**Министерство образования Республики Беларусь**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

**Кочева Ильи Александровича**

Отчет по лабораторной работе № 1,

студента 3-го курса 3-ой группы

2025

**Отчет по заданию: Построение морфологического скелета**

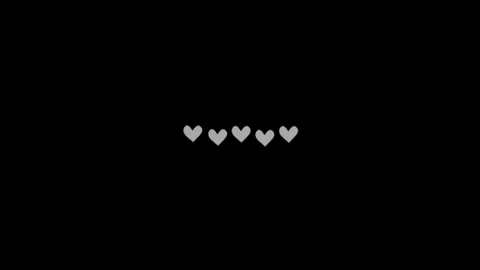
**1. Цель работы**

Разработать программу для построения морфологического скелета бинарного изображения и восстановления изображения по морфологическому скелету с использованием различных структурирующих элементов.

**2. Исходные данные**

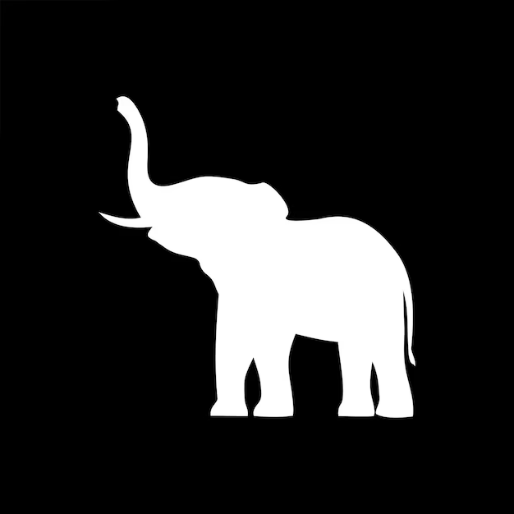
В качестве исходных данных использовались следующие бинарные изображения:

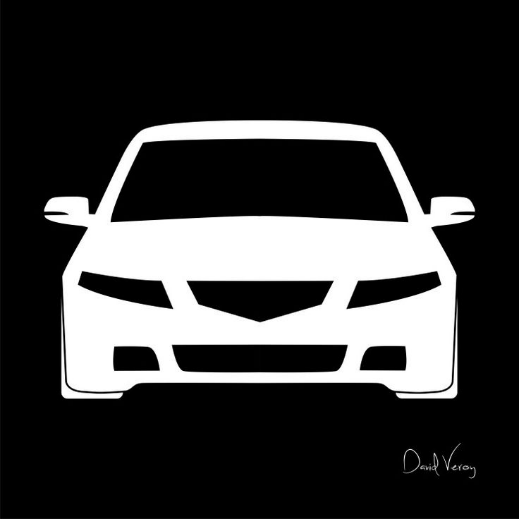
**Исходное изображение №1**  


**Исходное изображение №2**  


**Исходное изображение №3**

**Исходное изображение №4**



**Исходное изображение №5**

**3. Код программы**Реализация выполнена на Python в среде Jupyter Notebook с использованием нескольких базовых структурирующих элементов: Крест, Прямоугольник, Эллипс, Круг.  
 Программный код для построения морфологического скелета и восстановления изображения приведен ниже.

import numpy as np

import cv2

from matplotlib import pyplot as plt

from scipy.ndimage import binary\_erosion, binary\_dilation

def show\_images(images, columns=2, figsize=(10, 10)):

    if isinstance(images, np.ndarray):

        images = [images]

    rows = (len(images) + columns - 1) // columns

    fig, axes = plt.subplots(rows, columns, figsize=figsize)

    axes = np.array(axes).flatten()

    for i, img in enumerate(images):

        axes[i].imshow(img, cmap='gray', interpolation='none')

        axes[i].axis('off')

    for i in range(len(images), len(axes)):

        axes[i].axis('off')

    plt.tight\_layout()

    plt.show()

image\_path = 'car.png'

original\_image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

binary\_image = original\_image > 127

show\_images(binary\_image, columns=1, figsize=(6, 6))

def create\_structuring\_element(shape, size):

    if shape == 'cross':

        element = cv2.getStructuringElement(cv2.MORPH\_CROSS, (size, size))

    elif shape == 'rectangle':

        element = cv2.getStructuringElement(cv2.MORPH\_RECT, (size, size))

    elif shape == 'ellipse':

        element = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (size, size))

    elif shape == 'circle':

        element = np.array([

            [

                np.hypot(row, col) <= size for col in range(-size, size + 1)

            ] for row in range(-size, size + 1)

        ], dtype=np.uint8)

    else:

        raise ValueError(f"Неизвестная форма: {shape}")

    return element

structuring\_elements = {

    'Крест': create\_structuring\_element('cross', 5),

    'Прямоугольник': create\_structuring\_element('rectangle', 5),

    'Эллипс': create\_structuring\_element('ellipse', 5),

    'Круг': create\_structuring\_element('circle', 3)

}

show\_images(list(structuring\_elements.values()), columns=4, figsize=(12, 4))

def compute\_skeleton(binary\_img, struct\_elem, return\_all\_steps=False):

    binary\_img = binary\_img.astype(bool)

    struct\_elem = struct\_elem.astype(bool)

    skeleton = np.zeros\_like(binary\_img, dtype=bool)

    steps = [] if return\_all\_steps else None

    while True:

        eroded = binary\_erosion(binary\_img, structure=struct\_elem)

        if not eroded.any():

            break

        temp = binary\_dilation(eroded, structure=struct\_elem)

        diff = binary\_img & ~temp

        skeleton |= diff

        if return\_all\_steps:

            steps.append(diff)

        binary\_img = eroded

    if return\_all\_steps:

        steps.append(binary\_img)

        return steps

    else:

        return skeleton

def reconstruct\_image(steps, struct\_elem):

    reconstructed = np.zeros\_like(steps[0], dtype=bool)

    for step in reversed(steps):

        reconstructed = binary\_dilation(reconstructed, struct\_elem) | step

    return reconstructed

for name, struct\_elem in structuring\_elements.items():

    print(f"Обработка с использованием структурирующего элемента: {name}")

    skeleton\_steps = compute\_skeleton(binary\_image, struct\_elem, return\_all\_steps=True)

    skeleton\_image = np.any(skeleton\_steps, axis=0)

    reconstructed\_image = reconstruct\_image(skeleton\_steps, struct\_elem)

    show\_images(

        [binary\_image, skeleton\_image, reconstructed\_image],

        columns=3,

        figsize=(12, 4)

    )

|  |  |  |  |
| --- | --- | --- | --- |
| Структурирующий элемент | Исходное изображение | Морфологический скелет | Восстановленное изображение |
| Крест |  | | |
| Прямоугольник |  | | |
| Эллипс |  | | |
| Круг |  | | |

**4. Результаты**

Изображение №1

|  |  |  |  |
| --- | --- | --- | --- |
| Структурирующий элемент | Исходное изображение | Морфологический скелет | Восстановленное изображение |
| Крест |  | | |
| Прямоугольник |  | | |
| Эллипс |  | | |
| Круг |  | | |

Изображение №2

|  |  |  |  |
| --- | --- | --- | --- |
| Структурирующий элемент | Исходное изображение | Морфологический скелет | Восстановленное изображение |
| Крест |  | | |
| Прямоугольник |  | | |
| Эллипс |  | | |
| Круг |  | | |

Изображение №3

|  |  |  |  |
| --- | --- | --- | --- |
| Структурирующий элемент | Исходное изображение | Морфологический скелет | Восстановленное изображение |
| Крест |  | | |
| Прямоугольник |  | | |
| Эллипс |  | | |
| Круг |  | | |

Изображение №4

|  |  |  |  |
| --- | --- | --- | --- |
| Структурирующий элемент | Исходное изображение | Морфологический скелет | Восстановленное изображение |
| Крест |  | | |
| Прямоугольник |  | | |
| Эллипс |  | | |
| Круг |  | | |

Изображение №5

**5. Выводы**

Разработана программа для построения морфологического скелета бинарного изображения.

Выполнено восстановление изображения по построенному скелету.

Проведено тестирование программы с использованием различных структурирующих элементов (например, крестовидный и квадратный элементы).

Результаты показали корректность работы алгоритмов морфологического анализа.