МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет»

Кафедра инфокоммуникаций

Отчѐт по практическому занятию №3.10

«Цифровая обработка бинарных изображений»

по дисциплине «Теории распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1
Коновалова Виктория Николаевна « »
_____20_____г.
Подпись студента _____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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

Ход работы

Проработаны примеры из методических указаний.

Примеры лабораторной работы

Задание 4.1. Изменить размер изображения.

In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
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
```

In [3]:
```
image = cv2.imread('cat.jpeg',0)
img = cv2.imread('cat.jpeg',1)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Первый способ изменения размера задается в процентах

In [4]:
```
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)
```

Индивидуальное задание

Найти на черно-белом изображении наибольший по площади контур, который не является контуром всего изображения. Затем нужно выделить найденный контур на изображении и и вырезает изображение внутри этого контура.

---

1) Импортируем необходимые библиотеки

In [1]:
```
import cv2
import matplotlib.pyplot as plt
```

2)Загрузим изображение и преобразуем в оттенки серого

In [2]:
```
image = cv2.imread("circ.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

3)Применяем бинаризацию для выделения черных кругов. Пороговое значение равно 10.

In [3]:
```
ret, thresh = cv2.threshold(gray, 10, 255, cv2.THRESH_BINARY)
```

4)Найти все контуры на изображении

Находим контуры на изображении с помощью функции findContours. Здесь используется метод RETR_TREE для извлечения всех контуров и иерархии и метод CHAIN_APPROX_SIMPLE для упрощения полученных контуров.

In [4]:
```
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

5)Находим контур с наибольшей площадью в списке контуров с помощью цикла for. Здесь используется функция contourArea, чтобы определить площадь контура.

In [5]:
```
max_area = 0
max_contour = None
for contour in contours:
    area = cv2.contourArea(contour)
    x, y, w, h = cv2.boundingRect(contour)
    if area > max_area and not (x == 0 and y == 0 and w == image.shape[1] and h == image.shape[0]):
        max_area = area
        max_contour = contour
```

6) Нарисовать контур на изображении.

In [6]:
```
if max_contour is not None:
    cv2.drawContours(image, [max_contour], -1, (0, 255, 0), 3)

    # 7)Найти ограничивающий прямоугольник для контура и вырезать этот участок
    x, y, w, h = cv2.boundingRect(max_contour)
    cropped_image = image[y:y+h, x:x+w]
```

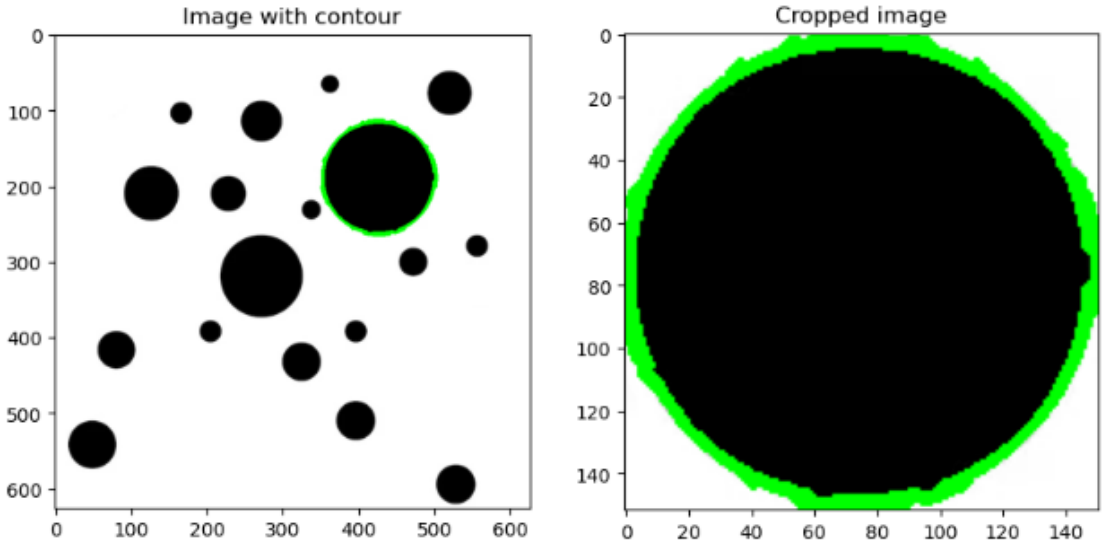
8)Создать фигуру и задать размер и отобразить изображение с контуром и вырезанную область

In [7]:
```
fig = plt.figure(figsize=(10, 5))

ax1 = fig.add_subplot(1, 2, 1)
ax1.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
ax1.set_title("Image with contour")

ax2 = fig.add_subplot(1, 2, 2)
ax2.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
ax2.set_title("Cropped image")

plt.show()
```



ВОПРОСЫ

1. Что такое аффинное преобразование?

Аффинное преобразование - это преобразование плоскости, которое сохраняет прямые и параллельность. Аффинное преобразование может выполнять повороты, масштабирование, сдвиги и отражения относительно прямых.

Отображение плоскости или пространства в себя, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся - в пересекающиеся, скрещивающиеся - в скрещивающиеся.

2. Как реализуется вращение изображения?

Поворот изображения на некоторый угол θ достигается с помощью матрицы $[^{cos\theta \quad sin\theta}$

$sin\theta \quad cos\theta$

У функции вращения cv.getRotationMatrix2D(,) первые два аргумента – координаты центра, третий аргумент – угол поворота.

3. Как реализуется смещение местоположения объекта?

Матрица смещения в плоскости (x, y) имеет вид:

$$M = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{pmatrix}$$

где (tx, ty) – вектор смещения. В программе сначала определяем количество столбцов (rows) – это ширина, затем количество строк (cols) – это высота. С помощью функции cv.warpAffine () изображение сдвигается в направлении (tx, ty).

4. Аффинная трансформация изображения. Нахождение матрицы преобразования.

При аффинном преобразовании все параллельные линии исходного изображения остаются параллельными и в выходном изображении. Чтобы найти матрицу преобразования, нам нужны три точки из входного изображения и их соответствующие местоположения в выходном изображении. Затем функция cv.getAffineTransform создаст матрицу 2x3, которая передается функции cv.warpAffine.

5. Функция cv.warpAffine, ее синтаксис.

Mat getAffineTransform

(InputArray src, // представляет три точки ввода

InputArray dst // представляет три точки вывода)

Эта функция в основном используется для создания матрицы аффинного преобразования.

- Произвольное аффинное преобразование можно выразить как Умножьте на матрицу (линейное преобразование), а затем добавьте вектор (перевод).

- Таким образом, мы можем использовать аффинное преобразование для выражения:
- Вращение (линейное преобразование)
- Перевод (добавление вектора)
- Операция масштабирования (линейное преобразование)
- Теперь вы можете знать, что на самом деле аффинное преобразование представляет собой отношение между двумя изображениями.
- Обычно мы используем 2 × 3Матрица для представления аффинного преобразования.

6. Охват объекта повернутым прямоугольником

Поворот прямоугольника, который ограничивает объект, позволяет минимизировать площадь этого прямоугольника. Функция cv2.drawContours() возвращает структуру box, которая содержит следующие аргументы: верхний левый угол (x, y), ширину, высоту, угол поворота. Чтобы нарисовать прямоугольник, нужны 4 угла прямоугольника, которые задаются функцией cv2.boxPoints ()

7. Заключение изображения в круг, эллипс с минимальной площадью.

Окружность с минимальной площадью, охватывающей объект, нарисуем с помощью функции cv2.minEnclosingCircle ().

Используя функцию cv2.ellipse(), можно вписать изображение в эллипс с минимальной площадью.

8. Аппроксимация контура

Аппроксимация контура - это процесс приближения кривой (контура) определенной формы (например, прямой линии или окружности) другой

кривой, состоящей из более простых форм (например, полигона). Этот процесс может быть полезен при обработке изображений для уменьшения количества точек, описывающих контур, без значительной потери информации. Это может ускорить вычисления и снизить объем памяти, необходимый для хранения данных.

Функция cv2.approxPolyDP(cnt,epsilon,True) позволяет аппроксимировать контур. Первый аргумент cnt = contours [i] – массив с координатами пикселей контура, аргумент epsilon задается в процентах, с уменьшением epsilon максимальное расстояние между ломаной прямой, аппроксимирующей контур, и самим контуром также уменьшается. Значение этого аргумента вычисляется функцией

epsilon = 0.1*cv2.arcLength(cnt,True).