```python
# Python Face_Recognition
#   https://face-recognition.readthedocs.io/en/latest/readme.html#
# conda create -n facer opencv pillow cmake scipy scikit-learn natsort
# conda activate facer
# conda install -c conda-forge dlib
# conda install -c akode face_recognition_models
# pip install --no-dependencies face_recognition

import face_recognition
import os, sys, cv2, natsort

FRAME_THICKNESS = 3
FONT_THICKNESS = 2
MODEL = "cnn"   # "hog"
COLOR_GRAY = (200,200,200)

def recog_face_from_driectory(known_faces_dir="known", unknown_faces_dir="unknown", tolerance=0.6) :
    # tolerance = 0.6 # 낮을수록 엄밀한 비교
    print("알고 있는 얼굴 로딩...")
    known_faces = []
    known_names = []

    for name in os.listdir(known_faces_dir):
        for filename in os.listdir(f"{known_faces_dir}/{name}"):
            image = face_recognition.load_image_file(f"{known_faces_dir}/{name}/{filename}")
                # 영상내의 각각의 얼굴에 대하여 128차원의  face encoding정보 구하기
            encoding = face_recognition.face_encodings(image)[0]
            known_faces.append(encoding)
            known_names.append(name)

    print("Wt폴더명:",known_names)
    print("모르는 얼굴 로딩...")
    for filename in os.listdir(unknown_faces_dir):
        print("Wt",filename,"......")
        image = face_recognition.load_image_file(f"{unknown_faces_dir}/{filename}")
        locations = face_recognition.face_locations(image) # (top, right, bottom, left)
        print("얼굴영역:",locations)
        encodings = face_recognition.face_encodings(image,locations)
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        for face_encoding, face_location in zip(encodings, locations):
            results = face_recognition.compare_faces(known_faces, face_encoding, tolerance)
            distances = face_recognition.face_distance(known_faces, face_encoding)
            print("거리:",distances)

            match = None
            if True in results:
                match = known_names[results.index(True)]
                print(f"Match 발견: {match}")

                top_left = (face_location[3], face_location[0])
                bottom_right = (face_location[1], face_location[2])
                color = [0, 255, 0]
                cv2.rectangle(image, top_left, bottom_right, color, FRAME_THICKNESS)

                top_left = (face_location[3], face_location[2])
                bottom_right = (face_location[1], face_location[2]+20)
                cv2.rectangle(image, top_left, bottom_right, color, cv2.FILLED)
                cv2.putText(image, match, (face_location[3]+10, face_location[2]+15),
                    cv2.FONT_HERSHEY_SIMPLEX, 3.0, COLOR_GRAY, FONT_THICKNESS)

        cv2.namedWindow(filename,cv2.WINDOW_NORMAL)
        cv2.resizeWindow(filename,1024,768)
        cv2.moveWindow(filename,100,50)

```

```
        cv2.imshow(filename, image)
        cv2.waitKey(1)
        cv2.destroyWindow(filename)

def recog_face_from_cam(cam_no, known_faces_dir = "known", tolerance=0.6) :
    # tolerance = 0.6 # 낮을수록 엄밀한 비교
    cap = cv2.VideoCapture(cam_no)

    print("알고 있는 얼굴 로딩...")
    known_faces = []
    known_names = []

    for name in os.listdir(known_faces_dir):
        for filename in os.listdir(f"{known_faces_dir}/{name}"):
            image = face_recognition.load_image_file(f"{known_faces_dir}/{name}/{filename}")
                # 영상내의 각각의 얼굴에 대하여 128차원의  face encoding정보 구하기
            encoding = face_recognition.face_encodings(image)[0]
            known_faces.append(encoding)
            known_names.append(name)

    while True :
        ret, frame = cap.read()

        # 카메라 영상, BGR형식에서 RGB형식으로 변환
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        locations = face_recognition.face_locations(image, model="hog") # (top, right, bottom, left)
        encodings = face_recognition.face_encodings(image, locations)

        print("얼굴영역:",locations)
        encodings = face_recognition.face_encodings(image,locations)
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        for face_encoding, face_location in zip(encodings, locations):
            results = face_recognition.compare_faces(known_faces, face_encoding, tolerance)
            distances = face_recognition.face_distance(known_faces, face_encoding)
            print("거리:",distances)

            match = None
            if True in results:
                match = known_names[results.index(True)]
                print(f"Match 발견: {match}")

                top_left = (face_location[3], face_location[0])
                bottom_right = (face_location[1], face_location[2])
                color = [0, 255, 0]
                cv2.rectangle(image, top_left, bottom_right, color, FRAME_THICKNESS)

                top_left = (face_location[3], face_location[2])
                bottom_right = (face_location[1], face_location[2]+20)
                cv2.rectangle(image, top_left, bottom_right, color, cv2.FILLED)
                cv2.putText(image, match, (face_location[3]+10, face_location[2]+15),
                    cv2.FONT_HERSHEY_SIMPLEX, 3.0, COLOR_GRAY, FONT_THICKNESS)

        cv2.namedWindow("CAM",cv2.WINDOW_NORMAL)
        cv2.resizeWindow("CAM",1024,768)
        cv2.moveWindow("CAM",100,50)

        cv2.imshow("CAM", image)
        inKey = cv2.waitKey(1)
        if inKey == 27 :
            cv2.destroyWindow("CAM")
            break

def capture_face_from_cam(camNo, capture_faces_dir = "captured", tolerance=0.6) :
    # tolerance = 0.6 # 낮을수록 엄밀한 비교
    cap = cv2.VideoCapture(camNo)
```

```
    if not os.path.isdir(capture_faces_dir):
        os.mkdir(capture_faces_dir)

    fileList = os.listdir(capture_faces_dir)
    if len(fileList) == 0 :
        nextNo = 1
    else :
        orderedFileList = natsort.natsorted(fileList)
        print("orderedFileList",orderedFileList)
        nextNo = int(orderedFileList[len(orderedFileList)-1][:-4])+1

    while True :
        ret, frame = cap.read()

        # 카메라 영상, BGR형식에서 RGB형식으로 변환
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        cv2.imshow("CAM", frame)
        inKey = cv2.waitKey(1)
        if inKey == 27 :
            cv2.destroyWindow("CAM")
            break
        elif inKey == ord('s') :
            filename = capture_faces_dir+"/"+str(nextNo)+".png" # JPG 말고 PNG로
            print(filename," is saved")
            cv2.imwrite(filename, frame)
            nextNo += 1

if __name__ == "__main__"   :
    if len(sys.argv) > 1:
        if len(sys.argv)>2 :
            camNo = int(sys.argv[2])
        else :
            camNo = 0

        if sys.argv[1] == 'dir':
            recog_face_from_driectory("known", "unknown", 0.6)
        elif sys.argv[1] == 'cam' :
            recog_face_from_cam(camNo, "known", 0.6)
        elif sys.argv[1] == 'capture' :
            capture_face_from_cam(camNo, "captured")
    else  :
        print("사용법: python facerecog_cam.py dir/cam/capture")
        print("사용법: python facerecog_cam.py capture cam_no")
```