Работа 4. Детектирование границ документов на кадрах видео

автор: Хонер П.Д.
дата: 2022-03-22T23:48:43

https://github.com/pavelkhoner/OpenCV/tree/master/khoner_p_d/prj.labs/lab04

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. бинаризцем 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 #1

Рис. 2. Исходное изображение #2

Рис. 3. Исходное изображение #3

Рис. 4. Цветоредуцированое изображение #1

Рис. 5. Цветоредуцированое изображение #2

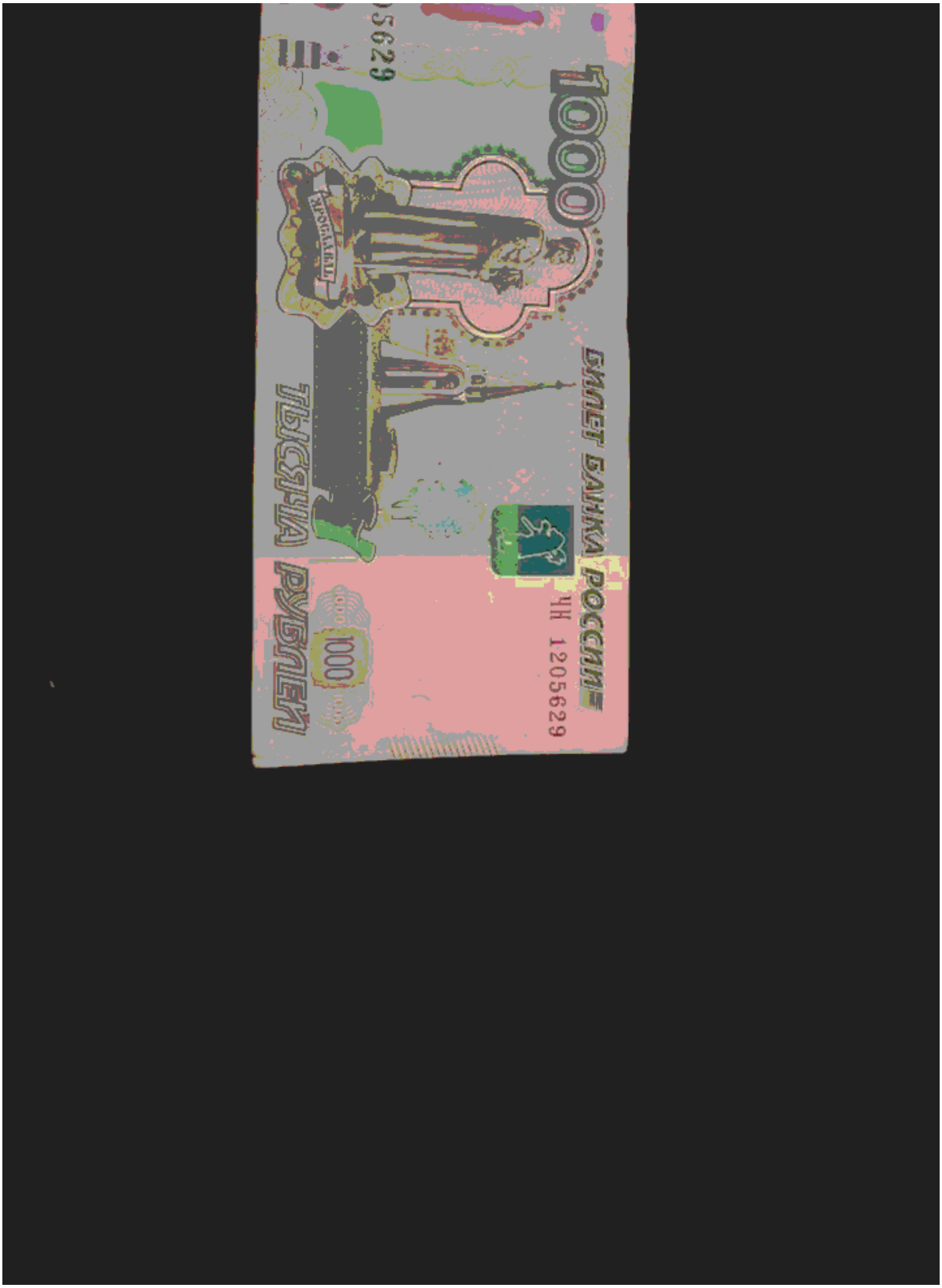

Рис. 6. Цветоредуцированое изображение #3

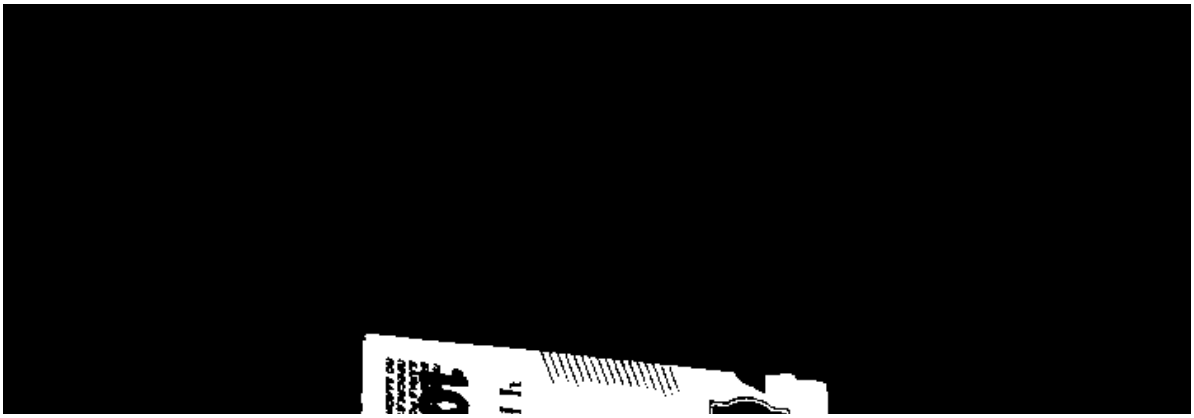

Рис. 7. Бинаризированное изображение #1

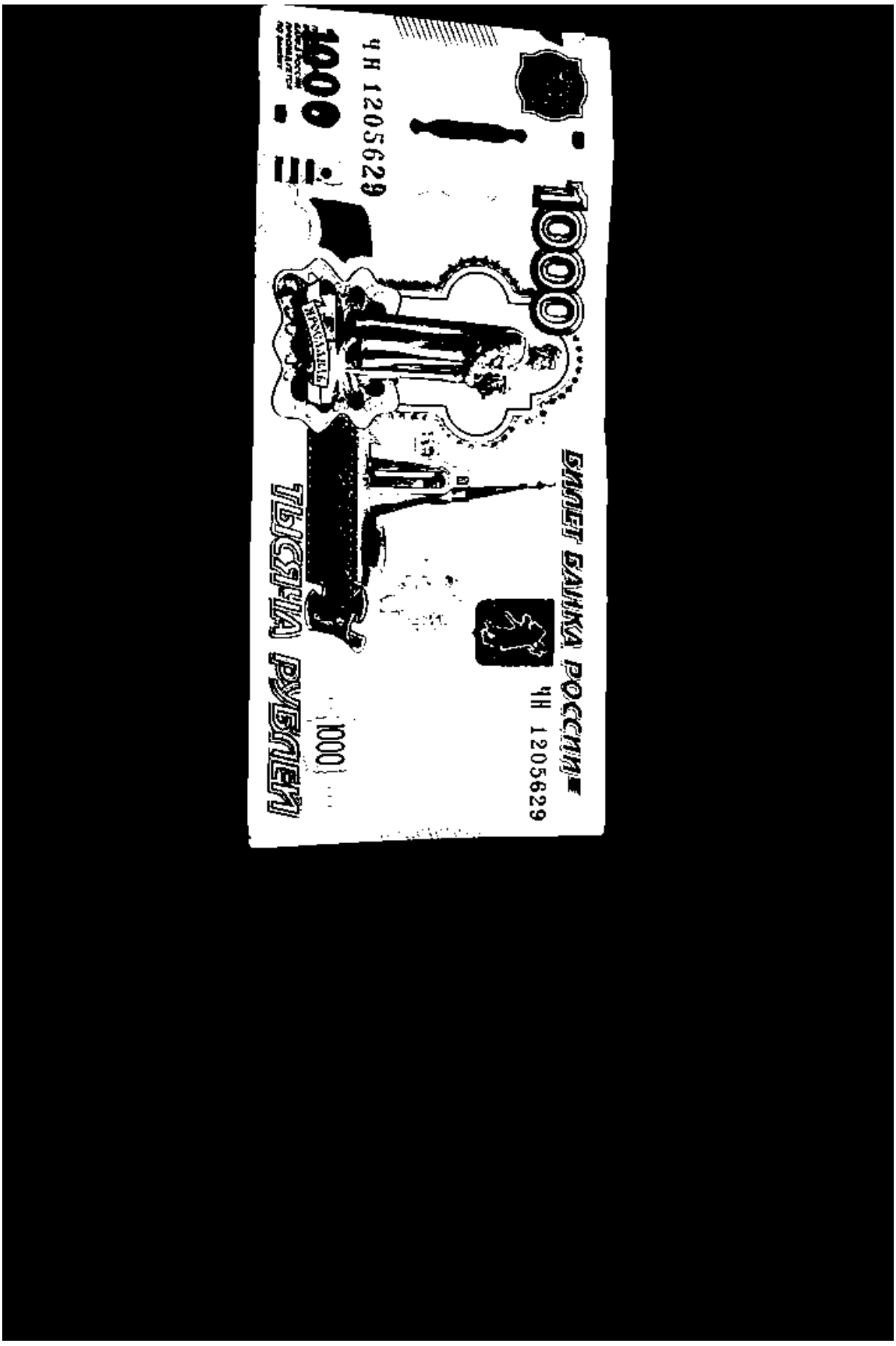

Рис. 8. Бинаризированное изображение #2

Рис. 9. Бинаризированное изображение #3

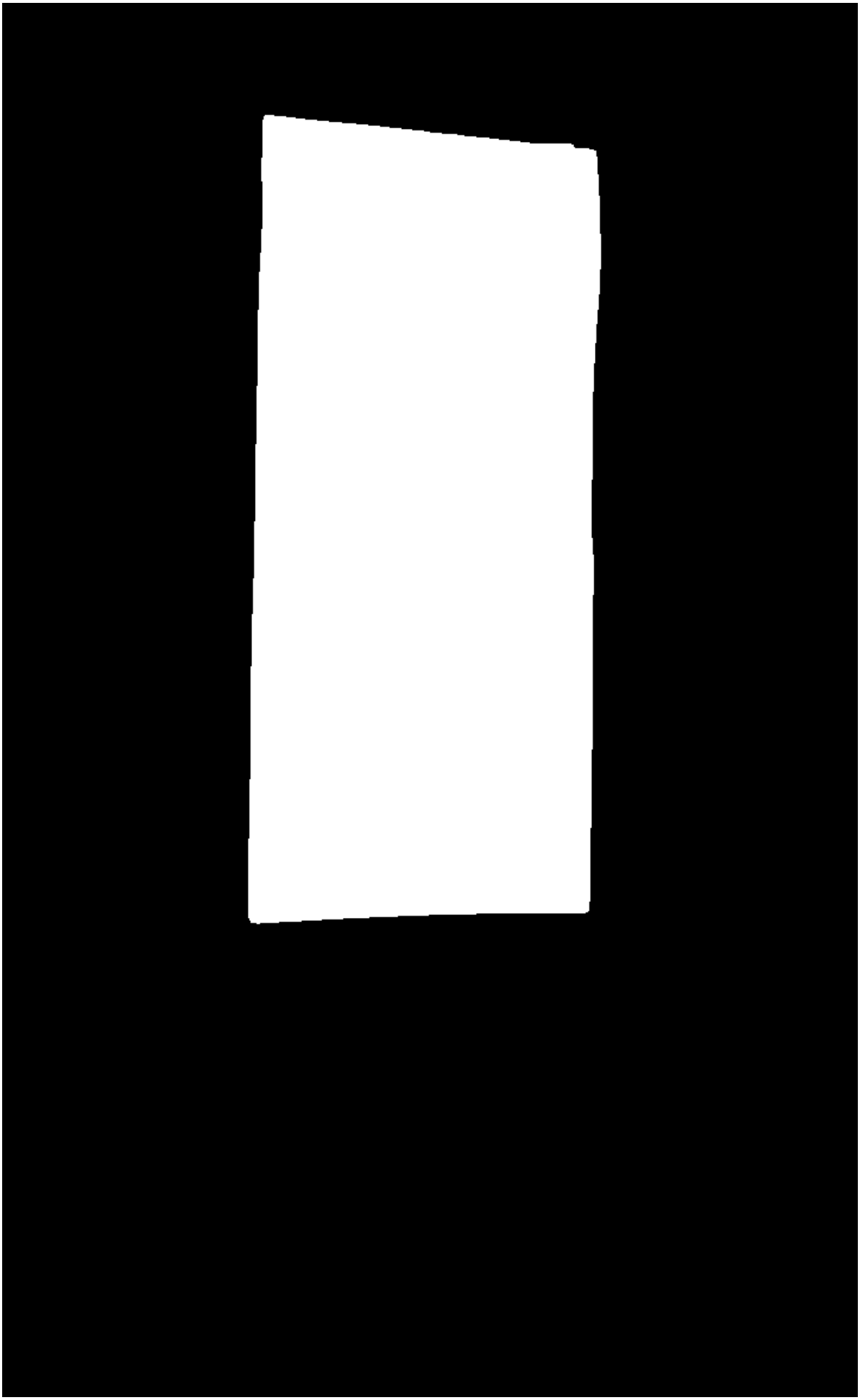

Рис. 10. Морфологически обработанное изображение #1

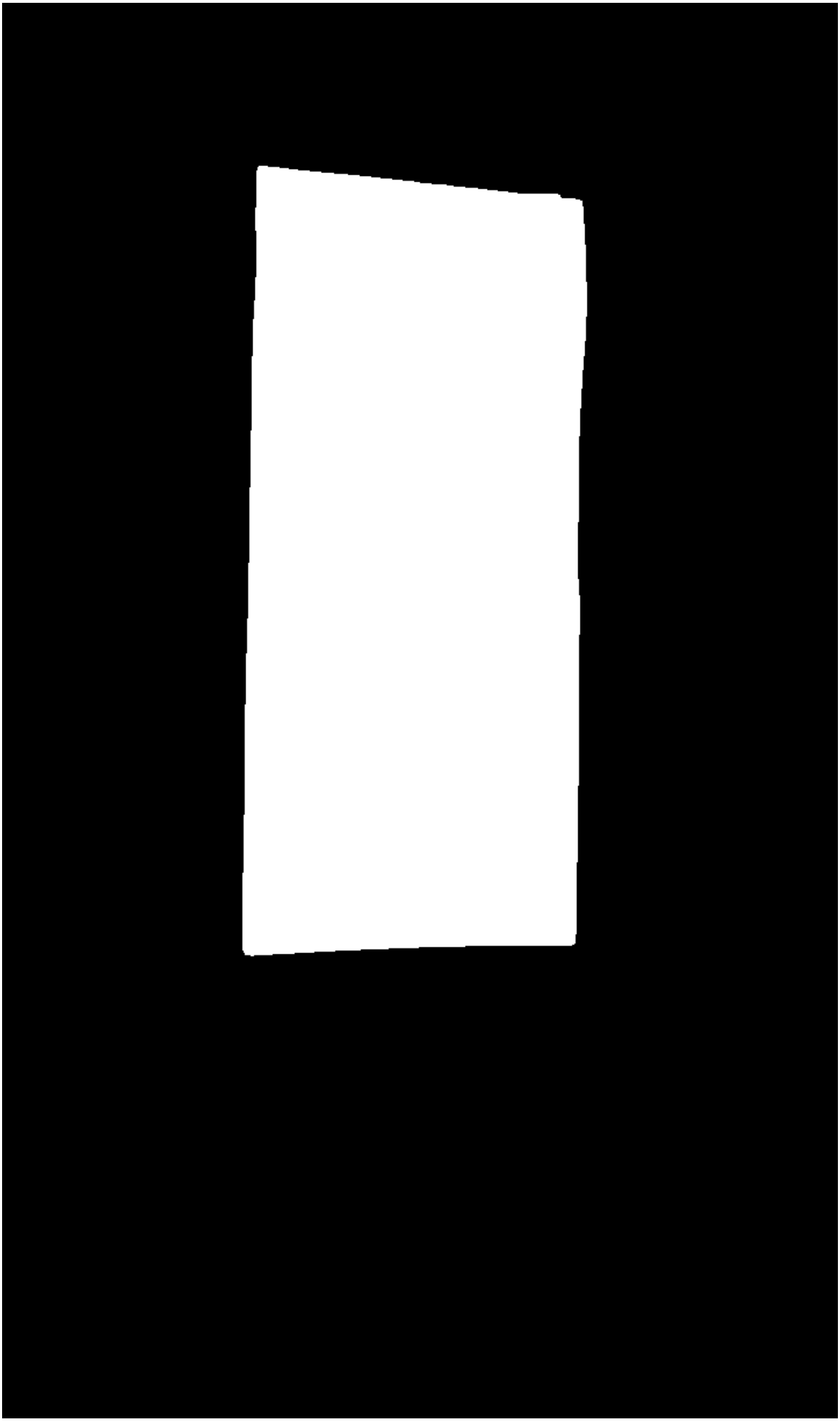

Рис. 11. Морфологически обработанное изображение #2

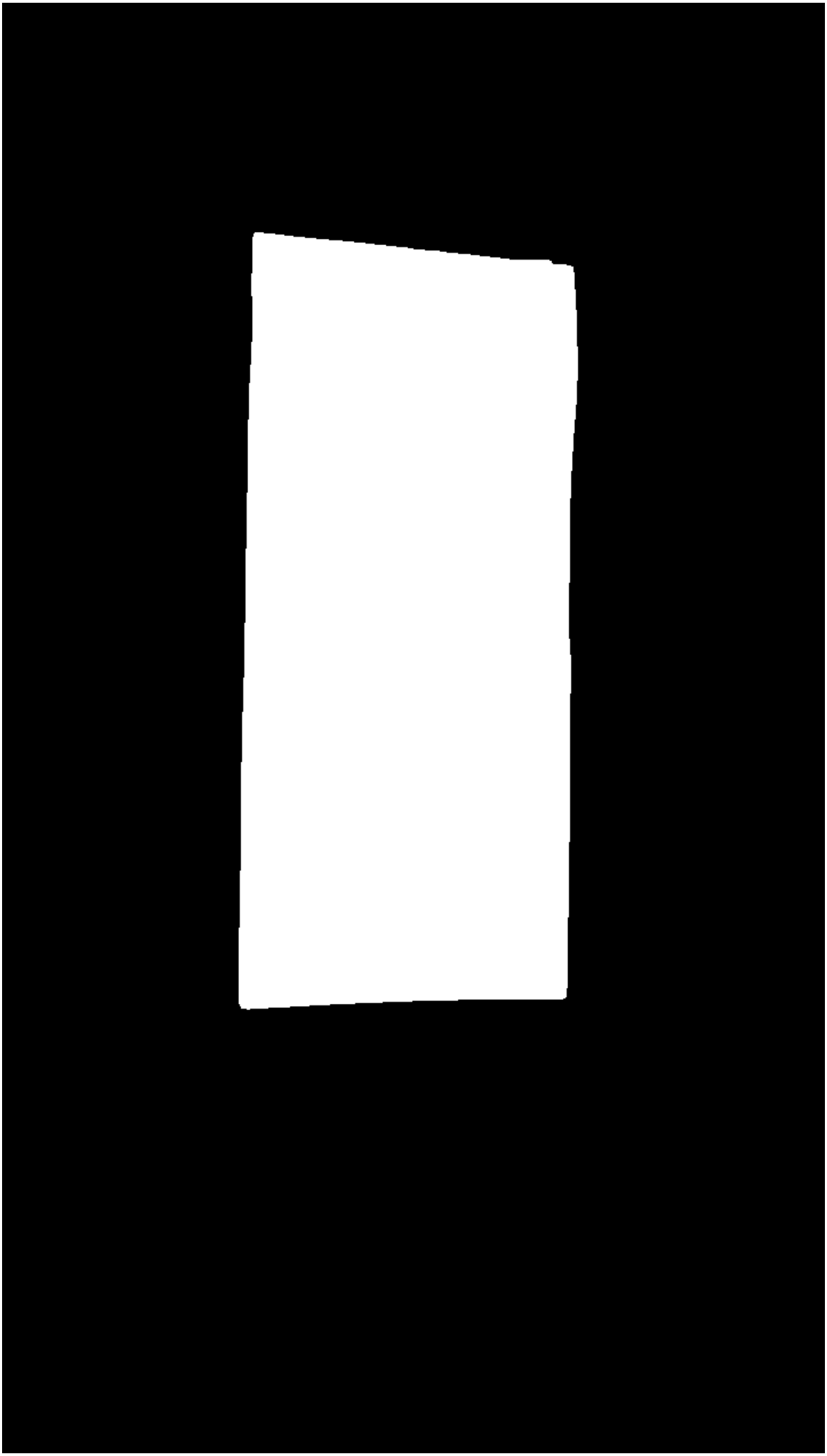

Рис. 12. Морфологически обработанное изображение #3

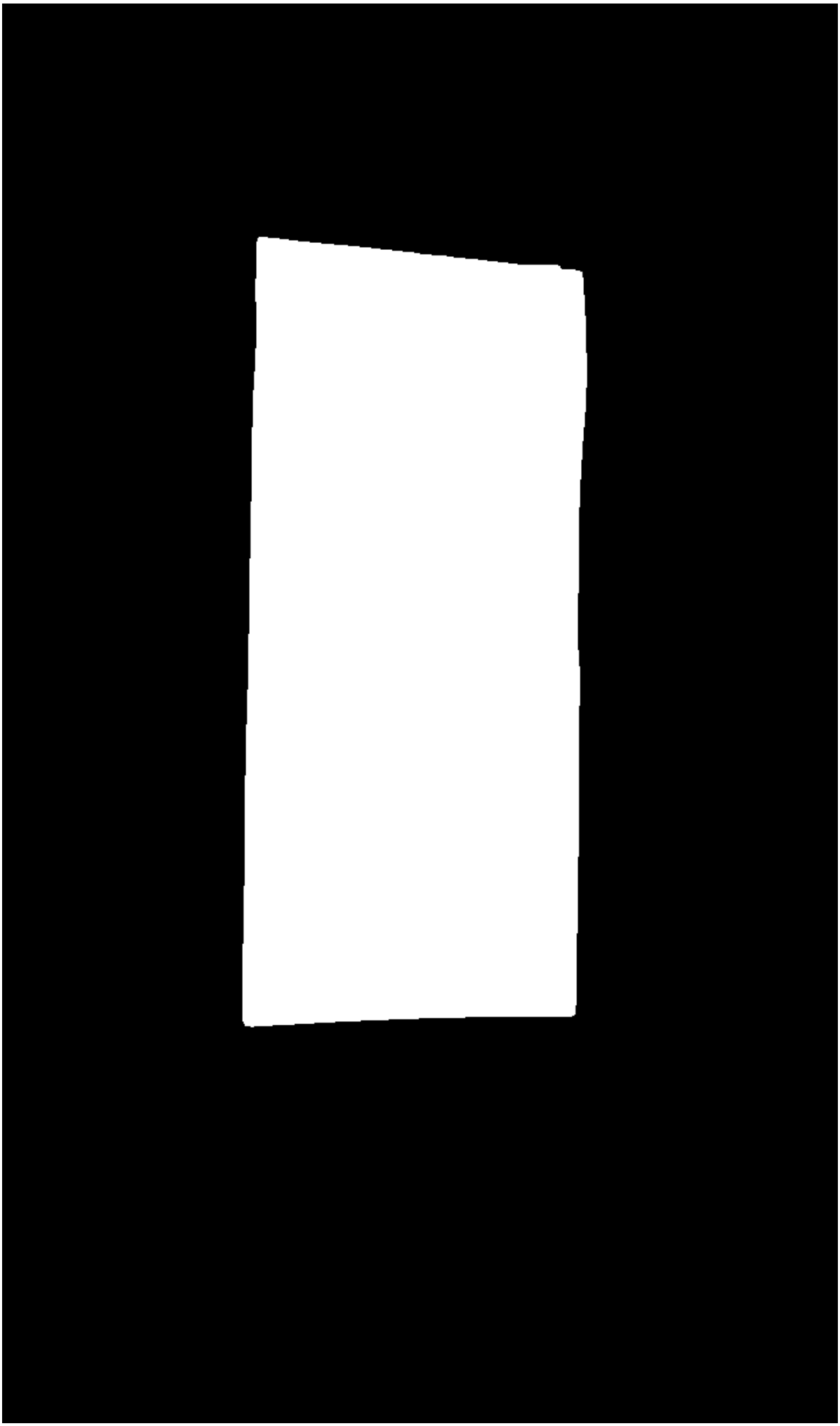

Рис. 13. Основная компонента связности #1

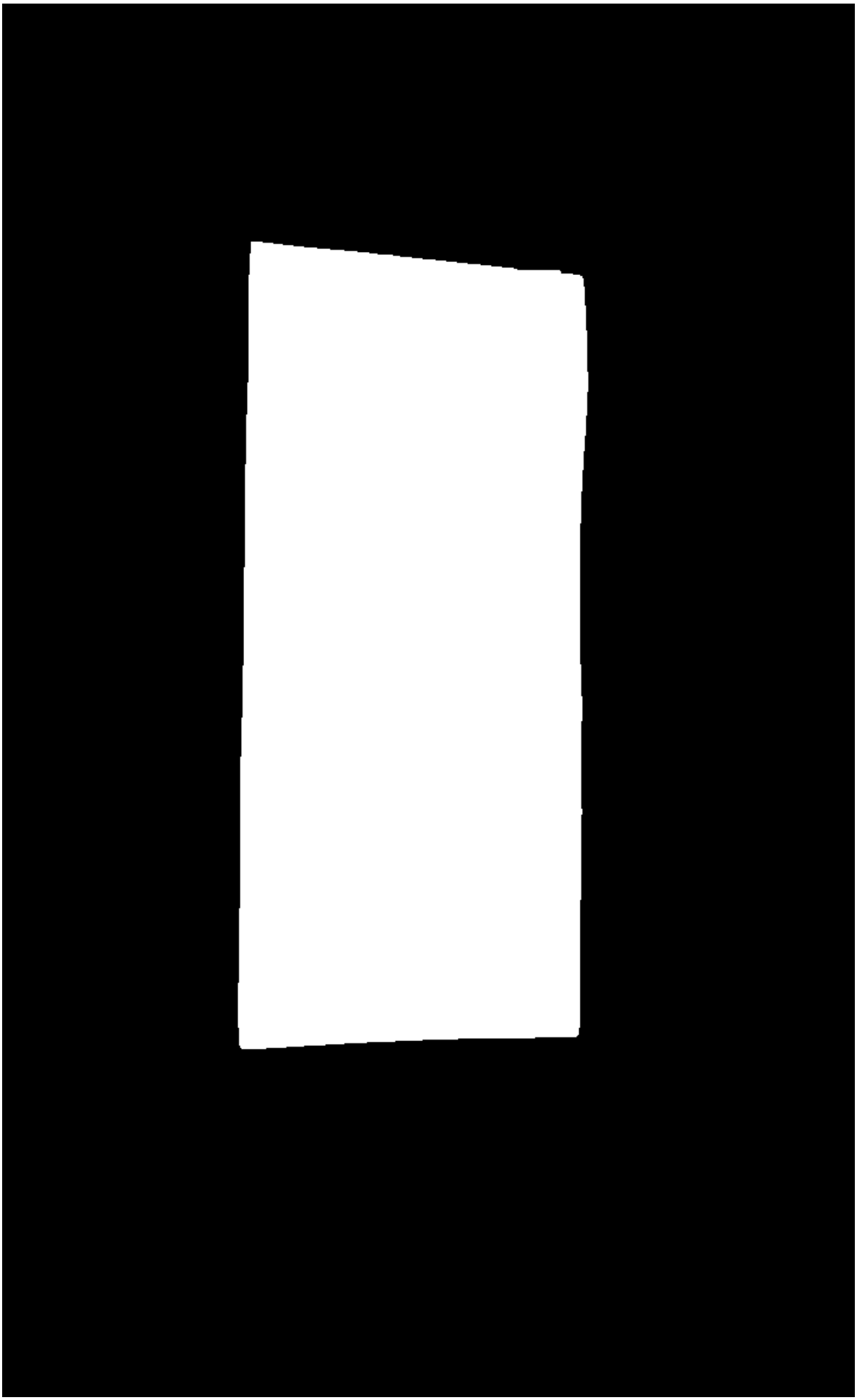

Рис. 14. Основная компонента связности #2

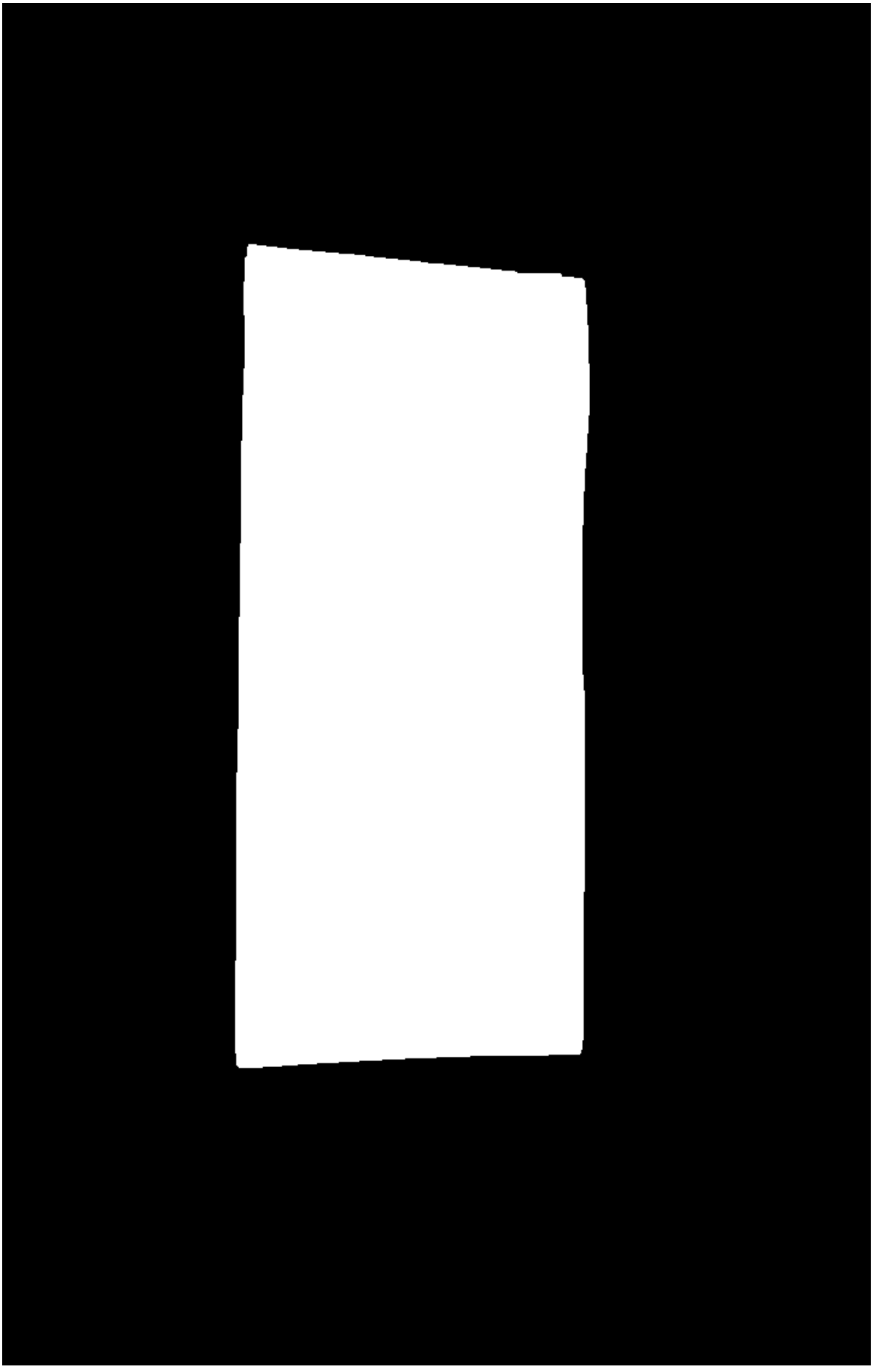

Рис. 15. Основная компонента связности #3

Текст программы

```
#include <iostream>
#include <opencv4/opencv2/opencv.hpp>
#include <string>

void Color_Reduction(cv::Mat& img, uchar div = 64)
{
    for (ptrdiff_t y{ 0 }; y < img.rows; ++y)
    {
        uchar* data = img.ptr(y);
        for (ptrdiff_t x{ 0 }; x < img.cols * img.channels(); ++x)
        {
            data[x] = data[x] - data[x] % div + div / 2;
        }
    }
}

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

int main()
{

    cv::VideoCapture video("../data/video4.MOV");
    if (!video.isOpened())
    {
        return -1;
    }

    cv::Mat frame[3];
    uint16_t frames_N = video.get(cv::CAP_PROP_FRAME_COUNT);

    for (ptrdiff_t i{ 0 }; i < frames_N; ++i)
    {
        if (i == (frames_N / 5) * 2)
        {
            video >> frame[0];
        }

        if (i == (frames_N / 5) * 3)
        {
            video >> frame[1];
        }
```

```
        if (i == (frames_N / 5) * 4)
        {
            video >> frame[2];
        }

        video.grab();
    }

    for (ptrdiff_t i{ 0 }; i < 3; ++i)
    {
        cv::imwrite("frame_" + std::to_string(i) + ".png", frame[i]);
    }

    cv::Mat Mask;

    for (ptrdiff_t i{ 0 }; i < 3; ++i)
    {
        Color_Reduction(frame[i]);
        cv::imwrite("frame_" + std::to_string(i) + "_CR.png", frame[i]);
        Binarisiotion(frame[i], 150);
        cv::imwrite("frame_" + std::to_string(i) + "_BINAR.png", frame[i]);
        cv::morphologyEx(frame[i], Mask, cv::MORPH_CLOSE,
cv::getStructuringElement(cv::MORPH_RECT, cv::Size(40, 40)));
        cv::imwrite("frame_" + std::to_string(i) + "_MORPH.png", Mask);

        cv::Mat centroids, stats, res;
        int n = cv::connectedComponentsWithStats(Mask, res, stats, centroids);
        std::vector<cv::Vec3b> labels(n);
        int max = 0, imax = 0;
        for (ptrdiff_t j = 1; j < n; j++)
        {
            if (max < stats.at<int>(j, cv::CC_STAT_AREA))
            {
                max = stats.at<int>(j, cv::CC_STAT_AREA);
                imax = j;
            }
        }
        for (ptrdiff_t j = 0; j < n; ++j)
        {
            labels[j] = cv::Vec3b(0, 0, 0);
        }
        labels[imax] = cv::Vec3b(255, 255, 255);
        cv::Mat exit(Mask.rows, Mask.cols, CV_8UC3);
        for (ptrdiff_t a = 0; a < exit.rows; ++a)
        {
            for (ptrdiff_t b = 0; b < exit.cols; ++b)
            {
                int label = res.at<int>(a, b);
                cv::Vec3b& pixel = exit.at<cv::Vec3b>(a, b);
                pixel = labels[label];
            }
        }
        cv::imwrite("frame_" + std::to_string(i) + "_CONECTED.png", exit);
    }

    return 0;
}
```