МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Цифровая обработка бинарных изображений»

Отчет по лабораторной работе № 10

по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2023г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель работы: изучить основные операции геометрических преобразований изображений, такие как изменение размера, сдвиг, вращение, аффинное преобразование и т. д

Ход работы:

Задание 4.1. Изменить размер изображения

Задание 4.1.

Изменить размер изображения.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
def img_print(orig, res):
    pose = [121, 122]
    signature = ["Оригинал", "Измененное"]
    img = [orig, res]
    i = 0
    while i < 2:
        plt.subplot(pose[i])
        plt.title(signature[i])
        plt.imshow(img[i])
        i += 1
    return 0
```

```
image = cv2.imread('cat.jpg',0)
img = cv2.imread('cat.jpg',1)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Первый способ изменения размера задается в процентах

```
scale_percent = 50 # процент изменения
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)

resized = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
print('Resized Dimensions : ', resized.shape)
img_print(img, resized);
```

```
Resized Dimensions :  (533, 800, 3)
```

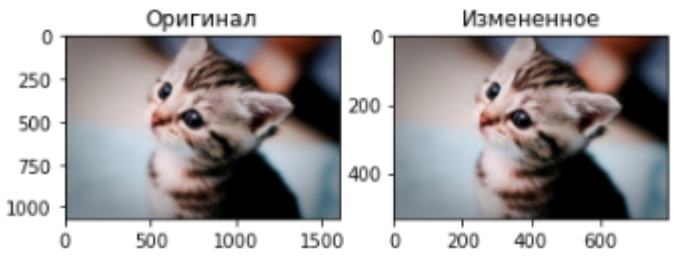


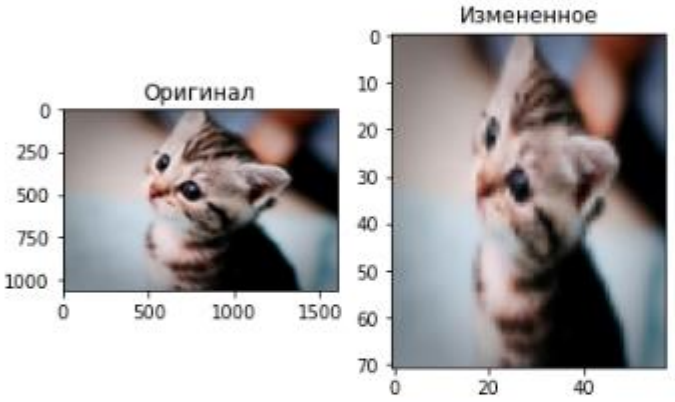
Рисунок 1 – Результат работы

Второй способ изменения размера задается вручную

```
print('Original Dimensions : ',img.shape)
width = 58
height = 71
dim1 = (width, height)

# resize image
resized1 = cv2.resize(img, dim1, interpolation=cv2.INTER_AREA)
print('Resized Dimensions : ', resized1.shape)
img_print(img, resized1);
```

```
Original Dimensions :  (1067, 1600, 3)
Resized Dimensions :  (71, 58, 3)
```



Третий способ: задается коэффициентом масштабирования

```
res = cv2.resize(img,None,fx=2, fy=2, interpolation = cv2.INTER_CUBIC)

#OR
#height, width = img.shape[:2]
#res = cv2.resize(img,(2*width, 2*height), interpolation = cv2.INTER_CUBIC)
img_print(img, res);
```

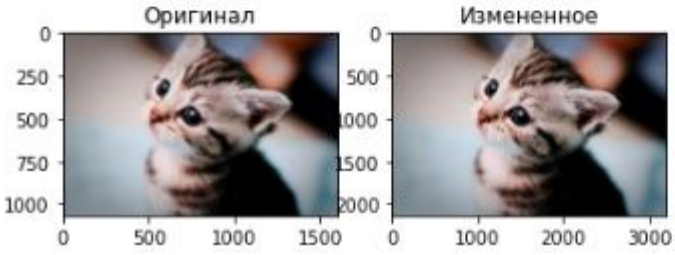


Рисунок 2 – Результат работы

Задание 4.2. Определить размер изображения и сдвинуть изображение на 100 столбцов и 50 строк.

Задание 4.2.

Определить размер изображения и сдвинуть изображение на 100 столбцов и 50 строк.

```
]: rows,cols,colors = img.shape
   M = np.float32([[1,0,100],[0,1,50]])
   dst = cv2.warpAffine(img,M,(cols,rows))
   plt.imshow(dst);
```

Рисунок 2 – Результат работы

Задание 4.3. Определить размер изображения, его центр и повернуть его на 90 градусов.

Задание 4.3.

Определить размер изображения, его центр и повернуть его на 90 градусов.

```
: M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
  dst = cv2.warpAffine(img,M,(cols,rows))
  plt.imshow(dst);
```

Рисунок 3 – Результат работы

Задание 4.4. Определить размер изображения, задать 3 точки, изменить их координаты и провести аффинное преобразование всего изображения по этим точкам

Задание 4.4.

Определить размер изображения, задать 3 точки, изменить их координаты и провести аффинное преобразование всего изображения по этим точкам

```
pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])

M = cv2.getAffineTransform(pts1,pts2)
dst = cv2.warpAffine(img,M,(cols,rows))

plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
```

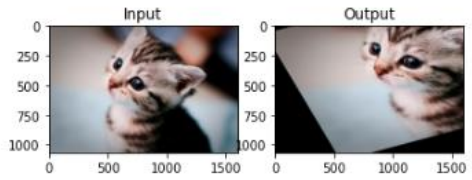


Рисунок 4 – Результат работы

Задание 4.5. Провести охват изображения в прямоугольник, повернутый так, чтобы площадь этого прямоугольника была минимальной. Перед выполнением заданий, используя вставку в Microsoft office «создать фигуры правильной формы», создайте фигуру бинарного изображения, скопируйте в Paint и сохраните в файле с расширением jpg или png. Созданное бинарное изображение должно быть инвертированным, т. е. эти фигуры должно быть темными, тогда прямоугольник, круг, эллипс, которые охватывают эти фигуры, будут хорошо видны.

Задание 4.5.

Провести охват изображения в прямоугольник, повернутый так, чтобы площадь этого прямоугольника была минимальной.
Перед выполнением заданий, используя вставку в Microsoft office «создать фигуры правильной формы», создайте фигуру бинарного изображения, скопируйте в Paint и сохраните в файле с расширением jpg или png. Созданное бинарное изображение должно быть инвертированным, т. е. эти фигуры должно быть темными, тогда прямоугольник, круг, эллипс, которые охватывают эти фигуры, будут хорошо видны.

```
image = cv2.imread('primer.png',0)
plt.axis('off')
plt.imshow(image);
```

```
ret, thresh = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
contours, hierarchy = cv2.findContours(thresh, 1, 1)

cnt = contours[0]
rect = cv2.minAreaRect(cnt)

box = cv2.boxPoints(rect)
box = np.int0(box)

imp = cv2.drawContours(image, [box], 0, (0, 0, 255), 2)
imp = cv2.cvtColor(imp, cv2.COLOR_BGR2RGB)

plt.axis('off')
plt.imshow(imp);
```

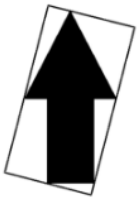


Рисунок 5 – Результат работы

Задание 4.6. Провести охват изображения в круг.

Задание 4.6.

Провести охват изображения в круг.

```
image = cv2.imread('primer.png',0)

(x,y),radius = cv2.minEnclosingCircle(cnt)
center = (int(x),int(y))
radius = int(radius)

imp = cv2.circle(image,center,radius,(0,255,0),2)
imp = cv2.cvtColor(imp, cv2.COLOR_BGR2RGB)

plt.axis('off')
plt.imshow(imp);
```

Рисунок 6 – Результат работы

Задание 4.7. Провести охват изображения в эллипс, повернутый так, чтобы площадь этого эллипса была минимальной.

Задание 4.7.

Провести охват изображения в эллипс, повернутый так, чтобы площадь этого эллипса была минимальной.

```
image = cv2.imread('primer.png',0)

ellipse = cv2.fitEllipse(cnt)
imag = cv2.ellipse(image,ellipse,(0,255,0),2)

imag = cv2.cvtColor(imag, cv2.COLOR_BGR2RGB)
plt.axis('off')
plt.imshow(imag);
```

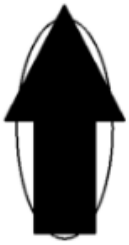


Рисунок 7 – Результат работы

Задание 4.8. Провести прямую линию вдоль оси симметрии изображения

Задание 4.8.

Провести прямую линию вдоль оси симметрии изображения

```
img = cv2.imread('primer.png',0)

ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
inv = cv2.bitwise_not(thresh)

contours, hierarchy = cv2.findContours(inv, 1, 1)
cnt = contours[0]
rows,cols = inv.shape[:2]

[vx,vy,x,y] = cv2.fitLine(cnt, cv2.DIST_L2,0,0.01,0.01)

lefty = int((-x*vy/vx) + y)
righty = int(((cols-x)*vy/vx)+y)

img = cv2.line(img,(cols-1,righty),(0,lefty),(0,255,0),2)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.axis('off')
plt.imshow(img);
```

Рисунок 8 – Результат работы

Задание 4.9. Нарисовать контур, охватывающий изображение, толщиной 2, вывести полученное изображение на экран.

Задание 4.9.

Нарисовать контур, охватывающий изображение, толщиной 2, вывести полученное изображение на экран.

```
image = cv2.imread('bin.jpg')
original_image = image
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50,200)
contours, hierarchy= cv2.findContours(edges.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
for cnt in contours:
    hull= cv2.convexHull(cnt)
    cv2.drawContours(image, [hull],0,(0,255,0),2)
plt.axis('off')
plt.imshow(image);
```

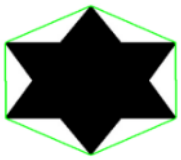


Рисунок 9 – Результат работы

Задание 4.10. Выполнить аппроксимацию контура, полагая epsilon = 1%, epsilon = 5% и epsilon = 10%.

Задание 4.10.

Выполнить аппроксимацию контура, полагая epsilon = 1%, epsilon = 5% и epsilon = 10%.

```
img = cv2.imread('bin2.jpg', 0)

ret, thresh = cv2.threshold(img, 0, 255, 0)
contours, hierarchy = cv2.findContours(thresh, 2, 3)

imag = cv2.imread('bin2.jpg')

f = plt.figure(figsize=(20,20))

plt.subplot(1,3,1)
plt.title('Original')
plt.imshow(img,'gray')

plt.subplot(1,3,2)
plt.title('Epsilon = 10%')
for i in range(np.size(contours)):
    cnt = contours [i]
    epsilon = 0.01 * cv2.arcLength(cnt, True)
    approx = cv2.approxPolyDP(cnt,epsilon,True)
    cv2.drawContours(imag,[approx],-1,(0,255,255),2)
plt.imshow(imag)

plt.subplot(1,3,3)
plt.title('Epsilon = 1%')
imAg = cv2.imread('bin2.jpg')
for i in range(np.size(contours)):
    cnt = contours [i]
    epsilon = 0.1 * cv2.arcLength(cnt, True)
    approx = cv2.approxPolyDP(cnt,epsilon,True)
    cv2.drawContours(imAg,[approx],-1,(0,255,255),2)
plt.imshow(imAg)
plt.show()
```

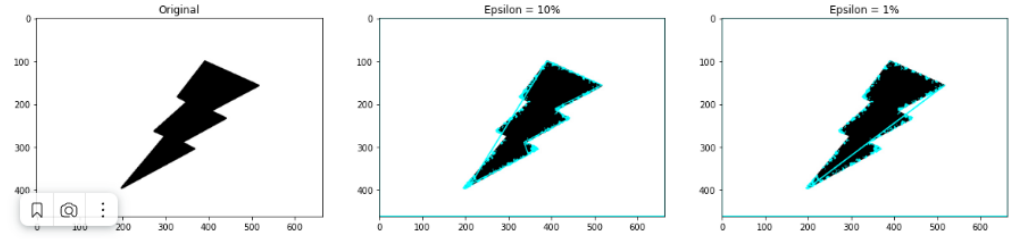


Рисунок 10 – Результат работы

Задание 4.11. Нарисовать прямоугольник в месте, где нужно вырезать фрагмент, вывести на экран фрагмент, ограниченный прямоугольником, увеличив этот фрагмент. Определить размер изображения, его центр и повернуть его на 90 градусов.

Задание 4.11.

Нарисовать прямоугольник в месте, где нужно вырезать фрагмент , вывести на экран фрагмент, ограниченный прямоугольником, увеличив этот фрагмент. Определить размер изображения, его центр и повернуть его на 90 градусов.

```
img = cv2.imread('avto.jpg', 1)
image = cv2.rectangle(img, (290, 470), (530, 550), (0, 0, 255), 2)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image)
```

```
200 100
```


```
crop = img[470:550, 290:530]
piece = cv2.resize(crop, (200,100),
interpolation=cv2.INTER_LINEAR)

(h, w) = piece.shape[:2]
print(w,h)

plt.imshow(piece);
```

```
200 100
```


```
center = (w / 2, h / 2)
M = cv2.getRotationMatrix2D(center, 90, 1)
rotated = cv2.warpAffine(piece, M, (150, 150))

plt.imshow(rotated)
```

```
<matplotlib.image.AxesImage at 0x15c4393f850>
```

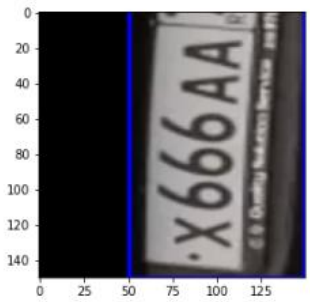


Рисунок 11 – Результат работы

Индивидуальное задание.

Считать цветное изобразение изображение и выполнитm:

1. Провести охват изображения в прямоугольник, повернутый так, чтобы площадь этого прямоугольника была минимальной
2. "Обрезать" изображение по минимальному квадрату
3. Привести изображение к размерау (200, 100)

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

Загрузим изображение и найдем его контур

```
img = cv2.imread('eye.jpg', 0)
ret,thresh = cv2.threshold(img,128,255, cv2.THRESH_BINARY)
contours, hierarchy = cv2.findContours(thresh, 5, 5)

cont = np.zeros_like(img)
cv2.drawContours(cont, contours, -1, 255, 1)

plt.subplot(121)
plt.title('contours')
plt.axis('off')
plt.imshow(cont, cmap='gray');

mask = np.zeros_like(img)
cv2.drawContours(mask, contours, -1, 255, -1)

plt.subplot(122)
plt.title('Original')
plt.axis('off')
plt.imshow(mask, cmap='gray');
```

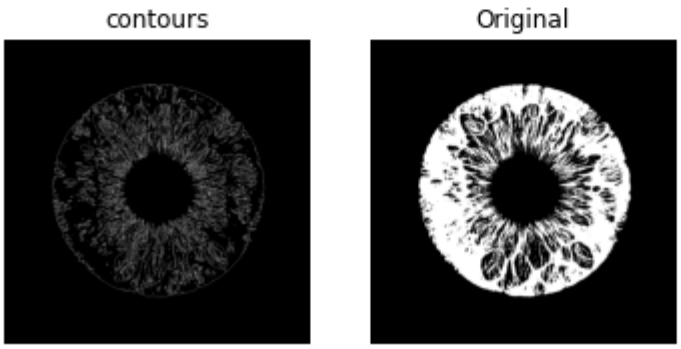


Рисунок 12 – Результат работы

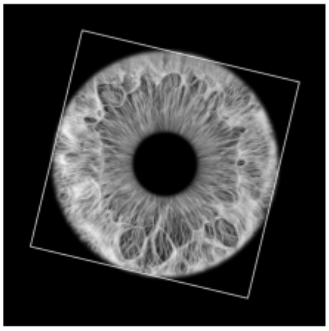
Проведем охват изображения в прямоугольник, повернутый так, чтобы площадь этого прямоугольника была минимальной.

```
cnt = contours[0]
rect = cv2.minAreaRect(cnt)

box = cv2.boxPoints(rect)
box = np.int0(box)

imp = cv2.drawContours(img, [box], 0,255, 3)

plt.axis('off')
plt.imshow(imp, cmap='gray');
```



Повернем изображение так, чтобы стороны прямоугольника были параллельны осям

```
rows,cols = img.shape
M = cv2.getRotationMatrix2D((cols/2,rows/2),13.5,1)
dst = cv2.warpAffine(img,M,(cols,rows))
plt.imshow(dst, cmap='gray');
```

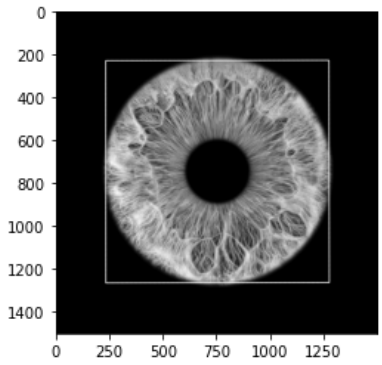


Рисунок 13 – Результат работы

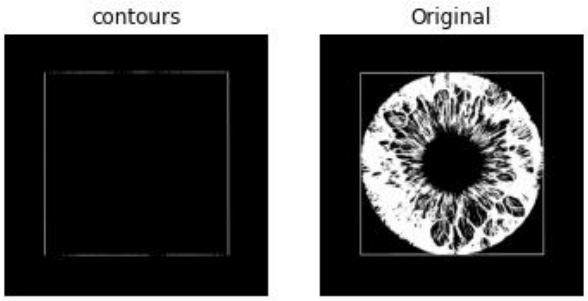
Заново найдем контур у уже объведенного изображения

```
ret,thresh = cv2.threshold(dst,128,255, cv2.THRESH_BINARY)
contours, hierarchy = cv2.findContours(thresh, 5, 5)

cont = np.zeros_like(img)
cv2.drawContours(cont, contours[0], -1, 255, 3)

plt.subplot(121)
plt.title('contours')
plt.axis('off')
plt.imshow(cont, cmap='gray');

plt.subplot(122)
plt.title('Original')
plt.axis('off')
plt.imshow(mask, cmap='gray');
```



"Обрезанем" изображение

```
out = np.zeros_like(dst)
out[mask == 255] = dst[mask == 255]

(y, x) = np.where(mask == 255)
min_y = np.min(contours[0],  axis=0)[0][1]
min_x = np.min(contours[0],  axis=0)[0][0]
max_y = np.max(contours[0],  axis=0)[0][1]
max_x = np.max(contours[0],  axis=0)[0][0]

out = out[min_y:max_y+1, min_x:max_x+1]

plt.axis('off')
plt.imshow(out, cmap='gray');
```

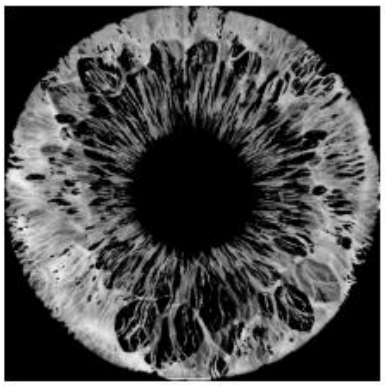


Рисунок 14 – Результат работы

И придаем определенный размер

```
piece = cv2.resize(out, (200,100), interpolation=cv2.INTER_LINEAR)
plt.imshow(piece, cmap='gray');
```

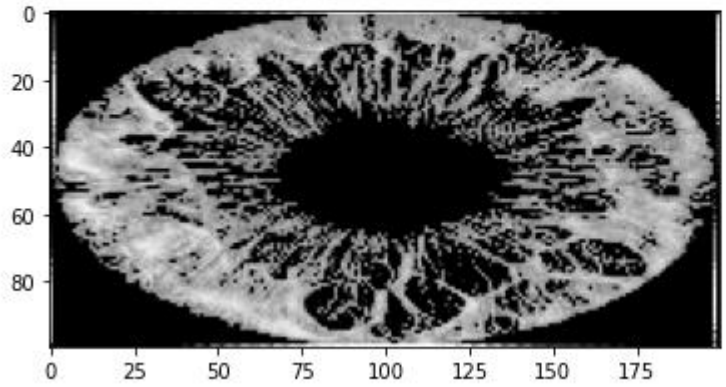


Рисунок 15 – Результат работы