

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

## Кафедра инфокоммуникаций

## Отчет по лабораторной работе № 3.12

**по дисциплине «Технологии распознавания образов»**

Выполнил студент группы ПИЖ-б-о-21-1

Трушева В. О. .«\_\_»\_\_\_\_2023г.

Подпись студента \_\_\_\_\_

Работа защищена «    »                      20   г.

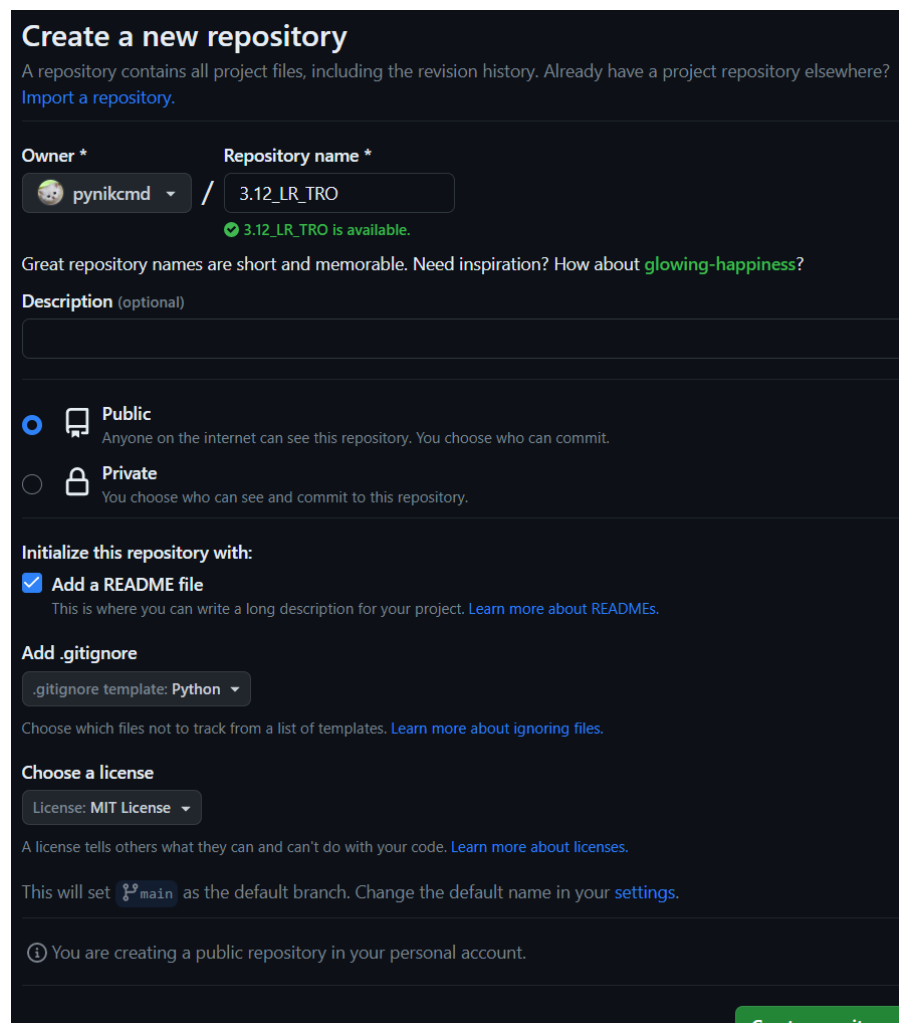
Проверила Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2023

Цель работы: сглаживание изображений с помощью различных фильтров нижних частот. Усвоение навыков применения 2D-свертки к изображениям.

## 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).



**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


**Owner \*** **Repository name \***


 pynikcmd / 3.12\_LR\_TRO

✔ 3.12\_LR\_TRO is available.

Great repository names are short and memorable. Need inspiration? How about [glowing-happiness?](#)

**Description** (optional)

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**

☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

**Add .gitignore**


.gitignore template: **Python**


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

**Choose a license**

License: **MIT License**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

[Create repository](#)

3. Выполните клонирование созданного репозитория на рабочий компьютер.

```
D:\fgit>git clone https://github.com/pynikcmd/3.12_LR_TR0.git
Cloning into '3.12_LR_TR0'...
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 соответствии с моделью ветвления git-flow.

```
D:\fgit\3.12_LR_TR0>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/3.12_LR_TR0/.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.

```

: # Создаем 3 возможных цвета - красный, зеленый и синий
red, green, blue = (255, 0, 0), (0, 255, 0), (0, 0, 255)
rgb = [red, green, blue] # Помещаем их в кортеж

: # Создаем функцию с параметрами (<наше изображение>, <вероятность зашумления>)
def sp_noise(image, prob):
    # Создаем массив нулей такого же размера и формата как исходное изображение
    output = np.zeros(image.shape, np.uint8)
    thres = 1- prob # Задаем порог
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rnd = random.random()
            if rnd > thres:
                # Задаем пикселю случайное значение из кортежа
                output[i][j] = random.choice(rgb)
            else:
                # Иначе оставляем пиксель без изменения
                output[i][j] = image[i][j]
    return output

: image = cv2.imread('img/cat.jpg')
image = cv2.resize(image, (900, 600))

: # Применяем к нашему изображению image, созданную функцию sp_noise,
# где 0.3 - вероятность зашумления пикселя
noise_img = sp_noise(image, 0.3)

# Объединяем оригинальное и зашумленное изображения
# в одно окно (для наглядности)
res = np.hstack((image, noise_img))

: plt.figure(figsize=(10, 10))
plt.axis('off')
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB));

```



Пример 1

### Задание 6.2.

Провести сглаживание изображения с помощью функции `cv2.filter2D()`, используя ядро  $5 \times 5$ .

```
img = cv2.imread('img/cat.jpg')

kernel = np.ones((5, 5), np.float32) / 25
dst = cv2.filter2D(img, -1, kernel)

plt.figure(figsize=(10, 10))
plt.subplot(121)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original')
plt.axis('off')
plt.subplot(122)
plt.imshow(cv2.cvtColor(dst, cv2.COLOR_BGR2RGB))
plt.title('Averaging')
plt.axis('off')
plt.show();
```

Original



Averaging



Пример 2

### Задание 6.3.

Провести усреднение изображения с помощью функции `cv2.blur()`, используя ядро  $5 \times 5$ .

```
: img = cv2.imread('img/cat.jpg')

: blur = cv2.blur(img, (50, 20))

: plt.figure(figsize=(10, 10))
: plt.subplot(121)
: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
: plt.title('Original')
: plt.axis('off')
: plt.subplot(122)
: plt.imshow(cv2.cvtColor(blur, cv2.COLOR_BGR2RGB))
: plt.title('Blurred')
: plt.axis('off')
: plt.show();
```

Original



Blurred



Пример 3

#### Задание 6.4.

Добавить к исходному изображению 20–30% шума. Провести фильтрацию изображения по Гауссу, используя ядро  $10 \times 10$ .

```
img = cv2.imread('img/cat.jpg')

blur = cv2.GaussianBlur(img, (5, 5), 20)

plt.figure(figsize=(10, 10))
plt.subplot(121)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original')
plt.axis('off')
plt.xticks([], plt.yticks([]))
plt.subplot(122)
plt.imshow(cv2.cvtColor(blur, cv2.COLOR_BGR2RGB))
plt.title('Blurred')
plt.axis('off')
plt.show();
```

Original



Blurred



Пример 4

#### Задание 6.5.

Добавить к исходному изображению 20–50% шума. Провести медианную фильтрацию изображения, используя ядро  $5 \times 5$ .

```
: img = cv2.imread('img/face.png')

: median = cv2.medianBlur(img,13)

: plt.figure(figsize=(10, 10))
: plt.subplot(121),plt.imshow(img), plt.title('Original')
: plt.axis('off')
: plt.subplot(122),plt.imshow(median), plt.title('Blurred')
: plt.axis('off')
: plt.show();
```

Original



Blurred



Пример 5

### Задание 6.6.

Создать файл с изображением, в котором обязательно присутствуют вертикальные и горизонтальные линии. С помощью оператора Собеля обнаружить и выделить эти линии.

```
img = cv2.imread('img/build.jpeg', 0)
img = cv2.resize(img, (900, 600))
```

```
# Функция Собеля для вычисления вертикальных линий
sobel_vertical = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)

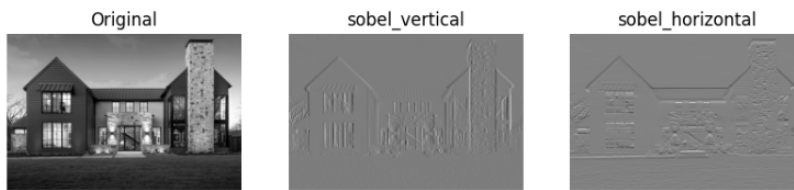
# То же самое, но 0, 1 означает, что теперь берем производную по y
sobel_horizontal = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)
```

```
plt.figure(figsize=(10, 10))
plt.subplot(131), plt.imshow(img, cmap = 'gray'), plt.title('Original')
plt.axis('off')

plt.subplot(132), plt.imshow(sobel_vertical, cmap = 'gray'), plt.title('sobel_vertical')
plt.axis('off')

plt.subplot(133), plt.imshow(sobel_horizontal, cmap = 'gray'), plt.title('sobel_horizontal')
plt.axis('off')

plt.show();
```



Пример 6

### Задание 6.7.

Сравнить оба способа для горизонтального фильтра Собеля с преобразованием в `cv2.CV_8U` и без него.

```
img = cv2.imread('img/build.jpeg', 0)
```

```
# Output dtype = cv2.CV_8U
sobelx8u = cv2.Sobel(img, cv2.CV_8U, 1, 0, ksize=5)

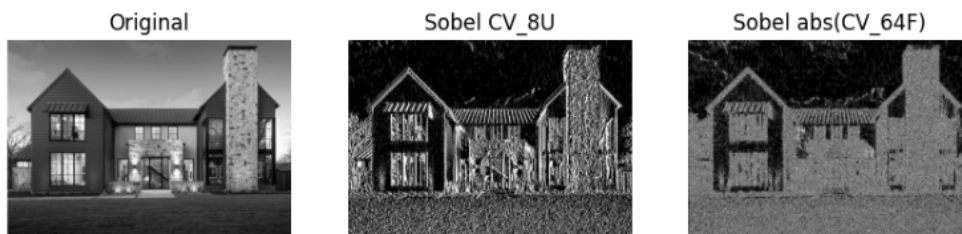
# Output dtype = cv2.CV_64F.
sobelx64f = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
abs_sobel64f = np.absolute(sobelx64f)
sobel_8u = np.uint8(abs_sobel64f)
```

```
plt.figure(figsize=(10, 10))
plt.subplot(131), plt.imshow(img, cmap = 'gray'), plt.title('Original')
plt.axis('off')

plt.subplot(132), plt.imshow(sobelx8u, cmap = 'gray'), plt.title('Sobel CV_8U')
plt.axis('off')

plt.subplot(133), plt.imshow(sobel_8u, cmap = 'gray'), plt.title('Sobel abs(CV_64F)')
plt.axis('off')

plt.show();
```



Пример 7

### Задание 6.8.

Создать файл с изображением, который обязательно содержит вертикальные и горизонтальные линии. С помощью оператора Превитта обнаружить и выделить эти линии.

```
] : img = cv2.imread('img/build.jpeg', 0)
img = cv2.resize(img, (900, 600))

] : xkernel = np.array([[ -1, -1, -1], [ 0, 0, 0], [ 1, 1, 1]]) # Создает ядро (маску) для x
ykernel = np.array([[ -1, 0, 1], [-1, 0, 1], [-1, 0, 1]]) # Создает ядро (маску) для Y

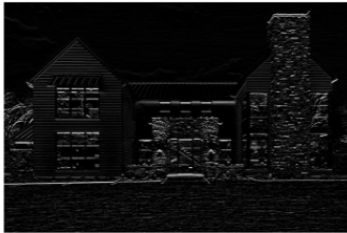
] : # Функция соединения изображения с ядром
img_prewittx = cv2.filter2D(img, -1, xkernel)
img_prewitty = cv2.filter2D(img, -1, ykernel)

] : plt.figure(figsize=(10, 10))
plt.subplot(121),plt.imshow(img_prewittx, cmap = 'gray'), plt.title('img_prewittx')
plt.axis('off')

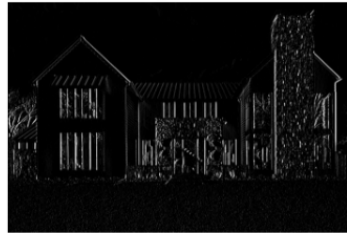
plt.subplot(122),plt.imshow(img_prewitty, cmap = 'gray'), plt.title('img_prewitty')
plt.axis('off')

plt.show();
```

img\_prewittx



img\_prewitty



Пример 8



### Задание 6.9.

Используя оператор Робертса, выделить линии на изображении.

```
img = cv2.imread('img/build.jpeg', 0)
img = cv2.resize(img, (900, 600))

kernel1 = np.array([[1, 0], [0, 1]])
kernel2 = np.array([[0, 1], [0, 1]])

img_robx = cv2.filter2D(img, -1, kernel1)
img_roby = cv2.filter2D(img, -1, kernel2)

output_image = img_robx + img_roby

plt.imshow(output_image, cmap = 'gray'), plt.title('output_image')
plt.axis('off')
plt.show();
```

output\_image



Пример 9

### Задание 6.10.

Создать файл с изображением, в котором присутствуют перепады изображения. С помощью оператора Лапласа обнаружить и выделить эти перепады.

```
: laplacian = cv2.Laplacian(img, cv2.CV_64F)

: plt.figure(figsize=(10, 10))
: plt.subplot(121),plt.imshow(img, cmap = 'gray'), plt.title('Original')
: plt.axis('off')

: plt.subplot(122),plt.imshow(laplacian, cmap = 'gray'), plt.title('Laplacian')
: plt.axis('off')

: plt.show();
```



Пример 10

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

Условие.

Создать файл с изображением, который обязательно содержит вертикальные и горизонтальные линии. С помощью оператора Превитта обнаружить и выделить эти линии. Создать комбинированный оператор Превитта, который объединяет обнаружение горизонтальных и вертикальных линий. Для этого можно использовать формулу:  $combined = \sqrt{img_{prewittx}^2 + img_{prewitty}^2}$

## Индивидуальное задание

Создать файл с изображением, который обязательно содержит вертикальные и горизонтальные линии. С помощью оператора Превитта обнаружить и выделить эти линии. Создать комбинированный оператор Превитта, который объединяет обнаружение горизонтальных и вертикальных линий. Для этого можно использовать формулу:  $combined = \sqrt{img\_prewittx^2 + img\_prewitty^2}$

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

```
9]: image = cv2.imread('img/build.jpeg', 0)
```

Оператора Превитта, который обнаруживает и выделяет линии

```
0]: xkernel = np.array([[ -1, -1, -1], [ 0, 0, 0], [ 1, 1, 1]]) # Создаем ядро (маску) для x
ykernel = np.array([[ -1, 0, 1], [-1, 0, 1], [-1, 0, 1]]) # Создаем ядро (маску) для Y

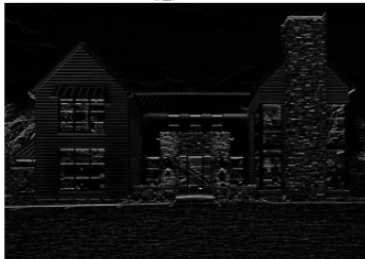
# Функция соединения изображения с ядром
img_prewittx = cv2.filter2D(image, -1, xkernel)
img_prewitty = cv2.filter2D(image, -1, ykernel)
```

```
1]: plt.figure(figsize=(10, 10))
plt.subplot(121),plt.imshow(img_prewittx, cmap = 'gray'), plt.title('img_prewittx')
plt.axis('off')

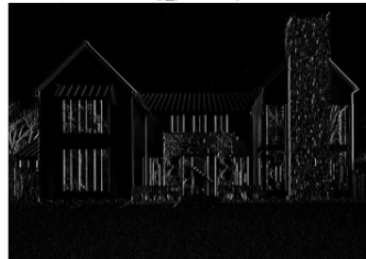
plt.subplot(122),plt.imshow(img_prewitty, cmap = 'gray'), plt.title('img_prewitty')
plt.axis('off')

plt.show();
```

img\_prewittx



img\_prewitty



Комбинированный оператор Превитта

```
: gradient_magnitude = np.sqrt(img_prewittx**2 + img_prewitty**2)
gradient_direction = np.arctan2(img_prewitty, img_prewittx)

: plt.figure(figsize=(10, 10))
plt.subplot(121),plt.imshow(image, cmap = 'gray'), plt.title('Original')
plt.axis('off')

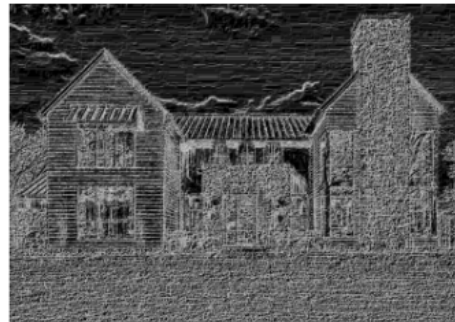
plt.subplot(122),plt.imshow(combined, cmap = 'gray'), plt.title('Combined')
plt.axis('off')

plt.show();
```

Original



Combined



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