МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и электроники

Кафедра технической кибернетики

Отчет по лабораторной работе №3

Дисциплина: «Системы обработки изображений»
Вариант №10

Выполнил: Новицкий Н. В.
Группа: 6132-010402D

Самара 2022

Задание на лабораторную работу

1. Создать средствами OpenCV бинарное изображение с простым рисунком.
2. Зашумить изображение импульсным шумом с вероятностью p (по вариантам).
3. Написать функции реализации эрозии и дилатации.
4. Выполнить операции эрозии и дилатации для зашумленного изображения со структурным элементом заданным по вариантам.
5. Отфильтровать зашумленное изображение при помощи морфологических операций вскрытия и закрытия (структурный элемент задан по вариантам). Подсчитать коэффициент шума для результат фильтрации.
6. Отфильтровать изображение при помощи логического фильтра. Подсчитать коэффициент шума для результат фильтрации.
7. На исходном изображении с помощью морфологических операций выделить контур объекта. Выяснить, когда контур получается внешним, внутренним, четырёхсвязным, восьмисвязным.
8. На исходном изображении с помощью морфологических операций выделить горизонтальные и вертикальные контуры объекта.

Процесс выполнения

Задание 1. Создать средствами OpenCV бинарное изображение с простым рисунком.

Код для создания простого рисунка средствами OpenCV:

```
img = np.zeros([200, 200], np.uint8)

# Квадраты
cv2.rectangle(img, (90, 90), (30, 30), (255, 255, 255), -1)
cv2.rectangle(img, (110, 110), (170, 170), (255, 255, 255), -1)

# Круги
cv2.circle(img, (140, 60), 32, (255, 255, 255), -1)
cv2.circle(img, (60, 140), 32, (255, 255, 255), -1)

show_image(img)
```

Результат:

Рисунок 1. Бинарное изображение

Задание 2. Зашумить изображение импульсным шумом с вероятностью p (по вариантам).

Вероятность p для нанесения шума на изображение в 10 варианте: 0,25.

Реализация импульсного шума на языке Python:

```
def pulse_noise(image, probability):
    """
    Импульсный шум
    """
    image_rows, image_cols = image.shape
    output_image = image.copy()

    for pixel in list(product(range(image_rows), range(image_cols))):
        if np.random.random() <= probability:
            output_image[pixel[0], pixel[1]] = 255

    return output_image
```

Результат:

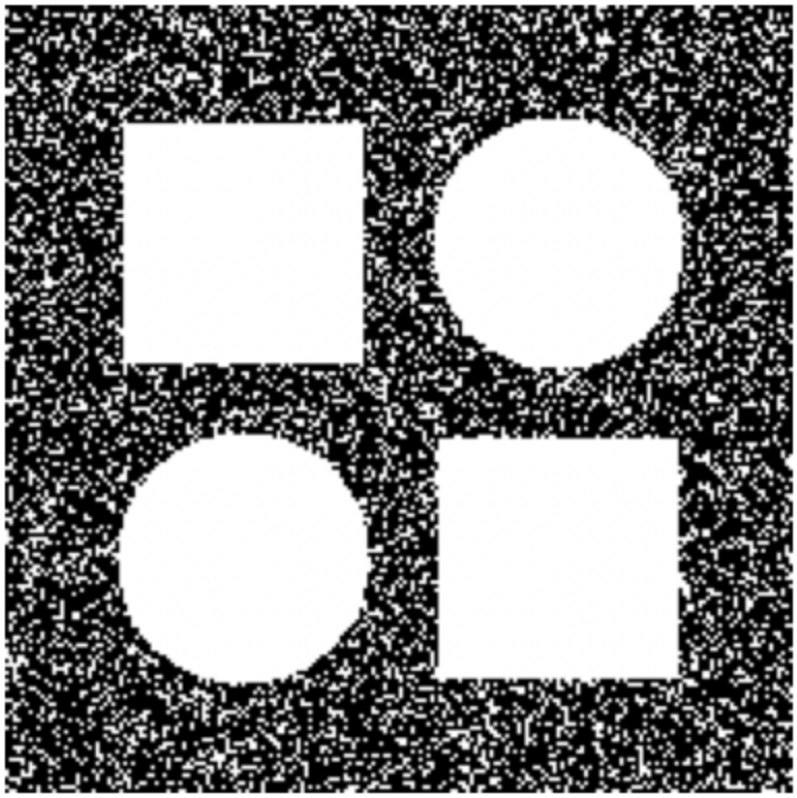

Рисунок 2. Зашумленное изображение

Задание 3. Написать функции реализации эрозии и дилатации.

Эрозия ("сужение" или "размывание") — одна из основных операций математической морфологии. Это операция, обратная операции дилатация. Фильтр **Эрозия** уменьшает область изображения, приводя к истончению пикселей, расширяя и усиливая светлые места на изображении. Суть данного преобразования состоит в том, что нежелательные вкрапления и шумы размываются, а большие и, соответственно, значимые участки изображения изменениям не подвергаются.

Дилатация ("расширение" или "наращивание") — одна из основных операций математической морфологии. Фильтр **Дилатация** увеличивает область изображения, расширяя его пиксели и тем самым способствуя объединению областей изображения, которые были разделены шумом и др. Изображение после фильтра становится светлее и слегка размытым. То есть темные детали ослабляются или вообще исчезают, что зависит от соотношения их размеров и яркостей с заданными параметрами фильтра.)

Реализация эрозии и дилатации на языке Python:

```
def get_value(current_window, kernel):
    """
    Вспомогательная функция
    """
    vals = []
    kernel_rows, kernel_cols = kernel.shape

    for i, j in np.ndindex((kernel_rows, kernel_cols)):
        if kernel[i, j] == 1:
            vals.append(current_window[i, j])

    return vals


def erosian(image, kernel):
    """
    Функция эрозии
    """
    image_rows, image_cols = image.shape
    kernel_rows, kernel_cols = kernel.shape
     temp_image = np.zeros((image_rows + 2 * (kernel_rows // 2), image_cols + 2 *
(kernel_cols // 2)))
    temp_image[
        (kernel_rows // 2): image_rows + 2 * (kernel_rows // 2) - (kernel_rows // 2),
        (kernel_cols // 2): image_cols + 2 * (kernel_cols // 2) - (kernel_cols // 2)
    ] = image
    output_image = image.copy()

    for i, j in tqdm(np.ndindex((image_rows, image_cols))):
        current_window = temp_image[i: i + kernel_rows, j: j + kernel_cols]
        output_image[i, j] = np.min(get_value(current_window, kernel))

    clear_output()
    return output_image
```

```
def dilation(image, kernel):
    """
    Функция дилатации
    """
    image_rows, image_cols = image.shape
    kernel_rows, kernel_cols = kernel.shape
     temp_image = np.zeros((image_rows + 2 * (kernel_rows // 2), image_cols + 2 *
(kernel_cols // 2)))
    temp_image[
        (kernel_rows // 2): image_rows + 2 * (kernel_rows // 2) - (kernel_rows // 2),
        (kernel_cols // 2): image_cols + 2 * (kernel_cols // 2) - (kernel_cols // 2)
    ] = image
    output_image = image.copy()

    for i, j in tqdm(np.ndindex((image_rows, image_cols))):
        current_window = temp_image[i: i + kernel_rows, j: j + kernel_cols]
        output_image[i, j] = np.max(get_value(current_window, kernel))

    clear_output()
    return output_image
```

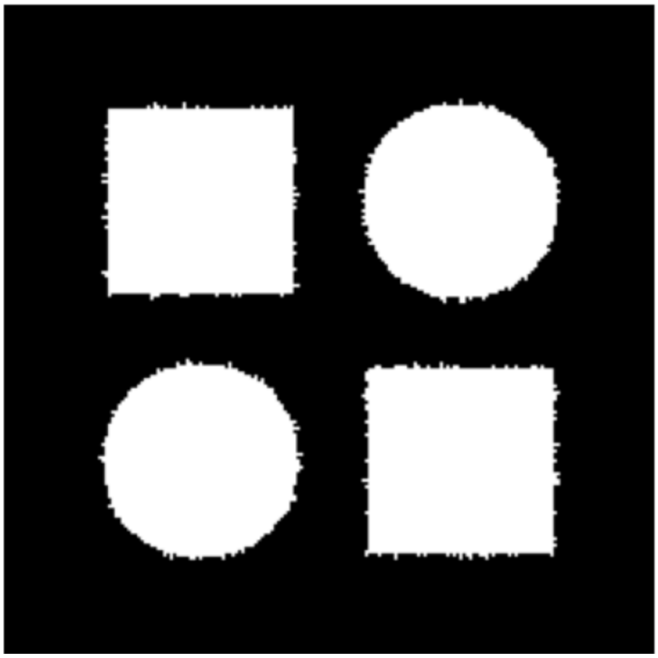

Рисунок 3. Изображение после операции эрозии

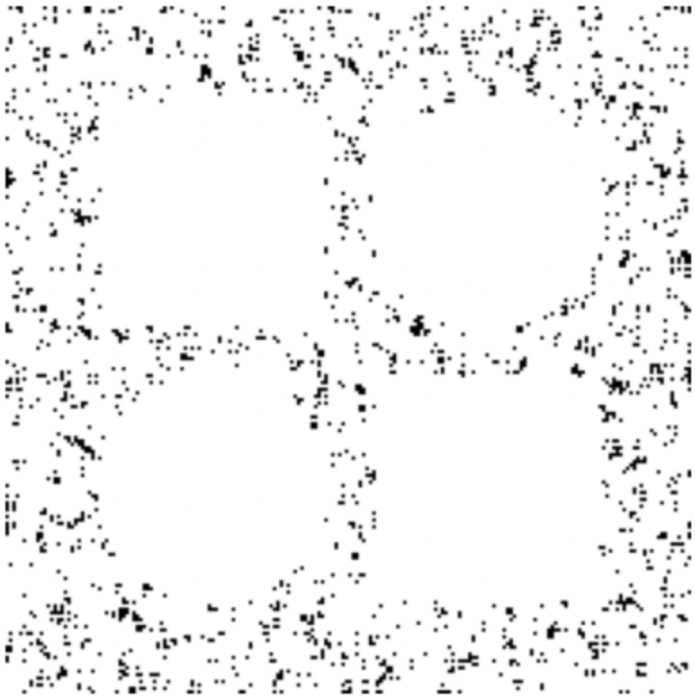

Рисунок 4. Изображение после операции дилатации

Задание 5. Отфильтровать зашумленное изображение при помощи морфологических операций вскрытия и закрытия (структурный элемент задан по вариантам). Подсчитать коэффициент шума для результат фильтрации.

Структурный элемент для фильтрации изображения:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

При использовании преобразования **открытие (дилатация эрозии изображения)** сначала к изображению применяется эрозия (сужение), а потом дилатация (расширение). При этом изображение может быть как черно-белым, так и полутоновым. Фильтр удаляет небольшие светлые детали на изображении, сохраняя общую и яркость крупных деталей. Т.е. он сглаживает и удаляет шумы, а также помогает избавиться от маленьких фрагментов, выступающих наружу области вблизи её границы. В основном данное преобразование используется для выравнивания неравномерной освещенности фона изображения. Размыкание применяется еще и для подсчёта участков на бинарном (черно-белом) изображении. Например, после порогового преобразования какого-либо изображения с помощью данного фильтра можно подсчитать количество частиц, расположенных ближе к друг другу, перед

подсчётом участков.)

При использовании преобразования закрытие (эрозия дилатации изображения) сначала к изображению применяется дилатация (расширение), а потом эрозия (сужение). То есть фильтр удаляет темные детали и делает изображение светлее, а затем уменьшает общую яркость до прежнего уровня, не восстанавливая темные детали. При этом изображение может быть как черно-белым, так и полутоновым. Фильтр Замыкание удаляет темные детали на изображении, почти не изменяя яркие детали, т.е. сглаживает и удаляет нежелательные шумы. Он помогает замкнуть внутренние отверстия области и устранить заливы вдоль границы области.

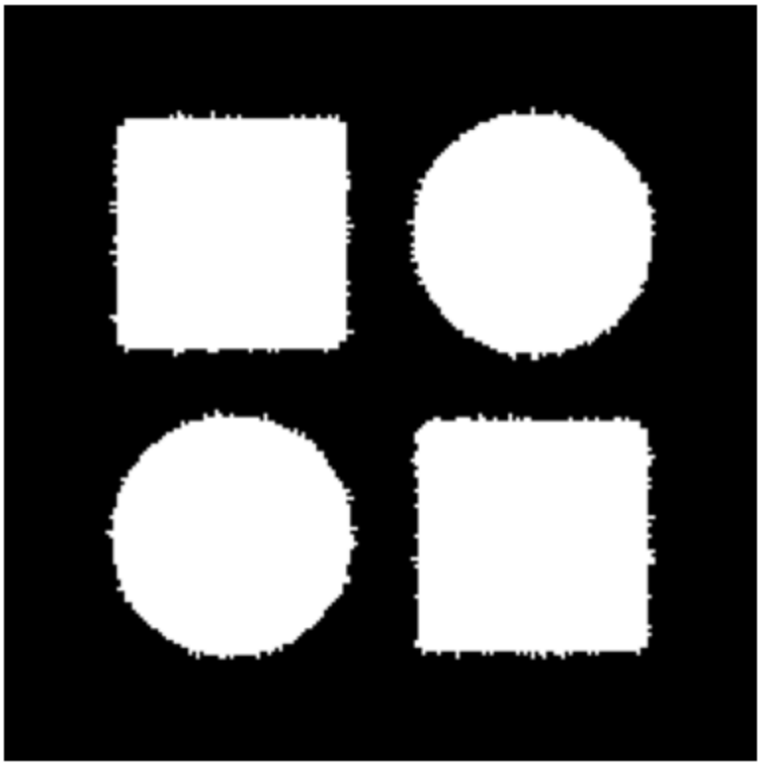

Рисунок 5. Изображение после операции открытия

Коэффициент шума после применения данной операции составил 0.66%, когда зашумленное изображение имело всего 16.125%.

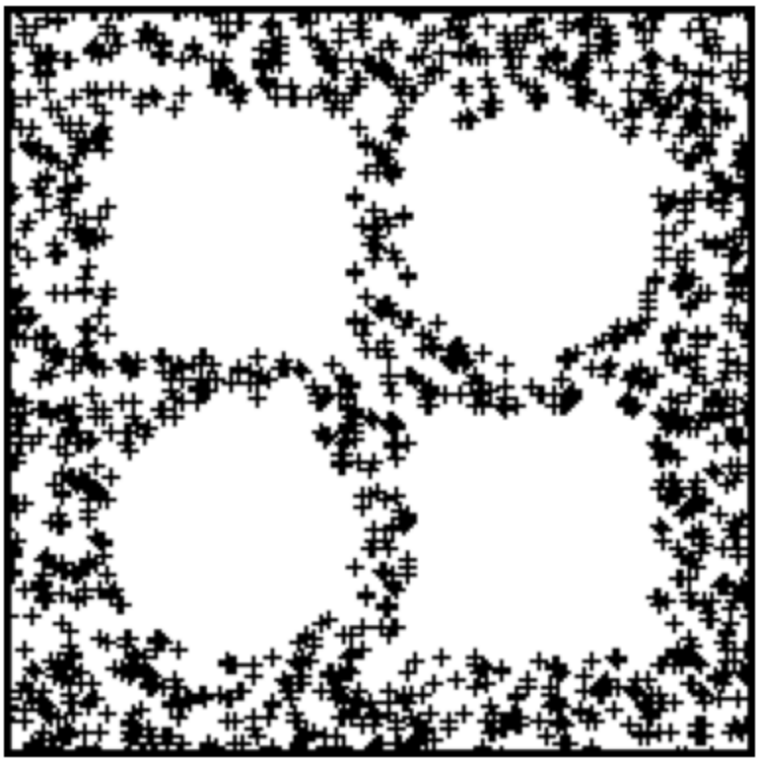

Рисунок 6. Изображение после операции закрытия

Коэффициент зашумленности изображения после операции закрытия составил 37.4254%.

Задание 6. Отфильтровать изображение при помощи логического фильтра. Подсчитать коэффициент шума для результат фильтрации.

Формула логического фильтра:

$$y = x_0 \wedge (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \vee \vee \neg x_0 \wedge (x_1 \wedge x_2 \wedge x_3 \wedge x_4)$$

Реализация логического фильтра на языке Python:

```
def value_for_logic_filter(image, i, j):
    """
      Вспомогательная функция, которая формирует значения для логического фильтра
    """
    filter_value = image[i, j] and (image[i - 1, j] or image[i, j - 1] or image[i, j + 1] or image[i + 1, j]) \
    or (not image[i, j]) and (image[i - 1, j] and image[i, j - 1] and image[i, j + 1] and image[i + 1, j])
    return int(filter_value)


def logic_filter(image):
    """
    Функция логического фильтра
```

```
    """
    image_rows, image_cols = image.shape
    output_image = np.copy(image)

    temp_image = np.zeros((image_rows + 2, image_cols + 2))
    temp_image[1: -1, 1: -1] = image
    temp_image = temp_image.astype(bool)

    for i, j in tqdm(np.ndindex((image_rows, image_cols))):
        output_image[i, j] = value_for_logic_filter(temp_image, i, j)

    clear_output()
    return output_image
```

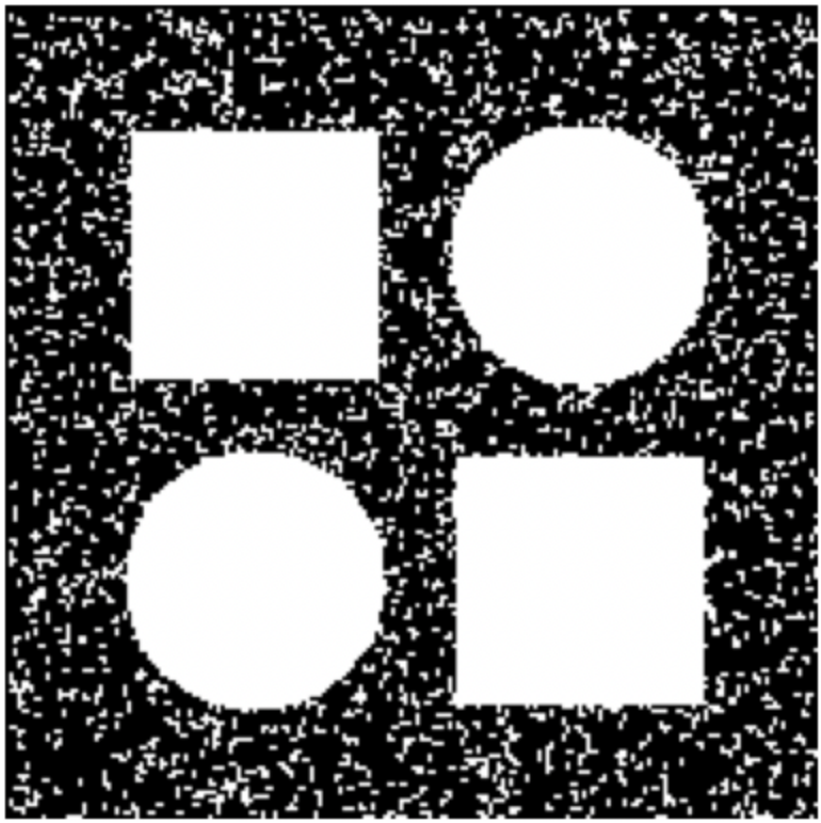

Рисунок 7. Изображение после фильтрации логическим фильтром

Коэффициент зашумленности после применения логического фильтра составил 46.585%.

Задание 7. На исходном изображении с помощью морфологических операций выделить контур объекта. Выяснить, когда контур получается внешним, внутренним, четырёхсвязным, восьмисвязным.

Внешний контур выделялся с помощью операции дилатации и со следующим структурным элементом:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

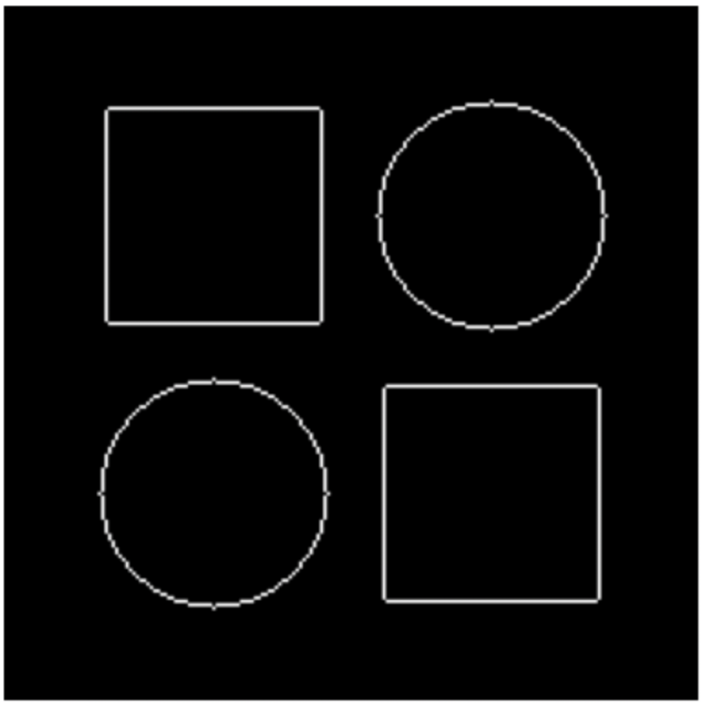

Рисунок 8. Выделение внешнего контура изображения

Внутренний контур выделялся с помощью операции эрозии со следующим структурным элементом:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

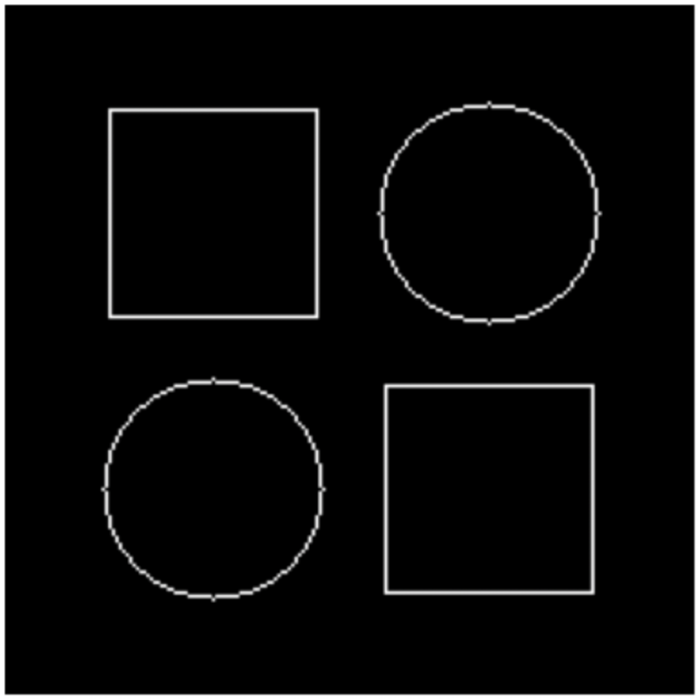

Рисунок 9. Выделение внутреннего контура изображения

Восьмисвязанный/четырехсвязанный внутренние контура выделялись с помощью эрозии, но с разными структурными элементами, для восьмисвязанного контура был взят элемент для выделения внешнего контура, а для четыресвязанного контура был взял элемент для выделения внутреннего контура.

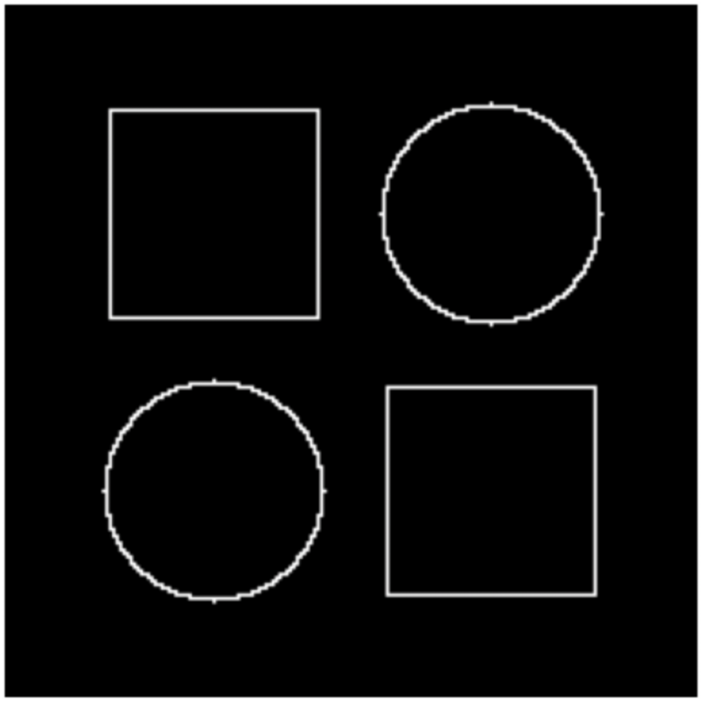

Рисунок 10. Четыресвязанный внутренний контур

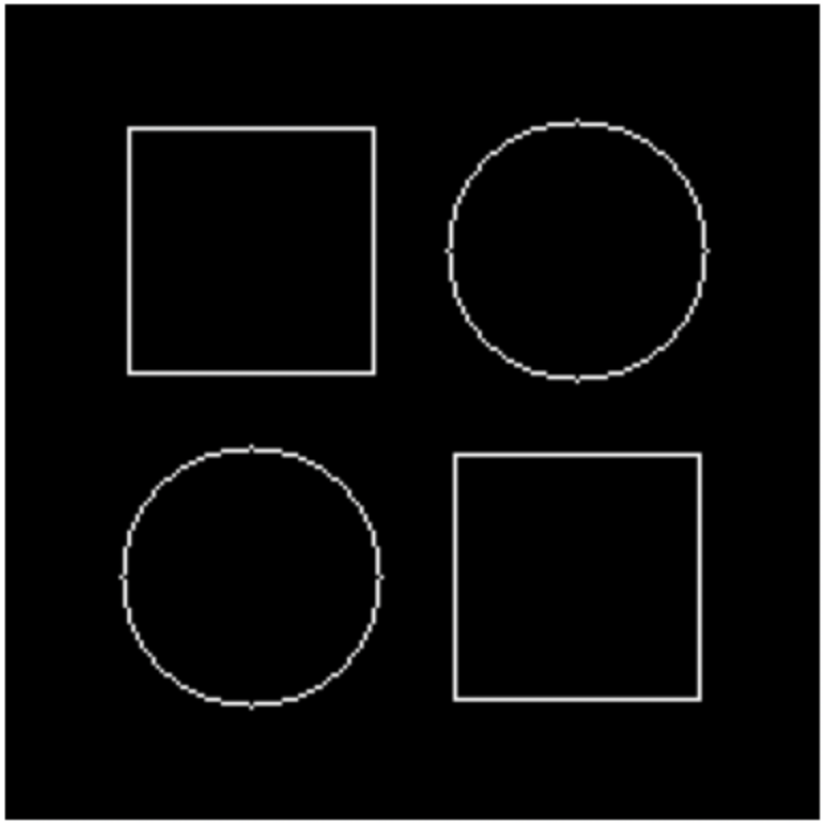

Рисунок 11. Восьмисвязанный внутренний контур

Задание 8. На исходном изображении с помощью морфологических операций выделить горизонтальные и вертикальные контуры объекта.

Вертикальные/горизонтальные контура изображения выделялись с помощью операции эрозии, но с разными структурными элементами.

Структурный элемент для выделения вертикальных контуров:

$$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Структурный элемент для выделения горизонтальных контуров:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Рисунок 12. Горизонтальные контуры изображения

Рисунок 13. Вертикальные контуры изображения

Заключение

В результате выполнения лабораторной работы были приобретены навыки использования морфологических операций над изображениями: эрозия, дилатация, открытие, закрытие и морфологический фильтр. Так же с помощью операций эрозии и далитации были выделены разные контуры изображения с помощью различных структурных элементов.

Исходный код можно найти в репозитории GitHub автора лабораторной работы: ссылка