurement Equipment
Laboratory Equipment
Layer3-Switches
other embedded systems

Linux kernel

High-performance computing (HPC)

Real-time computing (RTC)

Linux Process Scheduler
Linux Security Modules
Linux Network scheduler
Network stack
Netfilter
Linux device drivers
Linux file system drivers

Pool of free and open-source and proprietary software

Web server solution stacks (LAMP)
Distributed Computing
Routing daemons
Software Development
Package management systems
CAD, CAM & CAE Software
Office
Image Processing
Desktop Publishing (DTP)

Graphical User Interfaces (Shells)

Windowing Systems

Desktop UI

Touch UI

Wearable UI

Video processing software
3D computer graphics
Computer animation
Motion graphics

Digital Audio Workstation
DJ Mixing Software
Video games
Home cinema solutions

Debian software archives: 37,000 software packages

Lane Detection with OpenCV

- Ross Kippenbrock
- https://www.youtube.com/watch?v=VyLihutdsPk
- https://github.com/rkipp1210/pydata-berlin-2017/blob/master/finding-lane-lines-pydata-berlin-2017.pdf

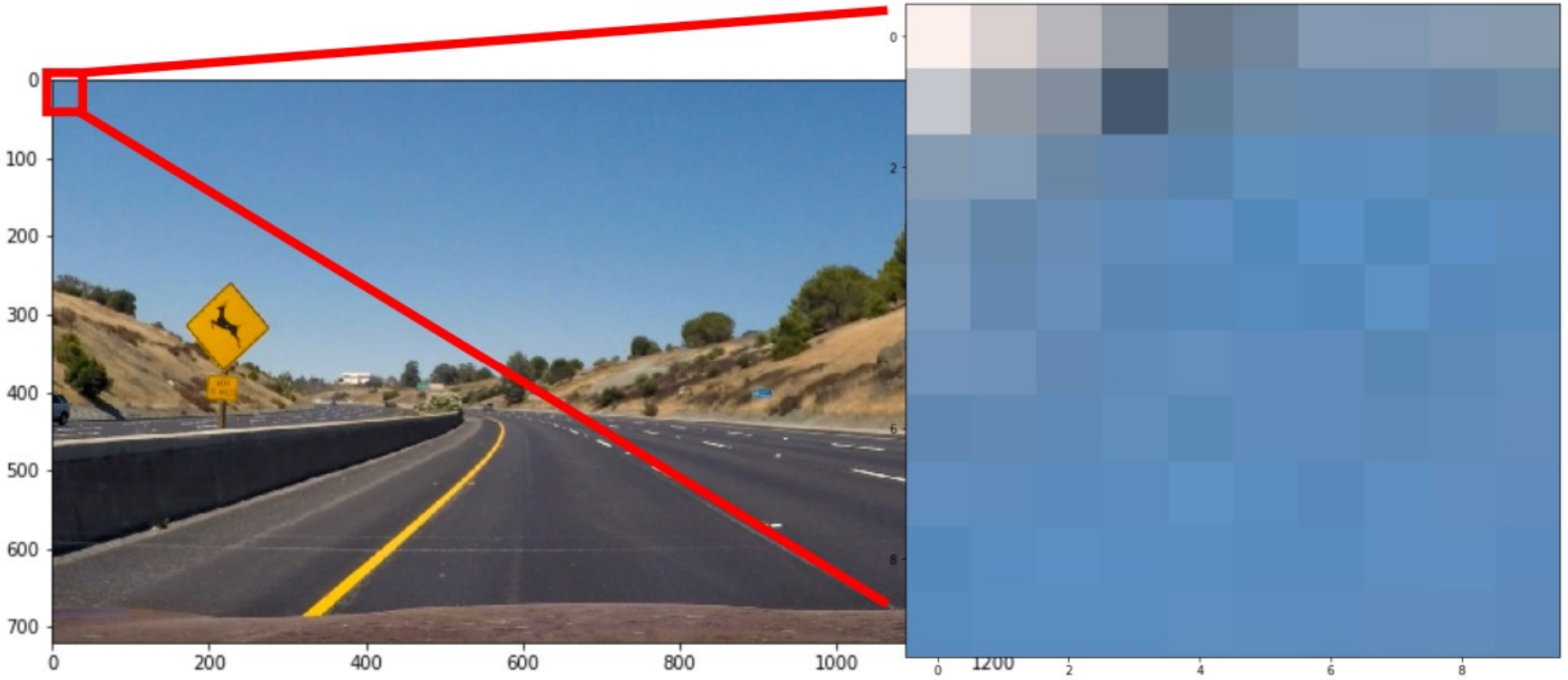
Instalación de OpenCV

- Actualizar sistema operativo (debian)
  - sudo apt-get update
  - Sudo apt-get upgrade
- Instalar dependencias
  - sudo apt-get install build-essential cmake pkg-config ….etc
- Descargar OpenCV
  - wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
  - unzip opencv.zip
- Configurar, compilar e Instalar
  - cmake guana guana guana
  - make -j 4 (muchas horas)
  - sudo make install

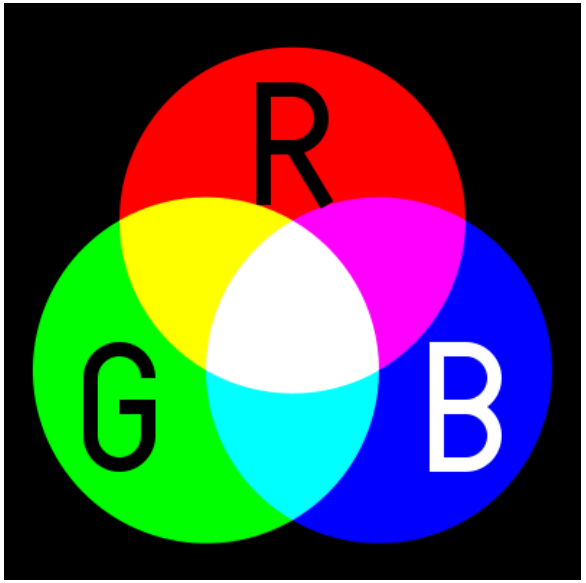
Imagenes

Anatomy of an image

```
img[0:10, 0:10, :]
```

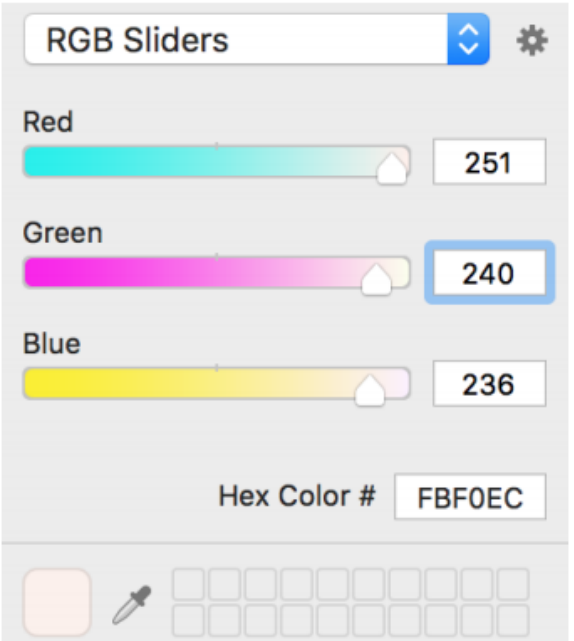


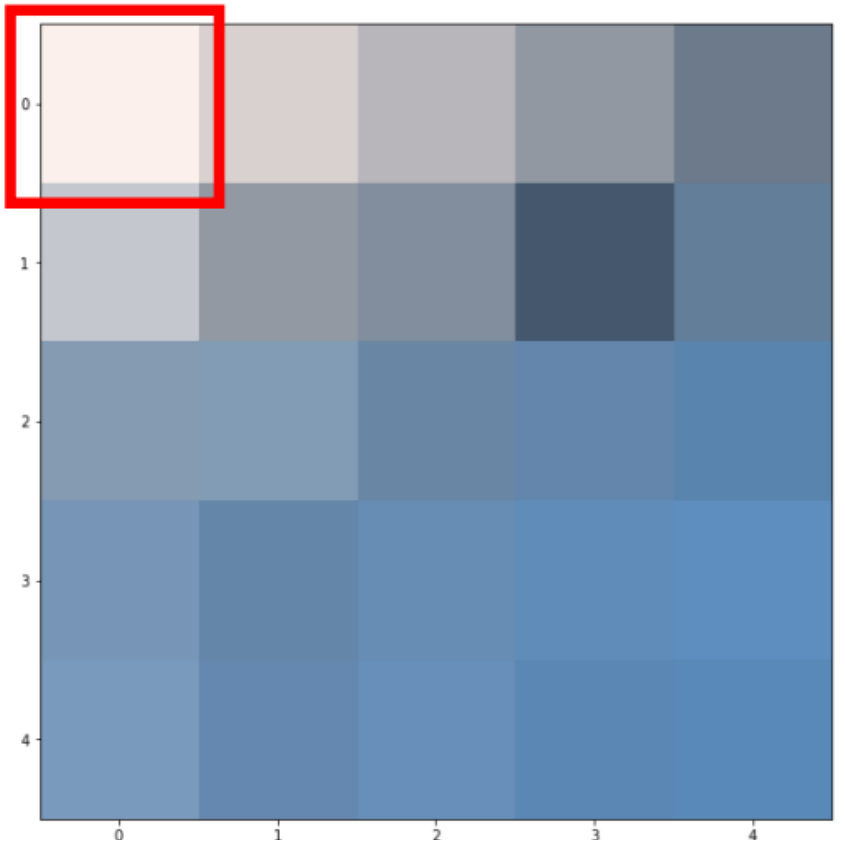
Imagenes




```
> img[0,0,:] # first pixel
array([251, 240, 236])
```

Red Green Blue





Ejercicios

- 01 – Captura de Imagenes
- 02 – Conversión de Color
- 03 – Color Space
- 04 – Binarizacion
- 05 – Filtrado
- 06 – Hough Lines

  Reta – Video streaming y otras cosas

  Romy - Redes neuronales

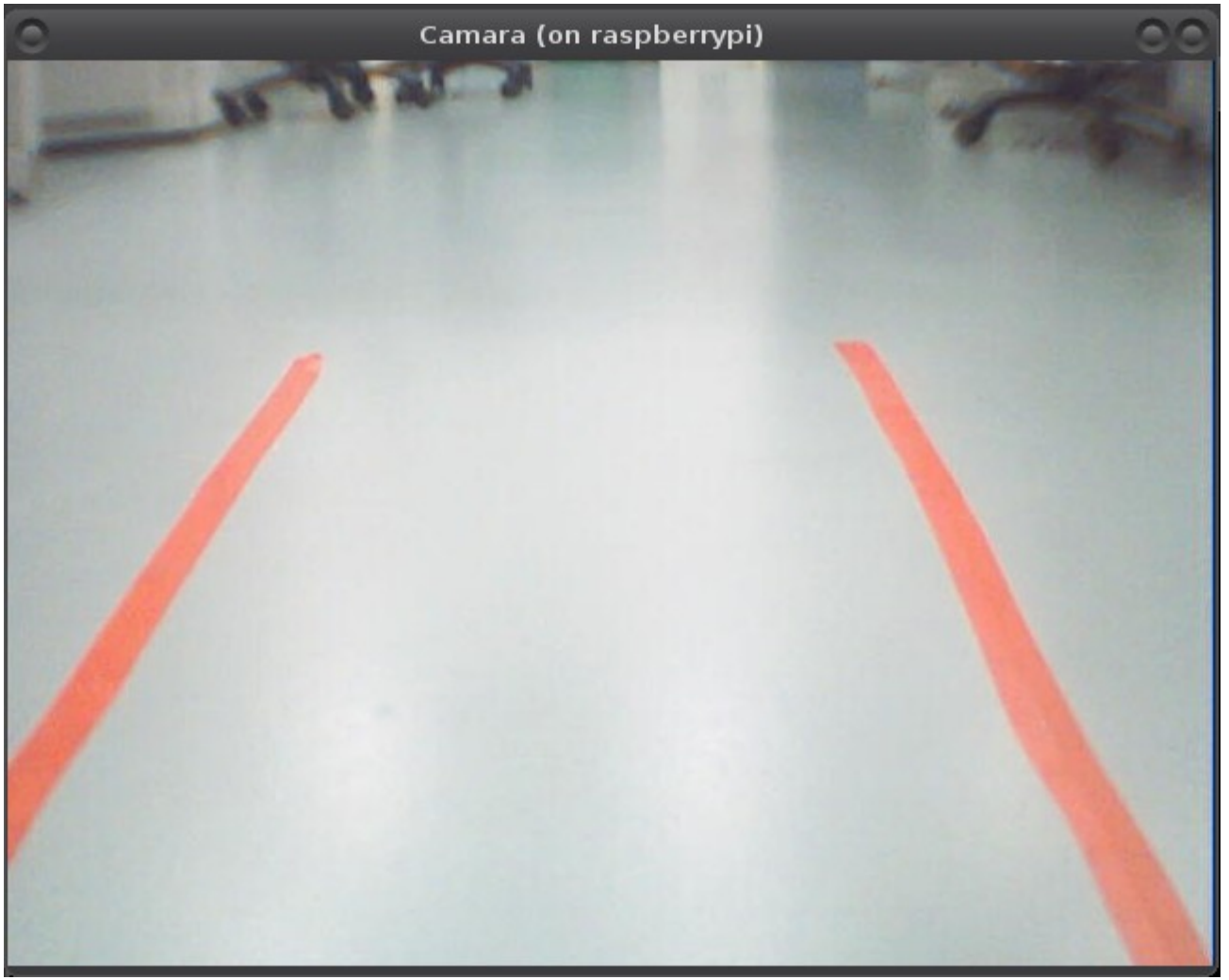
Ejercicio 01 - Captura

```
import numpy as np      # ← Numerical processing Python library
import cv2              # ← Computer Vision Library

cap = cv2.VideoCapture(0)

while(True):            # ← Siempre
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Display the resulting frame
    cv2.imshow('Camara',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Ejercicio 01 - Captura

```
import numpy as np      # ← Numerical processing Python library
import cv2              # ← Computer Vision Library

cap = cv2.VideoCapture(0)

while(True):            # ← Siempre
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Display the resulting frame
    cv2.imshow('Camara',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Ejercicio 01 - Captura



Ejercicio 02 – Conversion de Color

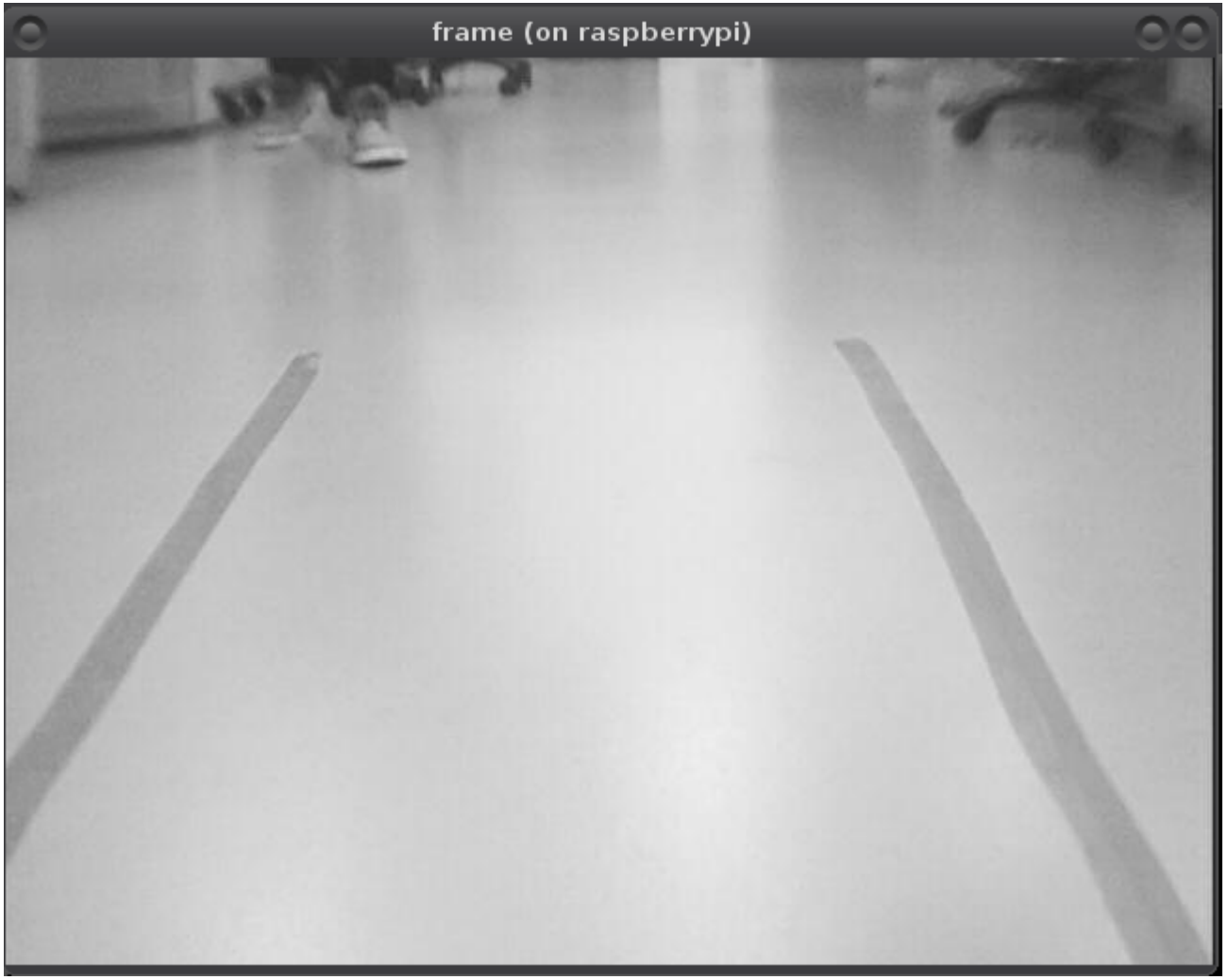
```
import numpy as np      # ← Numerical processing Python library
import cv2              # ← Computer Vision Library

cap = cv2.VideoCapture(0)

while(True):            # ← Siempre
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Aquí vamos a hacer el procesamiento de imagenes
    img=cv2.cvtColor(img, cv2.COLOR_CONVERSION)

    # Display the resulting frame
    cv2.imshow('Camara',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

- # When everything done, release the capture

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

Ejercicio 02 – Conversion de Color

- Gray conversion



Ejercicio 03 – Color spaces

- img=cv2.cvtColor(img, cv2.COLOR_**RGB2HLS)**
- img=cv2.cvtColor(img, cv2.COLOR_**RGB2GRAY**)
- img=cv2.cvtColor(img, cv2.COLOR_**RGB2YUV**)
- And many different combinations...

- https://docs.opencv.org/3.1.0/d7/d1b/group__imgproc__misc.html

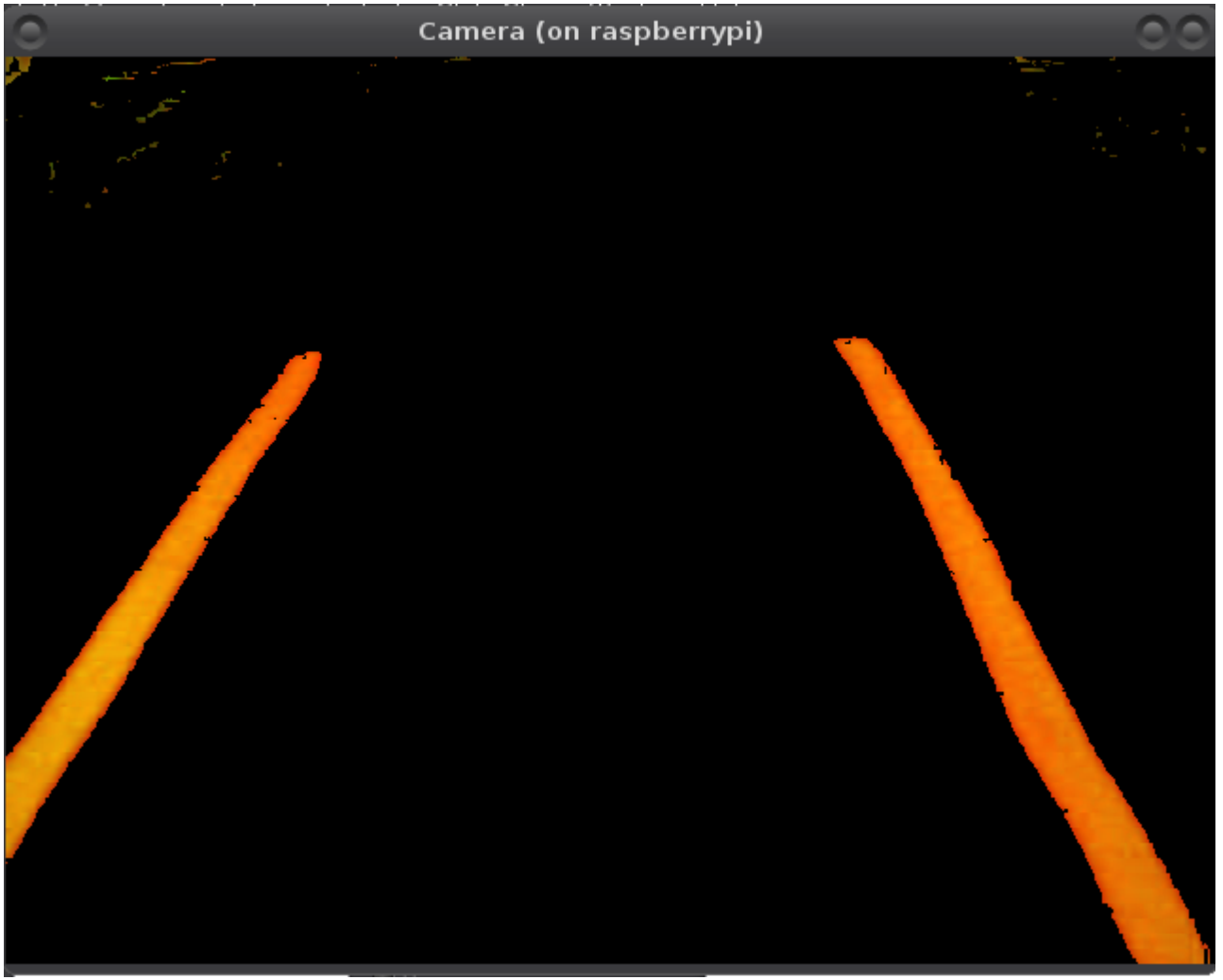
Ejercicio 03 – Color spaces

- Extraer RED de **R**GB

```
frame=cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
lower_red = np.array([0,50,50])          #example value
upper_red = np.array([10,255,255])    #example value
mask = cv2.inRange(frame, lower_red, upper_red)
img_result = cv2.bitwise_and(frame, frame, mask=mask)
```

Ejercicio 03 – Color spaces

- Red Extract

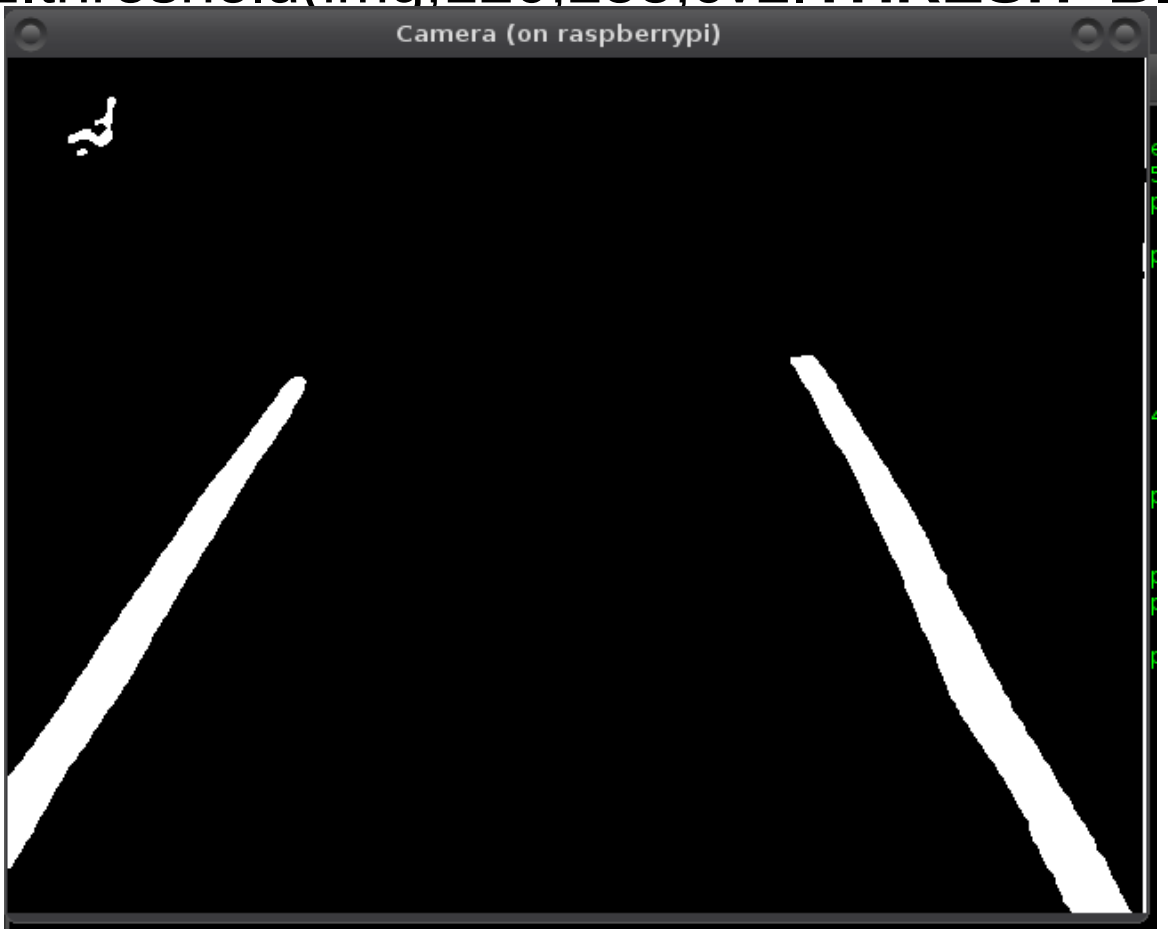


Ejercicio 04 - Binarizacion

- Convertir a Blanco y Negro

  # THRESHOLDING

  Result=cv2.threshold(img,120,255,cv2.**THRESH_BINARY**)

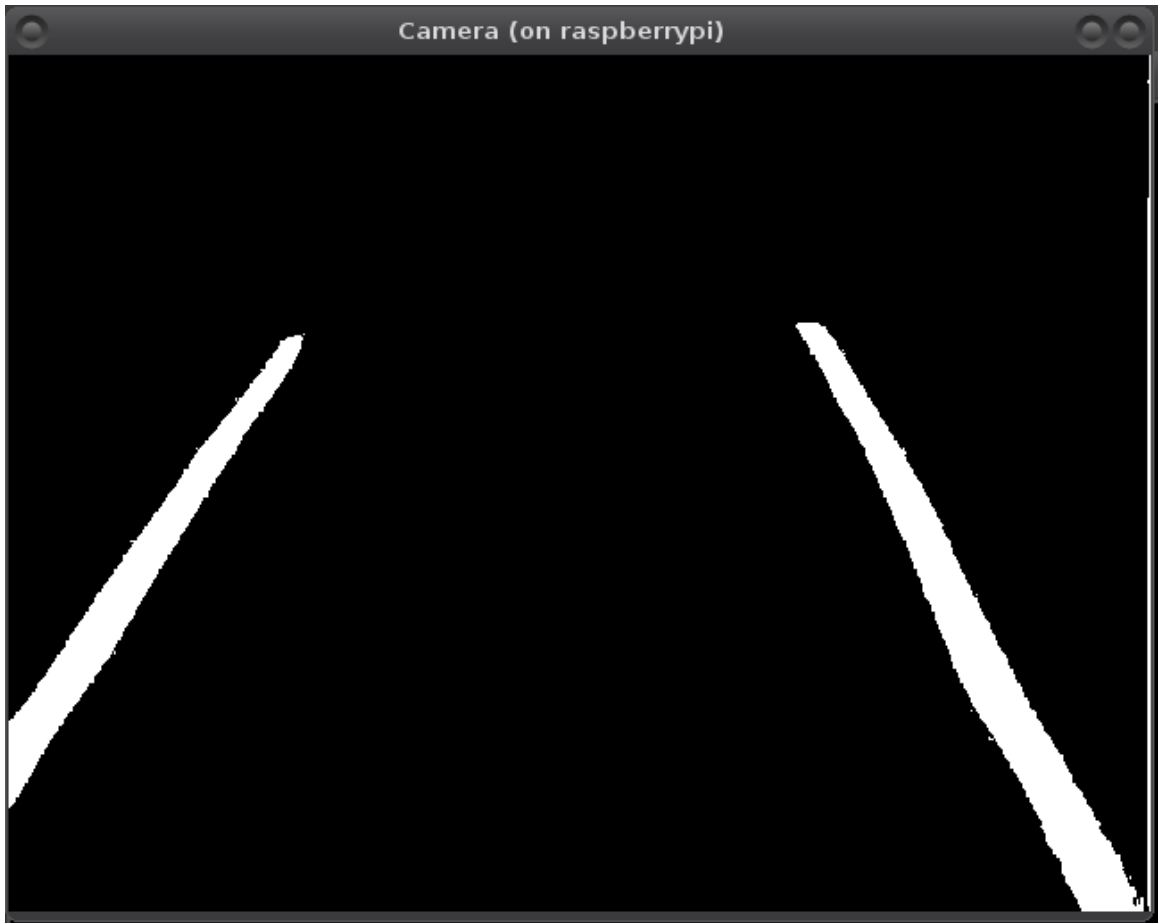


Ejercicio 05 - Filtrado

median = cv2.medianBlur(img_result,7)

blur = cv2.GaussianBlur(img,(5,5),0)

etc...

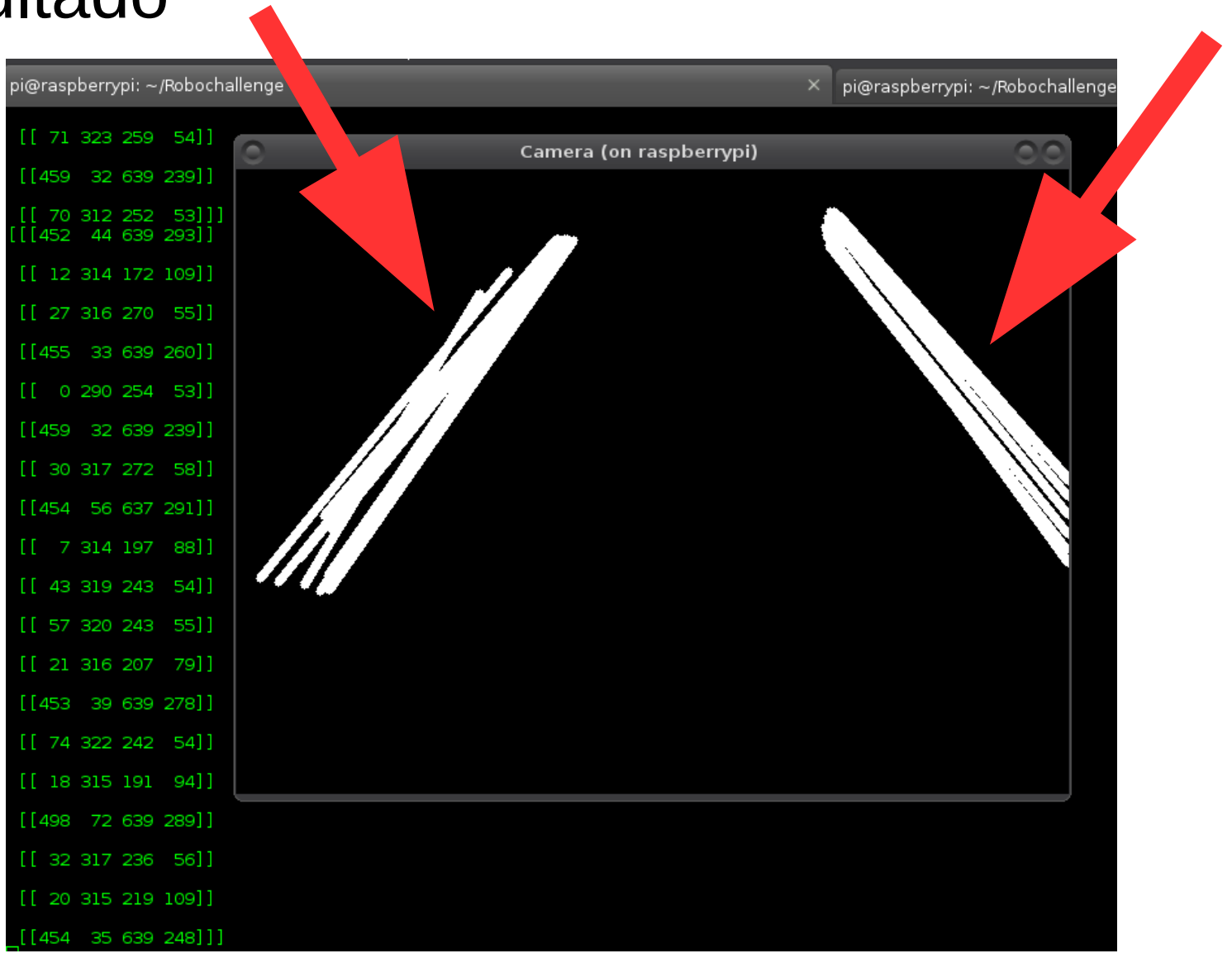


Ejercicio 06 – Hough Lines

```
rho = 1  # distance resolution in pixels of the Hough grid
theta = np.pi / 180  # angular resolution in radians of the Hough grid
threshold = 15  # minimum number of votes (intersections in Hough grid cell)
min_line_length = 50  # minimum number of pixels making up a line
max_line_gap = 20  # maximum gap in pixels between connectable line segments
line_image = np.copy(img) * 0  # creating a blank to draw lines on

# Run Hough on edge detected image
# Output "lines" is an array containing endpoints of detected line segments
lines = cv2.HoughLinesP(edges, rho, theta, threshold, np.array([]), min_line_length, max_line_gap)

for line in lines:
    for x1,y1,x2,y2 in line:
    cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),5)
```

Ejercicio 06 – Hough Lines

- Resultado



Ejercicio 06 – Hough Lines

- Lo importante es tener los valores numéricos de (X1,Y1) y (X2,Y2) para calcular dirección!