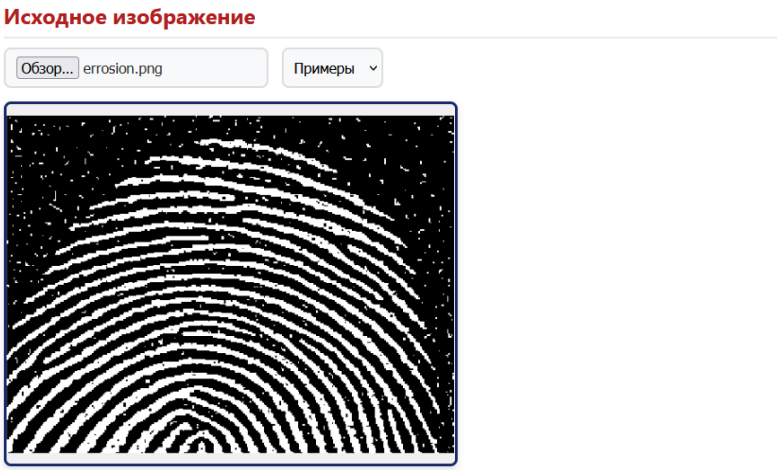
Вариант 16. В данной работе изучал применение различных обработок для улучшения качества изображений: морфологическая обработка, поэлементные операции и линейное контрастирование.

Код, как и в прошлой лабе, реализован на html, css и javascript.

Для каждой обработки подобрал по одному изображению, демонстрирующему корректность работы.

Для морфологической обработки, по сути реализующей математические операции над множеством пикселей, отлично сработал пример изображения со спектром, имеющем два пика, из презентации для отсеивания шума на изображении с отпечатком пальца.

Фото до:



Настройки: выбрал эрозию для удаления мусорных точек (у них очень маленькая граница). В качестве структурного элемента взял крест и размер ядра (длину стороны креста) равной 4:

```
applyMorphology(imageData) {
    const operation = document.querySelector('input[name="morphOperation"]:checked').value;
    const elementType = document.getElementById('structuringElement').value;
    const size = parseInt(document.getElementById('elementSize').value);
    const grayData = this.convertToGrayscale(imageData);
    const se = this.createStructuringElement(elementType, size);
    const result = new ImageData(new Uint8ClampedArray(grayData.data.length), grayData.width, grayData.height);

    for (let i = 0; i < grayData.data.length; i += 4) {
        result.data[i + 3] = 255;
    }


    for (let y = 0; y < grayData.height; y++) {
        for (let x = 0; x < grayData.width; x++) {
            const index = (y * grayData.width + x) * 4;
            let value;

            if (operation === 'erosion') {
                value = this.applyErosion(grayData, x, y, se);
            } else {
                value = this.applyDilation(grayData, x, y, se);
            }

            result.data[index] = value;
            result.data[index + 1] = value;
            result.data[index + 2] = value;
        }
    }
    return result;
}
```

После:

Результат обработки

Для реализации поэлементных операций избрал наивную коррекцию яркости, путём добавления в каждый параметр rgb фиксированную величину (это же аддитивная модель)

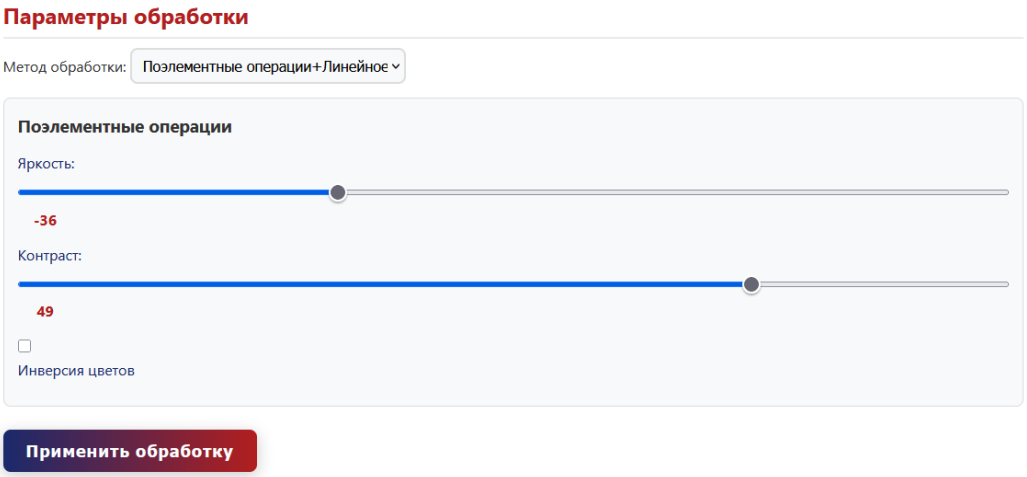
```
r = this.clamp(r + brightness);
g = this.clamp(g + brightness);
b = this.clamp(b + brightness);
```

функцию усиления контрастирования относительно центра 128 и инвертирование:

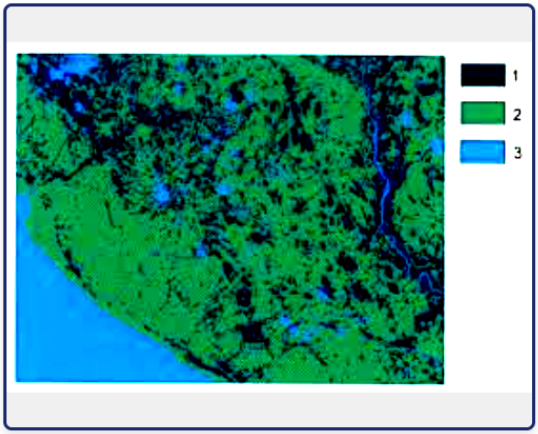
```
if (contrast !== 0) {
    const factor = (259 * (contrast + 1)) / (255 * (1 - contrast));
    r = this.clamp(factor * (r - 128) + 128);
    g = this.clamp(factor * (g - 128) + 128);
    b = this.clamp(factor * (b - 128) + 128);
}

if (invert) {
    r = 255 - r;
    g = 255 - g;
    b = 255 - b;
}
```

Пример применения из интернета на карте показывает пользу полученных операций. До:

Параметры:



После:



Выводы. Таким образом, я разобрался, как при помощи даже таких очевидных преобразований, как попиксильное увеличение яркости или дилатации со структурным элементом, делать какую-то нетривиальную обработку изображений с практическим применением.