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

«Пространственные методы обработки изображений»

по дисциплине «Технологии распознавания образов»

Выполнил студент группы
ПИЖ-б-о-21-1
Зиберов Александр
« » мая 2023 г.
Подпись студента_____
Работа защищена
« »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2023

Цель работы:

Изучить пространственные методы обработки изображений (зашумление, сглаживание, усреднение, фильтрация и другие).

Выполнение работы:

Проработать примеры лабораторной работы в отдельном ноутбуке.

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

Задание 6.1.

Создать файл с зашумлением изображения шумом типа соль-перец.

```
In [2]: red, green, blue = (255, 0, 0), (0, 255, 0), (0, 0, 255)
        rgb = [red, green, blue]

In [3]: def sp_noise(image, prob):
            output = np.zeros(image.shape, np.uint8)
            thres = 1- prob
            for i in range(image.shape[0]):
                for j in range(image.shape[1]):
                    rnd = random.random()
                    if rnd > thres:
                        output[i][j] = random.choice(rgb)
                    else:
                        output[i][j] = image[i][j]
            return output

In [4]: image = cv2.imread('pictures/gi.jpg')
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = cv2.resize(image, (900, 600))

In [5]: noise_img = sp_noise(image, 0.3)
        res = np.hstack((image, noise_img))

        plt.figure(figsize=(20,20))
        plt.axis("off")
        plt.imshow(res);
```

Рисунок 1 – Пример 1

Задание 6.2.

Провести сглаживание изображения с помощью функции cv2.filter2D (), используя ядро 5×5.

```
In [6]: img = cv2.imread('pictures/gi.jpg')
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

In [7]: kernel = np.ones((5, 5), np.float32) / 25
        dst = cv2.filter2D(img, -1, kernel)

In [8]: plt.figure(figsize=(20,20))
        plt.subplot(121),plt.imshow(img),plt.title('Original')
        plt.axis("off")
        plt.subplot(122),plt.imshow(dst),plt.title('Averaging')
        plt.axis("off")
        plt.show()
```

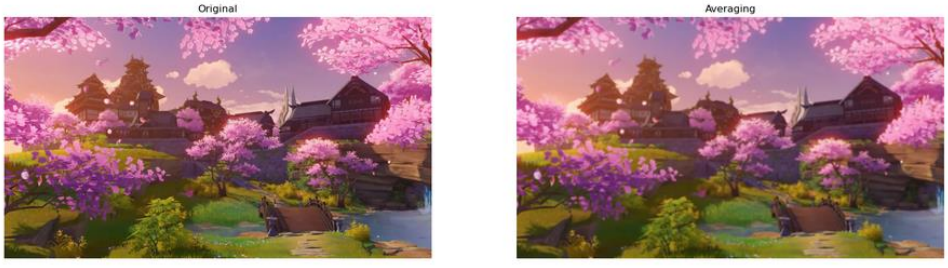


Рисунок 2 – Пример 2

Задание 6.3.

Провести усреднение изображения с помощью функции cv2.blur(), используя ядро 5×5.

```
In [9]: img = cv2.imread('pictures/gi.jpg')
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

In [10]: blur = cv2.blur(img, (5, 5))

In [11]: plt.figure(figsize=(20,20))
         plt.subplot(121),plt.imshow(img), plt.title('Original')
         plt.axis("off")
         plt.subplot(122),plt.imshow(blur), plt.title('Blurred')
         plt.axis("off")
         plt.show()
```

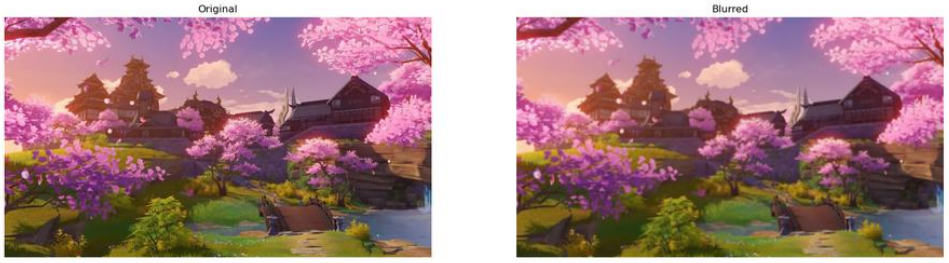


Рисунок 3 – Пример 3

Задание 6.4.

Добавить к исходному изображению 20–30% шума. Провести фильтрацию изображения по Гауссу, используя ядро 5×5.

```
In [12]: img = cv2.imread('pictures/gi.jpg')
         img = sp_noise(img, 0.25)
         img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

In [13]: blur = cv2.GaussianBlur(img, (5 ,5), 0)

In [14]: plt.figure(figsize=(20,20))
         plt.subplot(121),plt.imshow(img), plt.title('Original')
         plt.axis("off")
         plt.subplot(122),plt.imshow(blur), plt.title('Blurred')
         plt.axis("off")
         plt.show()
```

Рисунок 4 – Пример 4

Задание 6.5.

Добавить к исходному изображению 20–50% шума. Провести медианную фильтрацию изображения, используя ядро 5x5.

```
In [15]: img = cv2.imread('pictures/gi.jpg')
         img = sp_noise(img, 0.25)
         img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

In [16]: median = cv2.medianBlur(img,5)

In [17]: plt.figure(figsize=(20,20))
         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()
```

Рисунок 5 – 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.

In [18]:
```
img = cv2.imread('pictures/gi.jpg', 0)
img = cv2.resize(img, (900, 600))
```

In [19]:
```
sobel_vertical = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
sobel_horizontal = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)
```

In [20]:
```
plt.figure(figsize=(15,15))
plt.subplot(131),plt.imshow(img), plt.title('Original')
plt.axis("off")

plt.subplot(132),plt.imshow(sobel_vertical), plt.title('sobel_vertical')
plt.axis("off")

plt.subplot(133),plt.imshow(sobel_horizontal), plt.title('sobel_horizontal')
plt.axis("off")

plt.show()
```

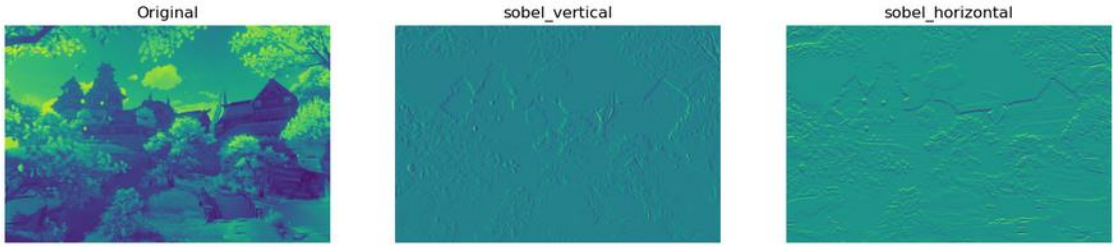


Рисунок 6 – Пример 6

Задание 6.7.

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

In [21]:
```
img = cv2.imread('pictures/gi.jpg', 0)
```

In [22]:
```
# 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)
```

In [23]:
```
plt.figure(figsize=(15,15))
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()
```

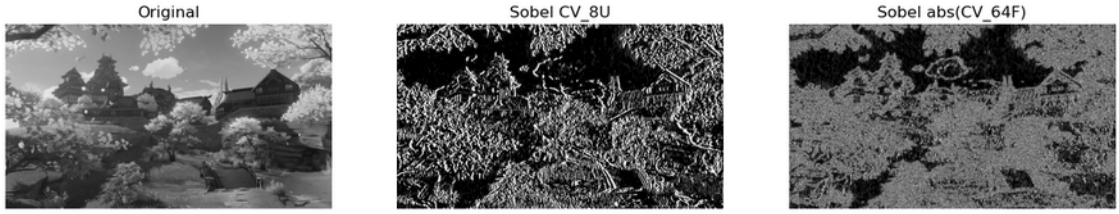


Рисунок 7 – Пример 7

Задание 6.8.

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

In [24]:
```
img = cv2.imread('pictures/gi.jpg', 0)
img = cv2.resize(img, (900, 600))
```

In [25]:
```
xkernel = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])
ykernel = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
```

In [26]:
```
img_prewittx = cv2.filter2D(img, -1, xkernel)
img_prewitty = cv2.filter2D(img, -1, ykernel)
```

In [27]:
```
plt.figure(figsize=(20,20))
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('Sobel img_prewitty')
plt.axis("off")

plt.show()
```

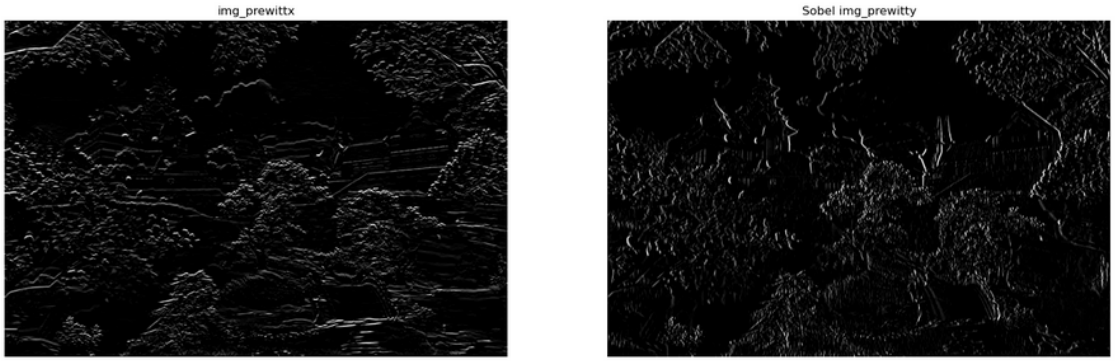


Рисунок 8 – Пример 8

Задание 6.7.

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

In [21]:
```
img = cv2.imread('pictures/gi.jpg', 0)
```

In [22]:
```
# 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)
```

In [23]:
```
plt.figure(figsize=(15,15))
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()
```

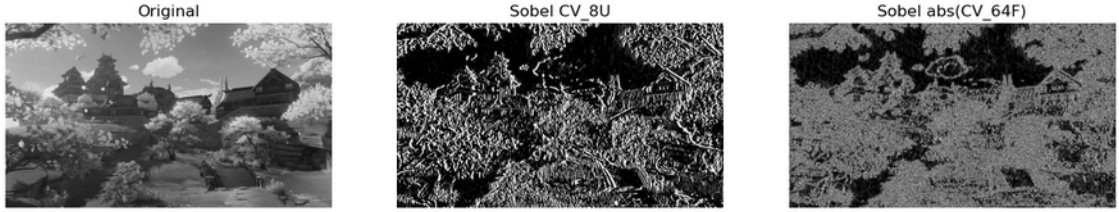


Рисунок 9 – Пример 9

Задание 6.8.

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

In [24]:
```
img = cv2.imread('pictures/gi.jpg', 0)
img = cv2.resize(img, (900, 600))
```

In [25]:
```
xkernel = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])
ykernel = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
```

In [26]:
```
img_prewittx = cv2.filter2D(img, -1, xkernel)
img_prewitty = cv2.filter2D(img, -1, ykernel)
```

In [27]:
```
plt.figure(figsize=(20,20))
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('Sobel img_prewitty')
plt.axis("off")

plt.show()
```

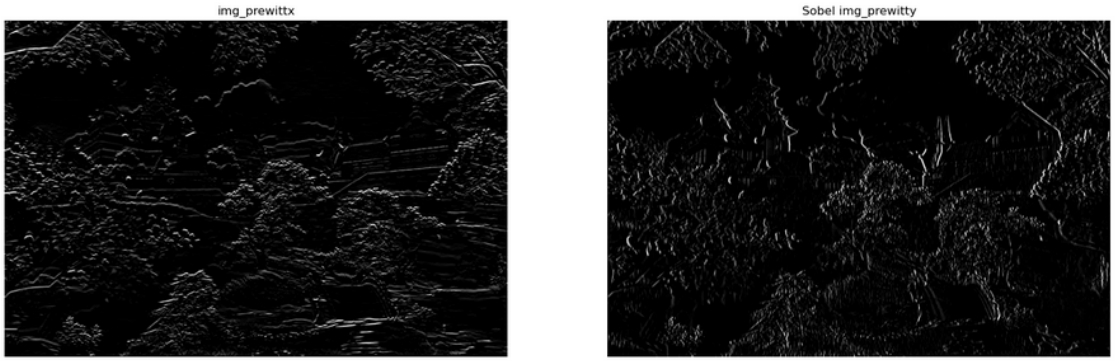


Рисунок 10 – Пример 10

Задание 6.9.

Используя оператор Робертса, выделить линии на изображении.

In [28]:
```
kernel1 = np.array([[1, 0], [0, 1]])
kernel2 = np.array ([[0, 1],[0, 1]])
```

In [29]:
```
img_robx = cv2.filter2D(img, -1, kernel1)
img_roby = cv2.filter2D(img, -1, kernel2)

output_image = img_robx + img_roby
```

In [30]:
```
plt.imshow(output_image, cmap = 'gray'), plt.title('output_image')
plt.axis("off")
plt.show()
```

Рисунок 11 – Пример 11

Задание 6.10.

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

In [31]:
```
laplacian = cv2.Laplacian(img, cv2.CV_64F)
```

In [32]:
```
plt.figure(figsize=(20,20))
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()
```

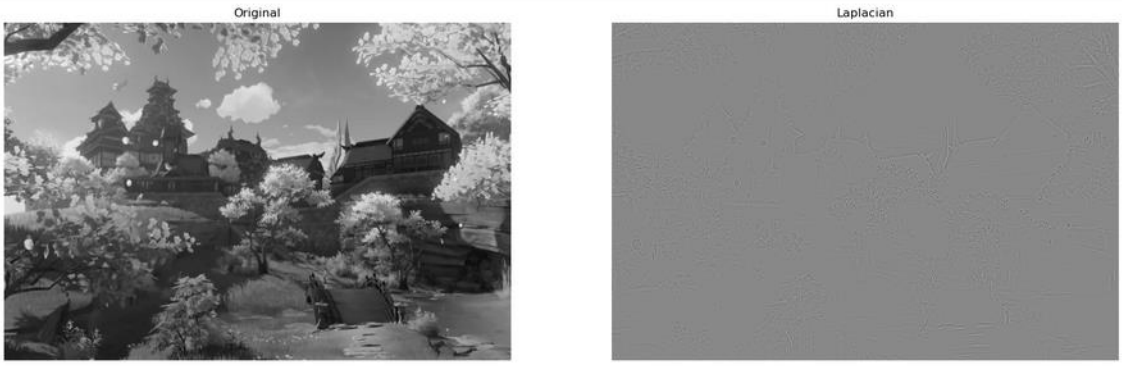


Рисунок 12 – Пример 12

Индивидуальное задание

Задание:

1. Добавить гауссовый шум на фото, а затем произвести его удаление разными способами
2. Выделить границы изображения с использованием операторов Собеля и Превитта

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

def show2compare(first, second, f_sign, s_sign, isGray):
    plt.figure(figsize=(15,15))
    if isGray:
        plt.subplot(121),plt.imshow(first, cmap='gray'), plt.title(f_sign), plt.axis("off")
        plt.subplot(122),plt.imshow(second, cmap='gray'), plt.title(s_sign), plt.axis("off")
    else:
        plt.subplot(121),plt.imshow(first), plt.title(f_sign), plt.axis("off")
        plt.subplot(122),plt.imshow(second), plt.title(s_sign), plt.axis("off")
    plt.show();
img_path = 'pictures/backrooms.jpg'
```

Добавление шума

Добавим гауссовый шум на изображение. Сгенерируем шум и добавим на изображение функцией cv2.add():

In [2]:
```
img = cv2.imread(img_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Рандомная генерация гауссова шума
mean, stddev = 10, 80
noise = np.zeros(img.shape, np.uint8)
cv2.randn(noise, mean, stddev)

# Добавляем шум
noisy_img = cv2.add(img, noise)

show2compare(img, noisy_img, "Оригинал", "Добавлен гауссовый шум", False)
```

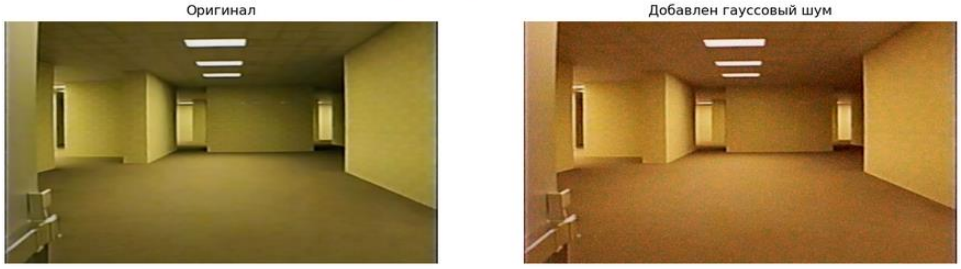


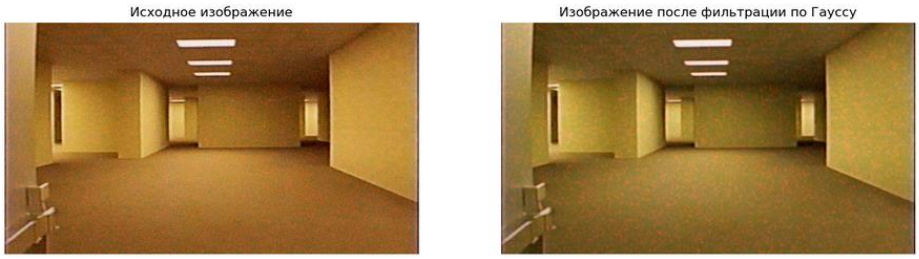
Рисунок 13 – Индивидуальное задание (1)

Удаление шума

Попробуем удалить шум при помощи медианного фильтра. Он вычисляет медианное значение всех пикселей, окружающих центральный пиксель, и его значение заменяется медианным.
Метод реализуется встроенной функцией cv2.medianBlur(src, size), где size - размер ядра.

In [3]:
```
median = cv2.medianBlur(noisy_img,5)

show2compare(noisy_img, median, "Исходное изображение", "Изображение после фильтрации по Гауссу", False)
```



Удаление шума методом нелокального усреднения. Данная функция реализует алгоритм, который вычисляет веса в зависимости от расстояния между блоками вокруг пикселей.

Выходное изображение после шумоподавления NLM определяется следующим образом:

$$NL[v](i) = \sum_{j \in I} w(i,j)v(j),$$

где I - область поиска, а $w(i,j)$ - вес, который определяется сходством совпадающих блоков.

In [4]:
```
denoised_NLM = cv2.fastNlMeansDenoisingColored(img, None, 10, 10)

show2compare(noisy_img, denoised_NLM, "Исходное изображение", "Удаление шума методом NLM", False)
```

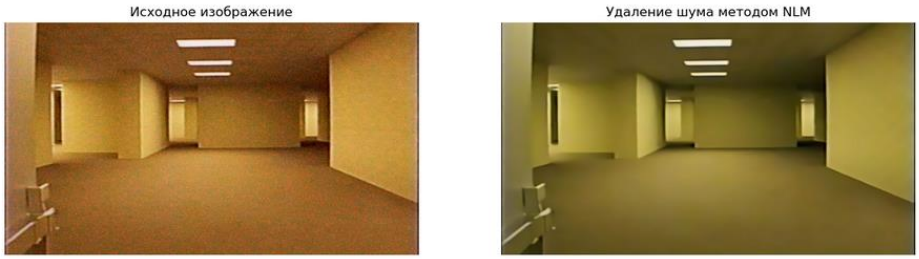


Рисунок 14 – Индивидуальное задание (2)

Нахождение вертикальных и горизонтальных границ изображения

Для начала найдем границы встроенной функцией cv2.Sobel(src, dst, dx, dy), параметры которой означают: src - исходное (входное) изображение, dst - целевое (выходное) изображение, dx и dy - производные соответственно x и y.

Затем выделим границы при помощи оператора Превитта, для этого создаем ядро и применяем его на изображении функцией cv2.filter2D.

In [5]:
```
img = cv2.imread(img_path, 0)

# Оператор Собеля
sobel_vertical = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
sobel_horizontal = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)
show2compare(sobel_horizontal, sobel_vertical, "Горизонтальные границы", "Вертикальные границы", True)

# Ядро (маска)
xkernel = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])
ykernel = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])

# Наложение маски на изображение
img_prewittx = cv2.filter2D(img, -1, xkernel)
img_prewitty = cv2.filter2D(img, -1, ykernel)

show2compare(img_prewittx, img_prewitty, "Выделение гориз. границ", "Выделение верт. границ", True)
```

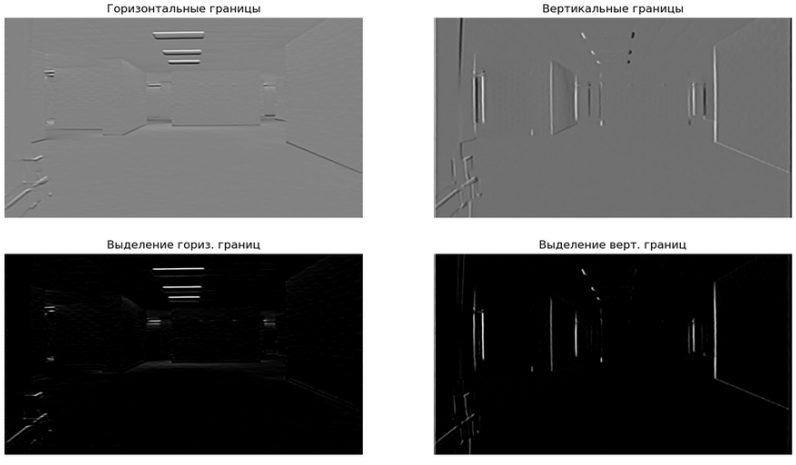


Рисунок 15 – Индивидуальное задание (3)

Вывод: В результате выполнения работы были изучены методы обработки изображений.

1. Что применяют для улучшения изображений и выделения их характерных особенностей?

Для улучшения изображений, выделения характерных особенностей применяют локальные преобразования в окрестности некоторого пикселя.

2. Что такое маска?

Маска – это локальный оператор, определяющий новое значение любого пикселя, которое равно линейной комбинации интенсивностей в окрестности пикселей.

3. Что такое веса?

Коэффициенты линейного преобразования называются весами.

4. Как возникает импульсивный шум (шум типа соль-перец)?

Такой шум возникает при случайном изменении интенсивностей пикселей, которые принимают минимальные или максимальные значения динамического диапазона пикселей. На изображении эти пиксели распределяются случайным образом. Это приводит к появлению на изображении черных и белых пикселей.

5. Что используют для сглаживания изображения? Для сглаживания изображений используют усредняющий фильтр. Усреднение участка изображения проводится с помощью ядра, например, 5x5. Для сглаживания (размытия) изображения используют также свертку с маской, при этом размерность маски должна быть равна сумме весов.

6. Как осуществляется Гауссова фильтрация?

Фильтрация с ядром Гаусса выполняется маской, веса которой находятся с помощью функции Гаусса. Рассматриваемая фильтрация осуществляется с помощью функции cv2.GaussianBlur(). В скобках второй аргумент – это ширина и высота ядра, которые должны быть положительными и нечетными. Указывается также стандартное отклонение в направлениях X и Y, sigmaX и sigmaY соответственно.

7. Как осуществляется медианная фильтрация?

В процессе этой фильтрации функция cv2.medianBlur() вычисляет медианное значение всех пикселей, окружающих центральный пиксель, и его значение заменяется медианным значением. Это очень эффективно для устранения шума соли и перца. Размер ядра должен быть положительным нечетным целым числом.