Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Нахождение и обработка контуров»

ОТЧЕТ
по лабораторной работе №13
дисциплины
«Основы распознавания образов»

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Нахождение и обработка контуров

Цель работы: обнаружение и выделение контуров на изображении, анализ контуров. Изучение функций cv2.findContours().

Ход работы:

Задание 7.1

С помощью функции cv2.findContours найти все контуры изображения.

```
img = cv2.imread('images/jaguar.jpg')
image = cv2.imread('images/jaguar.jpg', 0)
image = cv2.medianBlur(image, 5)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.subplot(131), plt.imshow(img), plt.title('Original')
plt.axis('off')

thresh = cv2.adaptiveThreshold(
    image,
    255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY,
    11,
    2
)

plt.subplot(132), plt.imshow(thresh, 'gray'), plt.title('Thresh')
plt.axis('off')

contours, hierarchy = cv2.findContours(
    thresh.copy(),
    cv2.RETR_LIST,
    cv2.CHAIN_APPROX_SIMPLE
)

for cnt in contours:
    cv2.drawContours(img, cnt, -1, (0, 255, 0), 3)

plt.subplot(133), plt.imshow(img), plt.title('Contours')
plt.axis('off')
plt.show()
```

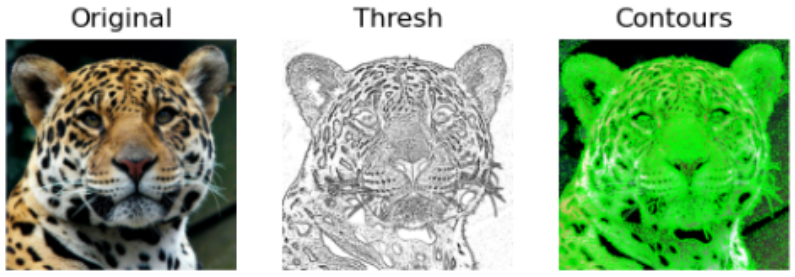

Рисунок 1 – Задания

Задание 7.2

Протестировать функцию поиска контура cv2.findContours с аргументом cv2.CHAIN_APPROX_SIMPLE, который экономит память.

```
img = cv2.imread('images/bmw.jpg')
image = cv2.imread('images/bmw.jpg', 0)
image = cv2.medianBlur(image, 5)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Превратим исходное изображение в бинарное инвертированое

```
thresh = cv2.adaptiveThreshold(
    image, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY_INV, 3, 10
)
```

Использование аргумента cv2.CHAIN_APPROX_SIMPLE в функции cv2.findContours экономит память

```
contours, hierarchy = cv2.findContours(
    thresh.copy(),
    cv2.RETR_LIST,
    cv2.CHAIN_APPROX_SIMPLE
)

for cnt in contours:
    cv2.drawContours(img, cnt, -1, (0, 255, 0), 3)

plt.subplot(121), plt.imshow(thresh), plt.title('Thresh')
plt.axis('off')
plt.subplot(122), plt.imshow(img), plt.title('Contours')
plt.axis('off')
plt.show();
```

Рисунок 2 – Задания

7.3 Выделить границу методом Канни.

Метод Канни обнаруживает границы связанной области изображения, выполняя поиск локальных максимумов градиента f(x, y). Градиент вычисляется после применения фильтра Гаусса. Метод использует два порога для нахождения сильных и слабых краев. Слабые края включаются в выход, если они связаны с сильными.

```
img = cv2.imread('images/Titanic.jpg', 0)
edges = cv2.Canny(img, 10, 200, apertureSize = 3)
```

2 аргумент функции cv2.Canny - минимальный градиент интенсивности; 3 аргумент функции cv2.Canny - максимальный градиент интенсивности

```
plt.imshow(edges)
plt.axis('off')
plt.show();
```

Рисунок 3 – Задания

Нахождение контуров изображения функцией cv2.findContours и функцией Кенни

```
image = cv2.imread('images/grand_staircase.jpg', 0)
img = cv2.imread('images/grand_staircase.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Функция cv2.adaptiveThreshold() определяет порог пикселя на основе небольшой области вокруг него. Это хорошо, если изображение имеет разные уровни освещения в разных областях. Таким образом, мы получаем разные пороги для разных областей одного и того же изображения, что дает лучшие результаты для изображений с разным освещением.

```
thresh = cv2.adaptiveThreshold(
    image,
    255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY,
    11,
    2
)
```

Функция cv2.findContours() возвращает контуры изображения и их иерархию.

```
contours, hierarchy = cv2.findContours(
    thresh.copy(),
    cv2.RETR_LIST,
    cv2.CHAIN_APPROX_SIMPLE
)
```

Нариусем контуры изображения с помощью функции cv2.drawContours()

```
for cnt in contours:
    cv2.drawContours(img, cnt, -1, (0, 255, 0), 1)
```

Рисунок 4 – Индивидуальное задание

Теперь воспользуемся функцией cv2.Canny(). Метод Кенни обнаруживает границы связанной области изображения, выполняя поиск локальных максимумов градиента f(x, y). Метод использует два порога для нахождения сильных и слабых краев. Слабые края включаются в выход, если они связаны с сильными.

```
img2 = cv2.imread('images/grand_staircase.jpg', 0)
edges = cv2.Canny(img2, 100, 300, apertureSize = 3)


plt.subplot(121), plt.imshow(img)
plt.axis('off')
plt.subplot(122), plt.imshow(edges)
plt.axis('off')
plt.show();
```

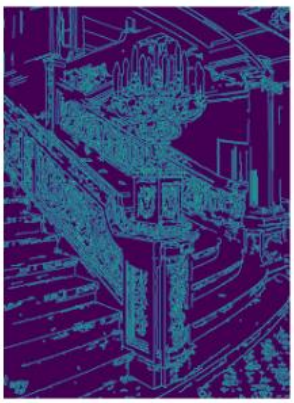

Рисунок 5 – Индивидуальное задание

Вывод: в ходе лабораторной работы изучены обнаружение и выделение контуров на изображении, анализ контуров. Изучена функция cv2.findContours(), cv2.drawContours ()

Ответы на контрольные вопросы:

1. Какие параметры выводит функция cv2.findContours()? Функция cv2.findContours() выводит массив изображения, массив контуров и иерархию контуров.

2. Каковы параметры функции cv2.drawContours()? Первым аргументом функции cv2.drawContours() является исходное изображение, вторым аргументом являются контуры, третьим аргументом является индекс контуров.

3. Каким образом можно нарисовать все контуры изображения? Чтобы нарисовать все контуры изображения, нужно использовать функцию img = cv2.drawContours(img, contours,-1, (0,255,0),3).

4. Как изменить параметры метода аппроксимации контура в функции cv2.findContours()? Для изменения параметров метода аппроксимации контура в функции cv2.findContours() нужно указать третий аргумент, который определяет метод аппроксимации. Доступны следующие методы cv2.CHAIN_APPROX_NONE, cv2.CHAIN_APPROX_SIMPLE, cv2.CHAIN_APPROX_TC89_L1, cv2.CHAIN_APPROX_TC89_KCOS.

5. Почему хранение всех координат пикселей контура не рационально? Хранение всех координат пикселей контура занимает много памяти, что неэффективно.

6. Каким образом можно сохранить только начальную и конечную координаты отрезка, если контур включает в себя прямые отрезки? Для сохранения только начальной и конечной координаты отрезка в контуре можно использовать метод аппроксимации cv2.CHAIN_APPROX_SI MPLE в функции cv2.findContours()

7. Что означает аргумент cv2.CHAIN_APPROX_SIMPLE в функции cv2.findContours()? Аргумент cv2.CHAIN_APPROX_SIMPLE в функции cv2.findContours() позволяет удалять все лишние точки в контуре и сжимать его, что экономит память.

8. Каким образом работает метод Канни? Метод Канни обнаруживает границы связанной области изображения, выполняя поиск локальных максимумов градиента f(x, y). Градиент вычисляется после применения фильтра Гаусса.

9. Какие аргументы принимает функция Канни cv2.Canny()? Функция Канни cv2.Canny(img, T_lower, T_upper,apertureSize = 3) содержит в скобках следующие аргументы: img – входное изображение, T_lower: нижний порог, T_upper: верхний порог, apertureSize – размер апертуры оператора Собеля.

10. Каковы этапы алгоритма Канни? Предварительное сглаживание по Гауссу для уменьшения шума, вычисление амплитуды и направления градиента изображения, обычно выбирается один из четырех возможных углов: 0º, 45°, 90° и 135°, не максимальное подавление: исключаются некраевые пиксели, это позволяет утончать края, использование порогового гистерезиса, для которого необходимы два порога: высокий и низкий порог. Если амплитуда градиента исследуемых пикселей больше, чем значения высокого порога, то эти пиксели резервируются как краевые пиксели. Если амплитуда положения пикселя меньше нижнего порога, то пиксель исключается. Если амплитуда интенсивности пикселя находится между двумя порогами, пиксель сохраняется только тогда, когда он подключен к пикселю выше верхнего порога.