

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 3.11
по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Трушева В. О. .«__»_____2023г.

Подпись студента_____

Работа защищена « __ »_____20__г.

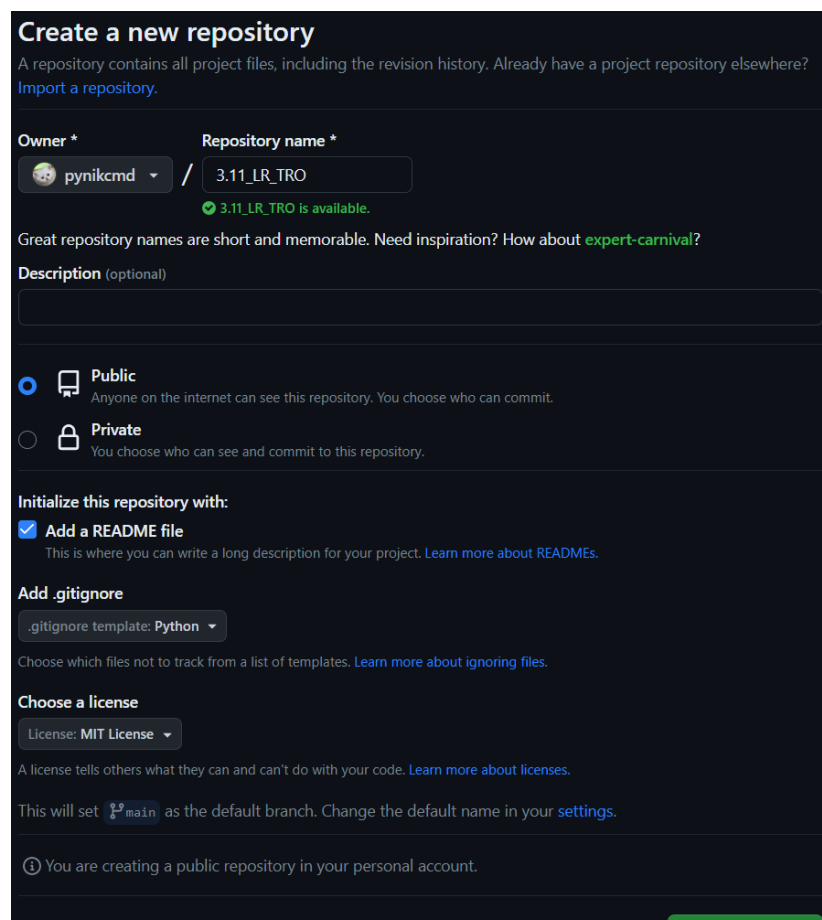
Проверила Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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

Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).



The screenshot shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main input fields: 'Owner' (set to 'pynikcmd') and 'Repository name' (set to '3.11_LR_TRO'). A green checkmark indicates that the repository name is available. There is a text box for 'Description (optional)'. Below the description, there are two radio buttons for 'Public' (selected) and 'Private'. Underneath, there is a section 'Initialize this repository with:' with a checked box for 'Add a README file'. Below that, there is a section 'Add .gitignore' with a dropdown menu set to 'Python'. Further down, there is a section 'Choose a license' with a dropdown menu set to 'MIT License'. At the bottom, there is a note that says 'This will set main as the default branch. Change the default name in your settings.' and a final note that says 'You are creating a public repository in your personal account.'

3. Выполните клонирование созданного репозитория на рабочий компьютер.

```
D:\fgit>git clone https://github.com/pynikcmd/3.11_LR_TRO.git
Cloning into '3.11_LR_TRO'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
D:\fgit\3.11_LR_TRO>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/fgit/3.11_LR_TRO/.git/hooks]
```

5. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.

6. Проработать примеры лабораторной работы в отдельном ноутбуке.

Задание 5.1

Для трех значений порога $70 + N_0$, $140 + N_0$, $210 + N_0$, где N_0 – номер по списку группы (21), провести пороговую обработку полутонового изображения с плавным изменением интенсивности.

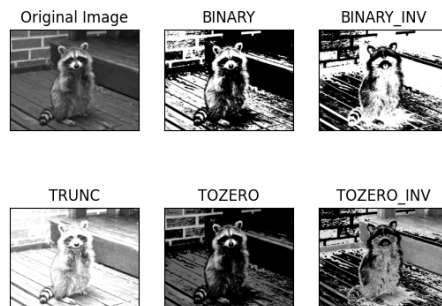
```
In [1]: import cv2
import numpy as np
from matplotlib import pyplot as plt

In [2]: img = cv2.imread('img.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

In [3]: ret, thresh1 = cv2.threshold(img, 91, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 91, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 91, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 91, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 91, 255, cv2.THRESH_TOZERO_INV)

In [7]: title = ['Original Image', 'BINARY', 'BINARY_INV', 'TRUNC', 'TOZERO', 'TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i], 'gray')
    plt.title(title[i])
    plt.xticks([], plt.yticks([]))
plt.show();
```



Пример 1

Задание 5.2.

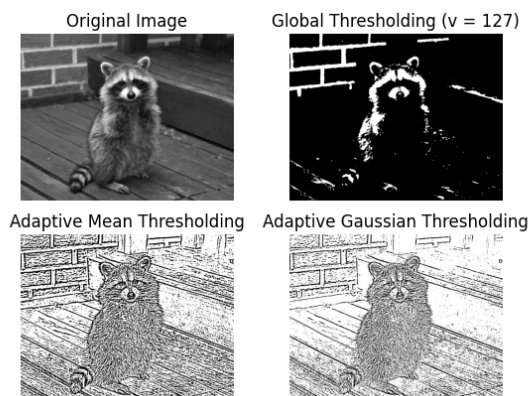
Протестировать функции с адаптивным порогом, задавая последовательно два значения порога, примерно 1/3 и 2/3 от максимума интенсивности. Проанализировать результат пороговой обработки изображения.

```
6]: img = cv2.imread('img.jpg',0)
img = cv2.medianBlur(img,5)

7]: 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)

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

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i], 'gray')
    plt.title(title[i])
    plt.axis('off')
plt.show();
```



Пример 2

Задание 5.3

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

```
: import random
from PIL import Image, ImageDraw

: image = Image.open('img.jpg')
draw = ImageDraw.Draw(image) # Создаем инструмент для рисования

: width = image.size[0] # Определяем ширину
height = image.size[1] # Определяем высоту
pix = image.load() # Выгружаем значения пикселей

: 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("median.png", "JPEG") # сохранить изображение

imag = cv2.imread('img.jpg')
imag = cv2.cvtColor(imag, cv2.COLOR_BGR2RGB)
img = cv2.imread('median.png')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.subplot(121),plt.imshow(imag),plt.title('Input')
plt.axis('off')
plt.subplot(122),plt.imshow(img),plt.title('Output')
plt.axis('off')
plt.show();
```

Input



Output



Пример 3

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. Сделать анализ того, как фильтрация шума улучшает результат.

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

```
ret1,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY) # Глобальная обработка
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU) # Обработка Otsu
blur = cv2.GaussianBlur(img,(5,5),0)
ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU) # Обработка Otsu's после фильтра Гаусса
```

```
images = [img, 0, th1, img, 0, th2, blur, 0, th3]
titles = ["Original Noisy Image","Histogram","GlobalThresholding (v=127)","Original Noisy Image", "Histogram",
         "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.axis('off')
    plt.subplot(3,3,i*3+2), plt.hist(images[i*3].ravel(),256)
    plt.title(titles[i*3+1])
    plt.axis('off')
    plt.subplot(3,3,i*3+3), plt.imshow(images[i*3+2], "gray")
    plt.title(titles[i*3+2])
    plt.axis('off')
    plt.show()
```

Original Noisy Image Histogram GlobalThresholding (v=127)



Original Noisy Image Histogram Otsu's Thresholding



Gaussian filtered Image Histogram Otsu's Thresh-olding



Пример 4

7. Создать ноутбук, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.).

Простой пороговый метод: выделить текст на изображении

```
: import cv2
: import numpy as np
: from matplotlib import pyplot as plt

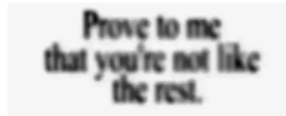
: image = cv2.imread('Img\text.jpg', 0)

: # Применение порогового метода Otsu для определения оптимального порогового значения
: ret, threshold = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

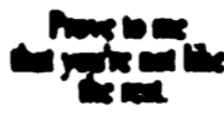
: plt.subplot(1,2,1),plt.imshow(image, 'gray')
: plt.title('Original Image'),plt.axis('off')
: plt.subplot(1,2,2),plt.imshow(threshold, 'gray')
: plt.title('Thresholded Image'), plt.axis('off')

plt.show();
```

Original Image



Thresholded Image



Применение адаптивного метода для улучшения качества текста

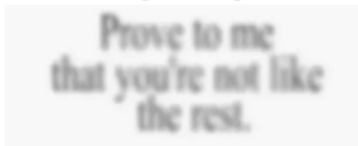
```
# Применение адаптивного метода
adaptive_threshold = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)

# Отображение результатов
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.axis('off')

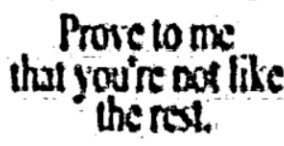
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(adaptive_threshold, cv2.COLOR_BGR2RGB))
plt.title('Adaptive Threshold Image')
plt.axis('off')

plt.tight_layout()
plt.show();
```

Original Image



Adaptive Threshold Image



Применить глобальный пороговый метод и адаптивный пороговый метод к изображению

```
image2 = cv2.imread('Img\face2.jpg', 0)

# Применение глобального порогового метода
ret, global_threshold = cv2.threshold(image2, 200, 255, cv2.THRESH_BINARY)

# Применение адаптивного порогового метода
adaptive_threshold = cv2.adaptiveThreshold(image2, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)

plt.figure(figsize=(10, 10))
plt.subplot(1,3,1),plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
plt.title('Original Image'),plt.axis('off')
plt.subplot(1,3,2),plt.imshow(global_threshold, 'gray')
plt.title('Global Thresholding'), plt.axis('off')
plt.subplot(1,3,3),plt.imshow(adaptive_threshold, 'gray')
plt.title('Adaptive Thresholding'), plt.axis('off')

plt.show();
```



8. Зафиксируйте сделанные изменения в репозитории.
9. Выполните слияние ветки для разработки с веткой main(master).
10. Отправьте сделанные изменения на сервер GitHub.

Контрольные вопросы

1. Тип порогового значения cv.THRESH_BINARY.

Для формирования на выходе бинарного изображения:

$$b(x, y) = \begin{cases} 255, & \text{если } f(x, y) > p; \\ 0, & \text{если } f(x, y) < p. \end{cases}$$

2. Тип порогового значения cv.THRESH_BINARY_INV.

Для формирования на выходе инвертированного бинарного изображения:

$$b(x,y) = \begin{cases} 0, & \text{если } f(x,y) > p; \\ 255, & \text{если } f(x,y) < p. \end{cases}$$

3. Тип порогового значения `cv.THRESH_TRUNC`.

На выходе:

$$b(x,y) = \begin{cases} threshold, & \text{если } f(x,y) > p; \\ src(x,y), & \text{если } f(x,y) < p. \end{cases}$$

4. Тип порогового значения `cv.THRESH_TOZERO`.

На выходе:

$$b(x,y) = \begin{cases} src(x,y), & \text{если } f(x,y) > p; \\ 0, & \text{если } f(x,y) < p. \end{cases}$$

5. Тип порогового значения `cv.THRESH_TOZERO_INV`.

На выходе:

$$b(x,y) = \begin{cases} 0, & \text{если } f(x,y) > p; \\ src(x,y), & \text{если } f(x,y) < p. \end{cases}$$

6. Что такое адаптивный пороговый метод?

Адаптивный пороговый метод – это метод бинаризации изображений, при котором пороговое значение рассчитывается и применяется независимо для каждого пикселя на основе локальных характеристик окрестности пикселя. В отличие от глобального порогового метода, где одно пороговое значение применяется ко всем пикселям изображения, адаптивный метод позволяет более гибко учитывать изменения яркости и контрастности в различных областях изображения.

7. Что такое простой пороговый метод?

В процессе пороговой обработки изображения все значения интенсивности пикселей по очереди сравниваются с пороговым значением. Старое значение интенсивности пикселя удаляется, и в зависимости от того, больше или меньше интенсивность пикселя порогового значения, этому пикселю присваивается новое значение интенсивности. Если интенсивность пикселя больше порогового значения, то новое значение интенсивности будет равно 255, если интенсивность пикселя меньше порогового значения, то новое значение интенсивности будет равно 0. В результате получим бинарное изображение.

8. Что такое бинаризация Оцу?

Если объект отличается по яркости от фона, то можно ввести порог, чтобы разделить изображение на светлый объект и темный фон. Объект – это множество пикселей, яркость которых превышает порог $I > p$, а фон – множество остальных пикселей, яркость которых ниже порога $I < p$. Метод Оцу для расчета порога использует гистограмму изображения. Гистограмма показывает, как часто встречается на данном изображении то или иное значение пикселя. Зная яркость каждого пикселя, подсчитаем сколько пикселей имеют такую яркость.

Применение бинаризации Оцу может быть полезным в различных задачах обработки изображений, таких как выделение объектов, обнаружение контуров, сегментация и других приложениях, где необходимо разделить изображение на фон и объекты.

9. Для чего модуль PIL?

PIL является библиотекой Python для работы с изображениями. Она предоставляет различные функции и классы для обработки, создания, изменения и сохранения изображений.

10. Что такой глобальный пороговый метод?

Глобальный пороговый метод — это метод бинаризации изображений, при котором одно пороговое значение применяется ко всем пикселям изображения. Этот метод основывается на глобальном анализе гистограммы яркости изображения и позволяет разделить пиксели на два класса: фон и объекты.