Wykład Specjalizujący

Zadanie 5

Wykrywanie obiektów za pomocą OpenCV

Autor:

Jerzy Dębowski 151266

Klasyfikatory

Do poprawnego działania należy zdefiniować klasyfikatory z biblioteki OpenCV

```
face_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
)

smile_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_smile.xml'
)

eye_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_eye.xml'
)
```

1. Wykrywanie twarzy na zdjęciu

Do zadania użyłem losowego zdjęcia z internetu:

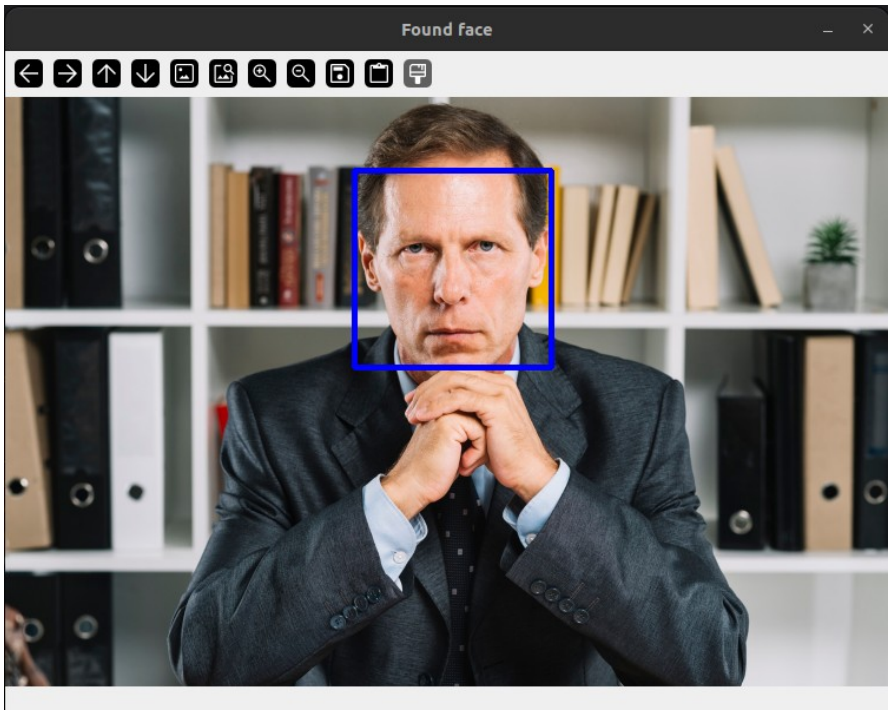
Aby wykryć twarz należy najpierw przeskalować obrazek funkcją cv2.resize z atrybutami dsize=None, fx=0.5, fy=0.5 oraz interpolation=cv2.INTER_AREA

Następnie na przeskalowanym obrazku szuakmy twarzy przy użyciu klasyfikatora face_cascade. Podajemy atrybuty scaleFactor=1.3 oraz minNeighbours=3.

Na końcu w miejscu znalezionych twarzy rysujemy prostokąty metodą cv2.rectangle

```python
def find_face(img_path, scaling_factor=0.5, scale_factor=1.3, min_neighbours=3):
    frame = cv2.imread(img_path)
    frame = cv2.resize(
        frame,
        dsize: None,
        fx=scaling_factor,
        fy=scaling_factor,
        interpolation=cv2.INTER_AREA,
    )
    face_rects = face_cascade.detectMultiScale(
        frame,
        scaleFactor=scale_factor,
        minNeighbors=min_neighbours,
    )

    for (x, y, w, h) in face_rects:
        cv2.rectangle(
            frame,
            (x, y),
            (x + w, y + h),
            (255, 0, 0),
            3,
        )
    cv2.imshow( winname: 'Found face', frame)
```



2. Wykrywanie twarzy, uśmiechów oraz oczu na zdjęciu wielu osób

Do zadania użyłem losowego zdjęcia z internetu:

Pierwszym krokiem jest wykrycie twarzy. Działa to identycznie jak w poprzednim zadaniu. Musiałem jedynie manewrować atrybutami scaleFactor oraz minNeighbours. Optymalnymi wartościami okazały się odpowiednio 1.2 oraz 8

Następnie na wykrytych twarzach należy wykryć uśmiechy oraz oczy. Służą do tego klasyfikatory smile_cascade oraz eye_cascade. Przyjmują one takie same parametry jak klasyfikator do wykrywania twarzy. Do klasyfikatora oczu przyjąłem parametry scaleFactor=1.1 oraz minNeighbours=6, a dla klasyfikatora uśmiechu odpowienio 1.1 oraz 55. Dodatkowo do klasyfikatora uśmiechu podałem atrybut minSize=(25,25) aby zwiększyć dokładność.

Aby zliczyć liczbę osób na zdjęciu wystarczy znać ilość wykrytych twarzy (len(faces))

```
jerzy
def find_smiles(img_path):
    image = cv2.imread(img_path)
    gray_filter = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray_filter, scaleFactor: 1.2, minNeighbors: 8)

    for (x, y, w, h) in faces:
        cv2.rectangle(
            image,
            (x, y),
            (x + w, y + h),
            (255, 0, 0),
            2
        )
        roi_gray = gray_filter[y:y + h, x:x + w]
        roi_color = image[y:y + h, x:x + w]
        smile = smile_cascade.detectMultiScale(roi_gray, scaleFactor: 1.1, minNeighbors: 55, minSize=(25, 25))
        eye = eye_cascade.detectMultiScale(roi_gray, scaleFactor: 1.1, minNeighbors: 6)
        for (sx, sy, sw, sh) in smile:
            cv2.rectangle(
                roi_color,
                (sx, sy),
                (sx + sw, sy + sh),
                (0, 255, 0),
                1
            )
        for (ex, ey, ew, eh) in eye:
            cv2.rectangle(
                roi_color,
                (ex, ey),
                (ex + ew, ey + eh),
                (0, 0, 255),
                1
            )
    print(f'Znaleziono: {len(faces)} osób')
    cv2.imshow( winname: 'Smiles', image)
```

Klasyfikator ma problem z rozpaznaniem oczu, gdy osoba na zdjęciu nosi okulary. Nie udało mi się wykryć oczu u wszystkich osób na zdjęciu. Najbardziej optymalny wynik jaki udało mi się uzyskać znajduje się poniżej:


```
Znaleziono: 7 osób
```

3. Wykrywanie twarzy z kamery

Zasada działania jest identyczna, jak w przypadku wykrywania twarzy ze zdjęcia, z tą różnicą, że zamiast zdjęcia podajemy do funkcji pojedynczą klatkę z kamery, do której otrzymujemy dostęp poprzez cap=cv2.VideoCapture(0). Pojedyncza klatka jest dostępna poprzez funkcję cap.read().

```python
def capture_face_from_video():
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        face_rects = face_cascade.detectMultiScale(
            frame,
            scaleFactor=2,
            minNeighbors=3,
        )
        for (x, y, w, h) in face_rects:
            cv2.rectangle(
                frame,
                (x, y),
                (x + w, y + h),
                (255, 0, 0),
                3,
            )
        cv2.imshow( winname: 'Video', frame)
        print(f'Osoby na video: {len(face_rects)}')
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
```

4. Wykrywanie osób na video

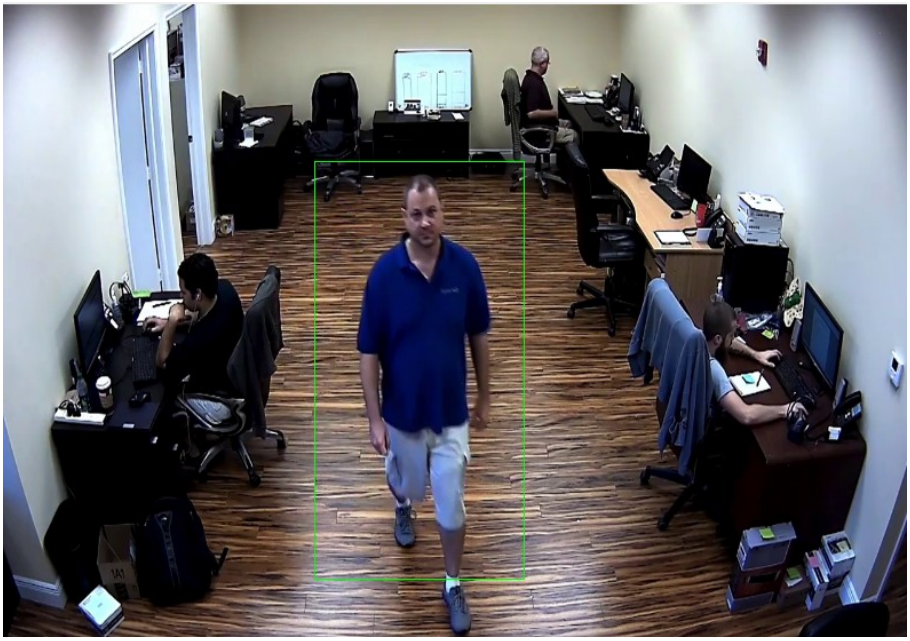
Do zadania wykorzystałem przykładowe nagranie z kamery dostępne na serwisie youtube: https://www.youtube.com/watch?v=9wxEmqyVlB8

Użylem fragmentu od 10 do 30 sekundy.
Do wykrywania osób na pojedynczej klatce służy klasyfikator cv2.HOGDescriptor()

Film odczytujemy poprzez cv2.VideoCapture(‘video.mp4’)
Aby ułatwić algorytmowi wykrycie osób zdecydowałem się na przeskalowanie klatek do rozmiaru (800, 560) oraz przekonwertowanie do koloru czarno-białego

```python
def find_people_on_video():
    cap = cv2.VideoCapture('video.mp4')
    hog = cv2.HOGDescriptor()
    hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
    while True:
        ret, frame = cap.read()
        frame = cv2.resize(frame, dsize: (800, 560))
        gray_filter = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        boxes, weights = hog.detectMultiScale(gray_filter, winStride=(8, 8), scale=1.1)
        boxes = numpy.array([[x, y, x + w, y + h] for (x, y, w, h) in boxes])
        for (xa, ya, xb, yb) in boxes:
            cv2.rectangle(
                frame,
                (xa, ya),
                (xb, yb),
                (0, 255, 0),
                1,
            )
        cv2.imshow( winname: 'Video', frame)
        print(f'Osoby na video: {len(boxes)}')
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
```

Algorytm ma problem z wykrywaniem osób siedzących przy biurku, co widać na załączonym obrazku:



```
Osoby na video: 1
```