

Guardar fotos y compararlas

Determinar si un cajón de estacionamiento está libre o no

Esta técnica se basa en la resta de imágenes o análisis de diferencias entre imágenes.

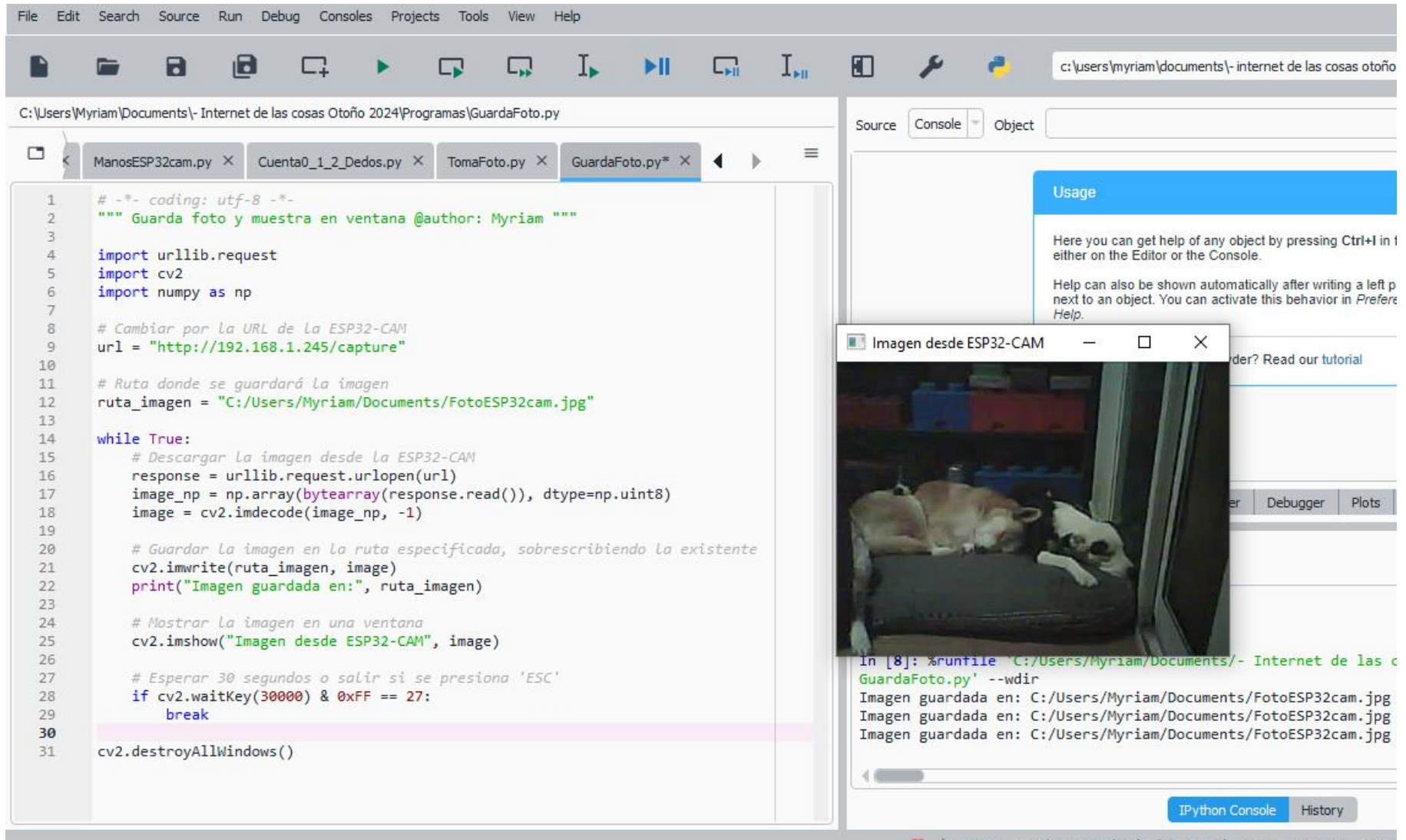
- 1) Tomar una foto del estacionamiento vacío como referencia y compararla con fotografías que vas capturando constantemente para determinar si hay un lugar libre.
- 2) Captura imágenes periódicamente del estacionamiento.
- 3) (Comparar) Resta la imagen actual de la imagen de referencia.
- 4) Analiza la diferencia resultante. Si hay cambios significativos (por ejemplo, un automóvil ocupa un espacio), las diferencias serán visibles. Para detectar las diferencias se pueden usar técnicas como el umbral para resaltar las áreas con cambios significativos.
- 5) Calcular la cantidad de píxeles que han cambiado para determinar si un espacio está ocupado.

Resultado:

Si hay diferencias significativas, determina que el espacio está ocupado.

Si no hay diferencias, el espacio está libre

GuardaFoto.txt



```
# -*- coding: utf-8 -*-
""" Guarda foto y muestra en ventana @author: Myriam """

import urllib.request
import cv2
import numpy as np

# Cambiar por la URL de la ESP32-CAM
url = "http://192.168.1.245/capture"

# Ruta donde se guardará la imagen
ruta_imagen = "C:/Users/Myriam/Documents/FotoESP32cam.jpg"

while True:
    # Descargar la imagen desde la ESP32-CAM
    response = urllib.request.urlopen(url)
    image_np = np.array(bytearray(response.read()), dtype=np.uint8)
    image = cv2.imdecode(image_np, -1)

    # Guardar la imagen en la ruta especificada, sobrescribiendo la existente
    cv2.imwrite(ruta_imagen, image)
    print("Imagen guardada en:", ruta_imagen)

    # Mostrar la imagen en una ventana
    cv2.imshow("Imagen desde ESP32-CAM", image)

    # Esperar 30 segundos o salir si se presiona 'ESC'
    if cv2.waitKey(30000) & 0xFF == 27:
        break

cv2.destroyAllWindows()
```

Usage

Here you can get help of any object by pressing Ctrl+I in the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences.

Imagen desde ESP32-CAM

In [8]: %runfile 'C:/Users/Myriam/Documents/Internet de las cosas/GuardaFoto.py' --wdir

Imagen guardada en: C:/Users/Myriam/Documents/FotoESP32cam.jpg

Imagen guardada en: C:/Users/Myriam/Documents/FotoESP32cam.jpg

Imagen guardada en: C:/Users/Myriam/Documents/FotoESP32cam.jpg

IPython Console History

Comparación de imágenes con Open CV

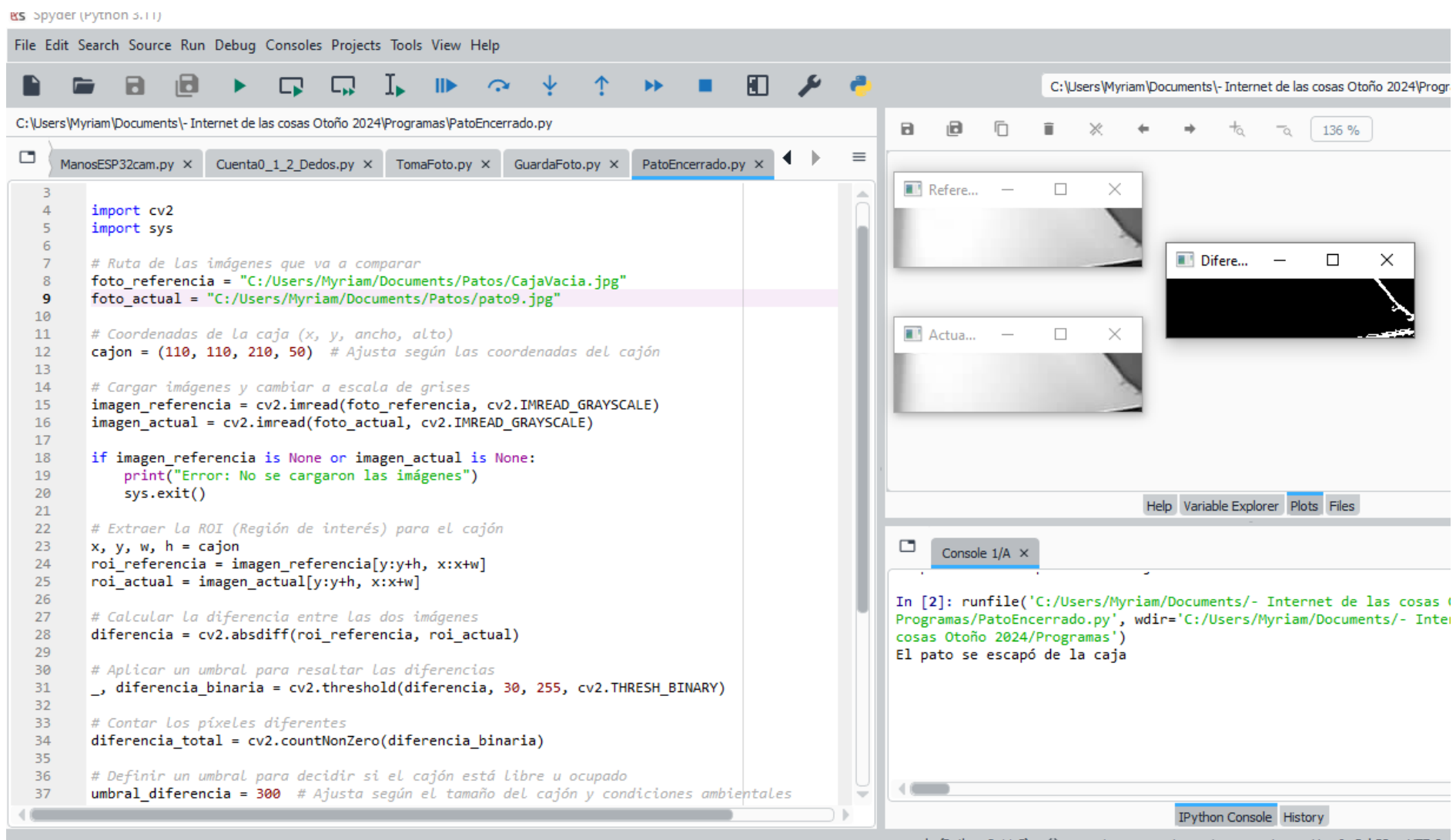
Python 3.11

The screenshot displays a Jupyter Notebook environment with the following components:

- File Explorer:** Shows the file path `C:\Users\Myriam\Documents\Internet de las cosas Otoño 2024\Programas\PatoEncerrado.py`.
- Code Editor:** Contains the following Python code:

```
3 import cv2
4 import sys
5
6 # Ruta de Las imágenes que va a comparar
7 foto_referencia = "C:/Users/Myriam/Documents/Patos/CajaVacía.jpg"
8 foto_actual = "C:/Users/Myriam/Documents/Patos/pato1.jpg"
9
10 # Coordenadas de la caja (x, y, ancho, alto)
11 cajon = (110, 110, 210, 50) # Ajusta según las coordenadas del cajón
12
13 # Cargar imágenes y cambiar a escala de grises
14 imagen_referencia = cv2.imread(foto_referencia, cv2.IMREAD_GRAYSCALE)
15 imagen_actual = cv2.imread(foto_actual, cv2.IMREAD_GRAYSCALE)
16
17 if imagen_referencia is None or imagen_actual is None:
18     print("Error: No se cargaron las imágenes")
19     sys.exit()
20
21 # Extraer la ROI (Región de interés) para el cajón
22 x, y, w, h = cajon
23 roi_referencia = imagen_referencia[y:y+h, x:x+w]
24 roi_actual = imagen_actual[y:y+h, x:x+w]
25
26 # Calcular la diferencia entre las dos imágenes
27 diferencia = cv2.absdiff(roi_referencia, roi_actual)
28
29 # Aplicar un umbral para resaltar las diferencias
30 _, diferencia_binaria = cv2.threshold(diferencia, 30, 255, cv2.THRESH_BINARY)
31
32 # Contar los píxeles diferentes
33 diferencia_total = cv2.countNonZero(diferencia_binaria)
34
35 # Definir un umbral para decidir si el cajón está libre u ocupado
36 umbral_diferencia = 300 # Ajusta según el tamaño del cajón y condiciones ambientales
37
```
- Image Viewer:** Displays three windows: 'Referencia...', 'Diferencia...', and 'Actual...'. The 'Diferencia...' window shows a binary image where the difference between the two regions of interest is highlighted in white against a black background.
- Console:** Shows the output of the script, including the message `El pato está atrapado en la caja`.

Comparación de imágenes con Open CV



The screenshot displays a Python IDE with a file named `PatoEncerrado.py` open. The code implements an image comparison algorithm using OpenCV. It reads two images, `CajaVacía.jpg` and `pato9.jpg`, from a specific directory. The images are converted to grayscale, and a region of interest (ROI) is extracted from each. The absolute difference between the two ROIs is calculated, and a binary threshold is applied to highlight differences. Finally, the total number of non-zero pixels in the difference image is counted to determine if the region is free or occupied.

```
3 import cv2
4 import sys
5
6 # Ruta de las imágenes que va a comparar
7 foto_referencia = "C:/Users/Myriam/Documents/Patos/CajaVacía.jpg"
8 foto_actual = "C:/Users/Myriam/Documents/Patos/pato9.jpg"
9
10 # Coordenadas de la caja (x, y, ancho, alto)
11 cajon = (110, 110, 210, 50) # Ajusta según las coordenadas del cajón
12
13 # Cargar imágenes y cambiar a escala de grises
14 imagen_referencia = cv2.imread(foto_referencia, cv2.IMREAD_GRAYSCALE)
15 imagen_actual = cv2.imread(foto_actual, cv2.IMREAD_GRAYSCALE)
16
17 if imagen_referencia is None or imagen_actual is None:
18     print("Error: No se cargaron las imágenes")
19     sys.exit()
20
21 # Extraer la ROI (Región de interés) para el cajón
22 x, y, w, h = cajon
23 roi_referencia = imagen_referencia[y:y+h, x:x+w]
24 roi_actual = imagen_actual[y:y+h, x:x+w]
25
26 # Calcular la diferencia entre las dos imágenes
27 diferencia = cv2.absdiff(roi_referencia, roi_actual)
28
29 # Aplicar un umbral para resaltar las diferencias
30 _, diferencia_binaria = cv2.threshold(diferencia, 30, 255, cv2.THRESH_BINARY)
31
32 # Contar los píxeles diferentes
33 diferencia_total = cv2.countNonZero(diferencia_binaria)
34
35 # Definir un umbral para decidir si el cajón está libre u ocupado
36 umbral_diferencia = 300 # Ajusta según el tamaño del cajón y condiciones ambientales
37
```

On the right side, three small windows show the images: `Referencia...`, `Diferencia...`, and `Actual...`. The `Diferencia...` window shows a black image with white highlights indicating the differences between the two regions of interest.

The bottom right pane shows the IPython Console with the following output:

```
In [2]: runfile('C:/Users/Myriam/Documents/- Internet de las cosas Otoño 2024/Programas/PatoEncerrado.py', wdir='C:/Users/Myriam/Documents/- Internet de las cosas Otoño 2024/Programas')
El pato se escapó de la caja
```


Rúbrica del proyecto final (para cada parte: ESP21, ESP32cam, Python y node-red)

0% valor	No funciona, no entregado o no está bien armado
25% valor	Funciona una parte, pero no se obtiene el resultado deseado
50% valor	Funciona, pero no se comunica con el resto del proyecto o funciona el programa pero el hardware falla o falta.
75% valor	Funciona una parte pero si se comunica con el resto del proyecto
100% valor	El proyecto funciona correctamente y se comunica con el resto del proyecto

Rúbrica del reporte final

(80%) Procedimiento: Explicar parte por parte los códigos en arduino (para ESP32 o R4 y para ESP32cam), Python y node-red

(10%) Conclusiones individuales: Cada miembro del equipo escribe una reflexión sobre lo que aprendieron en este proyecto, cómo lo pueden aplicar en su profesión, cómo escalar o mejorar este proyecto.

(10%) presentación y organización de la información