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

Институт цифрового развития
Кафедра инфокоммуникаций

«Цифровая обработка бинарных изображений»

ОТЧЕТ
по лабораторной работе №11
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.

Пороговая обработка изображений

Цель работы: изучение алгоритмов порогового преобразования. Рассмотрение методов адаптивного определения порога, нахождение порогового значения Оцу. Изучение функций cv.threshold , cv.adaptiveThreshold.

Ход работы:

Задание 5.1

Для трех значений порога 70 + No, 140 + No, 210 + No, где No – номер по списку группы, провести пороговую обработку полутонового изображения с плавным изменением интенсивности.

```
img = cv2.imread ('images/grad.jpg' , 0)

ret, thresh1 = cv2.threshold (
    img,
    88,
    255,
    cv.THRESH_BINARY
)
ret, thresh2 = cv2.threshold (
    img,
    158,
    255,
    cv.THRESH_BINARY_INV
)
ret, thresh3 = cv2.threshold (
    img,
    228,
    255,
    cv.THRESH_TRUNC
)

images = [img, thresh1, thresh2, thresh3]

for i, item in enumerate(images):
    plt.subplot(2, 2, i+1), plt.imshow(item, 'gray')
    plt.axis('off')

plt.show()
```

Рисунок 1 – Задания

Задание 5.2

Протестировать функции с адаптивным порогом, задавая последовательно два значения порога, примерно 1/3 и 2/3 от максимума интенсивности.

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

ret1, th1 = cv2.threshold(
    img,
    127,
    255,
    cv2.THRESH_BINARY
)
th2 = cv2.adaptiveThreshold(
    img,
    255,
    cv2.ADAPTIVE_THRESH_MEAN_C,
    cv2.THRESH_BINARY,
    11,
    2
)
th3 = cv2.adaptiveThreshold(
    img,
    255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY,
    11,
    2
)
```

Рисунок 2 – Задания

```
titles = [
    'Original Image',
    'Global Thresholding (v = 127)',
    'Adaptive Mean Thresholding',
    'Adaptive Gaussian Thresholding'
]
images = [img, th1, th2, th3]

for i, item in enumerate(images):
    plt.subplot(2, 2, i+1), plt.imshow(item, 'gray')
    plt.title(titles[i])
    plt.axis('off')
```

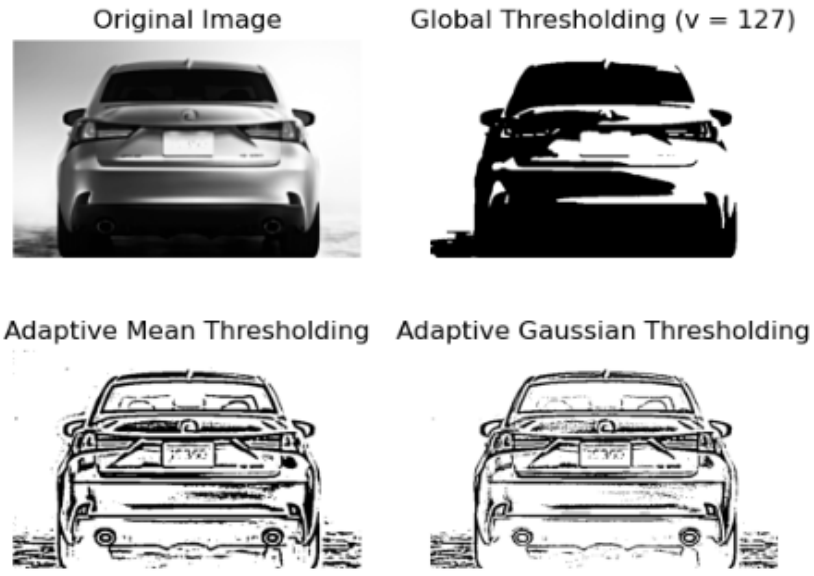


Рисунок 3 – Задания

Задание 5.3

Загрузить модули cv2, random, PIL. Создать зашумленное изображение.

```
image = Image.open('images/lexus.jpg')
draw = ImageDraw.Draw(image)

width = image.size[0]
height = image.size[1]
pix = image.load()

for i in range(width):
    for j in range(height):
        rand = random.randint(0, 200)
        a = pix[i, j][0] + rand
        b = pix[i, j][1] + rand
        c = pix[i, j][2] + rand
        if (a > 255):
            a = 255
        if (b > 255):
            b = 255
        if (c > 255):
            c = 255
        draw.point((i, j), (a, b, c))

image.save("images/median.png", "JPEG")

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

Рисунок 4 – Задания

Задание 5.4

На вход программы пороговой обработки подается зашумленное изображение. Это изображение обрабатывается тремя способами. В первом случае используется глобальный порог со значением 127. Во втором случае напрямую применяется порог Оцу. В третьем случае изображение сначала удаляет шум фильтром с гауссовым ядром 5x5, затем применяется пороговая обработка Оцу.

```
img = cv2.imread('images/median.png', 0)
```

Применим глобальную обработку и обработку Оцу.

```
ret1, th1 = cv2.threshold(
    img,
    127,
    255,
    cv2.THRESH_BINARY
)
ret2, th2 = cv2.threshold(
    img,
    0,
    255,
    cv2.THRESH_BINARY+cv2.THRESH_OTSU
)
```

Применим фильтр Гаусса и сделаем обработку Оцу после блюра.

Рисунок 5 – Задания

Применим фильтр Гаусса и сделаем обработку Оцу после блюра.

```
blur = cv2.GaussianBlur(img, (5, 5), 0)

ret3, th3 = cv2.threshold(
    blur,
    0,
    255,
    cv2.THRESH_BINARY+cv2.THRESH_OTSU
)

images = [img, 0, th1, img, 0, th2, blur, 0, th3]

titles = [
    'Original Noisy Image',
    'Histogram',
    'Global Thresholding (v=127)',
    'Original Noisy Image',
    'Histo-gram',
    'Otsu's Thresholding',
    'Gaussian filtered Image',
    'Histogram',
    'Otsu's Thresh-olding'
]

for i in range(3):
    plt.subplot(3, 3, i * 3 + 1), plt.imshow(images[i * 3], 'gray')
    plt.title(titles[i * 3]), plt.xticks([]),
    plt.yticks([])

    plt.subplot(3, 3, i * 3 + 2), plt.hist(images[i * 3].ravel(), 45)
    plt.title(titles[i * 3 + 1]), plt.xticks([]), plt.yticks([])

    plt.subplot(3, 3,i * 3 + 3), plt.imshow(images[i * 3 + 2], 'gray')
    plt.title(titles[i * 3 + 2]), plt.xticks([]),
    plt.yticks([])

plt.show()
```

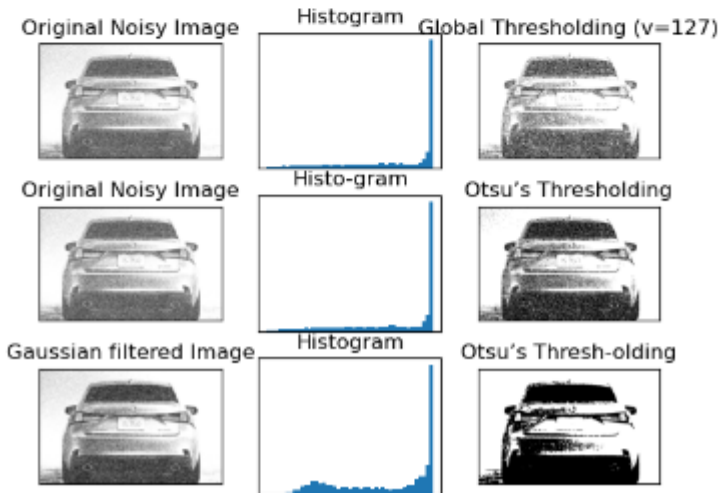


Рисунок 6 – Задания

Пороговая обработка Оцу на выделенном фрагменте фотографии

Если объект отличается по яркости от фона, то можно ввести порог, чтобы разделить изображение на светлый объект и темный фон. Метод Оцу для расчета порога использует гистограмму изображения. Гистограмма показывает, как часто встречается на данном изображении то или иное значение пикселя.

```
img = cv2.imread('images/bmw.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

image = cv2.rectangle(img, (155, 310), (240, 335), (0, 255, 0), 2)
plt.axis('off')
plt.title('Выделенный номер')
plt.imshow(image);
```

Выделенный номер

Сначала возьмём сам фрагмент

```
crop = img[315:331, 160:235]
piece = cv2.resize(
    crop,
    (200, 100),
    interpolation=cv2.INTER_LINEAR
)

cv2.imwrite ('images/o001oo.png', piece)

plt.axis('off')
plt.title('Выделенный номер')
plt.imshow(piece);
```

Выделенный номер

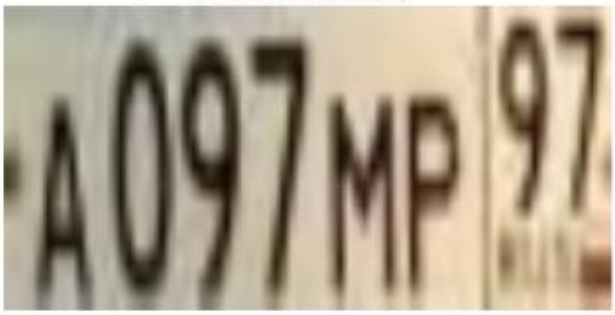


Рисунок 7 – Индивидуальное задание

Для эксперимента можно зашумить выделенный фрагмент и применить к этому изображению обработку Оцу на прямую и применить фильтр Гаусса + обработку Оцу

```
image = Image.open('images/o001oo.png')
draw = ImageDraw.Draw(image)

width = image.size[0]
height = image.size[1]
pix = image.load()

for i in range(width):
    for j in range(height):
        rand = random.randint(0, 50)
        a = pix[i, j][0] + rand
        b = pix[i, j][1] + rand
        c = pix[i, j][2] + rand
        if (a > 255):
            a = 255
        if (b > 255):
            b = 255
        if (c > 255):
            c = 255
        draw.point((i, j), (a, b, c))

image.save("images/numb_med.png", "JPEG")

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

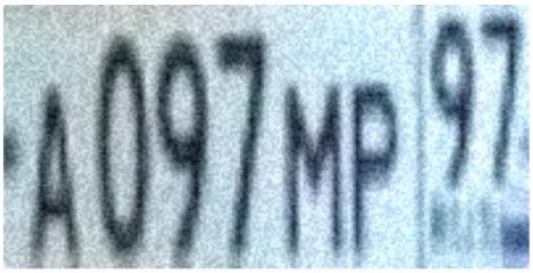


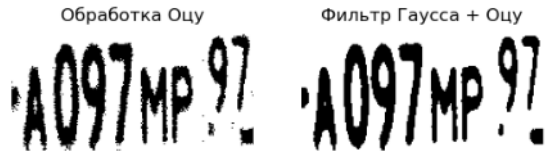
Рисунок 8 – Индивидуальное задание

```
img = cv2.imread('images/numb_med.png', 0)
blur = cv2.GaussianBlur(img, (5, 5), 0)


ret, th = cv2.threshold(
    img,
    0,
    255,
    cv2.THRESH_BINARY+cv2.THRESH_OTSU
)
plt.subplot(121),
plt.imshow(th, 'gray'),
plt.title('Обработка Оцу')
plt.axis("off")

ret2, th2 = cv2.threshold(
    blur,
    0,
    255,
    cv2.THRESH_BINARY+cv2.THRESH_OTSU
)
plt.subplot(122),
plt.imshow(th2, 'gray'),
plt.title('Фильтр Гаусса + Оцу')
plt.axis("off")

plt.show();
```



Как видно из результата, предварительное снижение шумов улучшает выходное изображение

Рисунок 12 – Индивидуальное задание

Вывод: в ходе лабораторной рабоыт изучены алгоритмы порогового преобразования. Рассмотрены методов адаптивного определения порога, нахождение порогового значения Оцу. Изучены функций cv.threshold , cv.adaptiveThreshold.

Ответы на контрольные вопросы:

1. Каким образом происходит получение бинарного изображения? Бинарное изображение получается путем сравнения интенсивности каждого пикселя с пороговым значением, удаления старого значения интенсивности и присвоения нового значения в зависимости от того, больше или меньше интенсивность пикселя порогового значения.

2. Что происходит с интенсивностью пикселя, если ее значение больше порогового значения? Если интенсивность пикселя больше порогового значения, то новое значение интенсивности будет равно 255, а при меньшей интенсивности порогового значения - новое значение интенсивности будет равно 0.

3. Для чего в функции cv.threshold(img,127, 255, cv.THRESH) используется третий и четвертый аргументы? Третий аргумент в функции cv.threshold(img,127, 255, cv.THRESH) задает значение интенсивности на выходе функции, когда значение пикселя больше порогового значения, а четвертый параметр задает используемый тип порогового значения.

4. Чем отличаются функции cv2.adaptiveThreshold с параметром cv2.ADAPTIVE_THRESH_MEAN_C эcv2.ADAPTIVE_THRESH_GAUSSIAN _C? Функция cv2.adaptiveThreshold с параметром cv2.ADAPTIVE_THRESH_MEAN_C берет в качестве порогового значения среднее арифметическое всех пикселей в окрестности выделенного пикселя, а функция cv2.adaptiveThreshold с параметром cv2.ADAPTIVE_THRESH_GAUSSIAN_C берет взвешенную сумму значений окрестностей, где весовые коэффициенты определяются с помощью функции Гаусса.

5. Какие аргументы принимают на вход функции cv2.adaptiveThreshold? На вход функции cv2.adaptiveThreshold необходимо подать исходное изображение, желаемое значение интенсивности на выходе, метод расчета порогового значения (cv2.ADAPTIVE_THRESH_MEAN_C или cv2.ADAPTIVE_THRESH_GAUSSIAN_C), размер окрестности и константу C, используемую для настройки порогового значения.

6. Для чего используется последний аргумент в функциях cv2.adaptiveThreshold и как он помогает настроить пороговое значение? Последний аргумент - константа C, вычитаемая из вычисленного среднего или взвешенного среднего. Это позволяет точно настроить пороговое значение и сделать его более или менее чувствительным к изменениям в значениях пикселей вокруг выделенного пикселя.

7. Что такое порог и как он помогает разделить изображение на объект и фон? Порог - это яркостное значение, которое используется для разделения изображения на объект и фон. Объект - это множество пикселей с яркостью выше порога, а фон - множество остальных пикселей с яркостью ниже порога.

8. Как метод Оцу помогает вычислить оптимальное пороговое значение? Метод Оцу использует гистограмму изображения, которая показывает, сколько пикселей имеют определенную яркость. Если гистограмма имеет два пика, то это означает наличие двух различных классов пикселей: полезные и фоновые. Метод Оцу вычисляет оптимальное пороговое значение, которое разделяет эти два класса.

9. Что означает флаг cv.THRESH_OTSU в функции cv2.threshold и зачем он нужен? Флаг cv.THRESH_OTSU в функции cv2.threshold указывает на использование метода Оцу для вычисления оптимального порогового значения. Этот флаг позволяет автоматически выбрать наилучшее пороговое значение для разделения объекта и фона.

10. Какими способами можно использовать порог для обработки изображений? Порог можно использовать для бинаризации изображения,

выделения объектов, удаления шума и т.д., например, функция cv2.threshold может использоваться вместе с пороговыми флагами для различных методов бинаризации. Также пороговые методы могут быть использованы вместе с другими методами обработки изображений, такими как морфологические операции.