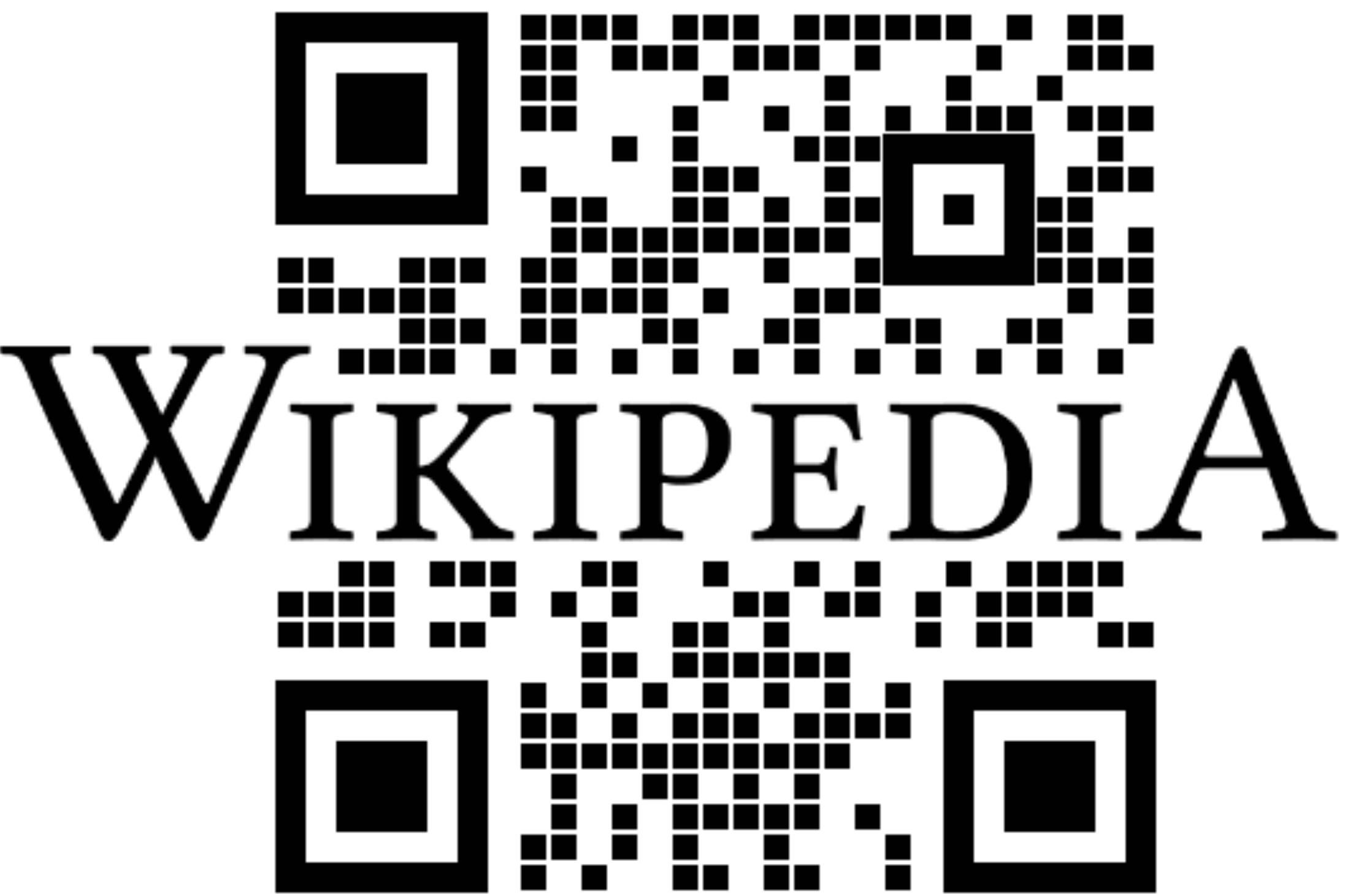


Распознавание QR-кода

работу выполнил:
Посевин М.Э.

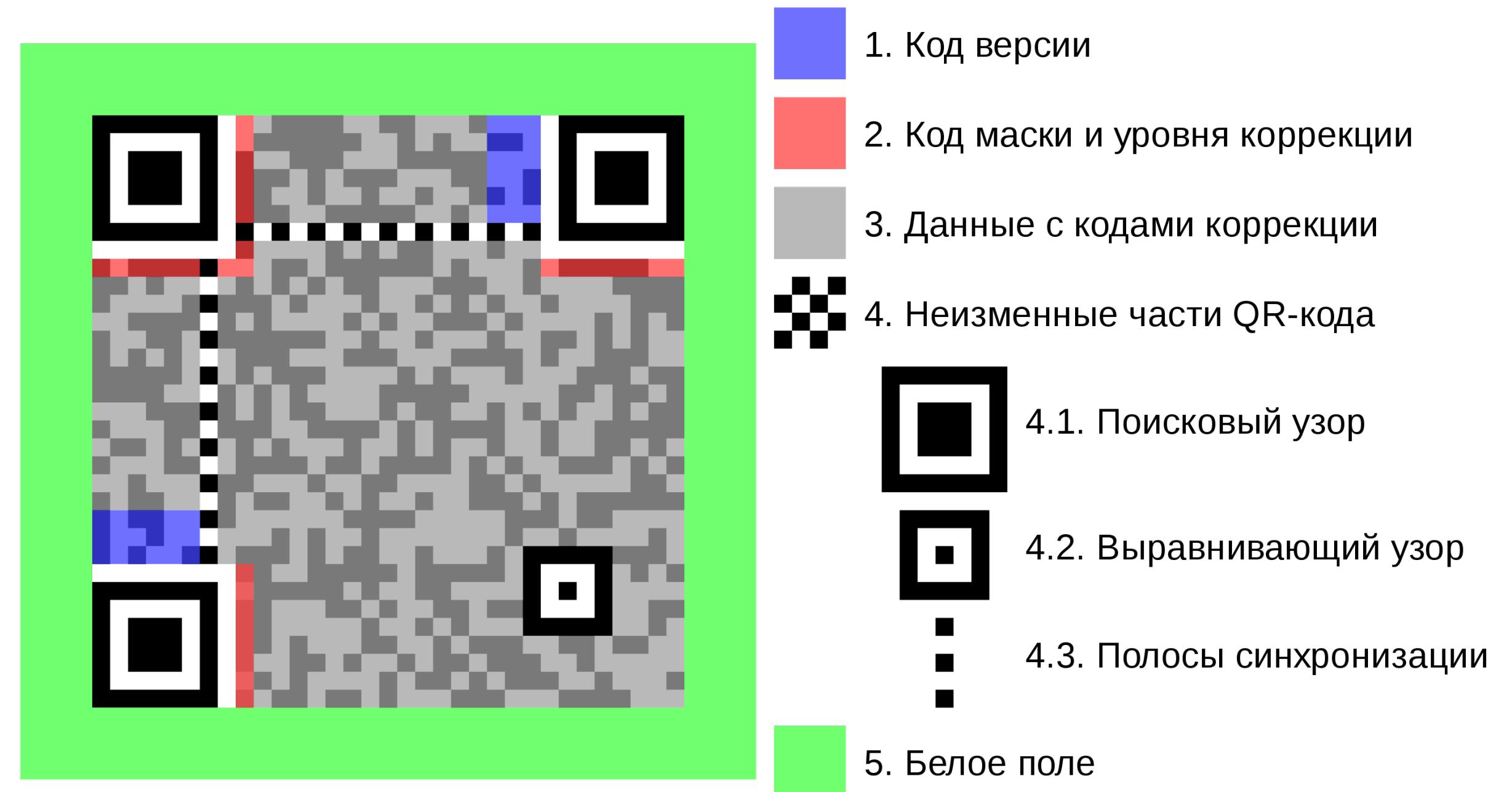
QR-код — тип матричных штрихкодов.
Штрихкод — считываемая машиной
оптическая метка, содержащая
информацию об объекте, к которому она
привязана.

QR-код состоит из чёрных квадратов,
расположенных в квадратной сетке на
белом фоне.



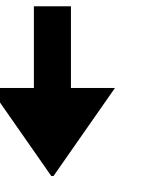
На QR-коде есть обязательные поля, они не несут закодированной информации, а содержат информацию для декодирования. Это:

- поисковые узоры
- выравнивающий узор
- полосы синхронизации
- код маски и уровня коррекции
- код версии

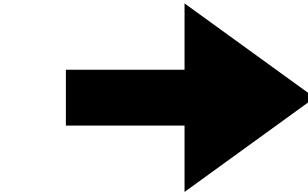


Алгоритм

1. Преобразование цветового режима изображения в оттенки серого
2. Применение оператора Собеля
3. Устранение шума
4. Бинаризация
5. Проведение ряда простых морфологических операций
6. Нахождение самого большого контура на изображении
7. Определение минимального ограничивающего четырехугольника
8. Кадрирование (+2%)
9. Уточнение границ
10. Обнаружение поисковых узоров
11. Вычисление размерности QR-кода по полосам синхронизации
12. Наложение сетки
13. Определение среднего цвета в элементах сетки

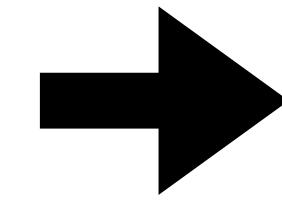


Преобразование цветового режима изображения в оттенки серого



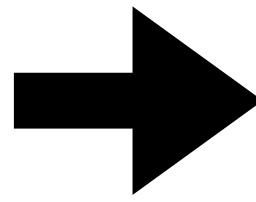
Применение оператора Собеля

Используем оператор Собеля, чтобы вычислить величину градиента серой картинки в вертикальном и горизонтальном направлениях. После этого мы складываем абсолютные значения X-градиента и Y-градиента оператора Собеля. После сложения мы получаем изображение с высоким значением горизонтального и вертикального градиента.



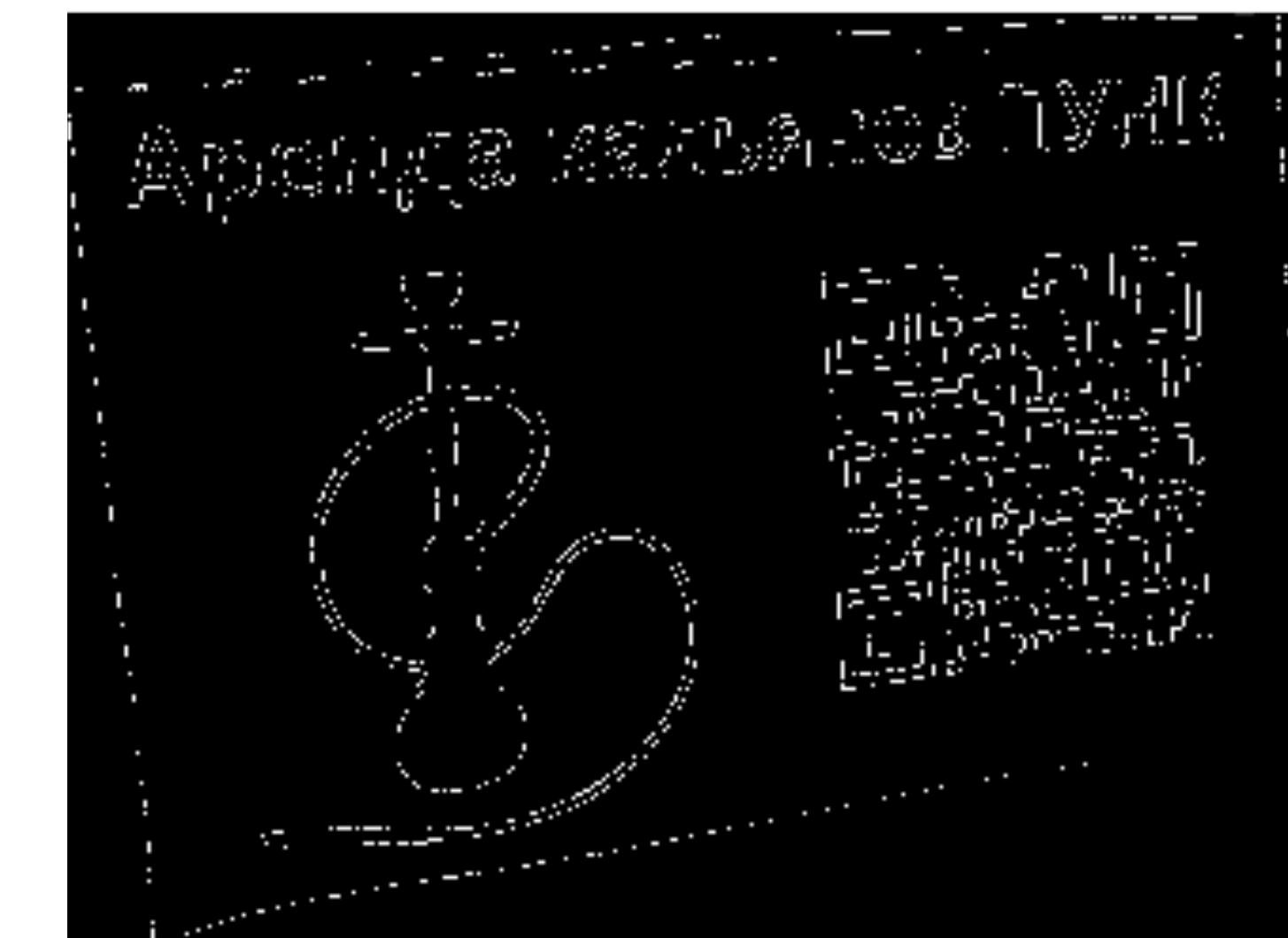
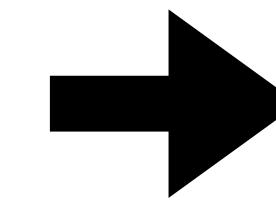
Устранение шума

Сглаживаем высокочастотный шум на изображении с градиентом.



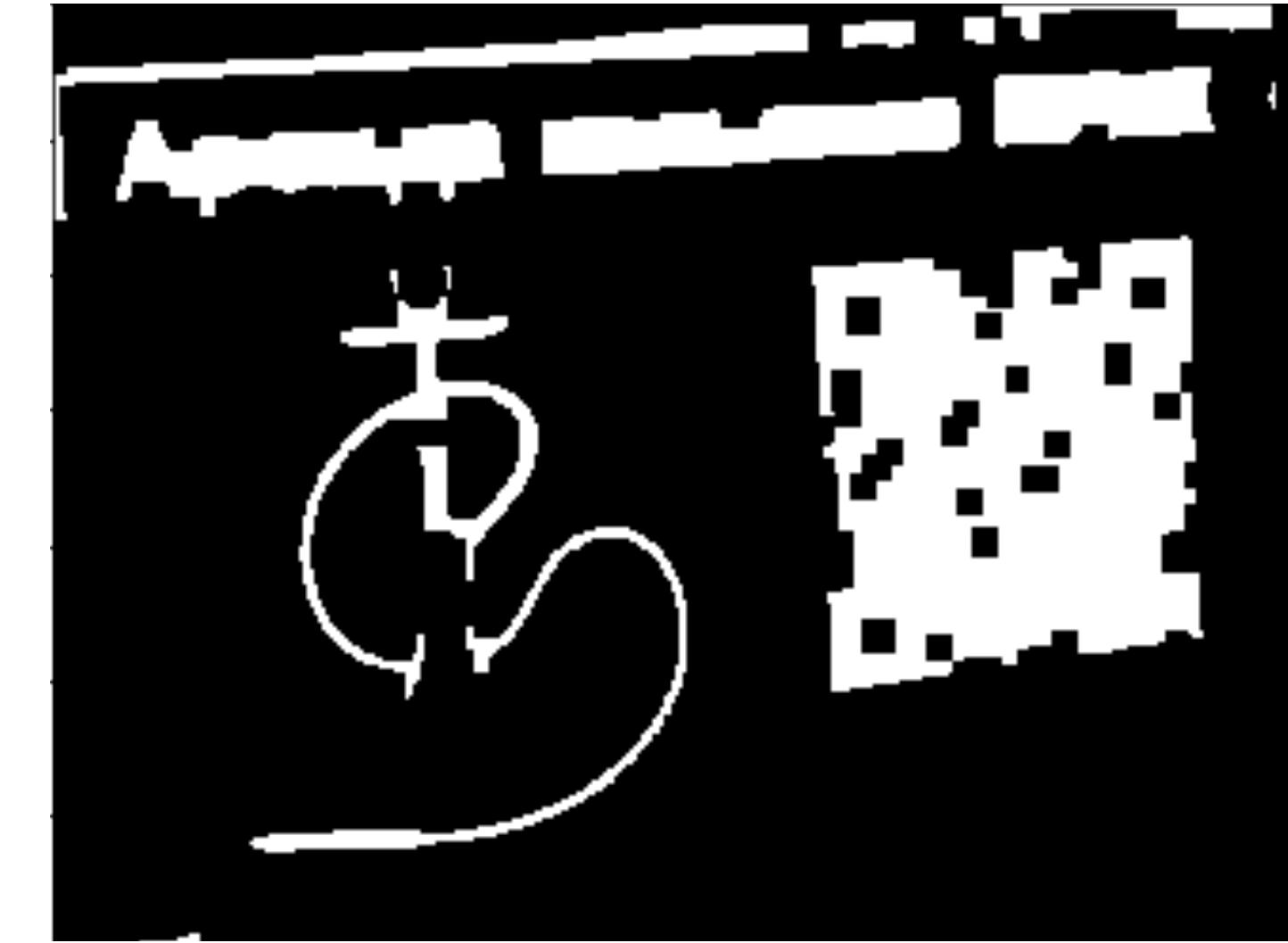
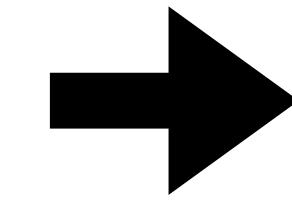
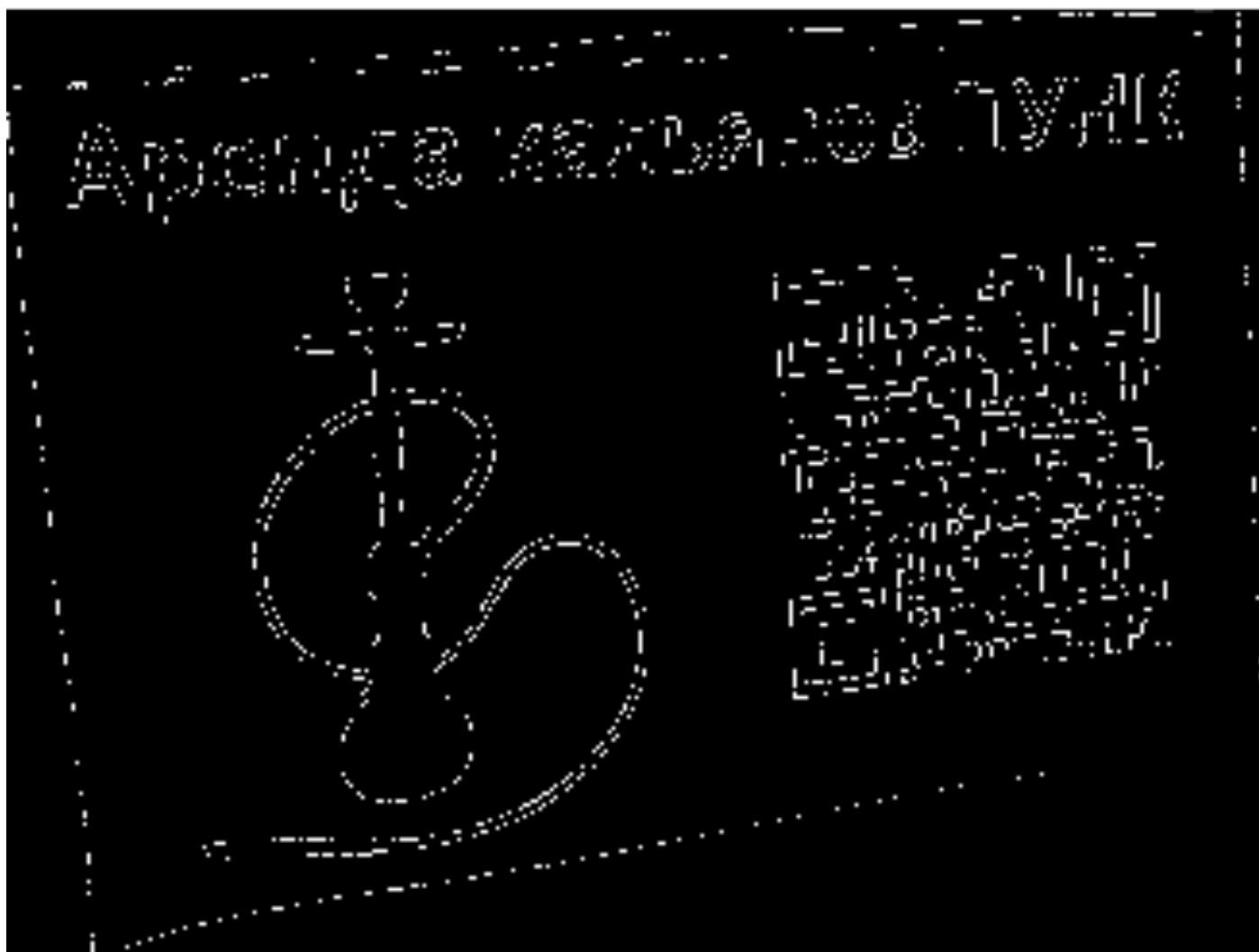
Бинаризация

Проводим бинаризацию размытого изображения. Каждый пиксель изображения со значением не выше 225 мы превратим в 0 (чёрный), а остальные — в 255 (белый).



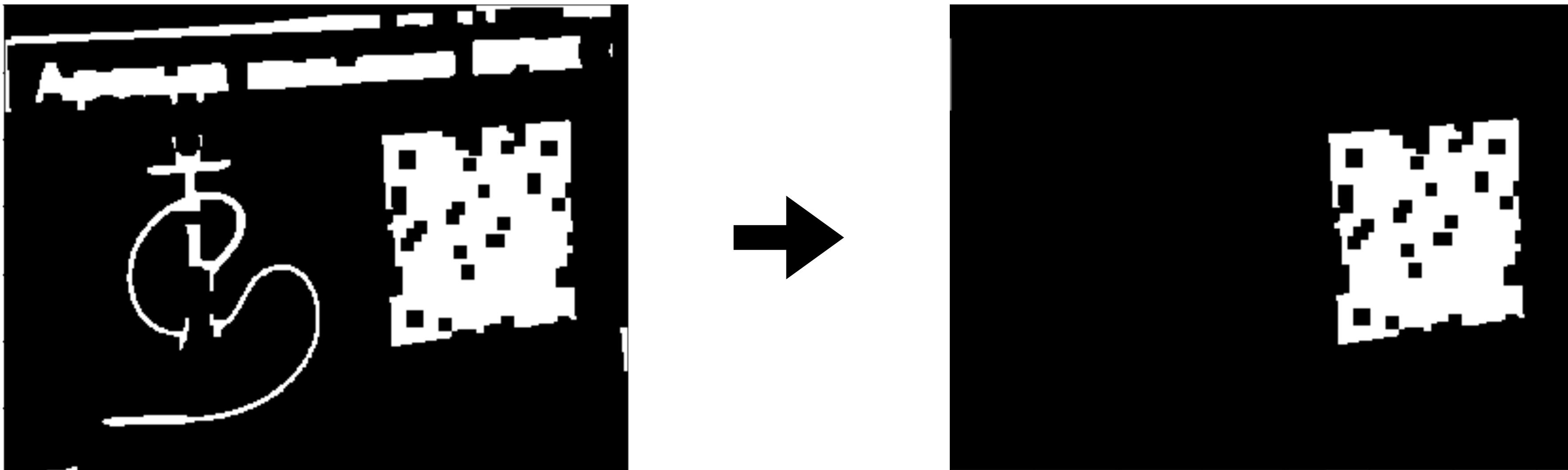
Проведение ряда простых морфологических операций

Произведем морфологическую операцию закрытие. Далее делаем 2 итерации эрозии, за которым следуют 2 итерации дилатации. Эрозия уберёт небольшие белые области, а дилатация не позволит крупным белым областям уменьшиться



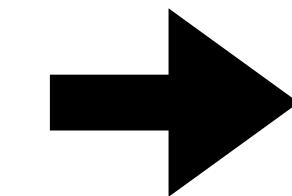
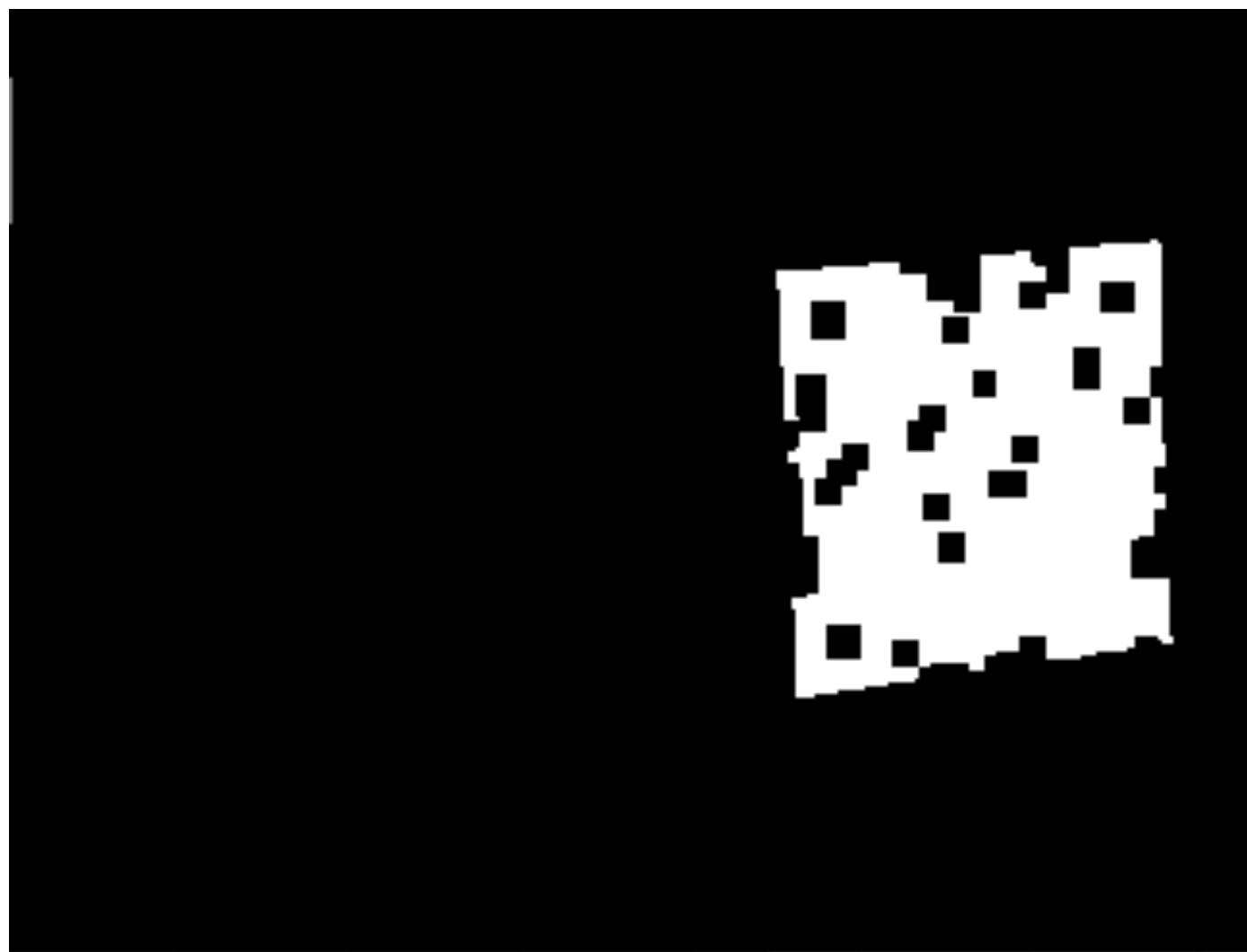
Нахождение самого большого контура на изображении

Находим самый большой контур на изображении с помощью cv2.findContours(), который (если обработка была произведена корректно) точно соотносится с областью штрихкода.



Определение минимального ограничивающего четырехугольника

С помощью cv2.minAreaRect() и cv2.BoxPoints() определяем минимальный ограничивающий прямоугольник, в который заключим самый большой контур. Вычисляем центр этого прямоугольника, определяем 2 самые удаленные от центра точки контура (верхнюю левую и нижнюю правую). Используя формулу для вычисления расстояния от точки до прямой, находим оставшиеся 2 точки.



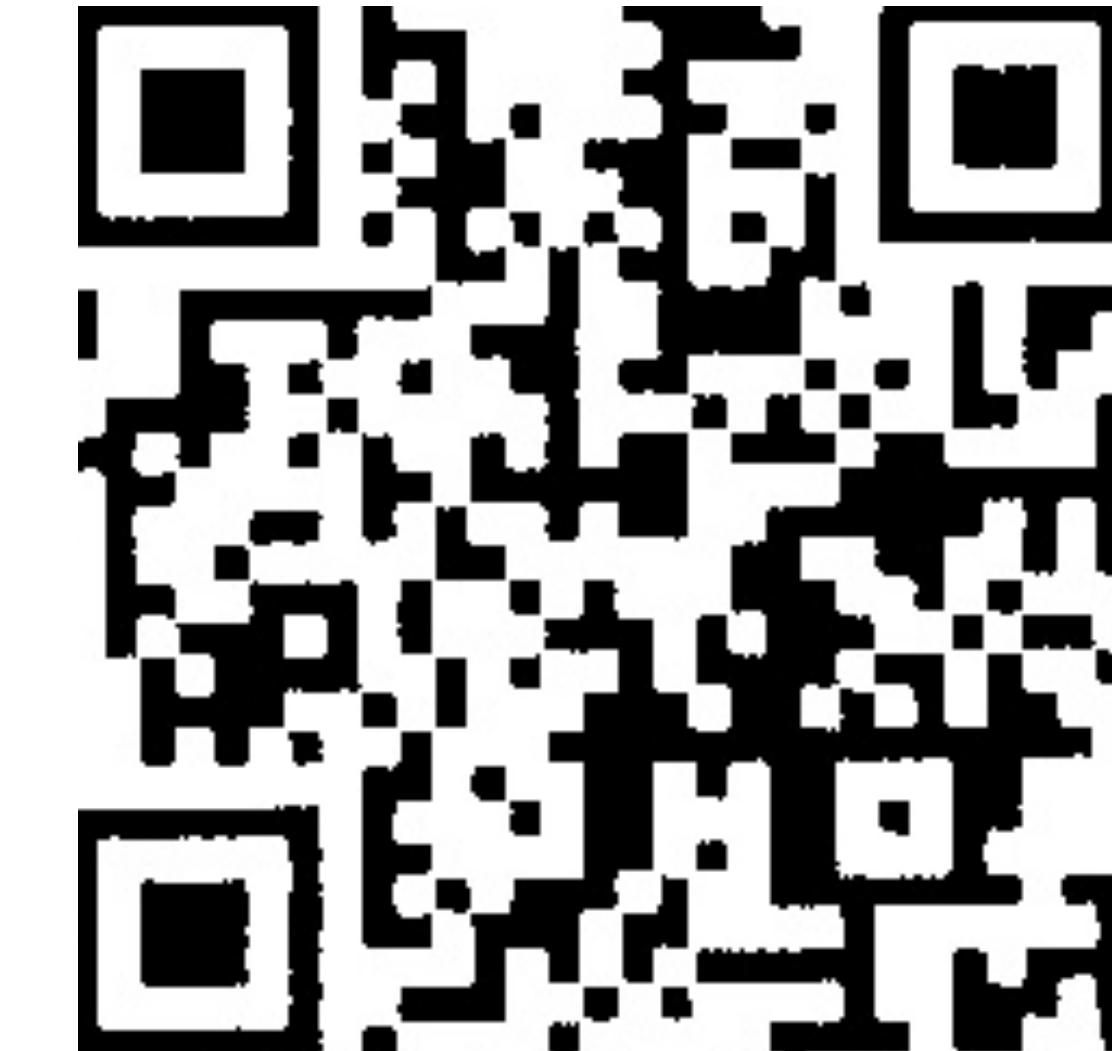
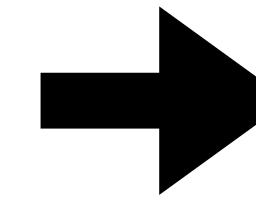
Кадрирование (+2%)

Увеличиваем периметр найденного четырехугольник на 2% и применяем перспективное преобразование изображения



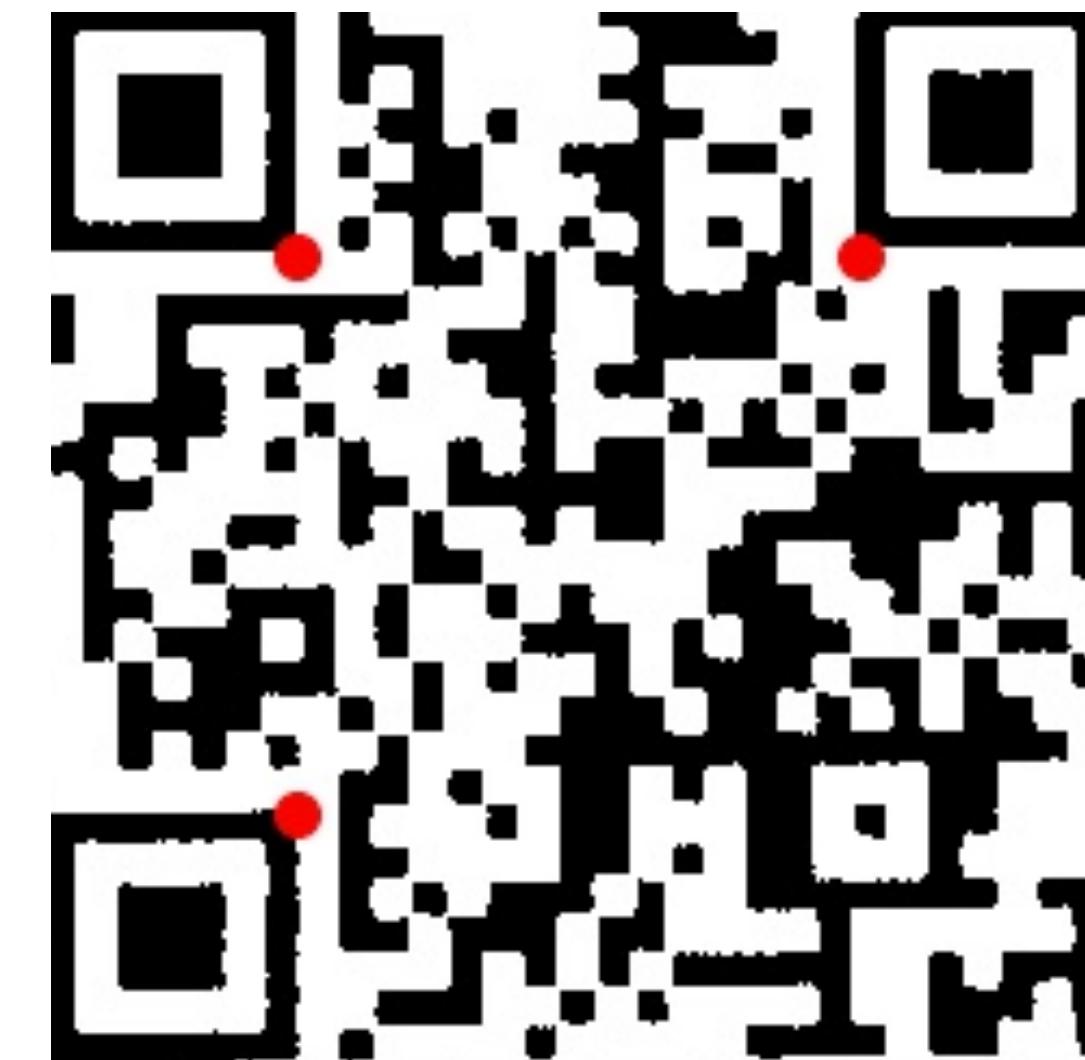
Уточнение границ

Бинаризуем кадрированное изображение и находим координаты черных пикселей, расположенных ближе всего к крайним точкам изображения. Координаты нижней правой точки дополнительно рассчитываются по формуле определения 4-ой точки параллелограмма (выбирается наиболее удаленная от центра). Снова применяем перспективное преобразование изображения.



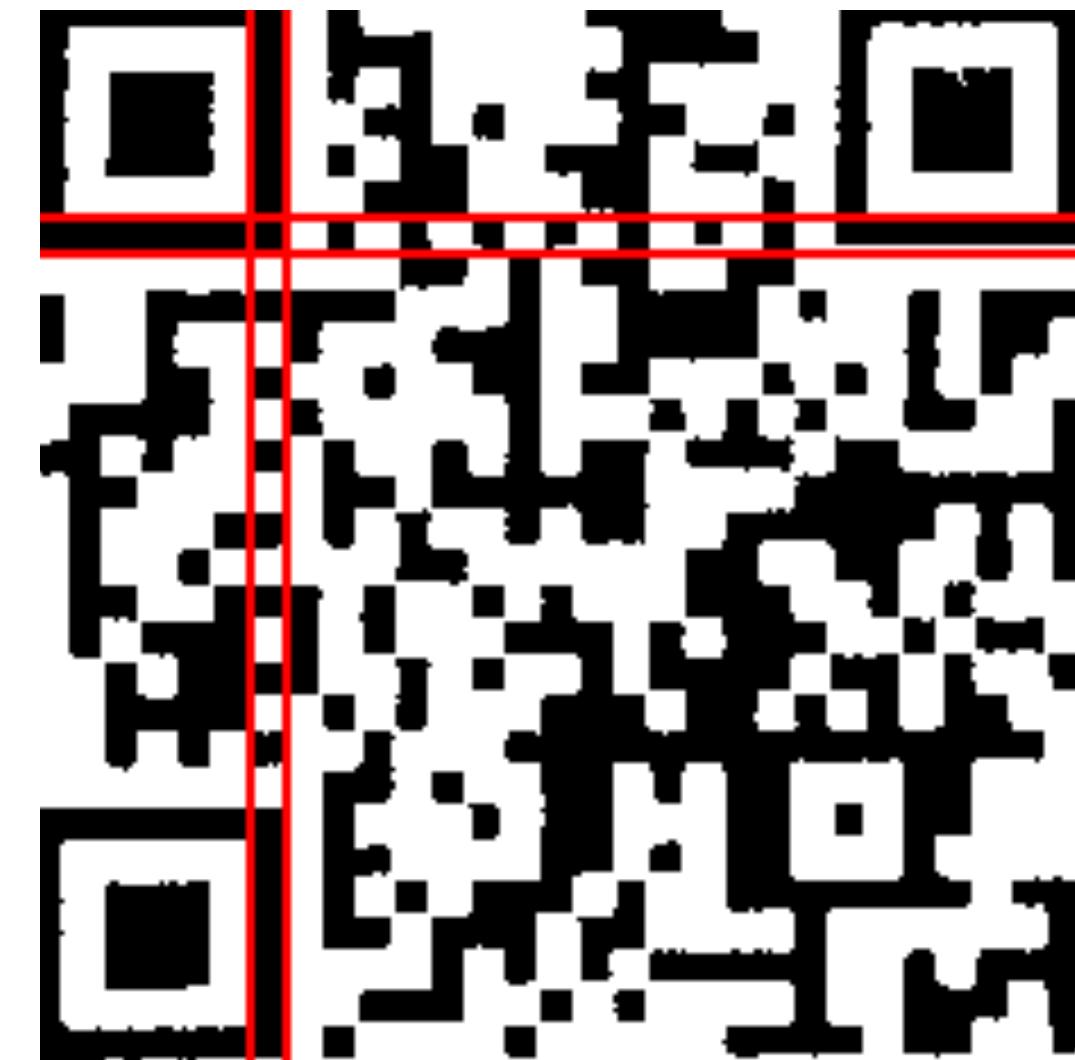
Обнаружение поисковых узоров

Находим координаты внутренних точек поисковых узоров. Для этого используется 6 разных алгоритмов и для их результатов считается среднее значение, это делается для того, чтобы если один алгоритм сработал неправильно, то за счет остальных неточность становится незначительной.



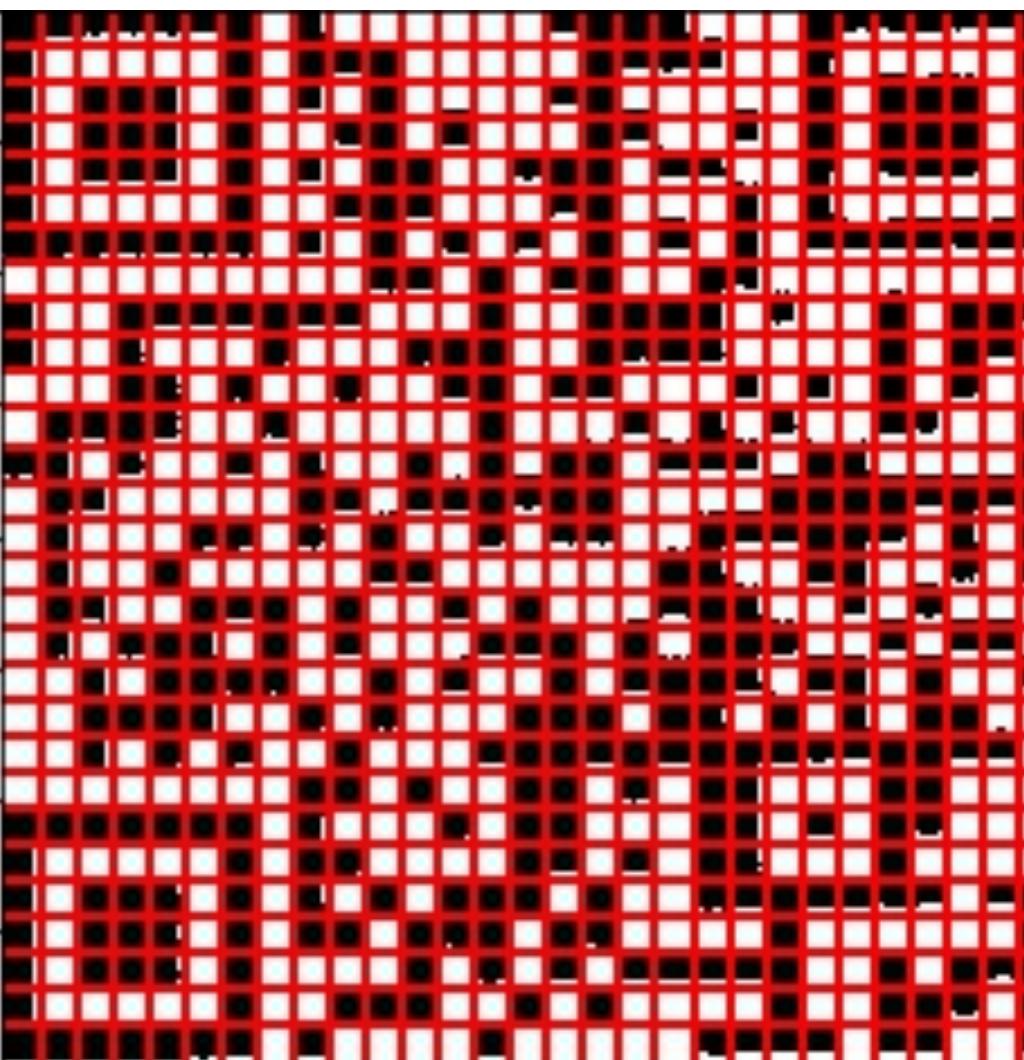
Вычисление размерности QR-кода по полосам синхронизации

С помощью ранее вычисленных координат поисковых узоров, находим расположение полос синхронизации и внутри находим самое частое число изменения цветов.



Наложение сетки

Поверх изображения строим сетку ранее вычисленной размерности



Определение среднего цвета в элементах сетки

Создаем нулевую квадратную матрицу размерности, соответствующей сетке. Внутри каждого малого квадрата сетки считаем среднее значение пикселей бинаризованного изображения и округляем его до ближайшего целого числа. Полученные значения сохраняем в соответствующих элементах нулевой матрицы. В итоге получаем квадратную матрицу состоящую из нулей и единиц, которую можно представить в виде бинарного изображения:



Другие примеры

