

1- Ejecuta detección en 3 imágenes (1 persona, grupo, rostros parciales).

Ejecución para una persona

The screenshot shows the CodeLab IDE interface. The left sidebar displays a file tree under 'CODELAB' with files like '1-MTCNN IMAGEN', 'codelab.py', 'rostroparcial.jpg', 'unapersona.jpg', and 'grupog.jpg'. The main editor window contains Python code for face detection using MTCNN. The terminal at the bottom shows the command 'PS C:\Users\Marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\tarea>'.

```
# Carga imagen (sube archivos en Colab o usa una URL y descárgala)
img = cv2.imread('unapersona.jpg') # reemplaza por tu archivo
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
detector = MTCNN() # puedes ajustar min_face_size
t0 = time()
res = detector.detect_faces(img_rgb)
t1 = time()

print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)*1000:.1f} ms')
for r in res:
    print(f'confidence: {r['confidence']}, r['box'], r['keypoints'].keys())
    vis = img_rgb.copy()
    for r in res:
        x, y, w, h = r['box']
        cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
        for name, (px,py) in r['keypoints'].items():
            cv2.circle(vis, (px,py), 2, (255,0,0), -1)
    plt.imshow(vis)
    plt.axis('off')
    plt.show()

detector = MTCNN() # su NMS interno filtra solapas
thr = 0.6
filtrados = [r for r in res if r['confidence'] >= thr]
print(f'Con {thr} quedan {len(filtrados)} rostro(s)')
```

El rostro fue reconocido correctamente

The screenshot shows the CodeLab IDE interface again. The file tree and code are identical to the previous screenshot. A preview window titled 'Figure 1' is open, displaying a portrait of a smiling man with a green bounding box around his head and red dots marking key points (nose, mouth, eyes). The terminal at the bottom shows the command 'PS C:\Users\Marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\tarea>'.

```
# Carga imagen (sube archivos en Colab o usa una URL y descárgala)
img = cv2.imread('unapersona.jpg') # reemplaza por tu archivo
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
detector = MTCNN() # puedes ajustar min_face_size
t0 = time()
res = detector.detect_faces(img_rgb)
t1 = time()

print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)*1000:.1f} ms')
for r in res:
    print(f'confidence: {r['confidence']}, r['box'], r['keypoints'].keys())
    vis = img_rgb.copy()
    for r in res:
        x, y, w, h = r['box']
        cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
        for name, (px,py) in r['keypoints'].items():
            cv2.circle(vis, (px,py), 2, (255,0,0), -1)
    plt.imshow(vis)
    plt.axis('off')
    plt.show()

detector = MTCNN() # su NMS interno filtra solapas
thr = 0.6
filtrados = [r for r in res if r['confidence'] >= thr]
print(f'Con {thr} quedan {len(filtrados)} rostro(s)')
```

En un tiempo de 161ms

Ejecución para un grupo

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** Shows a folder named "CODELAB" containing files like "1-MTCNN IMAGEN", "pycache_1-MTCNN.py", "Tarea", "grupo.jpg", "rostroparcial.jpg", "unapersona.jpg", "codelab.py", and "grupo.jpg".
- Code Cell:** Displays Python code for face detection using MTCNN. The code imports necessary libraries, reads an image, initializes the detector, and iterates through detected faces to draw bounding boxes and keypoints.
- Terminal:** Shows the command PS C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tarea>.
- Output:** Shows the output of the code execution, including a list of detected faces with their bounding boxes and confidence scores.
- Bottom Bar:** Includes status indicators like "Lin. 12, col. 12", "Espacios: 4", "UTF-8", "CRLF", "Python", "3.13.7 (env)", "Go Live", "Quokka", "Prettier", "Zero Two", and "Live Share".

Los rostros fueron reconocidos correctamente con cierta precisión en cuestión de ojos y nariz

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** Shows a folder named "CODELAB" containing files like "1-MTCNN IMAGEN", "pycache_1-MTCNN.py", "Tarea", "grupo.jpg", "rostroparcial.jpg", "unapersona.jpg", "codelab.py", and "grupo.jpg".
- Code Cell:** Displays Python code for face detection using MTCNN, similar to the previous screenshot but with additional filtering logic to remove false positives.
- Terminal:** Shows the command PS C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tarea>.
- Output:** Shows the output of the code execution, including a list of filtered detected faces and a visual representation of the group photo with green bounding boxes around each person's head and red dots marking key points (eyes and nose).
- Bottom Bar:** Includes status indicators like "Lin. 12, col. 12", "Espacios: 4", "UTF-8", "CRLF", "Python", "3.13.7 (env)", "Go Live", "Quokka", "Prettier", "Zero Two", and "Live Share".

Se detectaron los 15 rostros en un tiempo de 225ms

Ejecución para un rostro parcial

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** Shows a directory structure under "CODELAB" containing files like "1-MTCNN IMAGEN", "rostraparcial.jpg", "codelab.py", and "rostraparcial.jpg".
- Code Cell:** Displays Python code for face detection using MTCNN. The code imports cv2, numpy, and matplotlib, reads an image, performs detection, and displays the results.
- Output Cell:** Shows the command `(env) PS C:\Users\marlo\Documents\JUNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tareas>` followed by the output of the code execution.
- Terminal:** Shows the command `(env) PS C:\Users\marlo\Documents\JUNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tareas>` and the resulting numerical output related to TensorFlow operations.

No pudo reconocer el rostro parcial

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** Shows a directory structure under "CODELAB" containing files like "1-MTCNN IMAGEN", "Tarea", "rostraparcial.jpg", and "unapersona.jpg".
- Code Cell:** Displays Python code for face detection using MTCNN, similar to the previous screenshot but with a different input image.
- Output Cell:** Shows the command `(env) PS C:\Users\marlo\Documents\JUNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tareas>` and the resulting output image, which is a close-up of a smiling person's face.
- Terminal:** Shows the command `(env) PS C:\Users\marlo\Documents\JUNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tareas>` and the resulting numerical output related to TensorFlow operations.

2- Ajusta el umbral de confianza ($0.6 \rightarrow 0.95$) y reporta cuántos rostros quedan.

Ajuste para la imagen de una persona

The screenshot shows a Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows files like `codelab.py`, `rostopcial.jpg`, and `MTCNN IMAGEN`.
- Code Editor:** Displays the `codelab.py` script:

```
1- MTCNN IMAGEN > codelab.py > ...
3   import matplotlib.pyplot as plt
4   from mtcnn.mtcnn import MTCNN
5   from time import time
6
7   # Carga imagen (sube archivos en Colab o usa una URL y descarga)
8   img = cv2.imread('unapersona.jpg') # reemplaza por tu archivo
9   img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar min_face_size
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 ti = time()
15
16 print(f'Detected: {len(res)} rostro(s) * tiempo: {(ti - t0)*100}
17 for r in res:
18     print(r['confidence'], r['box'], r['keypoints'].keys())
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
23         for name, (px,py) in r['keypoints'].items():
24             cv2.circle(vis, (px,py), 2, (255,0,0), -1)
25     plt.imshow(vis)
26     plt.axis('off')
27     plt.show()
28
29 detector = MTCNN() # su NMS interno filtra solapas
```
- Terminal:** Shows command-line output for running the script.
- Output:** Shows the result of the script execution, including a bounding box around a person's face and keypoint detections.
- Status Bar:** Shows file path, line number (Line 15), and other status information.

Se reconoció perfectamente el rostro en un tiempo de 268ms

Ajuste para la imagen del grupo

The screenshot shows a Jupyter Notebook environment with the following details:

- Toolbar:** Archivo, Editor, Selección, Ver, Ejecutar, Terminal, ..., ↶, ↷, 🔍 CODELAB
- Left Sidebar:** EXPLORADOR, CODELAB, 1- MTCNN IMAGEN, __pycache__, Tarea, Strega.docx, entrega.docx, grupo.jpg, rostroacial.jpg, unapersonajpg, codelab.py, grupo.jpg.
- Code Cell:** A code cell titled "codelab.py x entrega.docx" containing the following Python script:

```
1- MTCNN IMAGEN > codelab.py > ...
  3 import matplotlib.pyplot as plt
  4 from mtcnn.mtcnn import MTCNN
  5 from time import time
  6
  7 # Carga imagen (sube archivos en Colab o usa una URL y descárgala)
  8 img = cv2.imread('grupo.jpg') # reemplaza por tu archivo
  9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
 10
 11 detector = MTCNN() # puedes ajustar min_face_size
 12 t0 = time()
 13 res = detector.detect_faces(img_rgb)
 14 t1 = time()
 15
 16 print(f"Detected: {len(res)} rostros * tiempo: ({t1 - t0})*1000")
 17 for r in res:
 18     print(r['confidence'], r['box'], r['keypoints'].keys())
 19     vis = img_rgb.copy()
 20     for r in res:
 21         x, y, w, h = r['box']
 22         cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
 23         for name, (px,py) in r['keypoints'].items():
 24             cv2.circle(vis, (px,py), 2, (255,0,0), -1)
 25     plt.imshow(vis)
 26     plt.axis('off')
 27     plt.show()
 28
 29 detector = MTCNN() # su NMS interno filtra solapas
```
- Output Cell:** Figure 1, showing a photograph of a group of people sitting on grass with green bounding boxes drawn around their faces, indicating where the algorithm detected them.
- Bottom Status Bar:** (env) PS C:\Users\marlo\Documents\IUNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\Tarea & "C:\Users\marlo\Documents\IUNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Scripts\python.exe" "c:/Users/marlo/Documents/IUNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/MTCNN/codelab.py".
- Bottom Icons:** cmd, Python Tarea, ESQUEMA, LÍNEA DE TIEMPO.

Se reconocieron los 15 rostros en un tiempo de 318ms

Para el rostro parcial sigue sin detectarlo

```

1-MTCNN IMAGEN > codelab.py ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga imagen (sube archivos en Colab o usa una URL y descárgala)
8 img = cv2.imread('rostroparcial.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar min_face_size
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 t1 = time()
15
16 print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)*1000:.1f} ms')
17 for r in res:
18     print(f'[confidence]: {r['box']}, {r['keypoints'].keys()')
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
23         for name, (px,py) in r['keypoints'].items():
24             cv2.circle(vis, (px,py), 2, (255,0,0), -1)
25     plt.imshow(vis)
26     plt.axis('off')
27     plt.show()
28
29 detector = MTCNN() # su NMS interno filtra solapas

```

(env) PS C:\Users\marlo\Documents\JUNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tarea> & "C:/Users/marlo/Documents/JUNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\env\Scripts\python.exe" "c:/Users/marlo/Documents/JUNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\codelab.py"

2025-09-21 22:23:04.523313: I tensorflow/core/util/port.cc(153) Ported TensorFlow operations to CPU. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.

2025-09-21 22:23:04.522118: I tensorflow/core/util/port.cc(153) OneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.

Traceback (most recent call last):

File "c:/Users/marlo/Documents/JUNIVALLE/8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\codelab.py", line 9, in <module>
 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
cv2.error: OpenCV(4.2.0) D:\opencv-python\opencv\modules\imgproc\src\color.cpp:199: error: (-215:Assertion Failed) _src.empty() in function 'cv::cvColor'

3- Mide tiempo de inferencia por imagen (promedia 5 corridas).

Una Persona

```

1-MTCNN IMAGEN > codelab.py ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga imagen (sube archivos en Colab o usa una URL y descárgala)
8 img = cv2.imread('unapersona.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar min_face_size
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 t1 = time()
15
16 print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)*1000:.1f} ms')
17 for r in res:
18     print(f'[confidence]: {r['box']}, {r['keypoints']')
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
23         for name, (px,py) in r['keypoints'].items():
24             cv2.circle(vis, (px,py), 2, (255,0,0), -1)
25     plt.imshow(vis)
26     plt.axis('off')
27     plt.show()
28
29 detector = MTCNN() # su NMS interno filtra solapas

```

(env) PS C:\Users\marlo\Documents\JUNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tarea> & "C:/Users/marlo/Documents/JUNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\env\Scripts\python.exe" "c:/Users/marlo/Documents/JUNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\codelab.py"

File "c:/Users/marlo/Documents/JUNIVALLE/8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\lib\site-packages\lz4\frame_init__.py", line 753, in flush
 self._fp.flush()
ValueError: I/O operation on closed file.
Exception ignored: IOError: [Errno 9] Bad file descriptor
Traceback (most recent call last):
File "c:/Users/marlo/Documents/JUNIVALLE/8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\lib\site-packages\lz4\frame_init__.py", line 753, in flush
 self._fp.flush()
ValueError: I/O operation on closed file.
Detected: 1 rostro(s) * tiempo: 163.7 ms
0.995415210723877 [7, 29, 161, 214] dict.keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

163ms

160ms

```

Explorador
  - 1-MTCNN IMAGEN
    - __pycache__
      - mtcnn.cpython-313.pyc
    - Tarea
      - ~Entrega.docx
      - entrega.docx
      - grupo.jpg
      - rostroparcial.jpg
      - rostroparcial2.jpg
      - unapersona.jpg
    - codelab.py
    - grupo.jpg
  - env
  - Introducción a TensorFlow y K...
  - MTCNN VIDEO
  - Reconocimiento de voz

codelab.py > entraga.docx

1- MTCNN IMAGEN > codelab.py > ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga Imagen (sube archivos en Colab o usa una URL de Internet)
8 img = cv2.imread('unapersona.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar min_face_size
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 t1 = time()
15
16 print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)}')
17 for r in res:
18     print(f'[confidence]: {r['confidence']}, r['box'], r['keypoints'].keys())
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
23         for name, (px,py) in r['keypoints'].items():
24             cv2.circle(vis, (px,py), 2, (255,0,0), -1)
25     plt.imshow(vis)
26     plt.axis('off')
27     plt.show()
28
29 detector = MTCNN() # su NMS interno filtra solapas

```

(env) PS C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTON IMAGEN\tarea> & "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\env\Scripts\python.exe" "c:/users/marlo/documents/univalle/8 semestre/proyecto integrador 2\codeLab\1- MTON IMAGEN\codelab.py"

File "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\env\site-packages\l24\frame_init_.py", line 753, in flush
self.fp.flush()
ValueError: I/O operation on closed file.
Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\env\lib\site-packages\l24\frame_init_.py", line 753, in flush
self.fp.flush()
ValueError: I/O operation on closed file.
Detected: 1 rostro(s) * tiempo: 160.4 ms
0.995415210723877 [7, 29, 161, 214] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

Lín. 22, col. 56 | Espacio: 4 | UTF-8 | CR/LF | Python | 3.13.7 (env) | Go Live | Quokka | Prettier | Zero Two | Live Share

165ms

```

Explorador
  - 1-MTCNN IMAGEN
    - __pycache__
      - mtcnn.cpython-313.pyc
    - Tarea
      - ~Entrega.docx
      - entrega.docx
      - grupo.jpg
      - rostroparcial.jpg
      - rostroparcial2.jpg
      - unapersona.jpg
    - codelab.py
    - grupo.jpg
  - env
  - Introducción a TensorFlow y K...
  - MTCNN VIDEO
  - Reconocimiento de voz

codelab.py > entraga.docx

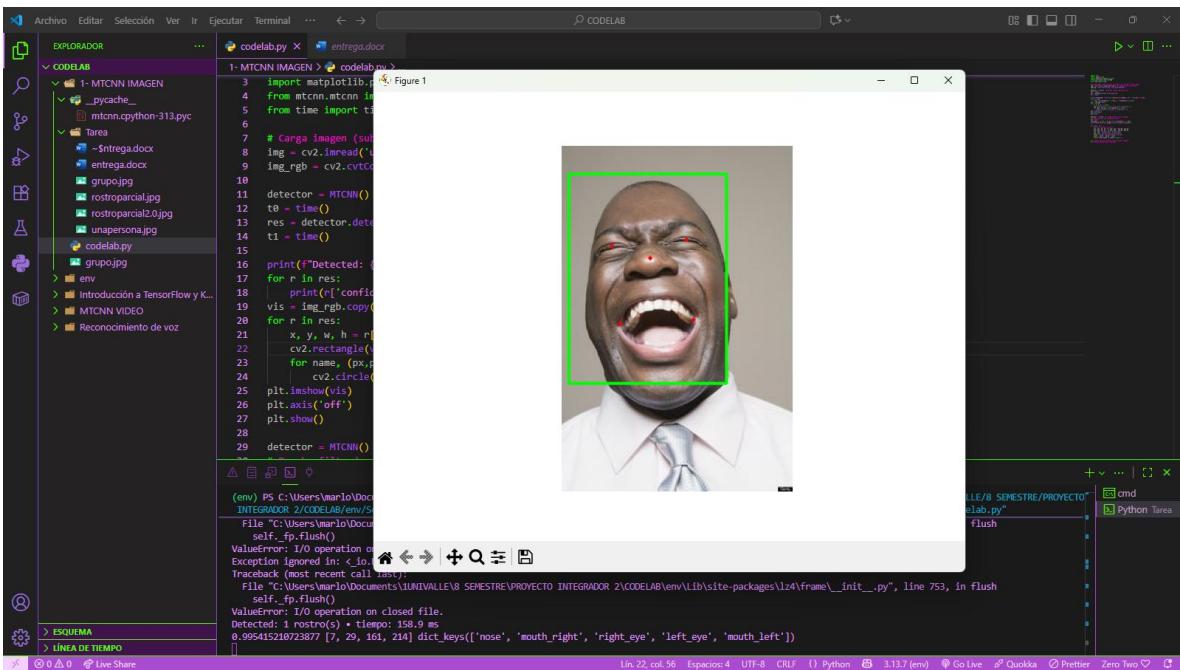
1- MTCNN IMAGEN > codelab.py > ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga Imagen (sube archivos en Colab o usa una URL de Internet)
8 img = cv2.imread('unapersona.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar min_face_size
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 t1 = time()
15
16 print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)}')
17 for r in res:
18     print(f'[confidence]: {r['confidence']}, r['box'], r['keypoints'].keys())
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
23         for name, (px,py) in r['keypoints'].items():
24             cv2.circle(vis, (px,py), 2, (255,0,0), -1)
25     plt.imshow(vis)
26     plt.axis('off')
27     plt.show()
28
29 detector = MTCNN() # su NMS interno filtra solapas

```

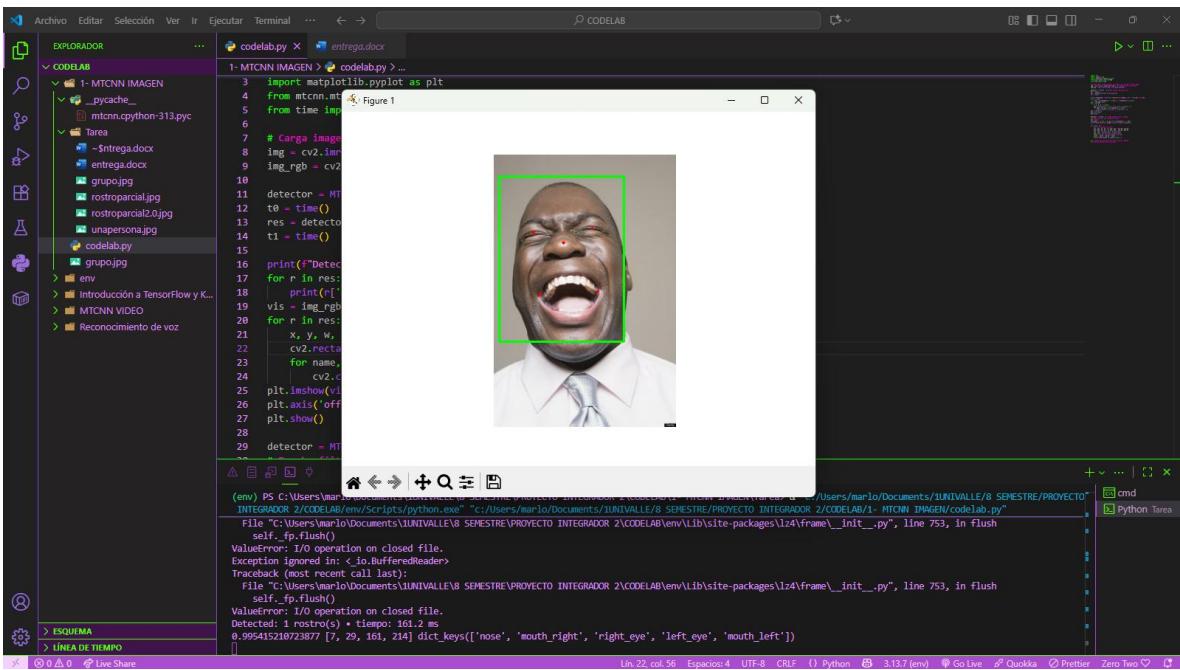
(env) PS C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTON IMAGEN\tarea> & "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\env\Scripts\python.exe" "c:/users/marlo/documents/univalle/8 semestre/proyecto integrador 2\codeLab\1- MTON IMAGEN\codelab.py"

File "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\env\site-packages\l24\frame_init_.py", line 753, in flush
self.fp.flush()
ValueError: I/O operation on closed file.
Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2\CODELAB\env\lib\site-packages\l24\frame_init_.py", line 753, in flush
self.fp.flush()
ValueError: I/O operation on closed file.
Detected: 1 rostro(s) * tiempo: 165.6 ms
0.995415210723877 [7, 29, 161, 214] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

Lín. 22, col. 56 | Espacio: 4 | UTF-8 | CR/LF | Python | 3.13.7 (env) | Go Live | Quokka | Prettier | Zero Two | Live Share



158ms



161ms

El promedio es de 161ms

Promedio de imagen de grupo

```
codelab.py x entrega.docx
1- MTCNN IMAGEN > codelab.py > ...
1 import matplotlib.pyplot as plt
2 from mtcnn.mtcnn import MTCNN
3 from time import time
4
5 # Carga imagen (sube archivos en Colab o usa una URL)
6 img = cv2.imread('grupo.jpg') # reemplaza por tu archivo
7 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
8
9 detector = MTCNN() # puedes ajustar min_face_size
10 t0 = time()
11 res = detector.detect_faces(img_rgb)
12 t1 = time()
13
14 print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)}')
15
16 for r in res:
17     print(f'[confidence] {r["box"]}, {r["keypoints"]}')
18
19 vis = img_rgb.copy()
20
21 for r in res:
22     x, y, w, h = r['box']
23     cv2.rectangle(vis, (x,y), (x+w,y+h), (255,0),
24     for name, (px,py) in r['keypoints'].items():
25         cv2.circle(vis, (px,py), 2, (255,0,0), -1)
26
27 plt.imshow(vis)
28 plt.axis('off')
29 plt.show()
30
31 detector = MTCNN() # su NIV interno filtra solapas
```

PS C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1- MTCNN IMAGEN\Tarea> & "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/env/lib/python3.7/site-packages/lz4/frame_init_py.py", line 753, in flush
self.fp.flush()
ValueError: I/O operation on closed file.
Exception: Abra archivo en el editor (ctrl + clic)
Traceback (most recent call last):
File "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/env/lib/python3.7/site-packages/lz4/frame_init_py.py", line 753, in flush
self.fp.flush()
ValueError: I/O operation on closed file.
Exception: Abra archivo en el editor (ctrl + clic)
Traceback (most recent call last):
File "C:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/env/lib/python3.7/site-packages/lz4/frame_init_py.py", line 753, in flush
self.fp.flush()

217ms

The screenshot shows a Microsoft Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "CODELAB".
- Code Editor:** Displays the file "codelab.py" which contains Python code for face detection using the MTCNN model.
- Terminal:** Shows the command "python codelab.py" being run, outputting the path to the image and the detection results.
- Output Panel:** Shows the execution results, including the bounding boxes and keypoints for each detected face.
- Image Preview:** A window titled "Figure 1" displays a group of people sitting outdoors with green bounding boxes and red keypoints overlaid on their faces, indicating the detected features.

230ms

The screenshot shows a Jupyter Notebook interface with several tabs open. The main code cell contains Python code for using MTCNN to detect faces in an image. The output of the code is a photograph of a group of people sitting on grass, with green bounding boxes and red dots highlighting detected faces and keypoints. Below the code cell, the terminal shows the command run and its output, which includes a traceback and execution time.

```
1- MTCNN IMAGEN > codelab.py >_
  3 import matplotlib.pyplot as plt
  4 from mtcnn.mtcnn import MTCNN
  5 from time import time
  6
  7 # Carga Imagen (sube archivos en Colab o usa una
  8 img = cv2.imread('grupo.jpg') # reemplaza por tu
  9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  10
  11 detector = MTCNN() # puedes ajustar min_face_size
  12 t0 = time()
  13 res = detector.detect_faces(img_rgb)
  14 t1 = time()
  15
  16 print(f'Detected: {len(res)} rostro(s) * tiempo:
  17 for r in res:
  18     print(r['confidence'], r['box'], r['keypoints'])
  19     vis = img_rgb.copy()
  20     for r in res:
  21         x, y, w, h = r['box']
  22         cv2.rectangle(vis, (x,y), (x+w,y+h), (255,
  23         for name, (px,py) in r['keypoints'].items():
  24             cv2.circle(vis, (px,py), 2, (255,0,0), -1)
  25     plt.imshow(vis)
  26     plt.axis('off')
  27     plt.show()
  28
  29 detector = MTCNN() # su NMS interno filtra solo
  (env) PS C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\scripts\python.exe "c:/Users/marlo/Downloads/codelab.py"
  Traceback (most recent call last):
  File "C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Lib\site-packages\lz4\frame\_init__.py", line 753, in flush
    self.fp.flush()
  ValueError: I/O operation on closed file.
  Exception ignored in: <_io.BufferedReader>
  Traceback (most recent call last):
  File "C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Lib\site-packages\lz4\frame\_init__.py", line 753, in flush
    self.fp.flush()
  ValueError: I/O operation on closed file.
  Detected: 15 rostro(s) * tiempo: 408.0 ms
  0.9999867768859 [143, 177, 36, 48] dict_keys(['nose', 'mouth right', 'right eye', 'left eye', 'mouth left'])
```

408ms

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar displays a file tree under the 'EXPLORADOR' tab, showing files like 'codelab.py', 'entrega.docx', 'MTCCN IMAGEN', '_pycache_/*', 'Tarea/*', and 'MTCCN VIDEO'. The main editor area contains a Python script named 'codelab.py' with code for face detection using the MTCNN model. The script imports cv2, matplotlib.pyplot, and mtcnn, then reads an image, initializes a detector, and processes it to find faces and keypoints. The output image shows a group of people sitting outdoors with green bounding boxes around their faces and red circles at key points like the nose and mouth. The bottom terminal window shows the command 'python codelab.py' being run and its output, which includes a traceback and a message indicating 15 faces were detected.

```
1- MTCCN IMAGEN > codelab.py > ...
2- import matplotlib.pyplot as plt
3- from mtcnn.mtcnn import MTCNN
4- from time import time
5-
6- # Carga imagen (sube archivos en Colab o usa una URL y descárgala)
7- img = cv2.imread('grupo.jpg') # reemplaza por tu archivo
8- img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
9-
10- detector = MTCNN() # puedes ajustar min_face_size
11- t0 = time()
12- res = detector.detect_faces(img_rgb)
13- t1 = time()
14-
15- print(f'Detected: {len(res)} rostro(s) • tiempo: {(t1 - t0)*1000:.1f}ms')
16- for r in res:
17-     print(f'{r["confidence"]}, {r["box"]}, {r["keypoints"] keys}')
18- vis = img_rgb.copy()
19- for r in res:
20-     x, y, w, h = r['box']
21-     cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
22-     for name, (px,py) in r['keypoints'].items():
23-         cv2.circle(vis, (px,py), 2, (255,0,0), -1)
24- plt.imshow(vis)
25- plt.axis('off')
26- plt.show()
27-
28- detector = MTCNN() # si NMS interno filtra solapas
29-
```

```
(env) PS C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\INTEGRADOR 2\CODELAB\env\Scripts\python.exe "c:/Users/marlo/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/main.py"
Traceback (most recent call last):
  File "C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\main.py", line 1, in <module>
    self._fp.flush()
ValueError: I/O operation on closed file.
Exception ignored: __io.BufferedReader__
Traceback (most recent call last):
  File "C:\Users\marlo\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Lib\site-packages\lz4\frame\_init__.py", line 753, in flush
    self._fp.flush()
ValueError: I/O operation on closed file.
Detected: 15 rostro(s) • tiempo: 248.4 ms
0.999986767768859 [143, 177, 36, 48] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])
```

248ms

```

EXPLORADOR ... codelab.py entraga.docx
CODELAB
1-MTCNN IMAGEN ...
    __pycache__ ...
        mtcnn.cpython-313.pyc
    tarea ...
        ~Entrega.docx
        grupo.jpg
        rostroparcial.jpg
        rostroparcial2.jpg
        unapersona.jpg
    codelab.py ...
        grupo.jpg
> env ...
    Introducción a TensorFlow y K...
    MTCNN VIDEO ...
    Reconocimiento de voz ...

```

```

1- MTCNN IMAGEN > codelab.py ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga imagen (sube archivo)
8 img = cv2.imread('grupo.jpg')
9 img_rgb = cv2.cvtColor(img, ...
10
11 detector = MTCNN() # pude
12 t0 = time()
13 res = detector.detect_faces(
14     img)
15
16 print(f'Detected: {len(res)}')
17 for r in res:
18     print(f'[confidence], {r["confidence"]}')
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x,y), ...
23             (x+w,y+h), (0,255,0))
24         for name, (px,py) in r['keypoints'].items():
25             cv2.circle(vis, (px,py), ...
26             2, (0,0,255))
27     plt.imshow(vis)
28     plt.axis('off')
29     plt.show()
30
31 detector = MTCNN() # su NPE

```

(env) PS C:\Users\marlio\Documents\JUNI
ValueError: I/O operation on closed file.
Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:\Users\marlio\Documents\JUNI\VALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Lib\site-packages\l2d\frame_init_.py", line 753, in flush
self.fp.flush()
ValueError: I/O operation on closed file.
Detected: 15 rostro(s) • tiempo: 232.6 ms
0.999986767768859 [143, 177, 36, 48] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])
0.999963641166867 [70, 195, 39, 49] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])
0.99978107213974 [495, 121, 38, 35] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])
0.999761164188385 [142, 79, 35, 46] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

232ms

El promedio de la imagen grupos es de 267ms

(Por cuestiones de practicidad y de completar la actividad se agregó una imagen que se asemeja a un rostro parcial por la cercanía de la imagen pero se reconoce sus ojos, boca y nariz)

Promedio rostro parcial

218ms

```

1- MTCNN IMAGEN > codelab.py ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga imagen (sube archivos en Colab o usa una URL y descárgalo)
8 img = cv2.imread('rostroparcial2.0.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar min_face_size
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 t1 = time()
15
16 print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)*1000}
17 for r in res:
18     print(f'[confidence]: {r['confidence']}, {r['box']}, {r['keypoints'].keys()})
19     vis = img_rgb.copy()
20
21 for r in res:
22     x, y, w, h = r['box']
23     cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
24     for name, (px,py) in r['keypoints'].items():
25         cv2.circle(vis, (px,py), 2, (255,0,0), -1)
26
27 plt.imshow(vis)
28 plt.axis('off')
29 plt.show()
30
31 detector = MTCNN() # su NMS interno filtra solapas

```

(env) PS C:\Users\marlio\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\1-MTCNN IMAGEN> python.exe "C:/Users/marlio/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/env/lib/site-packages/lz4/frame_init_.py"
self._fp.flush()
ValueError: I/O operation on closed file.
Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:/Users/marlio/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/env/lib/site-packages/lz4/frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.
Detected: 1 rostro(s) * tiempo: 218.8 ms
0.9227928519248962 [6, 0, 449, 599] dict.keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

210ms

```

1- MTCNN IMAGEN > codelab.py ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga imagen (sube archivos en Colab o usa una URL y descárgalo)
8 img = cv2.imread('rostroparcial.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar min_face_size
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 t1 = time()
15
16 print(f'Detected: {len(res)} rostro(s) * tiempo: {(t1 - t0)*1000}
17 for r in res:
18     print(f'[confidence]: {r['confidence']}, {r['box']}, {r['keypoints'].keys()})
19     vis = img_rgb.copy()
20
21 for r in res:
22     x, y, w, h = r['box']
23     cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
24     for name, (px,py) in r['keypoints'].items():
25         cv2.circle(vis, (px,py), 2, (255,0,0), -1)
26
27 plt.imshow(vis)
28 plt.axis('off')
29 plt.show()
30
31 detector = MTCNN() # su NMS interno filtra solapas

```

Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:/Users/marlio/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/env/lib/site-packages/lz4/frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.
Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:/Users/marlio/Documents/UNIVALLE/8 SEMESTRE/PROYECTO INTEGRADOR 2/CODELAB/env/lib/site-packages/lz4/frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.
Detected: 1 rostro(s) * tiempo: 210.3 ms
0.9227928519248962 [6, 0, 449, 599] dict.keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

218ms

```

Explorador
  -> Codelab
    -> 1-MTCNN IMAGEN
      -> __pycache__
        -> mtcnn.cpython-313.pyc
      -> Tarea
        -> -Sintegra.docx
        -> entregado.docx
        -> grupo.jpg
        -> rostroparcial.jpg
        -> rostroparcial2.jpg
        -> unapersona.jpg
      -> codelab.py
      -> grupo.jpg
    -> env
    -> Introducción a TensorFlow y K...
    -> MTCNN VIDEO
    -> Reconocimiento de voz

codelab.py <--> entregado.docx

1- MTCNN IMAGEN > codelab.py > ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga imagen (sube archivos en Colab o usa una URL y descárgala)
8 img = cv2.imread('rostroparcial2.0.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 t1 = time()
15
16 print(f'Detected: {len(res)} rostros')
17 for r in res:
18     print(f'[confidence]: {r["box"]}')
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
23         for name, (px,py) in r['keypoints']:
24             cv2.circle(vis, (px,py), 2, (0,0,255), -1)
25     plt.imshow(vis)
26     plt.axis('off')
27     plt.show()
28
29 detector = MTCNN() # su NMS interno

```

Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:\Users\marl\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Lib\site-packages\lz4\frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.
Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:\Users\marl\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Lib\site-packages\lz4\frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.
Detected: 1 rostro(s) * tiempo: 218.2 ms
0.9227928519248962 [8, 0, 449, 599] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

229ms

```

Explorador
  -> Codelab
    -> 1-MTCNN IMAGEN
      -> __pycache__
        -> mtcnn.cpython-313.pyc
      -> Tarea
        -> -Sintegra.docx
        -> entregado.docx
        -> grupo.jpg
        -> rostroparcial.jpg
        -> rostroparcial2.jpg
        -> unapersona.jpg
      -> codelab.py
      -> grupo.jpg
    -> env
    -> Introducción a TensorFlow y K...
    -> MTCNN VIDEO
    -> Reconocimiento de voz

codelab.py <--> entregado.docx

1- MTCNN IMAGEN > codelab.py > ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga imagen (sube archivos en Colab o usa una URL y descárgala)
8 img = cv2.imread('rostroparcial2.0.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN() # puedes ajustar
12 t0 = time()
13 res = detector.detect_faces(img_rgb)
14 t1 = time()
15
16 print(f'Detected: {len(res)} rostros')
17 for r in res:
18     print(f'[confidence]: {r["box"]}')
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x,y), (x+w, y+h), (0,255,0), 2)
23         for name, (px,py) in r['keypoints']:
24             cv2.circle(vis, (px,py), 2, (0,0,255), -1)
25     plt.imshow(vis)
26     plt.axis('off')
27     plt.show()
28
29 detector = MTCNN() # su NMS interno

```

Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:\Users\marl\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Lib\site-packages\lz4\frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.
Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:\Users\marl\Documents\UNIVALLE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\Lib\site-packages\lz4\frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.
Detected: 1 rostro(s) * tiempo: 229.4 ms
0.9227928519248962 [8, 0, 449, 599] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

```

1- MTCNN IMAGEN > cd colab.py ...
3 import matplotlib.pyplot as plt
4 from mtcnn.mtcnn import MTCNN
5 from time import time
6
7 # Carga imagen (sube archivos en Colab o usa una URL y descárgala)
8 img = cv2.imread('rostroparcial2.0.jpg') # reemplaza por tu archivo
9 img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 detector = MTCNN()
12 t0 = time()
13 res = detector.detect_faces(img)
14 t1 = time()
15
16 print(f'Detected: {len(res)}')
17 for r in res:
18     print(f'confidence: {r["confidence"]}')
19     vis = img_rgb.copy()
20     for r in res:
21         x, y, w, h = r['box']
22         cv2.rectangle(vis, (x, y), (x+w, y+h), (0, 255, 0), 2)
23         for name, (px, py) in r['landmarks'].items():
24             cv2.circle(vis, (px, py), 2, (0, 0, 255), -1)
25     plt.imshow(vis)
26     plt.axis('off')
27     plt.show()
28
29 detector = MTCNN()

```

Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:\Users\marlo\Documents\UNIVALE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\lib\site-packages\liz4\frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.
Exception ignored in: <_io.BufferedReader>
Traceback (most recent call last):
File "C:\Users\marlo\Documents\UNIVALE\8 SEMESTRE\PROYECTO INTEGRADOR 2\CODELAB\env\lib\site-packages\liz4\frame_init_.py", line 753, in flush
self._fp.flush()
ValueError: I/O operation on closed file.

Detectado: 1 rostro(s) * tiempo: 229.4 ms
0.9237928519248962 [6, 0, 449, 599] dict_keys(['nose', 'mouth_right', 'right_eye', 'left_eye', 'mouth_left'])

218ms

El promedio es de 218ms

4- Dibuja landmarks (ojos, nariz, boca) y explica un uso de alignment.

Las imágenes anteriores todas se manejaron con landmarks

El uso de alignment sirve para normalizar los rostros que no están perfectamente alineados a la perspectiva de la cámara y así poder que el sistema también incluya estos rostros.