Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

Отчёт

по лабораторной работе №4

Дисциплина: Техническое зрение

Тема: Распознавание образов на изображении при помощи контурного анализа

Студент гр. 3331506/70401 Архипов А.Е.

Преподаватель Варлашин В.В.

« »_____2020 г.

Санкт-Петербург
2020

Цель

Ознакомление со встроенными функциями для контурного анализа из библиотеки OpenCV, и их использование для нахождения необходимых объектов на заданных изображениях.

Задачи

1) Найти и обозначить примерный центр выделяющегося объекта на заданном изображении (рисунок 1);

Рисунок 1 — Задание 1

2) Найти и обозначить примерный центр выделяющегося объекта на заданном изображении (рисунок 2);

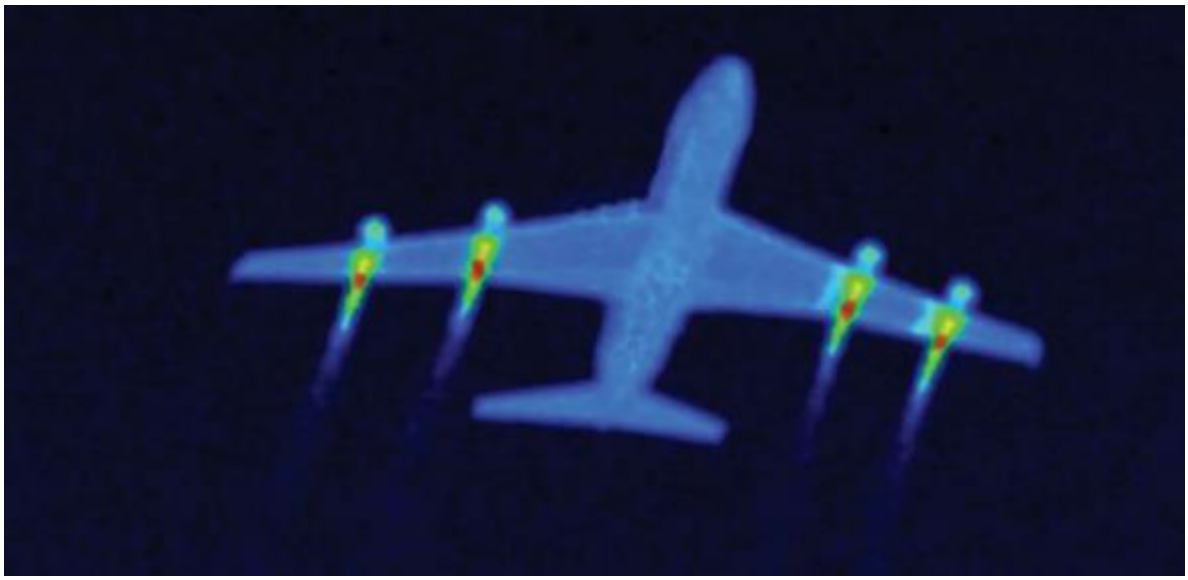

Рисунок 2 — Задание 2

3.1) На каждом роботе найти его цветную верхнюю крышку и обвести контуром цвета его команды;

3.2) Найти лампу, обозначить её как-нибудь;

3.3) Для каждой команды обозначить ближайшего робота к лампе, путём рисования его центра масс;

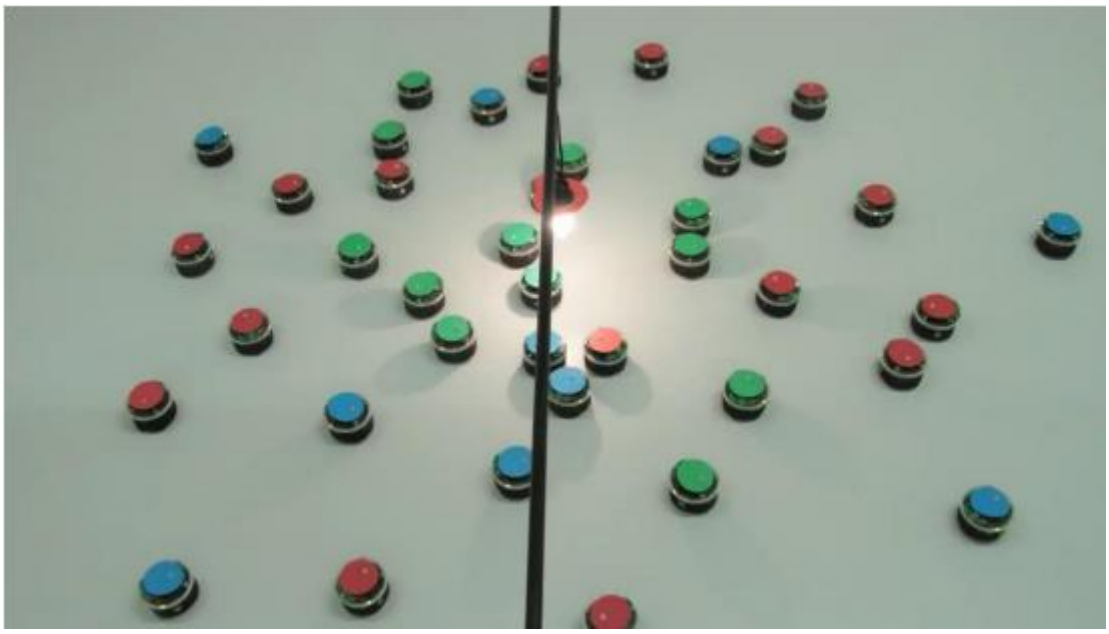

Рисунок 3 — Задание 3

4) Найти и обозначить на рисунке 4 правильные и бракованные гаечные ключи с помощью заданного шаблона.

Рисунок 4 — Задание 4

Ход работы

1. Решение задачи 1

Алгоритм решения данного задания:

1) Проводим пороговую фильтрацию с помощью функции threshold. Результат показан на рисунке 5.

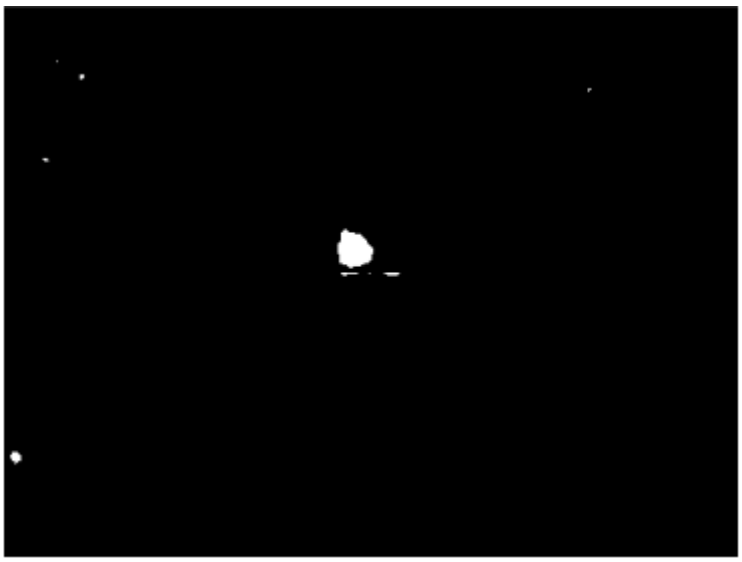

Рисунок 5 — Результат пороговой фильтрации

2) Для удаления мелких дефектов проводим эрозию и дилатацию функциями erode и dilate соответственно. Результат показан на рисунке 6.

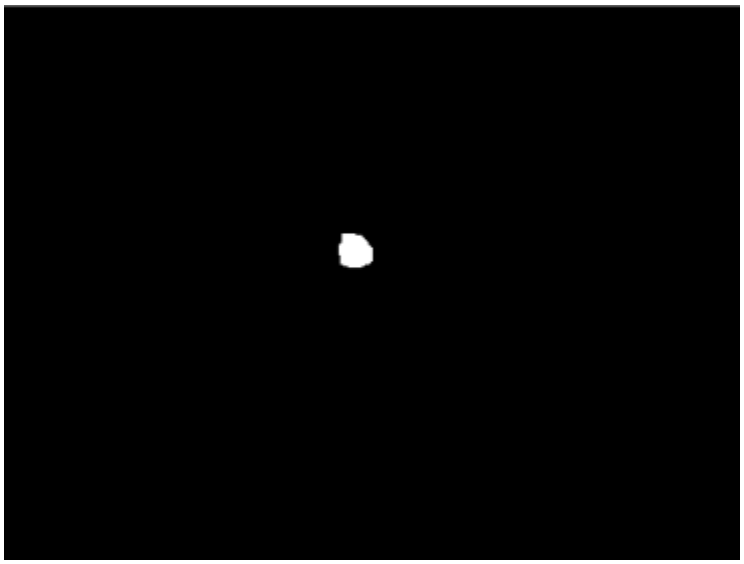

Рисунок 6 — Результат эрозии и дилатации

3) Функцией Canny выделяем границы, находим контур функцией findContours и рисуем его функцией polylines. Результат показан на рисунке 7.

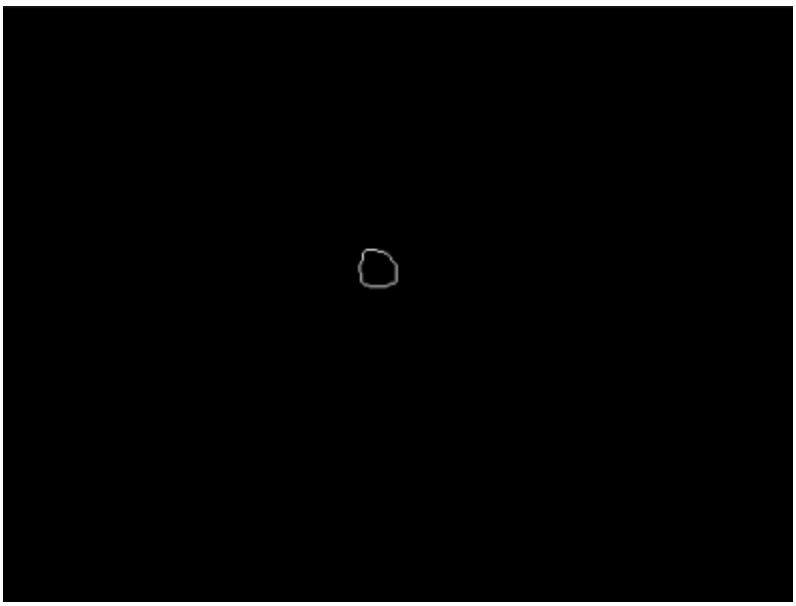

Рисунок 7 — Контур цели

4) Для нахождения центра масс контура используем класс Moments и функцию moments. Обозначаем центр масс красным крестом на исходной изображении. Результат показан на рисунке 8

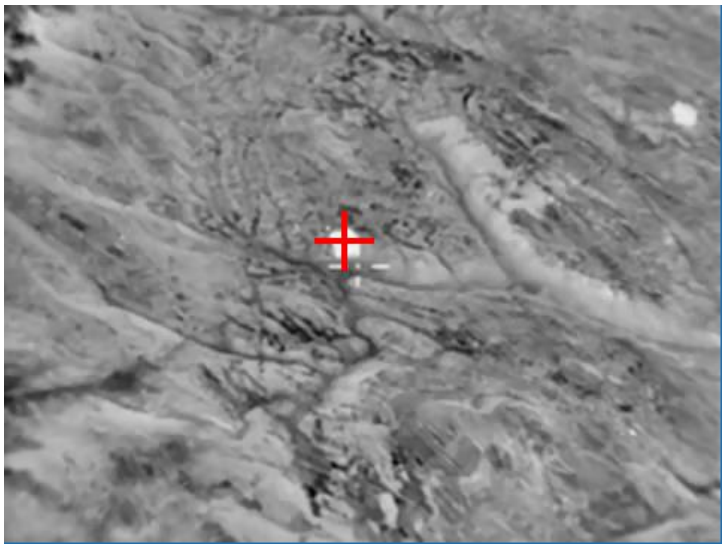

Рисунок 8 – Результат

2. Решение задачи 2

Алгоритм решения данного задания:

1) Переводим изображение в цветовое пространство HSV с помощью функции cvtColor. Проводим пороговую фильтрацию функцией inRange. Результат показан на рисунке 9.

Рисунок 9 – Результат пороговой фильтрации

2) Для удаления мелких дефектов проводим эрозию и дилатацию функциями erode и dilate. Результат показан на рисунке 10.

Рисунок 10 – Результат эрозии и дилатации

3) Функцией Canny выделяем границы, находим контуры функцией findContours и рисуем функцией polylines. Результат показан на рисунке 11.

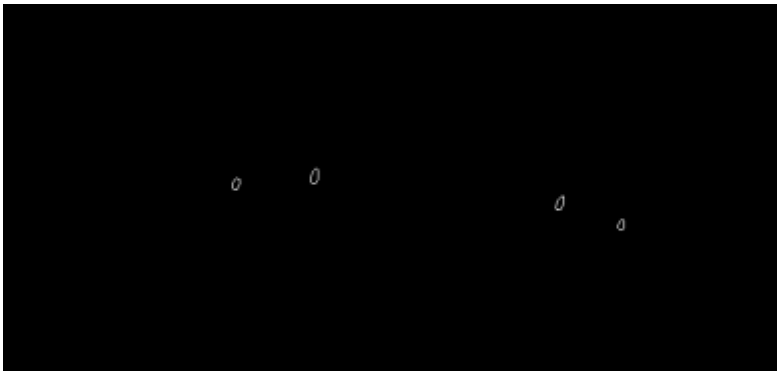

Рисунок 11 – Контуры целей

4) Для нахождения центра масс контуров используем класс Moments и функцию moments. Обозначаем центр масс черным крестом на исходной изображении. Результат показан на рисунке 12.

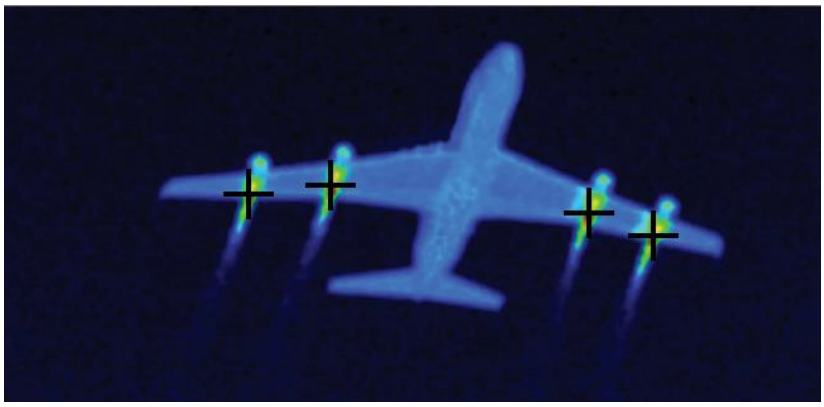

Рисунок 12 – Результат

На рисунке 13 показаны результаты работы алгоритма с другими примерами.

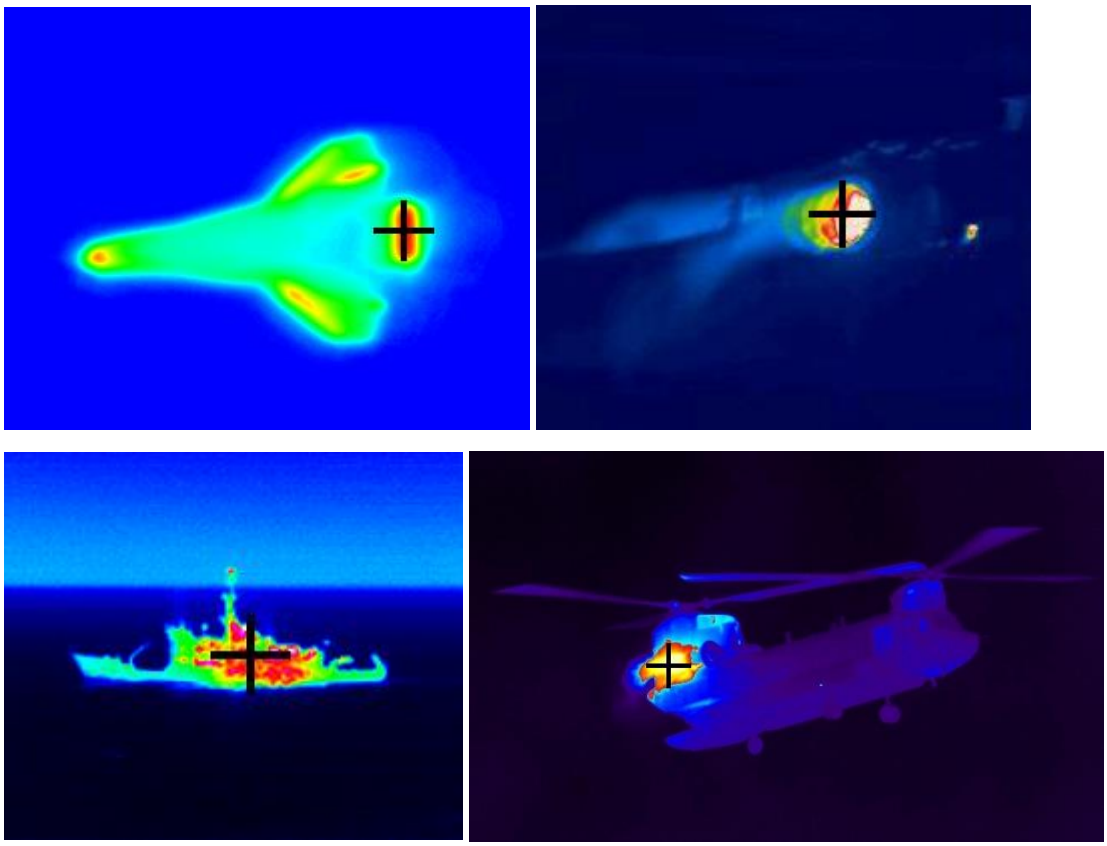

Рисунок 13 – Результаты обработки других изображений

3. Решение задачи 3

Алгоритм решения данного задания:

1) Переводим изображение в цветовое пространство HSV с помощью функции cvtColor. Проводим поиск лампы. Для этого используются следующие функции: inRange, erode, dilate, Canny, fintContours, moments, circle.

2) Производим поиск крышек трех цветов. Используются функции inRange, erode, dilate, Canny, findContours, polylines.

3) Поиск ближайшей до лампы крышки производится следующим образом:

- Создается вектор точек (координаты центров масс) всех контуров определенного цвета;
- Вычисляются расстояния между каждой точкой и лампой и находится наименьшее расстояние, реализовано функцией findMinIndex (рисунок 14);
- Центр масс контура, который находится ближе всего к лампе, соединяется с лампой линией.

```
int findMinIndex(Point lamp, vector<Point> var)
{
    vector<double> distances;
    for (int i = 0; i < var.size(); i++)
    {
        double x1 = lamp.x;
        double x2 = var[i].x;
        double y1 = lamp.y;
        double y2 = var[i].y;
        double result = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
        distances.push_back(result);
        cout << distances[i] << endl;
    }

    int indexMin = 0;
    for (int i = 0; i < distances.size(); i++)
    {
        if (distances[i] <= distances[indexMin])
        {
            indexMin = i;
            cout << i << endl;
        }
    }
    return indexMin;
}
```

Рисунок 14 – Функция findMinIndex

Результат работы всего алгоритма представлен на рисунке 15.

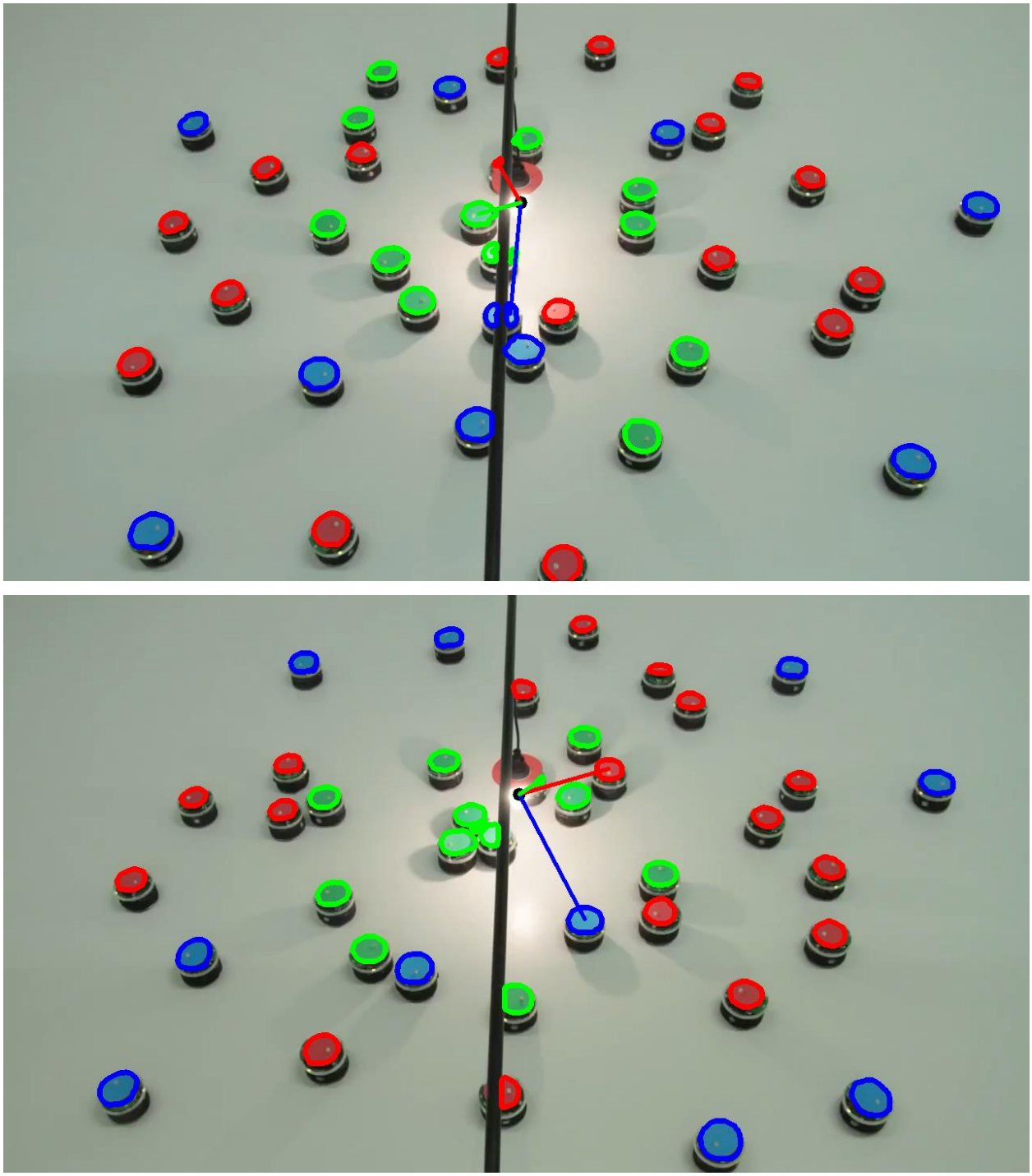

Рисунок 15 – Результат выполнения 3 задания

4. Решение задачи 4

Алгоритм решения данного задания:

1) Сначала производим обработку шаблона:

- Пороговая фильтрация функцией threshold;
- Поиск границ функцией Canny;
- Нахождение контура и его изображение функциями findContours и polylines.

2) Обработка изображения с ключами:

- Пороговая фильтрация функцией threshold;
- Эрозия и дилатация функциями erode и dilate;
- Поиск границ функцией Canny;
- Нахождение контура и его изображение функциями findContours и polylines.

3) Проводим сравнение контуров ключей с контуром шаблона с помощью функции matchShapes.

4) Обводим контуры правильных ключей и неправильных разными цветами. Результат показан на рисунке 16.

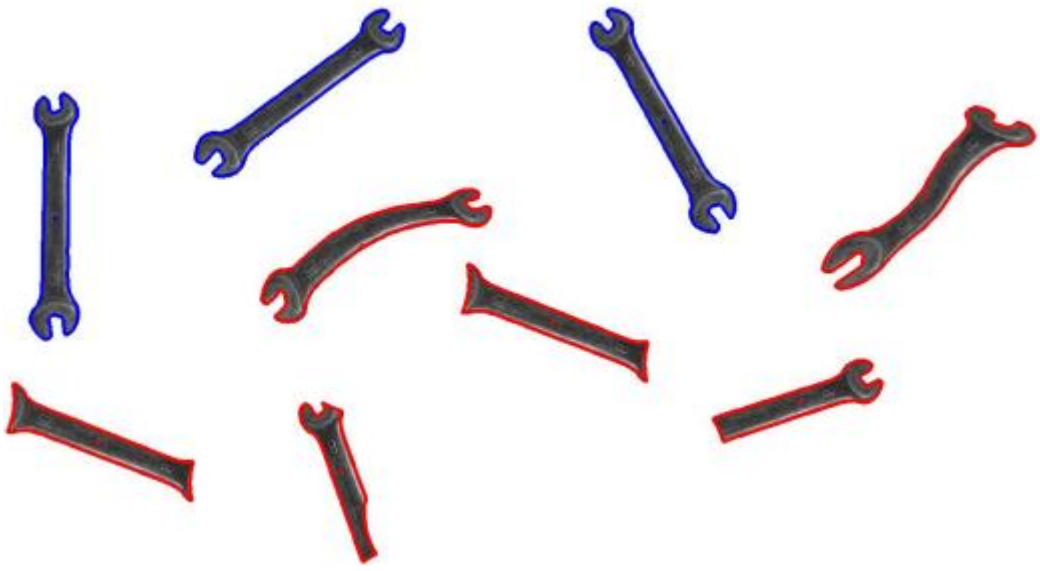

Рисунок 16 – Результат выполнения 4 задания

Вывод

В ходе работы были изучены основные функции контурного анализа из библиотеки OpenCV. С помощью данных функций были успешно распознаны образы объектов на заданных изображениях.

Защита

Нужно произвести сшивание перекрытых масок роботов независимо от цвета. Пример разделенной и сшитой маски робота приведен на рисунке 17.

Рисунок 17 — Пример сшивания маски робота

Алгоритм решения данного задания:

1) Нужно определить роботов, которые перекрыты трубкой.

- Сначала нужно определить контур для трубки. Воспользуемся преобразованием Хафа для поиска прямых линий. В OpenCV есть готовая функция HoughLines для поиска прямых линий. Результат работы преобразования приведен на рисунке 18.

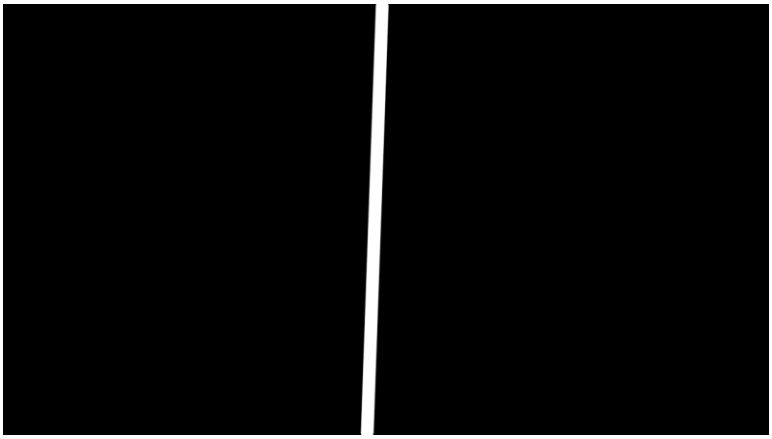

Рисунок 18 — Результат поиска трубки

- Наносим трубку на маски с роботами. Пример показан на рисунке 19.

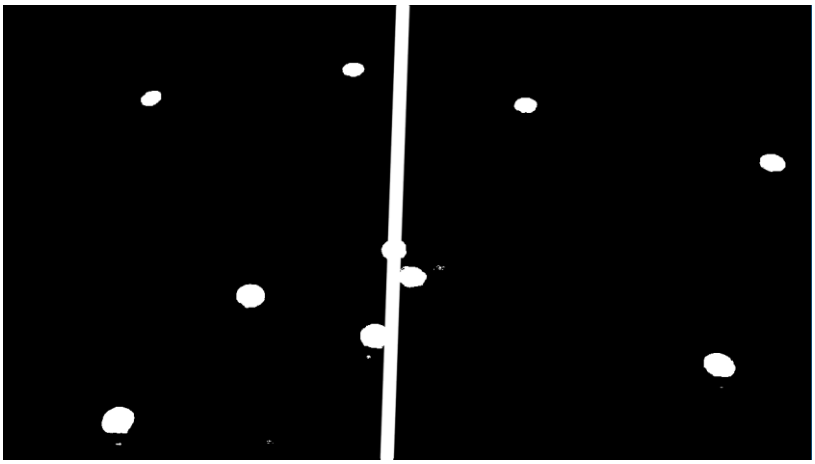

Рисунок 19 — Нанесение трубки на маску роботов

- Далее необходимо оставить маски роботов, которые пересекаются с трубкой. Это можно выполнить с помощью функции floodFill, которая делает заливку изолированных пикселей. Если в качестве точки выбрать любую точку на трубке, то выделятся все пиксели, которые соприкасаются друг с другом. Пример работы приведен на рисунке 20.

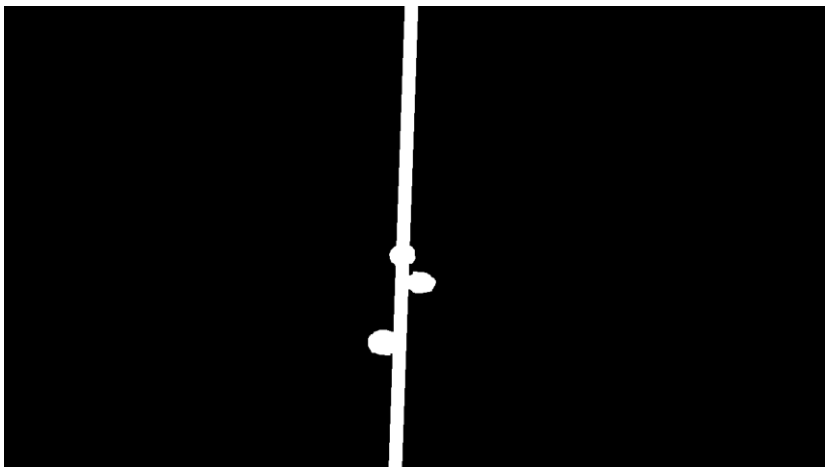

Рисунок 20 — Роботы, перекрытые трубкой, и сама трубка

2) Необходимо определить маску для робота с учетом перекрытия. Для этого воспользуемся методом Hit or Miss. Он позволяет искать шаблоны, которые задаются ядром, на изображении.

Для заполнения ядра написана функция initKernel, она принимает на вход размер ядра и заполняет его для поиска эллипсов на изображении.

Если шаблон был найден, то на выходе будет точка, которая соответствует центру ядра. В результате получится полотно на котором белыми точками обозначены центры найденных ядер. Чтобы восстановить эллипсы необходимо нарисовать их в каждой белой точки полотна и такого же размера, которым они были найдены. Результат работы приведен на рисунке 21.

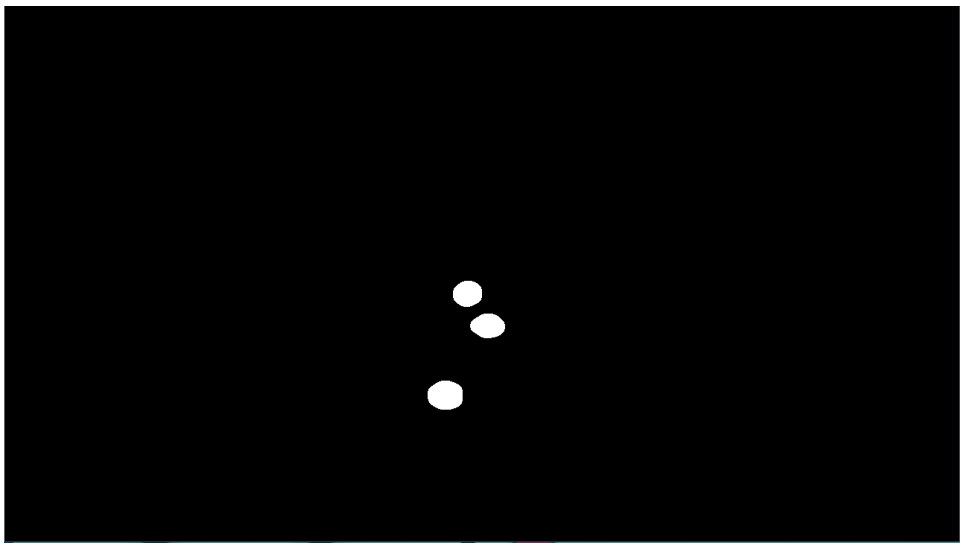

Рисунок 21 — Пример восстановления масок роботов

3) Нанесение получившихся масок на полотно с остальными масками роботов.

Результат работы представлен на рисунке 22.

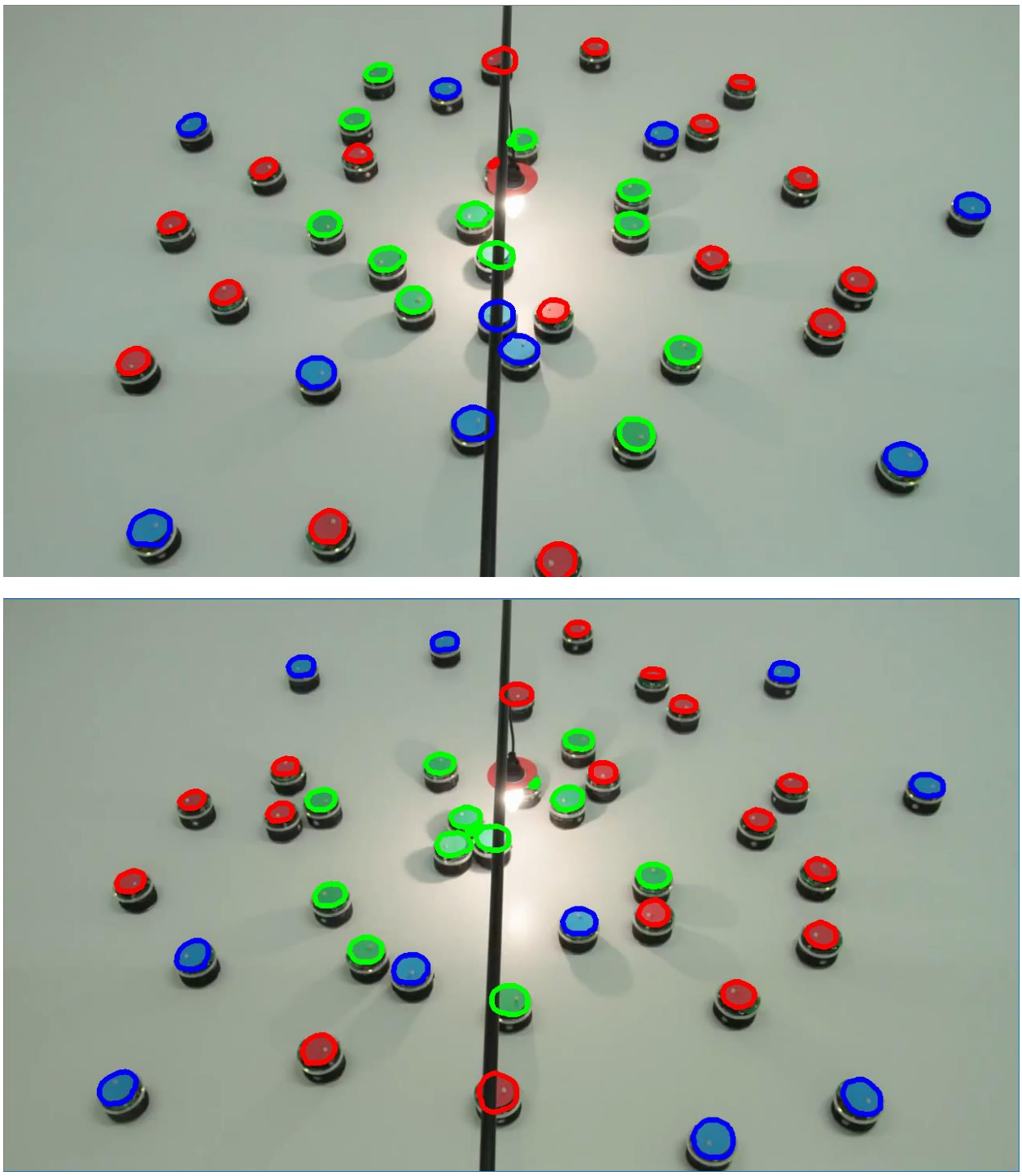

Рисунок 22 — Результат выполнения задания