

Spis treści

1 Treść zadania:	3
1.1 Opis metody wykonania (kod):	3
1.2 Wynik(wykres):	3
2 Treść zadania:	4
2.1 Opis metody wykonania (kod):	4
2.2 Wynik(wykres):	5
3 Treść zadania:	5
3.1 Opis metody wykonania (kod):	5
3.2 Wynik(wykres):	6
4 Treść zadania:	6
4.1 Opis metody wykonania (kod):	7
4.2 Wynik(wykres):	8
5 Treść zadania:	8
5.1 Opis metody wykonania (kod):	8
5.2 Wynik(wykres):	9
6 Treść zadania:	10
6.1 Opis metody wykonania (kod):	10
6.2 Wynik(wykres):	11
7 Treść zadania:	11
7.1 Opis metody wykonania (kod):	12
7.2 Wynik(wykres):	13
8 Treść zadania:	14
8.1 Opis metody wykonania (kod):	14
8.2 Wynik(wykres):	15

9 Treść zadania:	16
9.1 Opis metody wykonania (kod):	16
9.2 Wynik(wykres):	18
10 Treść zadania:	18
10.1 Opis metody wykonania (kod):	18
10.2 Wynik(wykres):	21
11 Treść zadania:	22
11.1 Opis metody wykonania (kod):	22
11.2 Wynik(wykres):	24

1 Treść zadania:

Wykorzystując funkcję imread wczytaj obrazek lena.png oraz wyświetl go za pomocą komendy imshow

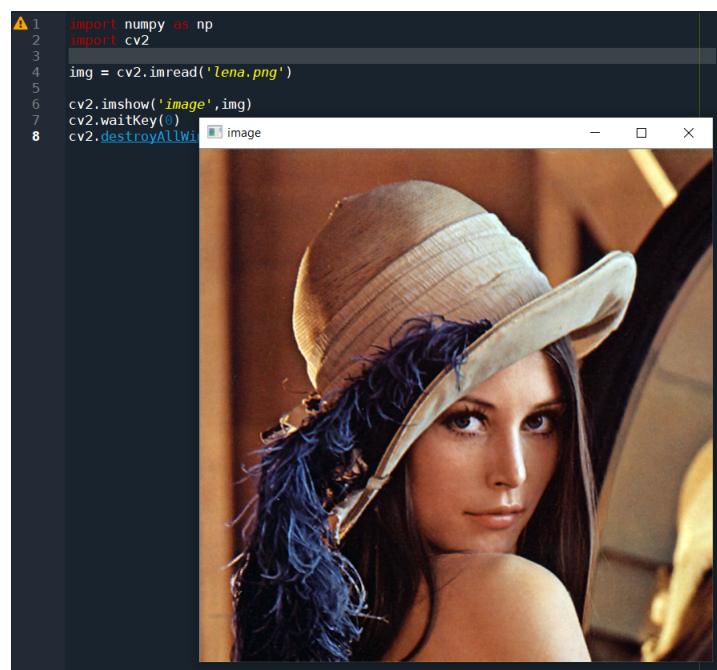
1.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

img = cv2.imread('lena.png')

cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1.2 Wynik(wykres):



Rysunek 1: A Figure1.

2 Treść zadania:

Wykorzystując funkcję rectangle zaznacz maksymalnie wysunięte części twarzy modelki czerwonym prostokątem. Za pomocą funkcji putText, w prawym dolnym rogu dodaj napis „Laboratoria SK”.

2.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

img = cv2.imread('lena.png')

cv2.rectangle(img,(210,210),(360,400),(0,0,255),3)

font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'Labolatorium SK',(305,500), font, 0.75,(255,255,255),2, cv2.LINE_AA)

cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.2 Wynik(wykres):



Rysunek 2: A Figure1.

3 Treść zadania:

Wykorzystując funkcję cvtColor z parametrem COLOR-BGR2GRAY dokonaj konwersji pliku do skali szarości.

3.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

img = cv2.imread('lena.png')

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.rectangle(img,(210,210),(360,400),(0,0,255),3)
```

```

font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'Labolatorium SK',(305,500), font, 0.75,(255,255,255),2, cv2.LINE_AA)

cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

3.2 Wynik(wykres):



Rysunek 3: A Figure1.

4 Treść zadania:

Wykorzystując funkcję imwrite zapisz powstały obrazek jako lena-gray.png

4.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

img = cv2.imread('lena.png')

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY )

cv2.rectangle(img,(210,210),(360,400),(0,0,255),3)

font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'Labolatorium SK',(305,500), font, 0.75,(255,255,255),2, cv2.LINE_AA)

cv2.imshow('image',img)
cv2.imwrite('lena_grey.png',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4.2 Wynik(wykres):



Rysunek 4: A Figure1.

5 Treść zadania:

Pamiętając o macierzowej reprezentacji obrazów w bibliotece OpenCV wytnij i zapisz zaznaczoną twarz modelki do pliku lena-gray-template.png.

5.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

img = cv2.imread('lena.png')

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY )
```

```

cv2.rectangle(img,(210,210),(360,400),(0,0,255),3)

font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'Labolatorium SK',(305,500), font, 0.75,(255,255,255),2, cv2.LINE_AA)

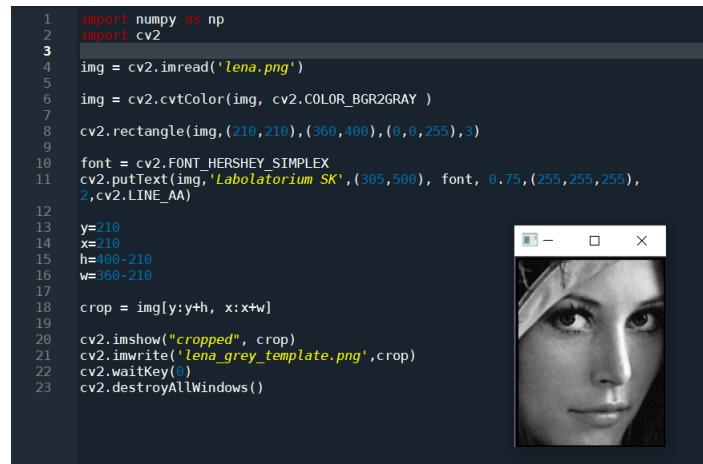
y=210
x=210
h=400-210
w=360-210

crop = img[y:y+h, x:x+w]

cv2.imshow("cropped", crop)
cv2.imwrite('lena_grey_template.png',crop)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

5.2 Wynik(wykres):



```

1 import numpy as np
2 import cv2
3
4
5 img = cv2.imread('lena.png')
6
7 img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY )
8
9 cv2.rectangle(img,(210,210),(360,400),(0,0,255),3)
10
11 font = cv2.FONT_HERSHEY_SIMPLEX
12 cv2.putText(img,'Labolatorium SK',(305,500), font, 0.75,(255,255,255),
13 2, cv2.LINE_AA)
14
15 y=210
16 x=210
17 h=400-210
18 w=360-210
19
20 crop = img[y:y+h, x:x+w]
21
22 cv2.imshow("cropped", crop)
23 cv2.imwrite('lena_grey_template.png',crop)
24 cv2.waitKey(0)
25 cv2.destroyAllWindows()

```

Rysunek 5: A Figure1.

6 Treść zadania:

Wykorzystując funkcje getRotationMatrix2D oraz warpAffine obróć oraz nadpisz obraz lena-gray-template.png o 15°.

6.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

img = cv2.imread('lena.png')

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY )

cv2.rectangle(img,(210,210),(360,400),(0,0,255),3)

font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img,'Labolatorium SK',(305,500), font, 0.75,(255,255,255),2, cv2.LINE_AA)

y=210
x=210
h=400-210
w=360-210

crop = img[y:y+h, x:x+w]

cv2.imwrite('lena_grey.png',crop)

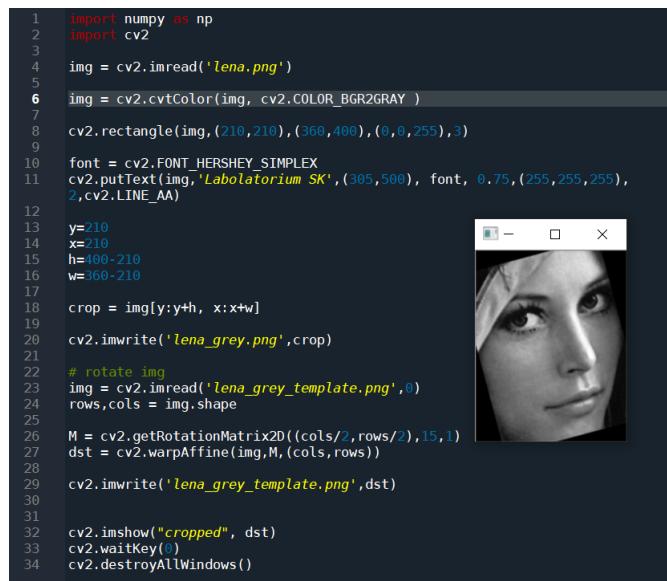
# rotate img
img = cv2.imread('lena_grey_template.png',0)
rows,cols = img.shape

M = cv2.getRotationMatrix2D((cols/2,rows/2),15,1)
dst = cv2.warpAffine(img,M,(cols,rows))
```

```
cv2.imwrite('lena_grey_template.png',dst)
```

```
cv2.imshow("cropped", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.2 Wynik(wykres):



```
1 import numpy as np
2 import cv2
3
4 img = cv2.imread('lena.png')
5
6 img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
7
8 cv2.rectangle(img, (210,210), (360,400), (0,0,255), 3)
9
10 font = cv2.FONT_HERSHEY_SIMPLEX
11 cv2.putText(img, "Labolatorium SK", (305,500), font, 0.75,(255,255,255),
12 2, cv2.LINE_AA)
13
14 y=210
15 x=210
16 h=400-210
17 w=360-210
18
19 crop = img[y:y+h, x:x+w]
20 cv2.imwrite('lena_grey.png',crop)
21
22 # rotate img
23 img = cv2.imread('lena_grey_template.png',0)
24 rows,cols = img.shape
25
26 M = cv2.getRotationMatrix2D((cols/2,rows/2),15,1)
27 dst = cv2.warpAffine(img,M,(cols,rows))
28
29 cv2.imwrite('lena_grey_template.png',dst)
30
31
32 cv2.imshow("cropped", dst)
33 cv2.waitKey(0)
34 cv2.destroyAllWindows()
```

Rysunek 6: A Figure1.

7 Treść zadania:

Wykorzystując funkcję matchTemplate sprawdź wyniki dopasowania lena-gray-template.png do lena-gray.png dla każdego z następujących parametrów:

CV-TM-SQDIFF,

CV-TM-SQDIFF-NORMED,

CV-TM-CCORR,

CV-TM-CCORR-NORMED,
CV-TM-CCOEFF,
CV-TM-CCOEFF-NORMED.

7.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

# load images
image = cv2.imread('lena_grey.png')
template = cv2.imread('lena_grey_template.png')

# resize images
image = cv2.resize(image, (0,0), fx=0.5, fy=0.5)
template = cv2.resize(template, (0,0), fx=0.5, fy=0.5)

# Convert to grayscale better detection
imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
templateGray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)
# 6 statistical method as 3rd argument in this method
# Find template (to find the most similar area in the image)
result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCOEFF)
# result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCOEFF_NORMED)
# result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_SQDIFF)
# result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_SQDIFF_NORMED)
# result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCORR)
# result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCORR_NORMED)

#Finds the global minimum and maximum in an array.
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
top_left = max_loc
h,w = templateGray.shape
bottom_right = (top_left[0] + w, top_left[1] + h)
```

```

cv2.rectangle(image,top_left, bottom_right,(0,0,255),4)

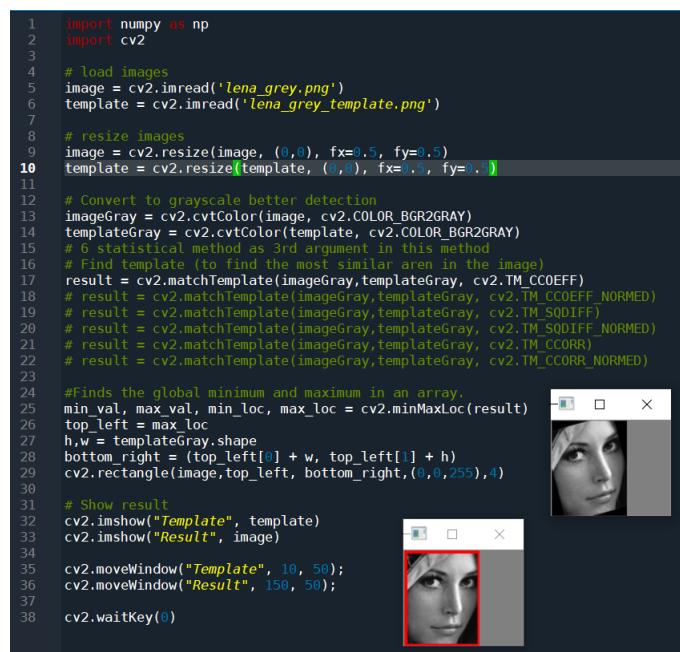
# Show result
cv2.imshow("Template", template)
cv2.imshow("Result", image)

cv2.moveWindow("Template", 10, 50);
cv2.moveWindow("Result", 150, 50);

cv2.waitKey(0)

```

7.2 Wynik(wykres):



```

1 import numpy as np
2 import cv2
3
4 # Load images
5 image = cv2.imread('lena_grey.png')
6 template = cv2.imread('lena_grey_template.png')
7
8 # Resize images
9 image = cv2.resize(image, (0,0), fx=0.5, fy=0.5)
10 template = cv2.resize(template, (0,0), fx=0.5, fy=0.5)
11
12 # Convert to grayscale better detection
13 imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
14 templateGray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)
15 # 6 statistical method as 3rd argument in this method
16 # Find template (to find the most similar area in the image)
17 result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCOEFF)
18 # result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCOEFF_NORMED)
19 # result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_SQDIFF)
20 # result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_SQDIFF_NORMED)
21 # result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCORR)
22 # result = cv2.matchTemplate(imageGray,templateGray, cv2.TM_CCORR_NORMED)
23
24 # Finds the global minimum and maximum in an array.
25 min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
26 top_left = max_loc
27 h,w = templateGray.shape
28 bottom_right = (top_left[0] + w, top_left[1] + h)
29 cv2.rectangle(image,top_left, bottom_right,(0,0,255),4)
30
31 # Show result
32 cv2.imshow("Template", template)
33 cv2.imshow("Result", image)
34
35 cv2.moveWindow("Template", 10, 50);
36 cv2.moveWindow("Result", 150, 50);
37
38 cv2.waitKey(0)

```

Rysunek 7: A Figure1.

8 Treść zadania:

Wyświetl obraz z kamery oraz wykorzystując klasyfikatory haarcascade-frontalface-default.xml oraz haarcascade-eye.xml znajdź twarz oraz oczy na uzyskanym obrazie. Zaznacz znalezione elementy za pomocą czerwonych prostokątów. Przydatne funkcje:

CascadeClassifier,
detectMultiScale.

8.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

# multiple cascades: https://github.com/Itseez/opencv/tree/master/data/haarcascades

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

cap = cv2.VideoCapture(0)

while 1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

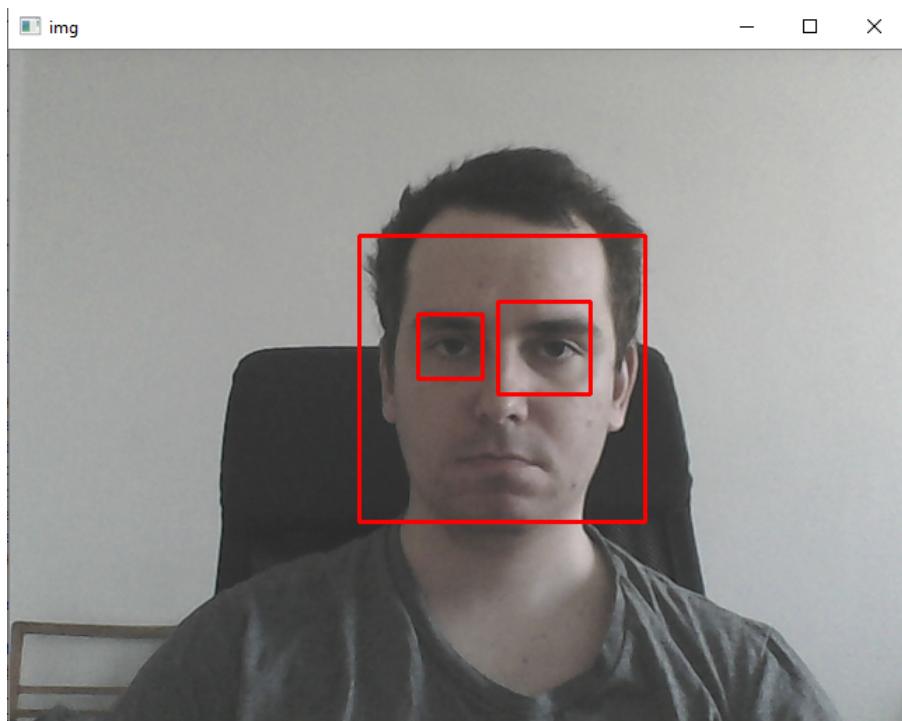
        eyes = eye_cascade.detectMultiScale(roi_gray)
        for (ex,ey,ew,eh) in eyes:
```

```
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,0,255),2)

cv2.imshow('img',img)
k = cv2.waitKey(30) & 0xff
if k == 27:
    break

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

8.2 Wynik(wykres):



Rysunek 8: A Figure1.

9 Treść zadania:

Wyświetl pozycję x, y znalezionych oczu na obrazie. Pamiętaj, aby wyświetlona wartość odnosiła się do środka znalezioneego obszaru.

9.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

# multiple cascades: https://github.com/Itseez/opencv/tree/master/data/haarcascades

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

cap = cv2.VideoCapture(0)

while 1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.2, 2)

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)

    for (ex,ey,ew,eh) in eyes:
        aaa=cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,0,255),2)
        font = cv2.FONT_HERSHEY_SIMPLEX
```

```

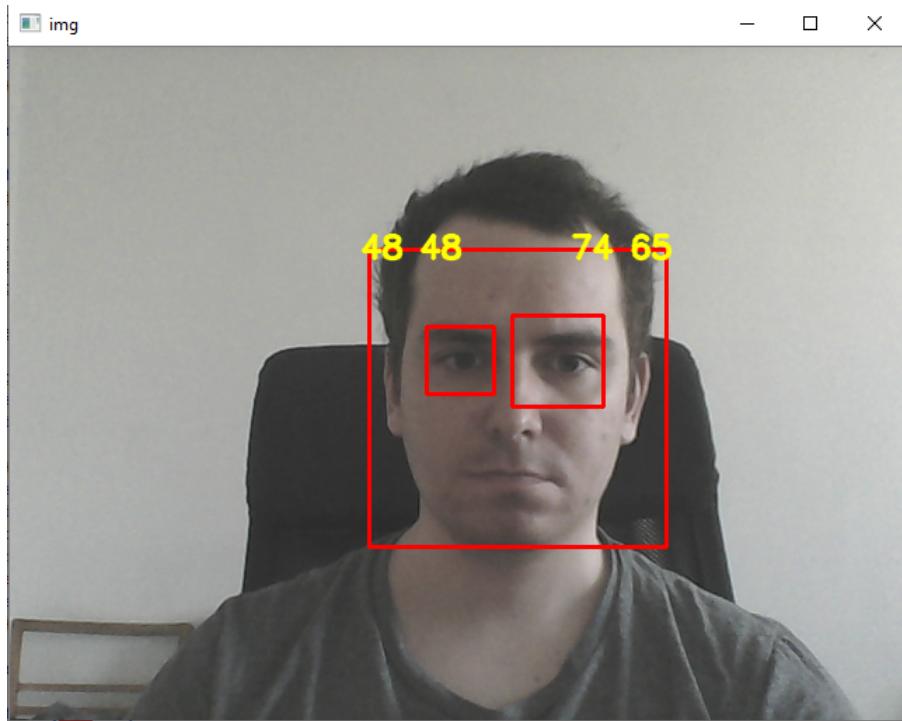
c0 = int((ex+ey)/2)
c1 = int((ew+eh)/2)
# print('Ball centre: {},{}'.format(c0,c1))
ca = str(c0)
cb = str(c1)
xxxx = ca+' '+cb
# print(eyes.shape)
if eyes.shape == (2,4):
    if ex in eyes[1,:]:
        cv2.putText(img,xxxx,(400,150), font, 0.75,(0,255,255),2, cv2.LINE_AA)

    if ex in eyes[0,:]:
        cv2.putText(img,xxxx,(250,150), font, 0.75,(0,255,255),2, cv2.LINE_AA)
else:
    break
cv2.imshow('img',img)
k = cv2.waitKey(30) & 0xff
if k == 27:
    break

cap.release()
cv2.destroyAllWindows()

```

9.2 Wynik(wykres):



Rysunek 9: A Figure1.

10 Treść zadania:

Napisz prosty interfejs człowiek-komputer wyświetlający na obrazie z kamery liczbę 1 gdy komputer wykryje na nim otwarte oczy oraz liczbę 0 w przeciwnym przypadku.

10.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

# multiple cascades: https://github.com/Itseez/opencv/tree/master/data/haarcascades
```

```

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

cap = cv2.VideoCapture(0)

while 1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.2, 2)

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)

    gray = cv2.medianBlur(gray, 5)
    rows = gray.shape[0]
    circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, rows / 8,
                               param1=100, param2=30,
                               minRadius=1, maxRadius=30)
    font = cv2.FONT_HERSHEY_SIMPLEX

    if circles is not None:
        circles = np.uint16(np.around(circles))
        for i in circles[0, :]:
            center = (i[0], i[1])
            # circle center
            cv2.circle(img, center, 1, (0, 100, 100), 3)
            # circle outline
            radius = i[2]

```

```

        cv2.circle(img, center, radius, (255, 0, 255), 3)

        cv2.putText(img,'1',(200,200), font, 2,(0,255,255),2,cv2.LINE_AA)

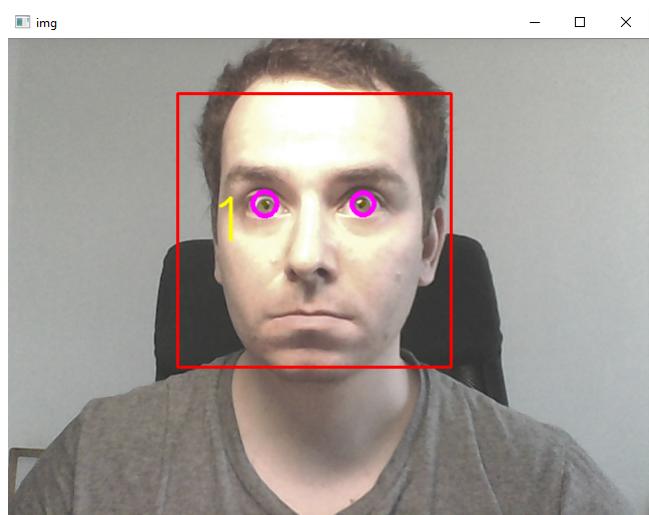
    else:
        cv2.putText(img,'0',(200,200), font, 2,(0,255,255),2,cv2.LINE_AA)
        break

cv2.imshow('img',img)
k = cv2.waitKey(30) & 0xff
if k == 27:
    break

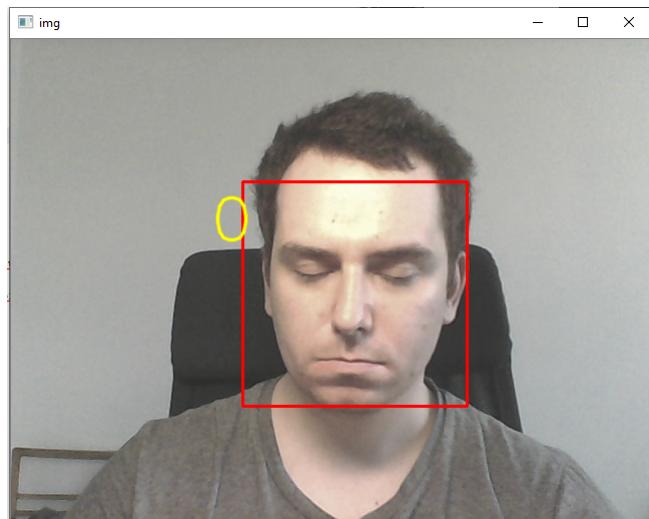
cap.release()
cv2.destroyAllWindows()

```

10.2 Wynik(wykres):



Rysunek 10: A Figure1.



Rysunek 11: A Figure1.

11 Treść zadania:

Napisz program obracający znalezioną twarz o 180° (góra na dół).

11.1 Opis metody wykonania (kod):

```
import numpy as np
import cv2

# multiple cascades: https://github.com/Itseez/opencv/tree/master/data/haarcascades

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

while 1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

```

for (x,y,w,h) in faces:

    crop = img[y:y+h, x:x+w]
    rotated=cv2.rotate(crop, cv2.ROTATE_180)

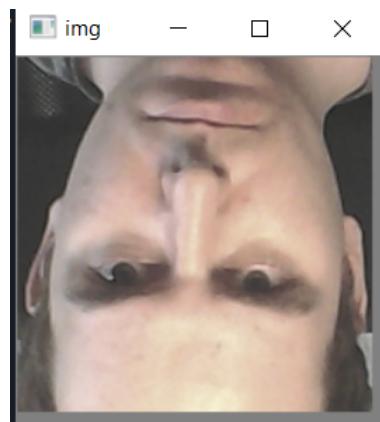
    cv2.rectangle(rotated,(x,y),(x+w,y+h),(0,0,255),2)

cv2.imshow('img',rotated)
k = cv2.waitKey(30) & 0xff
if k == 27:
    break

cap.release()
cv2.destroyAllWindows()

```

11.2 Wynik(wykres):



Rysunek 12: A Figure1.