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

Кафедра инфокоммуникаций

Отчёт по практическому занятию №3.10

«Цифровая обработка бинарных изображений»

по дисциплине «Теории распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1
Образцова М.Д. « »_____20__г.
Подпись студента _____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цифровая обработка бинарных изображений

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.

[3]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

[4]:
```
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
```

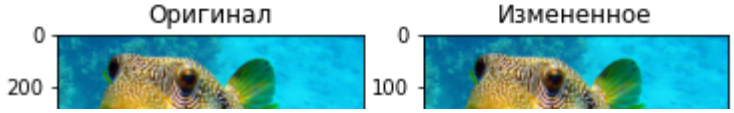
[5]:
```
image = cv2.imread('fish.jpg',0)
img = cv2.imread('fish.jpg',1)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Первый способ изменения размера задается в процентах

[6]:
```
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 :  (424, 600, 3)
```



Индивидуальное задание

Индивидуальная задача

На изображении, получаенной с камеры

Необходимо обрезать изображение до размеров знака, повернуть его на угол, равный наклону знака, улучшить качество изображения и обвести контур знака для его распознавания.

In [111…
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

In [126…
```
# загрузка изображения
imag = cv2.imread('sign.jpg')
imag = cv2.cvtColor(imag, cv2.COLOR_BGR2RGB)
```

In [143…
```
# обрезаем изображение до размеров знака
crop = imag[193:235, 995:1045]
piece = cv2.resize(crop, (50,50), interpolation=cv2.INTER_LINEAR)

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

In [165…
```
# определяем угол наклона знака
angle = 90

img = piece
```

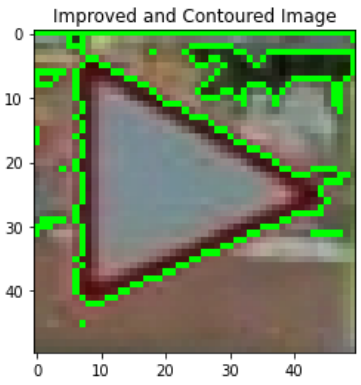
[169…
```
# переводим изображение в оттенки серого
gray = cv2.cvtColor(improved, cv2.COLOR_BGR2GRAY)

# бинаризуем изображение
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
```

[170…
```
# находим контуры
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# рисуем контуры на исходном изображении
cv2.drawContours(improved, contours, -1, (0, 255, 0), 1);
```

[171…
```
plt.title("Improved and Contoured Image")
plt.imshow(improved);
```



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. Что делает функция cv.resize?

Изменение размера изображения в OpenCV можно совершить функцией cv.resize (img, dim, interpolation=…).

3. Какие параметры принимает функция cv.resize?

Первый аргумент – матрица изображения, второй dim либо width, height – размер изображения, третий – метод интерполяции. Способы изменения размера следующие.

- Размер нового изображения указывается в процентах (например: 50%): scale_percent = 50.
- Размер изображения задается вручную: width=58, height=71.
- Размер изображения задается с помощью коэффициента масштабирования.

В процессе масштабирования используются разные методы интерполяции. Основные методы интерполяции таковы: cv.INTER_AREA – для сжатия, cv.INTER_CUBIC и cv.INTER_LINEAR – для масштабирования. По умолчанию используется метод интерполяции cv.INTER_LINEAR

4. Как реализуется вращение изображения?

Поворот изображения на некоторый угол θ достигается с помощью матрицы $M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$

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

5. 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).

6. 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.

7. 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.

8. 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 ()

9. 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.

10. 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).