Работа 4. Детектирование границ документов на кадрах видео

автор: Дворядкин К.А. дата: 2022-03-21T04:53:11
url: GitHub

Задание

0. текст, иллюстрации и подписи отчета придумываем самостоятельно
1. самостоятельно снимаем видео смартфоном

- объект съемки - купюры (рубли разного номинала), расправленные и лежащие на поверхности (проективно искаженны прямоугольник)
- количество роликов - от 5 шт.
- длительность - 5-7 сек
- условия съемки разные

2. извлекаем по 3 кадра из каждого ролика (делим кол-во кадров на 5 и берем каждый с индеком 2/5,3/5,4/5)
3. цветоредуцируем изображения
4. бинаризуем изображения
5. морфологически обрабатываем изображения
6. выделяем основную компоненту связности
7. руками изготавливаем маски (идеальная зона купюры)
8. оцениваем качество выделение зоны и анализируем ошибки

Результаты

Рис. 1. Исходное изображение для Video1 Frame0

Рис. 2. Цветоредуцированное изображение

Рис. 3. Бинаризованное изображение

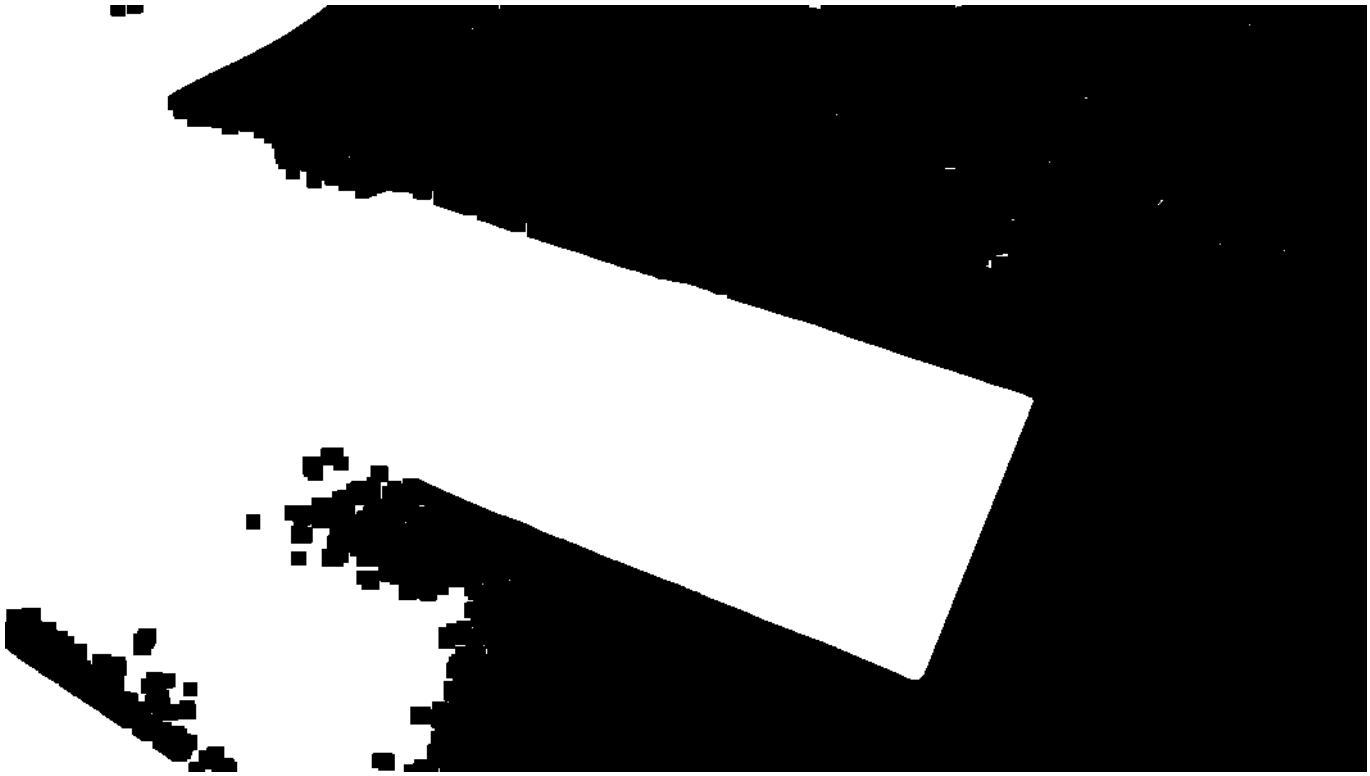

Рис. 4. Морфологически обработанное изображение

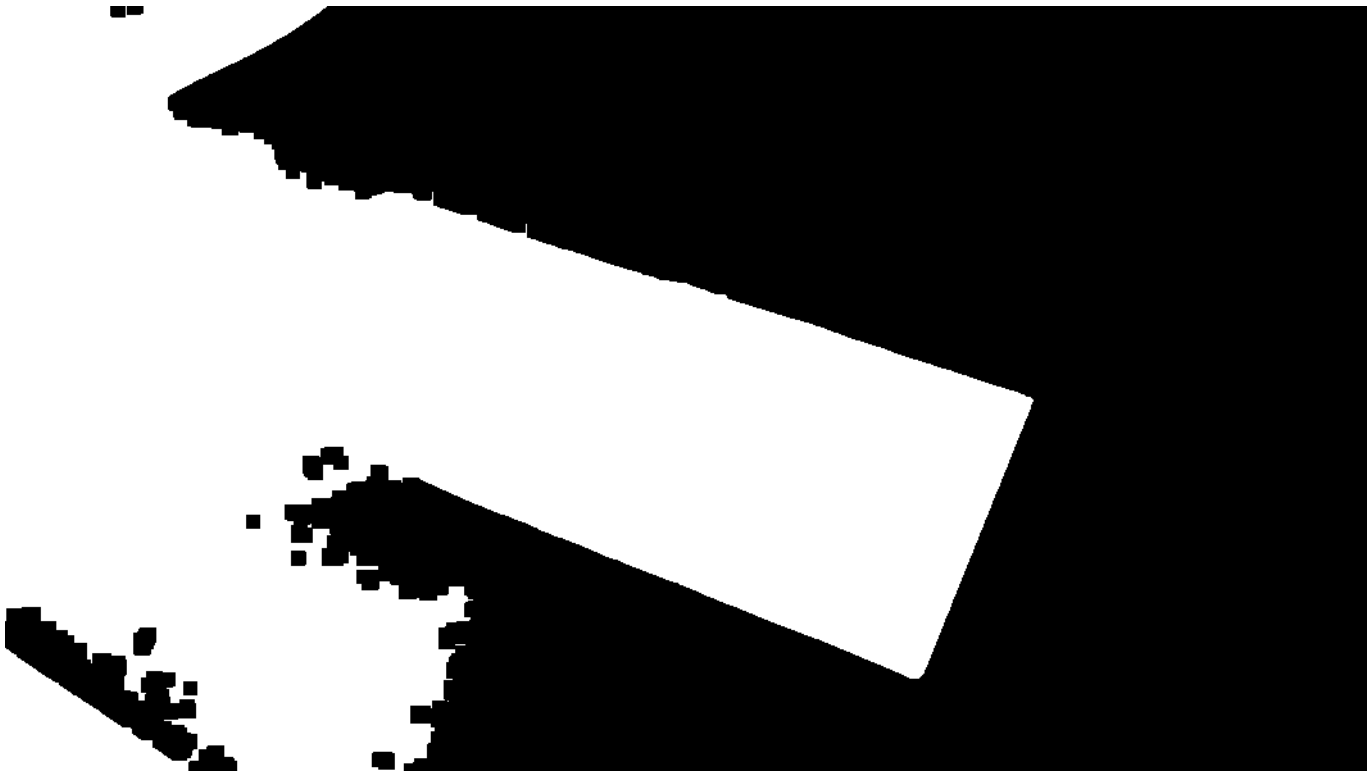

Рис. 5. Выделена основная компонента связности

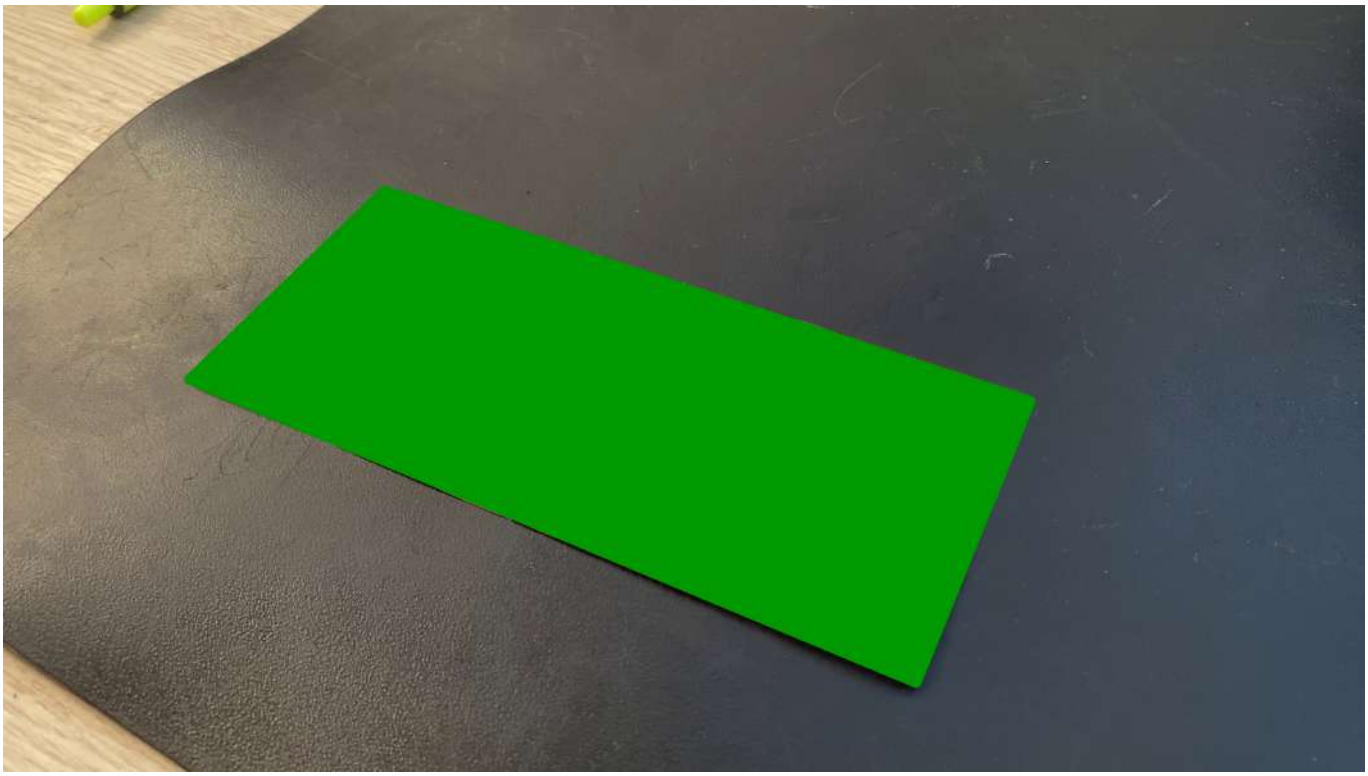

Рис. 6. Наложенная маска

Рис. 7. Исходное изображение для Video3 Frame1

Рис. 8. Цветоредуцированное изображение

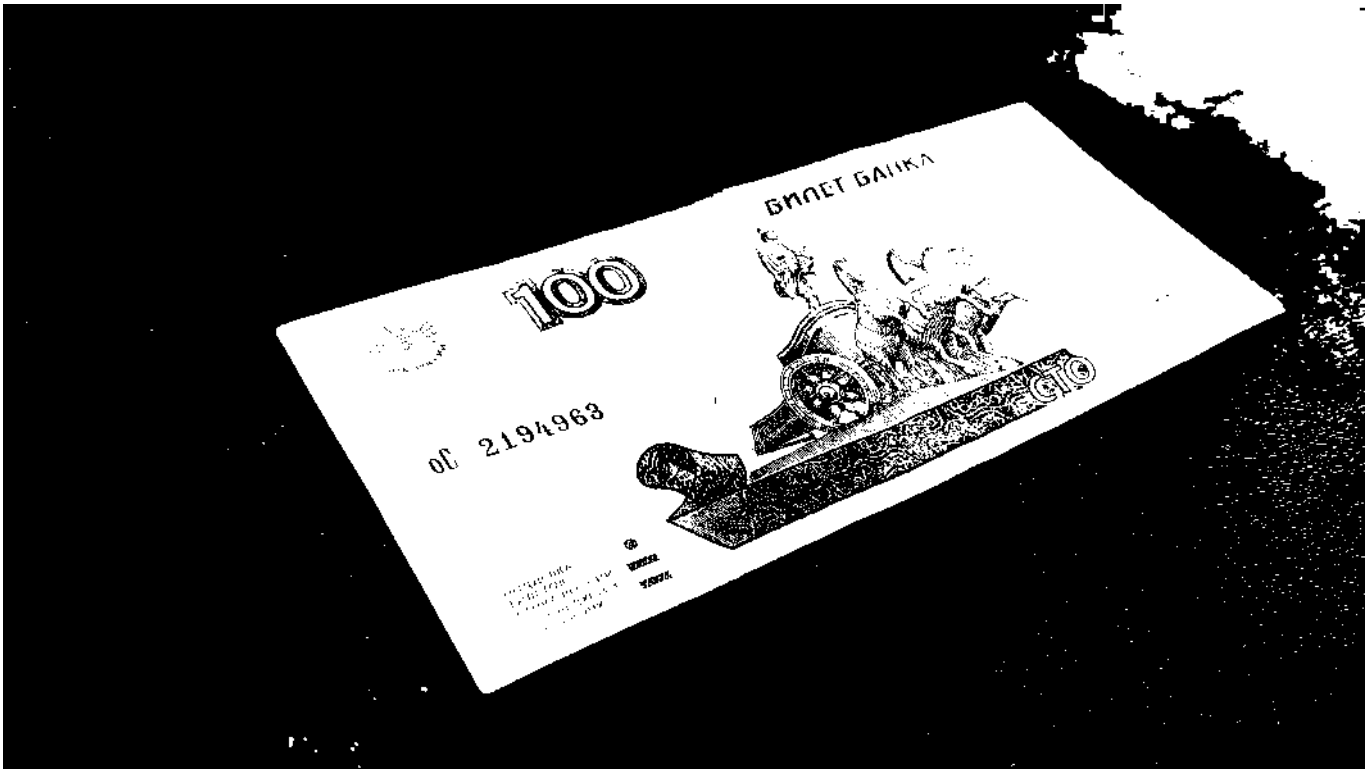

Рис. 9. Бинаризованное изображение

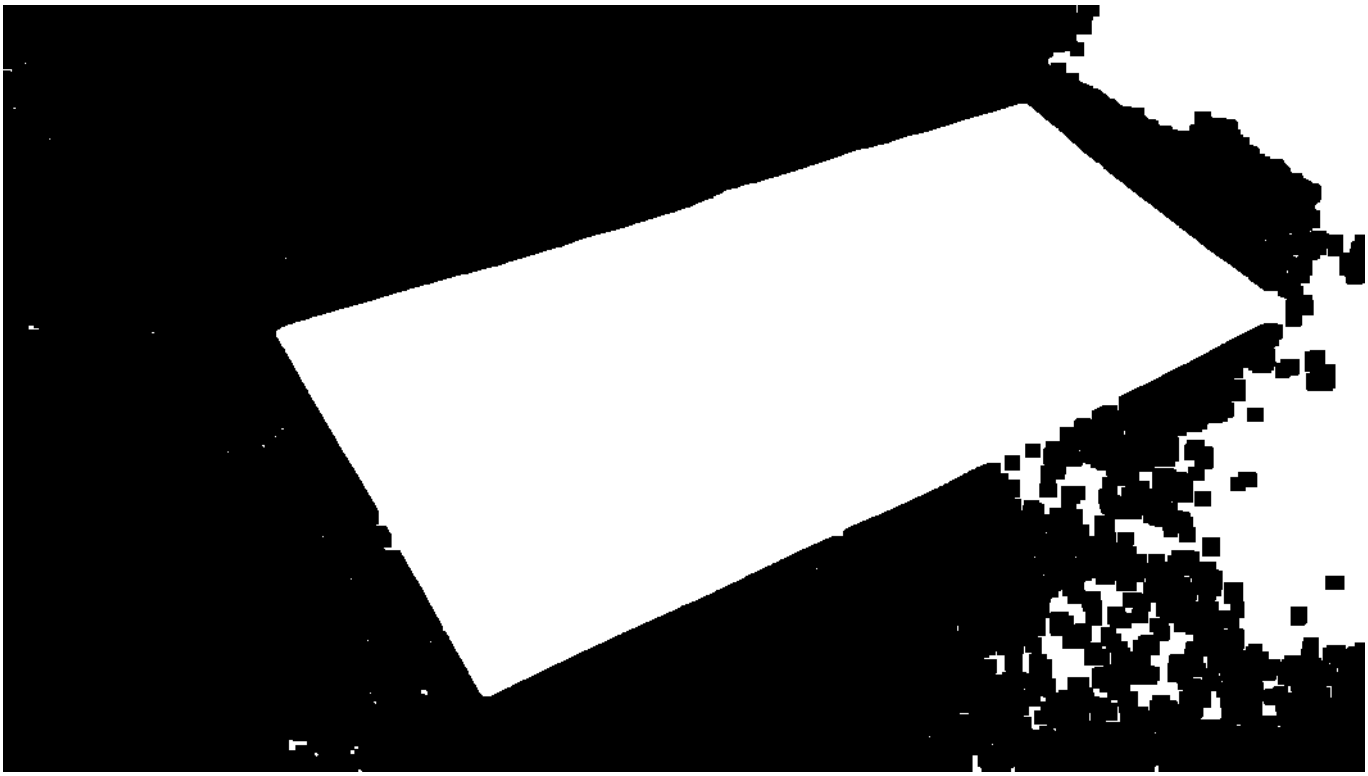

Рис. 10. Морфологически обработанное изображение

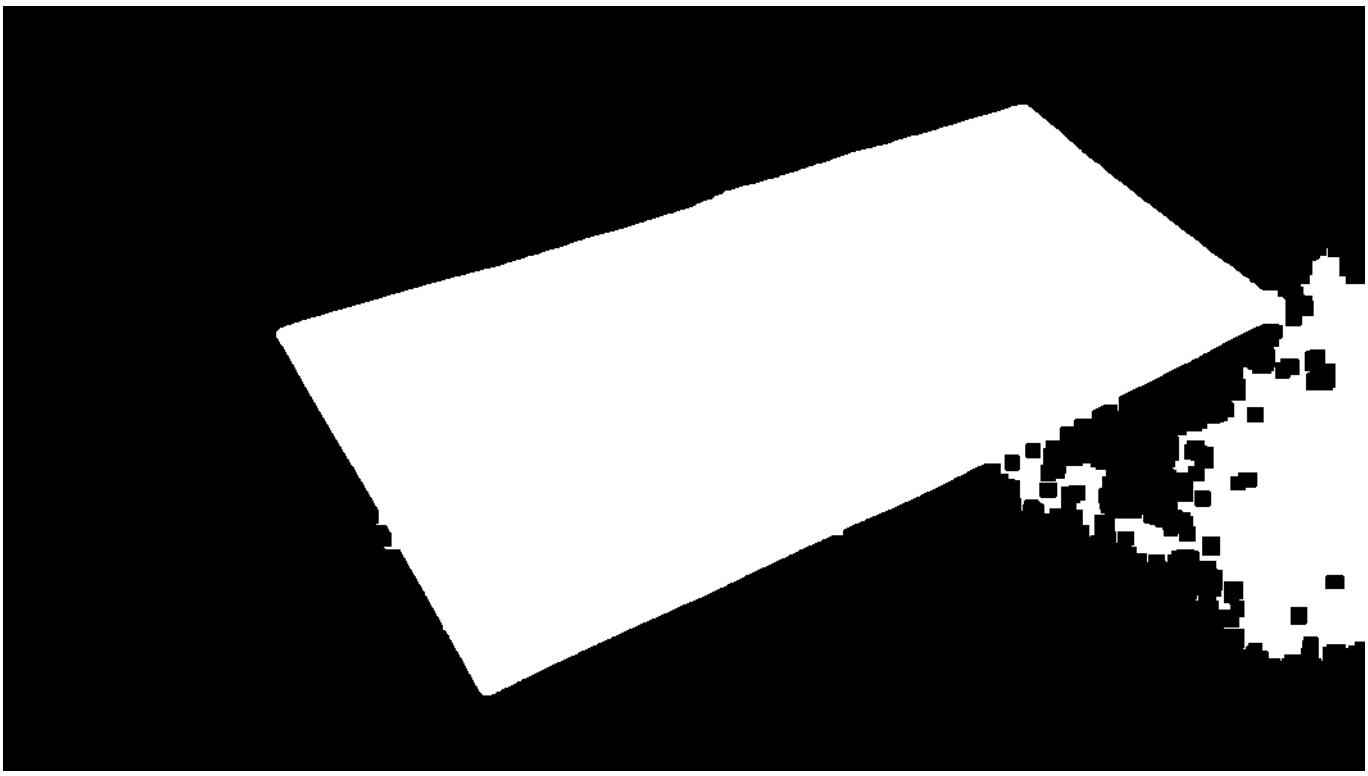

Рис. 11. Выделена основная компонента связности

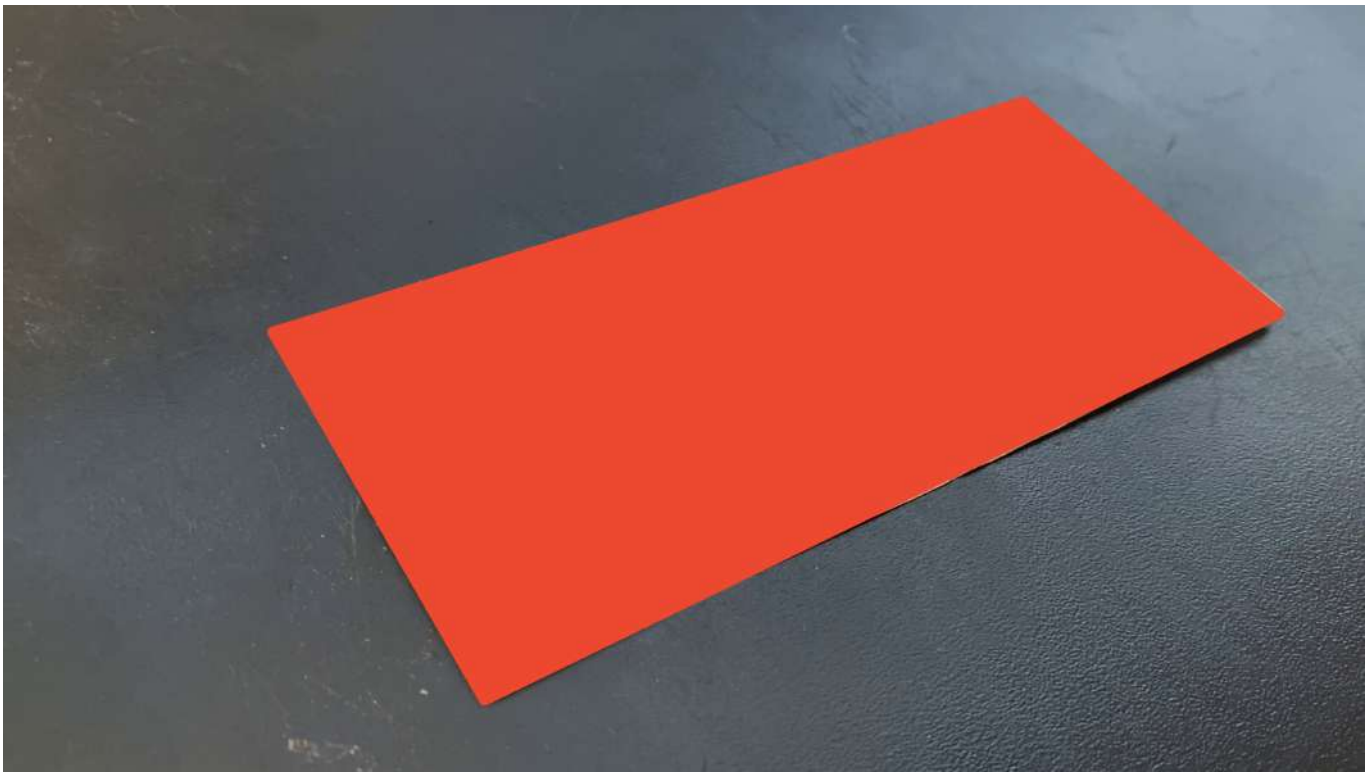

Рис. 12. Наложенная маска

Рис. 13. Исходное изображение для Video5 Frame2

Рис. 14. Цветоредуцированное изображение

Рис. 15. Бинаризованное изображение

Рис. 16. Морфологически обработанное изображение

Рис. 17. Выделена основная компонента связности

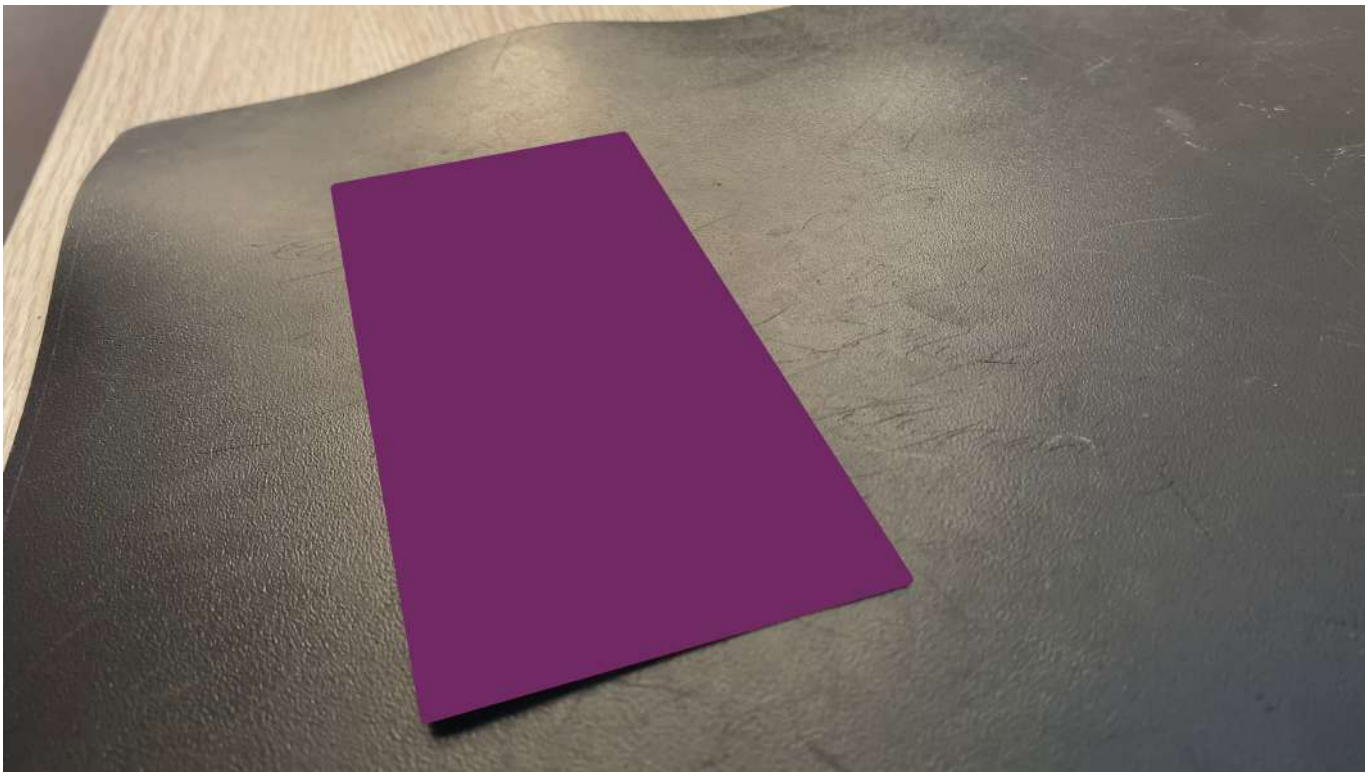

Рис. 18. Наложенная маска

Рис. 19. Исходное изображение для Video2 Frame0

Рис. 20. Цветоредуцированное изображение

Рис. 21. Бинаризованное изображение

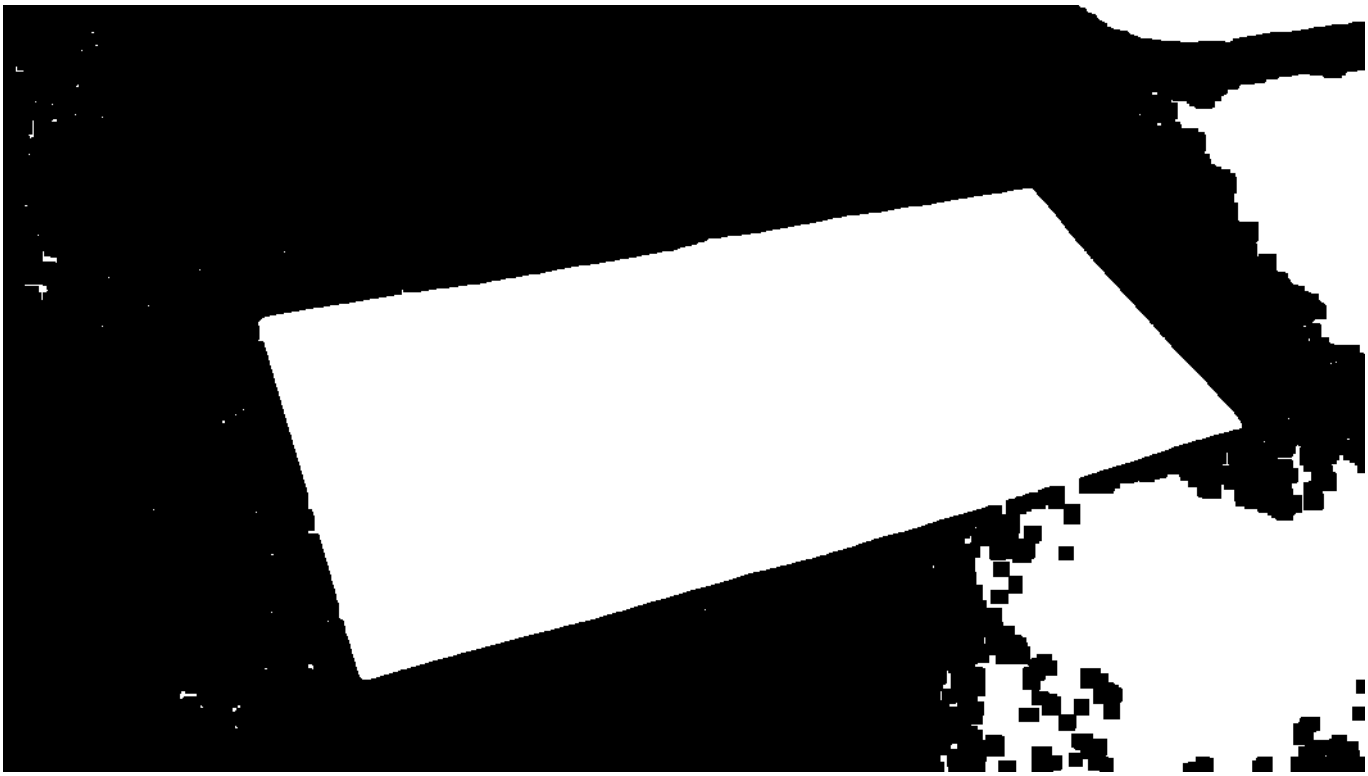

Рис. 22. Морфологически обработанное изображение

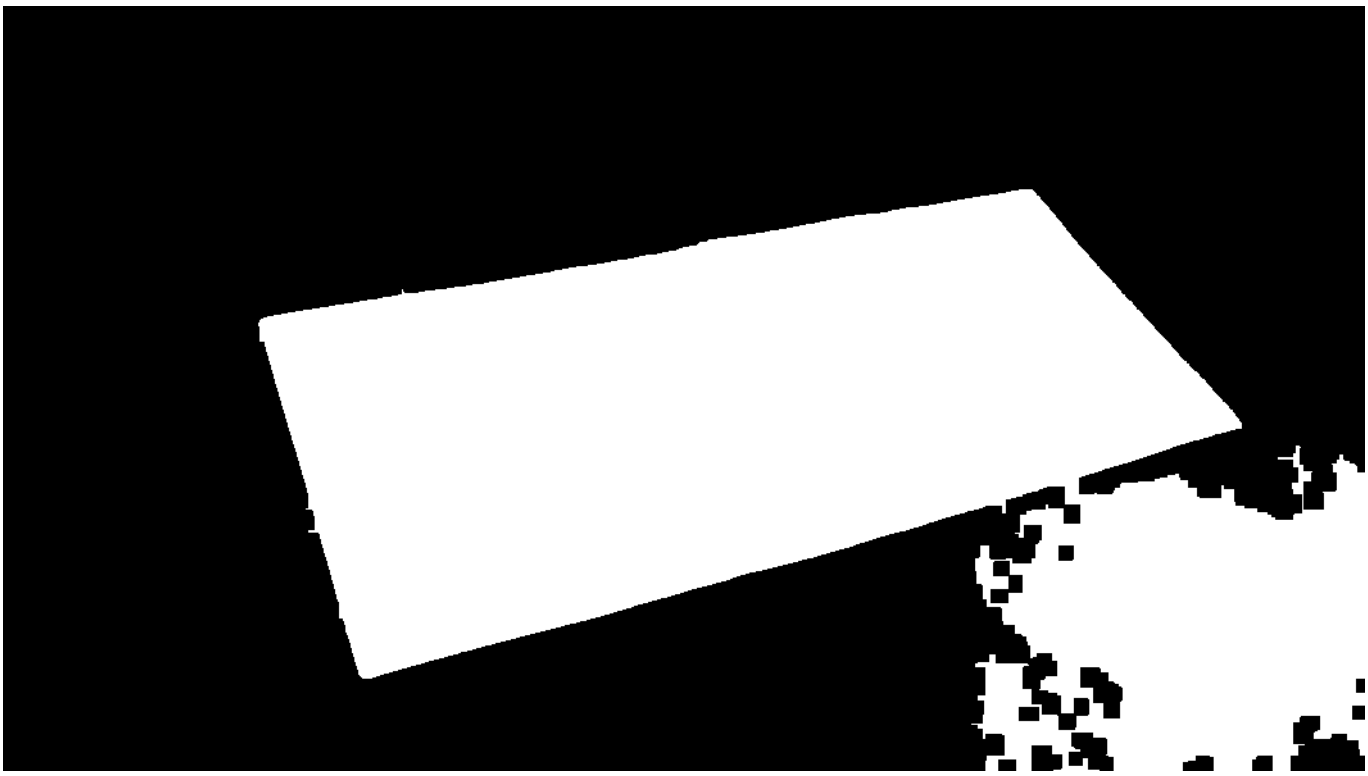

Рис. 23. Выделена основная компонента связности

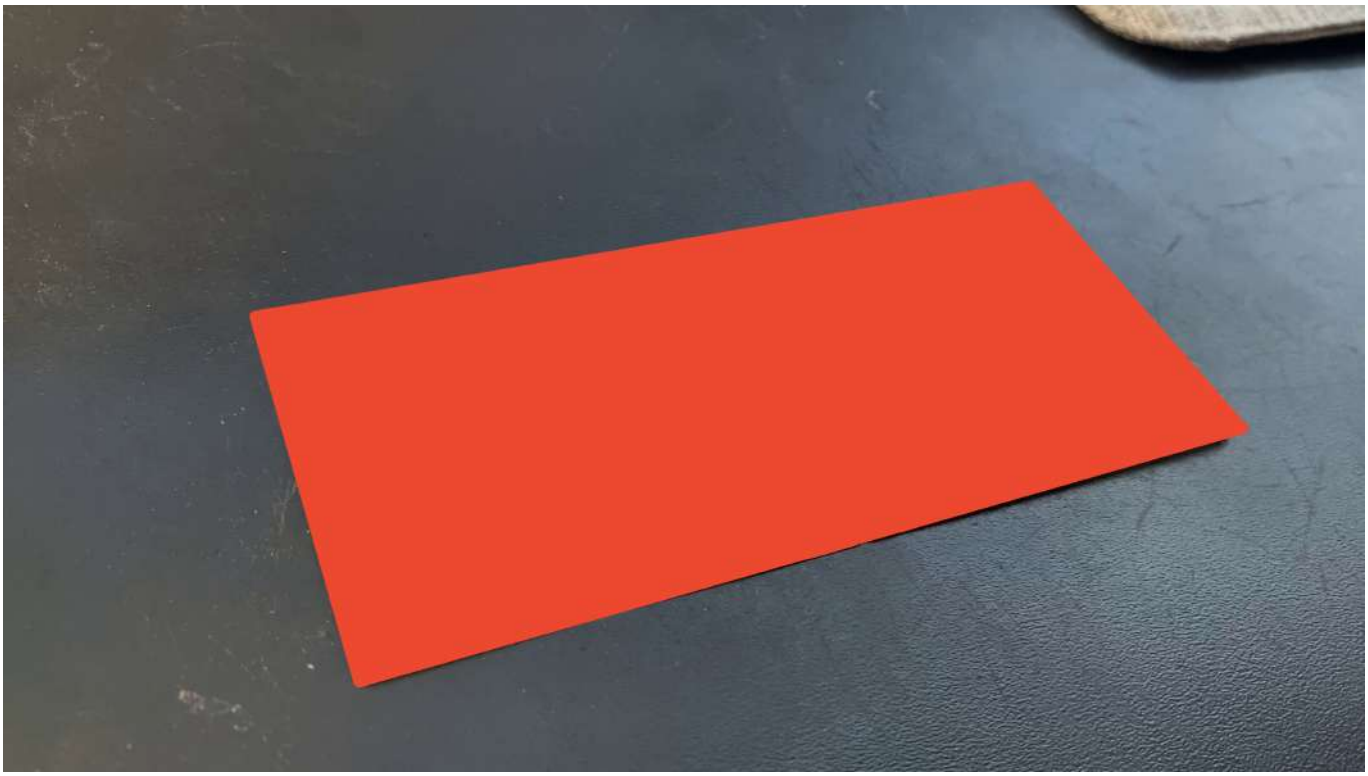

Рис. 24. Наложенная маска

Рис. 25. Исходное изображение для Video4 Frame2

Рис. 26. Цветоредуцированное изображение

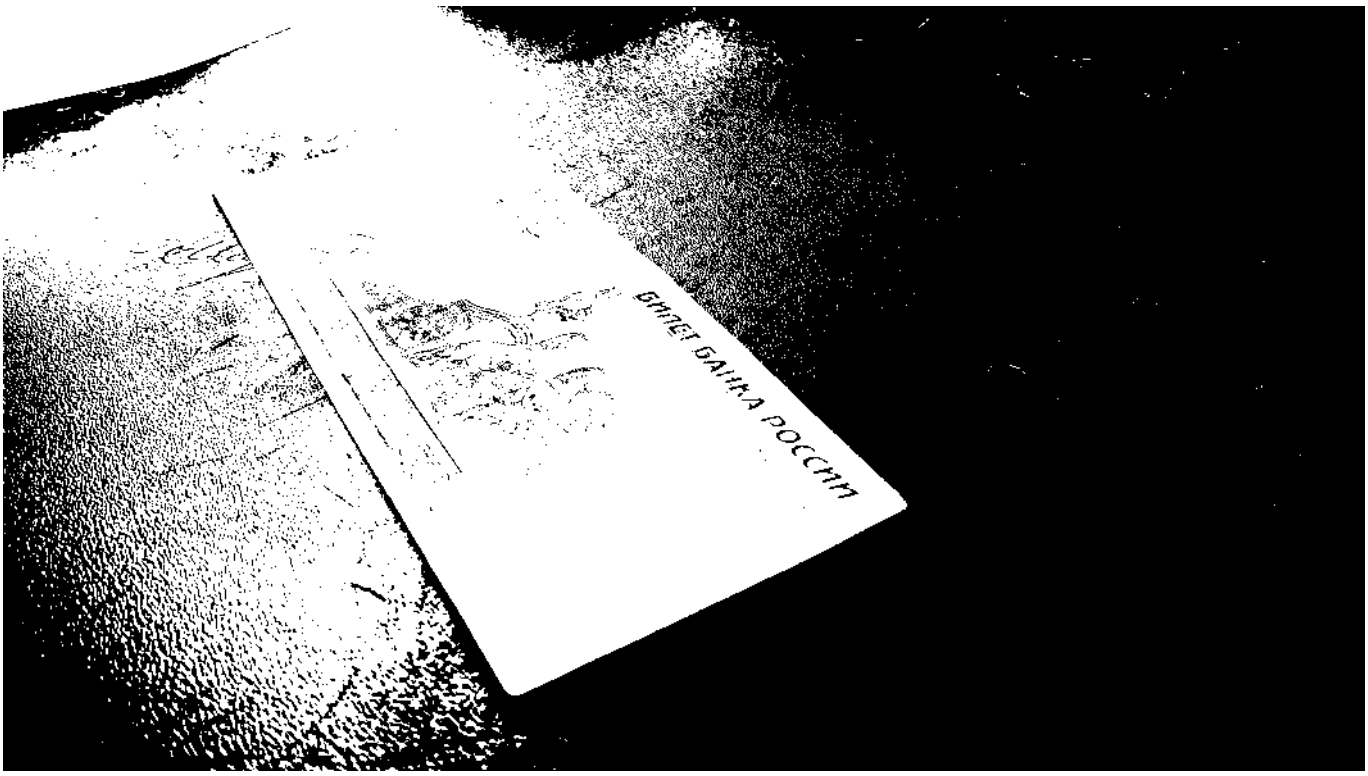

Рис. 27. Бинаризованное изображение

Рис. 28. Морфологически обработанное изображение

Рис. 29. Выделена основная компонента связности

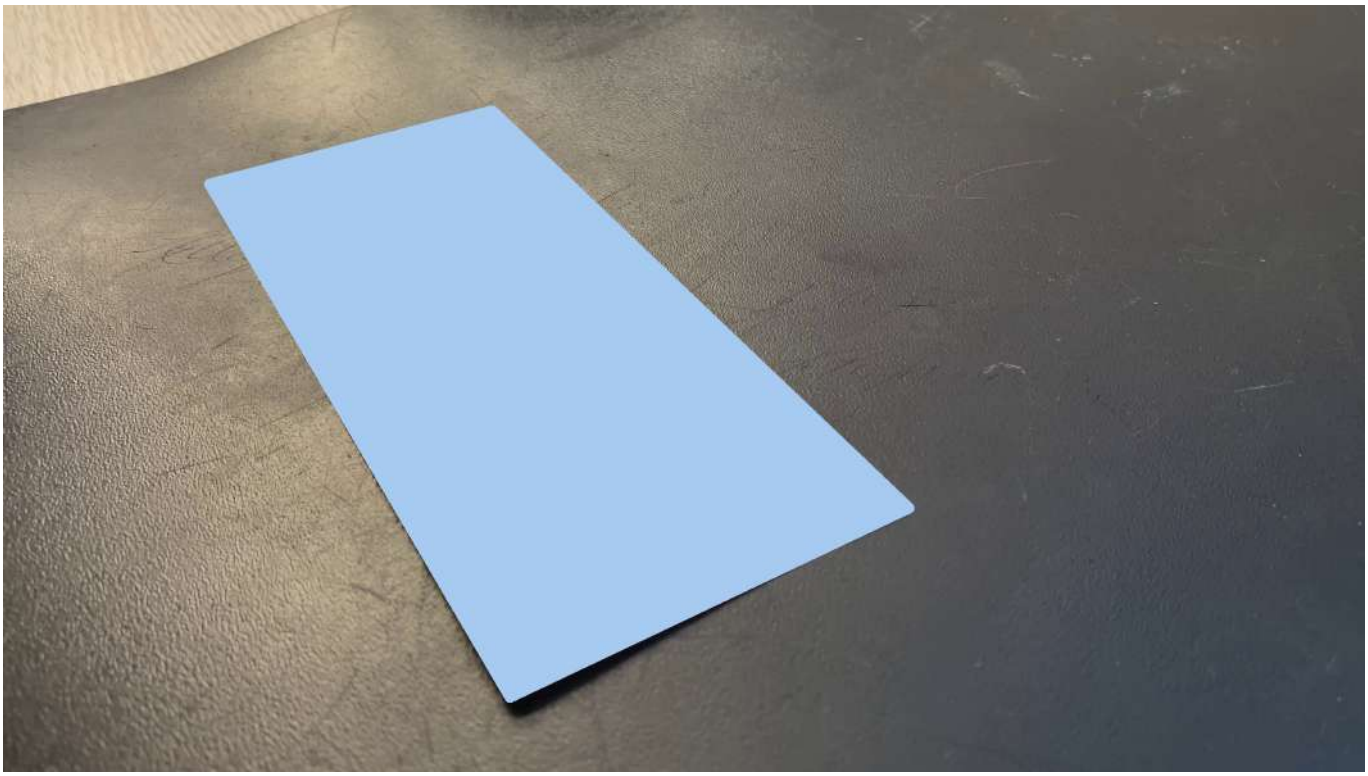

Рис. 30. Наложенная маска

В результате проделанной работы справдлив вывод, что на точность выделения компонент связности рассматриваемого объекта сильно влияет освещение фона, на котором расположен наш объект.

Текст программы

```
#include <opencv2/opencv.hpp>
#include <iostream>

std::vector<cv::Mat> split3(cv::VideoCapture& video, int v){
    std::vector<cv::Mat> res;
    res.resize(3);

    int countf = video.get(cv::CAP_PROP_FRAME_COUNT);
    for(int i = 0; i < countf; i++){
        if(i == (countf / 5) * 2){
            video >> res[0];
        }
        if(i == (countf / 5) * 3){
            video >> res[1];
        }
        if(i == (countf / 5) * 4){
            video >> res[2];
        }
        video.grab();
    }

    for (int i = 0; i < 3; i++){
        cv::imwrite("output/video" + std::to_string(v) + "_frame" +
```

```
std::to_string(i) + ".png", res[i]);
    }

    return res;
}

void clReduce(cv::Mat& img, int div=64){
        int lines = img.rows;
        int objs = img.cols * img.channels();

        for(int i = 0; i < lines; i++){
            uchar* data = img.ptr<uchar>(i);
            for(int j = 0; j < objs; j++){
                data[j] = data[j] / div * div + div / 2;
            }
        }
    }

void binarisation(cv::Mat& img, uchar val){
    cv::cvtColor(img, img, cv::COLOR_BGR2GRAY);
    cv::threshold(img, img, val, 255, cv::THRESH_BINARY);
}

int main() {

    // 1. самостоятельно снимаем видео смартфоном
    // - объект съемки - купюры (рубли разного номинала), расправленные и
лежащие на поверхности (проективно искаженны прямоугольник)
    // - количество роликов - от 5 шт.
    // - длительность - 5-7 сек
    // - условия съемки разные

    for (int v = 1; v <= 5; v++){
    cv::VideoCapture cap("./data/video" + std::to_string(v) + ".MOV");

    if(!cap.isOpened()){
        std::cout << "Error opening video stream or file" << std::endl;
        return -1;
    }

    // 2. извлекаем по 3 кадра из каждого ролика (делим кол-во кадров на 5
и берем каждый с индеком 2/5,3/5,4/5)

    std::vector<cv::Mat> frames = split3(cap, v);

    // 3. цветоредуцируем изображения

    for(int i = 0; i < 3; i++){
        clReduce(frames[i]);
        cv::imwrite("output/video" + std::to_string(v) + "_frame" +
std::to_string(i) + "_cr.png", frames[i]);
    }

    // 4. бинаризуем изображения
```

```
    for(int i = 0; i < 3; i++){
        binarisation(frames[i], 150);
        cv::imwrite("output/video" + std::to_string(v) + "_frame" +
std::to_string(i) + "_binar.png", frames[i]);
    }

    // 5. морфологически обрабатываем изображения

    // cv::Mat Mask;
    cv::Mat kernel = cv::getStructuringElement(cv::MORPH_RECT,
cv::Size(40, 40));
    for(int i = 0; i < 3; i++){
        // kernel = cv::getStructuringElement(cv::MORPH_RECT, cv::Size(40,
40));
        cv::morphologyEx(frames[i], frames[i], cv::MORPH_CLOSE, kernel);
        // cv::morphologyEx(frames[i], frames[i], cv::MORPH_OPEN, kernel);
        cv::imwrite("output/video" + std::to_string(v) + "_frame" +
std::to_string(i) + "_morph.png", frames[i]);
    }

    // 6. выделяем основную компоненту связности

    for(int i = 0; i < 3; i++){
        // cws(frames[i]);
        // cv::imwrite("output/frame" + std::to_string(i) + "_comps.png",
frames[i]);
        cv::Mat img = frames[i];
        cv::Mat labels, stats, centroids;
        int cws = cv::connectedComponentsWithStats(img, labels, stats,
centroids);
        int max=0, imax=0;
        for(int i = 1; i < cws; i++){
        if(stats.at<int>(i, cv::CC_STAT_AREA) > max){
            max = stats.at<int>(i, cv::CC_STAT_AREA);
            imax = i;
            }
        }
        // std::cout << max << " + " << imax << std::endl;
        std::vector<cv::Vec3b> handle(cws);
        for(int i = 0; i < cws; i++){
            handle[i] = cv::Vec3b(0, 0, 0);
        }
        handle[imax] = cv::Vec3b(255, 255, 255);
        cv::Mat res(img.rows, img.cols, CV_8UC3);
        for(int i = 0; i < res.rows; i++){
            for(int j = 0; j < res.cols; j++){
                int label = labels.at<int>(i, j);
                cv::Vec3b &pixel = res.at<cv::Vec3b>(i, j);
                pixel = handle[label];
            }
        }
        cv::imwrite("output/video" + std::to_string(v) + "_frame" +
std::to_string(i) + "_comps.png", res);
```

```
    }
    cap.release();
    }

    // 7. руками изготавливаем маски (идеальная зона купюры)

    std::vector<cv::Point> pts;

    //video1
    //frame0
    cv::Mat im = cv::imread("output/video1_frame0.png");
    pts.push_back(cv::Point(524,1054));
    cv::line(im, cv::Point(524,1054), cv::Point(1075,522), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(1075,522));
    cv::line(im, cv::Point(1075,522), cv::Point(2896,1115), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(2896,1115));
    cv::line(im, cv::Point(2896,1115), cv::Point(2574,1908), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(2574,1908));
    cv::line(im, cv::Point(2574,1908), cv::Point(524,1054), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    cv::fillPoly( im, pts, cv::Scalar(0, 153, 0) );
    cv::imwrite("output/video1_frame0_masked.png", im);
    pts.clear();

    //video1
    //frame1
    im = cv::imread("output/video1_frame1.png");
    pts.push_back(cv::Point(454,856));
    cv::line(im, cv::Point(454,856), cv::Point(1078,376), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(1078,376));
    cv::line(im, cv::Point(1078,376), cv::Point(2850,1065), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(2850,1065));
    cv::line(im, cv::Point(2850,1065), cv::Point(2388,1844), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(2388,1844));
    cv::line(im, cv::Point(2388,1844), cv::Point(454,856), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    cv::fillPoly( im, pts, cv::Scalar(0, 153, 0) );
    cv::imwrite("output/video1_frame1_masked.png", im);
    pts.clear();

    //video1
    //frame2
    im = cv::imread("output/video1_frame2.png");
    pts.push_back(cv::Point(530,778));
    cv::line(im, cv::Point(530,778), cv::Point(2414,1996), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(2414,1996));
    cv::line(im, cv::Point(2414,1996), cv::Point(3019,1181), cv::Scalar(0,
```

```
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(3019,1181));
    cv::line(im, cv::Point(3019,1181), cv::Point(1231,345), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    pts.push_back(cv::Point(1231,345));
    cv::line(im, cv::Point(1231,345), cv::Point(530,778), cv::Scalar(0,
153, 0), 28, cv::LINE_4);
    cv::fillPoly( im, pts, cv::Scalar(0, 153, 0) );
    cv::imwrite("output/video1_frame2_masked.png", im);
    pts.clear();

    //video2
    //frame0
    im = cv::imread("output/video2_frame0.png");
    pts.push_back(cv::Point(708,875));
    cv::line(im, cv::Point(708,875), cv::Point(2893,508), cv::Scalar(48,
72, 237), 28, cv::LINE_4);
    pts.push_back(cv::Point(2893,508));
    cv::line(im, cv::Point(2893,508), cv::Point(3496,1192), cv::Scalar(48,
72, 237), 28, cv::LINE_4);
    pts.push_back(cv::Point(3496,1192));
    cv::line(im, cv::Point(3496,1192), cv::Point(1002,1907),
cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    pts.push_back(cv::Point(1002,1907));
    cv::line(im, cv::Point(1002,1907), cv::Point(708,875), cv::Scalar(48,
72, 237), 28, cv::LINE_4);
    cv::fillPoly( im, pts, cv::Scalar(48, 72, 237) );
    cv::imwrite("output/video2_frame0_masked.png", im);
    pts.clear();

    cv::Point p1, p2, p3, p4;
    //video2
    //frame1
    im = cv::imread("output/video2_frame1.png");
    p1 = {758,920};
    p2 = {2876,271};
    p3 = {3598,868};
    p4 = {1356,1958};
    pts.push_back(p1);
    cv::line(im, p1, p2, cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    pts.push_back(p4);
    cv::line(im, p4, p1, cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    cv::fillPoly( im, pts, cv::Scalar(48, 72, 237) );
    cv::imwrite("output/video2_frame1_masked.png", im);
    pts.clear();

    //video2
    //frame2
    im = cv::imread("output/video2_frame2.png");
    p1 = {1217,1994};
```

```
    p2 = {554,892};
    p3 = {2750,184};
    p4 = {3504,786};
    pts.push_back(p1);
    cv::line(im, p1, p2, cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    pts.push_back(p4);
    cv::line(im, p4, p1, cv::Scalar(48, 72, 237), 28, cv::LINE_4);
    cv::fillPoly( im, pts, cv::Scalar(48, 72, 237) );
    cv::imwrite("output/video2_frame2_masked.png", im);
    pts.clear();

    cv::Scalar clr = {99, 41, 112};

    //video3
    //frame0
    im = cv::imread("output/video3_frame0.png");
    p1 = {929,463};
    p2 = {1724,320};
    p3 = {2555,1688};
    p4 = {1303,2063};
    pts.push_back(p1);
    cv::line(im, p1, p2, clr, 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, clr, 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, clr, 28, cv::LINE_4);
    pts.push_back(p4);
    cv::line(im, p4, p1, clr, 28, cv::LINE_4);
    cv::fillPoly( im, pts, clr );
    cv::imwrite("output/video3_frame0_masked.png", im);
    pts.clear();

    //video3
    //frame1
    im = cv::imread("output/video3_frame1.png");
    p1 = {986,538};
    p2 = {1770,407};
    p3 = {2558,1676};
    p4 = {1253,2041};
    pts.push_back(p1);
    cv::line(im, p1, p2, clr, 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, clr, 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, clr, 28, cv::LINE_4);
    pts.push_back(p4);
    cv::line(im, p4, p1, clr, 28, cv::LINE_4);
    cv::fillPoly( im, pts, clr );
    cv::imwrite("output/video3_frame1_masked.png", im);
    pts.clear();
```

```
    //video3
    //frame2
    im = cv::imread("output/video3_frame2.png");
    p1 = {937,516};
    p2 = {1748,369};
    p3 = {2552,1617};
    p4 = {1190,2009};
    pts.push_back(p1);
    cv::line(im, p1, p2, clr, 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, clr, 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, clr, 28, cv::LINE_4);
    pts.push_back(p4);
    cv::line(im, p4, p1, clr, 28, cv::LINE_4);
    cv::fillPoly( im, pts, clr );
    cv::imwrite("output/video3_frame2_masked.png", im);
    pts.clear();

    clr = {240, 202, 166};

    //video4
    //frame0
    im = cv::imread("output/video4_frame0.png");
    p1 = {788,587};
    p2 = {1549,363};
    p3 = {2782,1379};
    p4 = {1718,1926};
    pts.push_back(p1);
    cv::line(im, p1, p2, clr, 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, clr, 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, clr, 28, cv::LINE_4);
    pts.push_back(p4);
    cv::line(im, p4, p1, clr, 28, cv::LINE_4);
    cv::fillPoly( im, pts, clr );
    cv::imwrite("output/video4_frame0_masked.png", im);
    pts.clear();

    //video4
    //frame1
    im = cv::imread("output/video4_frame1.png");
    p1 = {638,413};
    p2 = {1442,219};
    p3 = {2579,1256};
    p4 = {1357,1755};
    pts.push_back(p1);
    cv::line(im, p1, p2, clr, 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, clr, 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, clr, 28, cv::LINE_4);
```

```
        pts.push_back(p4);
        cv::line(im, p4, p1, clr, 28, cv::LINE_4);
        cv::fillPoly( im, pts, clr );
        cv::imwrite("output/video4_frame1_masked.png", im);
        pts.clear();

        //video4
        //frame2
        im = cv::imread("output/video4_frame2.png");
        p1 = {580,503};
        p2 = {1379,296};
        p3 = {2555,1416};
        p4 = {1428,1950};
        pts.push_back(p1);
        cv::line(im, p1, p2, clr, 28, cv::LINE_4);
        pts.push_back(p2);
        cv::line(im, p2, p3, clr, 28, cv::LINE_4);
        pts.push_back(p3);
        cv::line(im, p3, p4, clr, 28, cv::LINE_4);
        pts.push_back(p4);
        cv::line(im, p4, p1, clr, 28, cv::LINE_4);
        cv::fillPoly( im, pts, clr );
        cv::imwrite("output/video4_frame2_masked.png", im);
        pts.clear();

        clr = {255, 244, 248};

        //video5
        //frame0
        im = cv::imread("output/video5_frame0.png");
        p1 = {263,794};
        p2 = {2791,535};
        p3 = {3527,1154};
        p4 = {356,1804};
        pts.push_back(p1);
        cv::line(im, p1, p2, clr, 28, cv::LINE_4);
        pts.push_back(p2);
        cv::line(im, p2, p3, clr, 28, cv::LINE_4);
        pts.push_back(p3);
        cv::line(im, p3, p4, clr, 28, cv::LINE_4);
        pts.push_back(p4);
        cv::line(im, p4, p1, clr, 28, cv::LINE_4);
        cv::fillPoly( im, pts, clr );
        cv::imwrite("output/video5_frame0_masked.png", im);
        pts.clear();

        //video5
        //frame1
        im = cv::imread("output/video5_frame1.png");
        p1 = {244,893};
        p2 = {2742,522};
        p3 = {3516,1116};
        p4 = {496,1936};
        pts.push_back(p1);
```

```
    cv::line(im, p1, p2, clr, 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, clr, 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, clr, 28, cv::LINE_4);
    pts.push_back(p4);
    cv::line(im, p4, p1, clr, 28, cv::LINE_4);
    cv::fillPoly( im, pts, clr );
    cv::imwrite("output/video5_frame1_masked.png", im);
    pts.clear();

    //video5
    //frame2
    im = cv::imread("output/video5_frame2.png");
    p1 = {367,929};
    p2 = {2810,595};
    p3 = {3545,1196};
    p4 = {581,1936};
    pts.push_back(p1);
    cv::line(im, p1, p2, clr, 28, cv::LINE_4);
    pts.push_back(p2);
    cv::line(im, p2, p3, clr, 28, cv::LINE_4);
    pts.push_back(p3);
    cv::line(im, p3, p4, clr, 28, cv::LINE_4);
    pts.push_back(p4);
    cv::line(im, p4, p1, clr, 28, cv::LINE_4);
    cv::fillPoly( im, pts, clr );
    cv::imwrite("output/video5_frame2_masked.png", im);
    pts.clear();

    // 8. оцениваем качество выделение зоны и анализируем ошибки

    return 0;
}
```