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 Паньков И.С.

Преподаватель Титов В.В.

«__» _____ 2020 г.

Санкт-Петербург
2020

Цель работы — применение контурного анализа средствами библиотеки OpenCV для распознавания и детектирования объектов на изображениях.

З а д а н и я

Задание 1. Найти на изображении укрытие противника и обозначить его центр для наведения БПЛА на цель.

Задание 2. Найти на изображении моторное отделение техники противника для наведения артиллерийской установки на уязвимое место цели.

Задание 3. На изображении с группой роботов:

- найти на каждом роботе его цветную верхнюю крышку и обвести контуром цвета его команды;
- найти лампу и обозначить её маркером;
- найти для каждой команды ближайшего робота к лампе и обозначить его путём рисования центра масс.

Задание 4. Найти на изображении исправные и бракованные гаечные ключи и обозначить их разными метками.

Выполнение работы

Задание 1. Поиск укрытий на изображениях

Первое задание заключается в поиске на полутоновом изображении наиболее яркой его части, то есть области с наибольшей интенсивностью пикселей. Сделать это можно, применив пороговую фильтрацию, а затем определив контур полученной области. Центр масс найденного контура и является центром искомого укрытия.

Для выполнения поиска укрытий была разработана функция `findEnemyCover`. С целью более гибкой настройки порога при выполнении пороговой фильтрации данная функция выводит изображение в окне и создаёт трекбар (англ. trackbar — «бегунок», «ползунок»). Текущее положение трекбара записывается в переменную `thresh`. Окно с выводимым изображением представлено на рисунке 1.

Рисунок 1 — Окно с выводимым функцией `findEnemyCover` изображением

При изменении положения трекбара текущее значение переменной `thresh` и указатель на некоторую служебную информацию `data` передаются во вспомогательную функцию `detectCover`, которая выполняет основную работу.

В функции `detectCover` сначала выполняется пороговая фильтрация изображения с помощью функции `threshold`, а затем выделение границ с

помощью морфологической операции поиска градиента `morphologyEx` с флагом `MorphTypes::MORPH_GRADIENT`. Для избавления от возникающих шумов перед этапом выделения границ дополнительно выполняется фильтрация изображения открывающим фильтром с помощью морфологической операции `morphologyEx` с флагом `MorphTypes::OPEN`. Результаты проведения пороговой фильтрации и выделения границ представлены на рисунке 2.

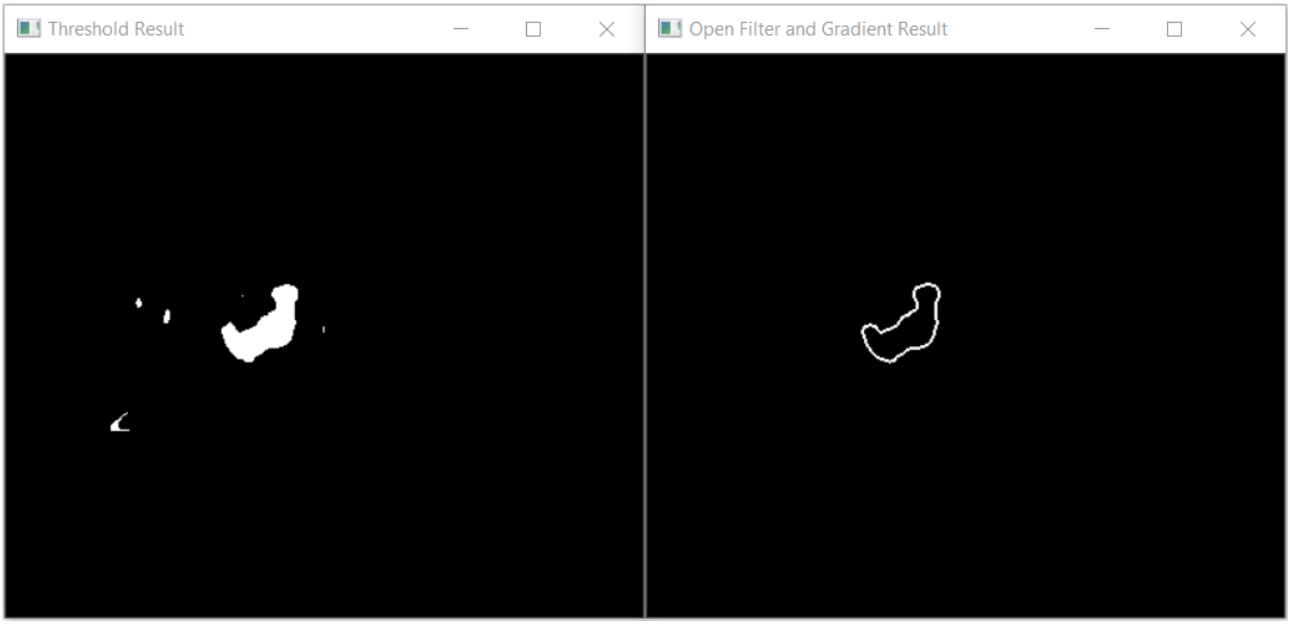

Рисунок 2 — Результаты проведения пороговой фильтрации (слева) и выделения границ полученной области (справа)

Результаты проведения поиска укрытий на всех изображениях представлены на рисунках 4 — 5.

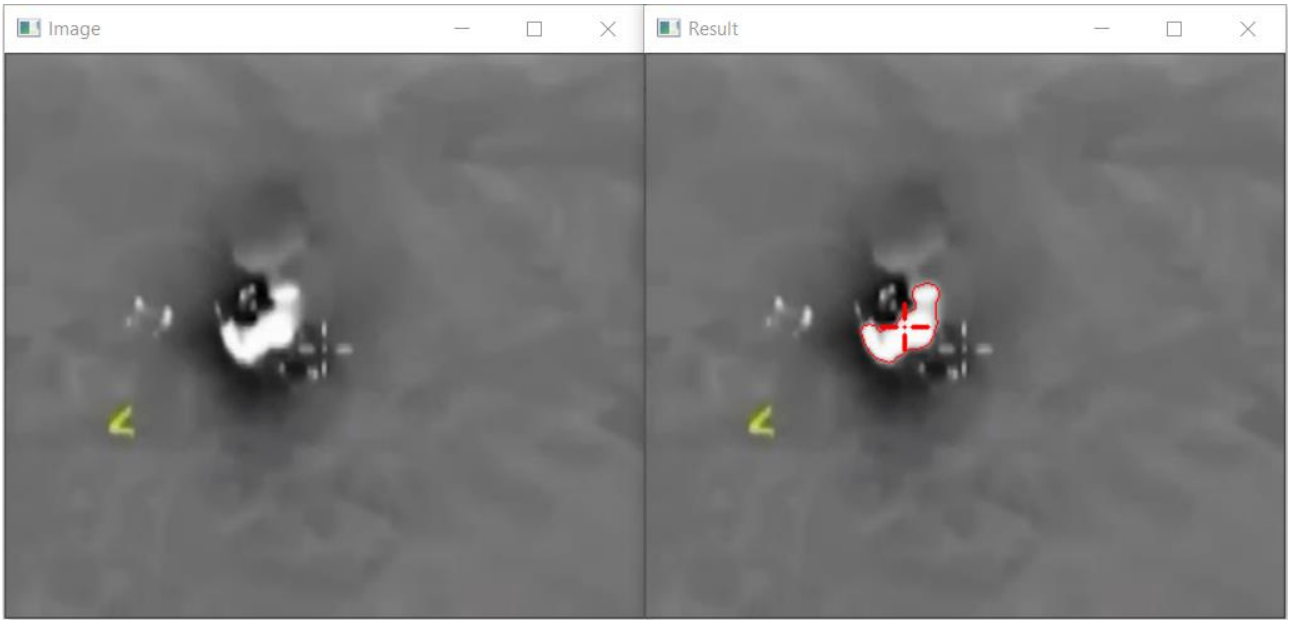

Рисунок 3 — Первое исходное изображение (слева) и результат поиска укрытий на нём (справа)

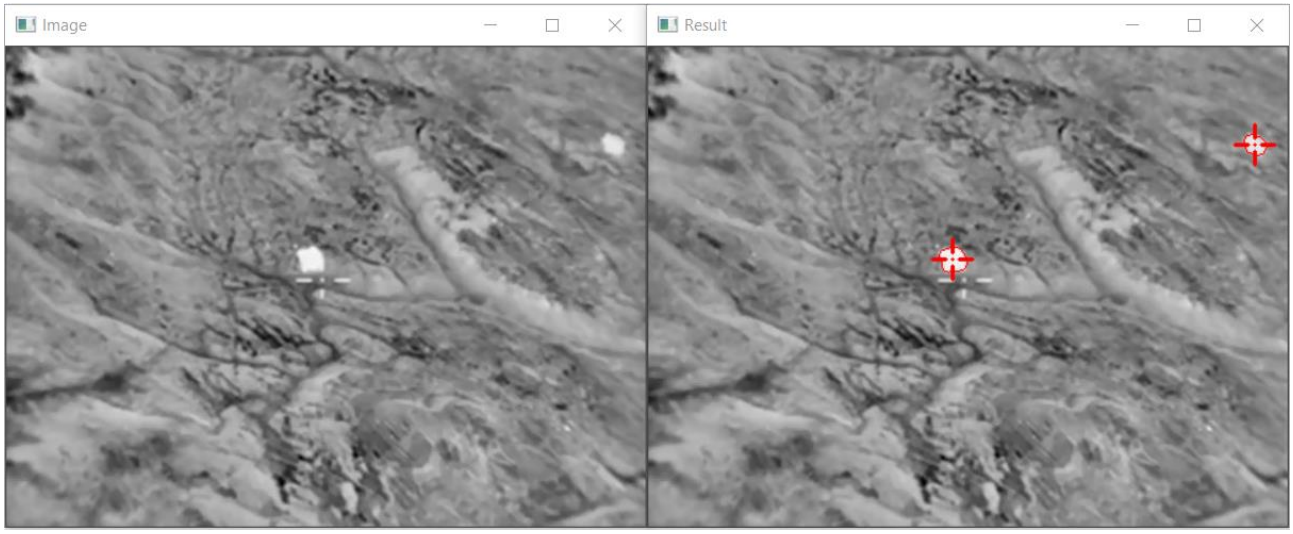

Рисунок 4 — Второе исходное изображение (слева) и
результат поиска укрытий на нём (справа)

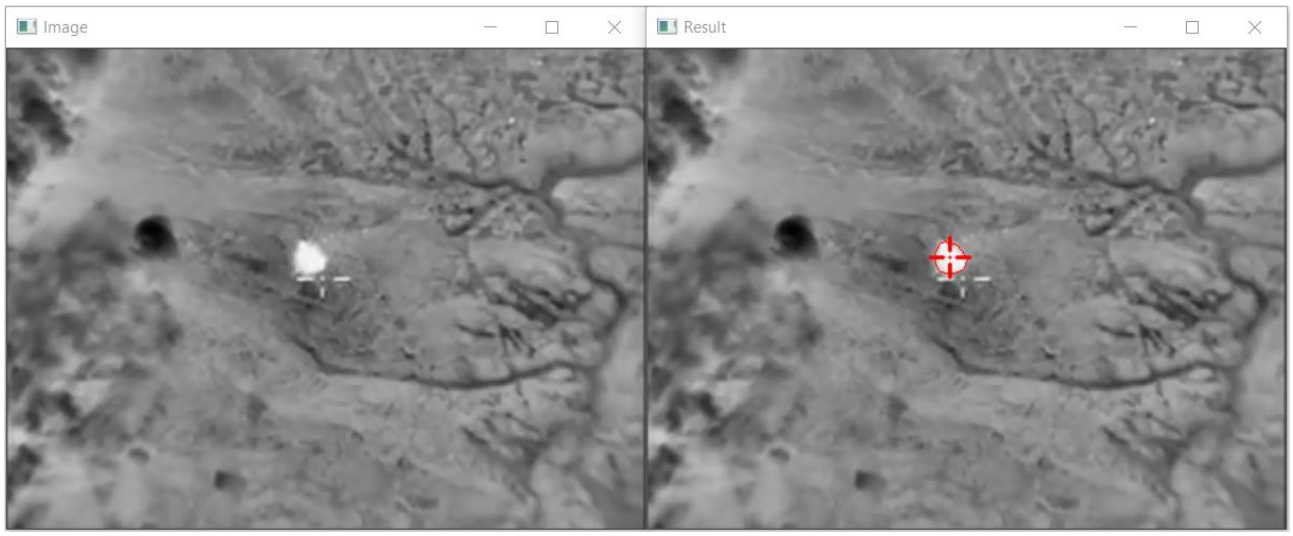

Рисунок 5 — Третье исходное изображение (слева) и
результат поиска укрытий на нём (справа)

Задание 2. Поиск уязвимых мест на изображениях

Второе задание заключается в поиске на цветном изображении области с определённой цветовой палитрой. Это удобно сделать, перейдя из цветового пространства RGB (англ. Red, Green, Blue — красный, зелёный, синий) в цветовое пространства HSV (англ. Hue, Saturation, Value — тон, насыщенность, значение). В этом цветовом пространстве гораздо проще указывать цвета, которые изменяются в диапазоне от 0 до 360 градусов, а также их насыщенность и яркость, которые измеряются в диапазоне от 0 до 100 процентов. Определив диапазоны тонов, насыщенности и яркости, которые требуется искать на изображении, необходимо выполнить пороговую фильтрацию и поиск контуров.

Для выполнения поиска уязвимых мест была разработана функция `findEnemyTarget`. С целью более гибкой настройки диапазонов тонов, насыщенностей и яркостей цветов при выполнении пороговой фильтрации данная функция также выводит изображение в окне и создаёт шесть трекбаров — по два для каждого из параметров для настройки центра диапазона и его ширины. Текущие положения трекбаров записываются соответственно в переменные `hue`, `hueWidth`, `sat`, `satWidth`, `val` и `valWidth`. Окно с выводимым изображением представлено на рисунке 6.

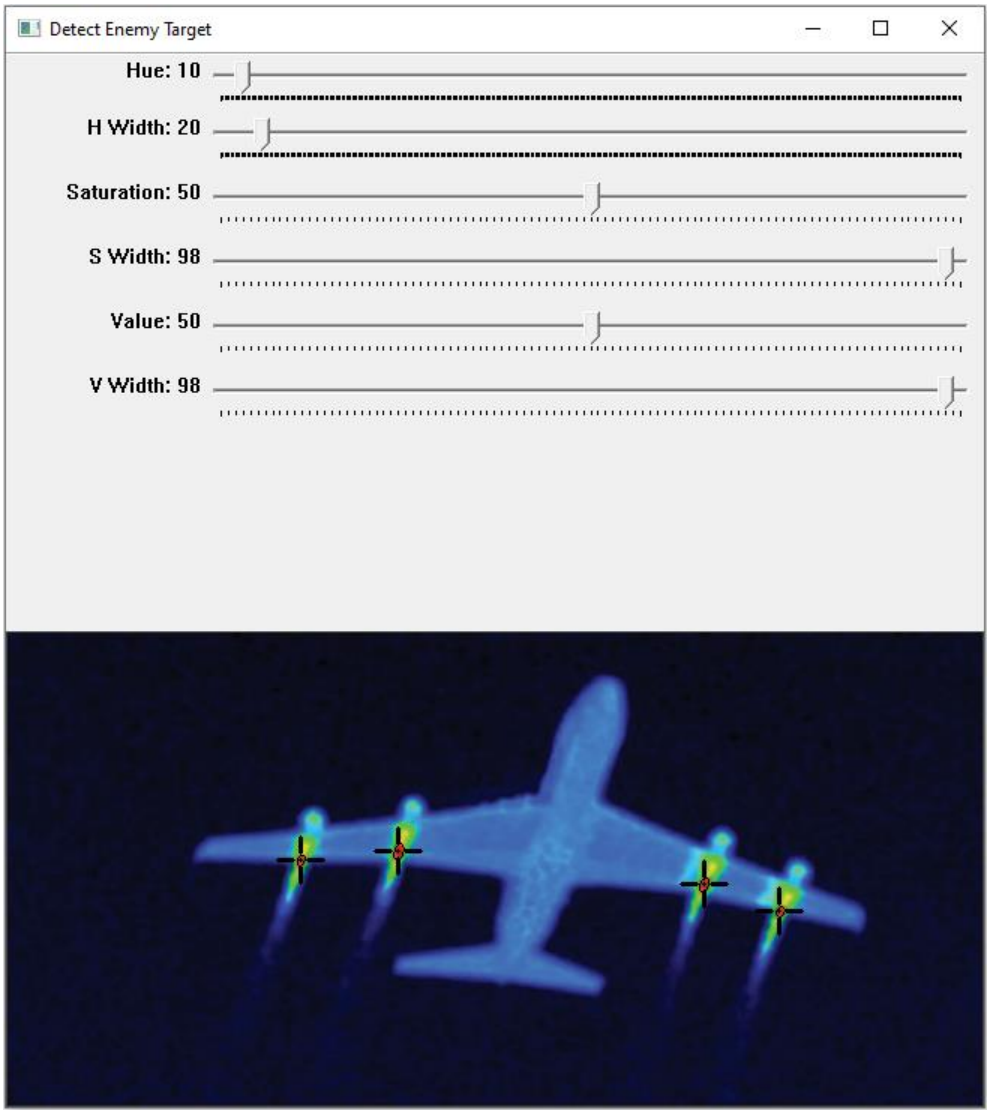

Рисунок 6 — Окно с выводимым функцией `findEnemyTarget` изображением

При изменении положений трекбаров указатели на текущие значения переменных и некоторая служебная информация с помощью указателя `data` передаются во вспомогательную функцию `detectTarget`, которая выполняет основную работу.

В функции `detectTarget` определяются текущие границы диапазонов значений тонов, насыщенностей и яркостей цветов, а затем с помощью функции `inRange` выполняется пороговая фильтрация изображений и производится выделение границ с помощью морфологической операции взятия градиента `morphologyEx` с флагом `MorphTypes::MORPH_GRADIENT`. Для избавления от возникающих шумов перед этапом выделения границ дополнительно выполняется фильтрация изображения открывающим фильтром с помощью морфологической операции `morphologyEx` с флагом `MorphTypes::OPEN`. Результаты проведения пороговой фильтрации и выделения границ представлены на рисунке 7.

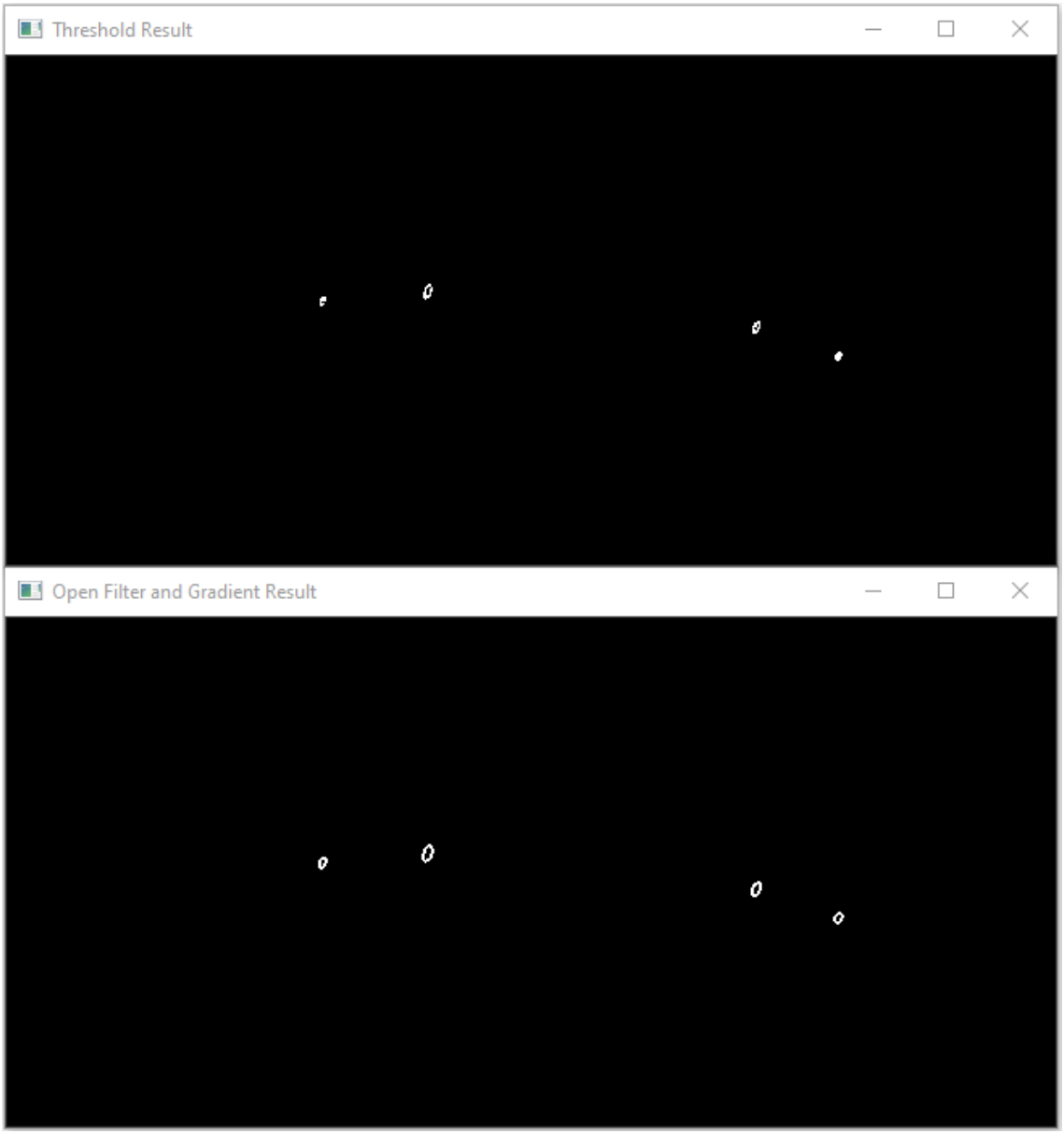

Рисунок 7 — Результаты проведения пороговой фильтрации (сверху) и выделения границ полученных областей (снизу)

Результаты проведения поиска уязвимых мест на всех изображениях представлены на рисунках 8 — 12.

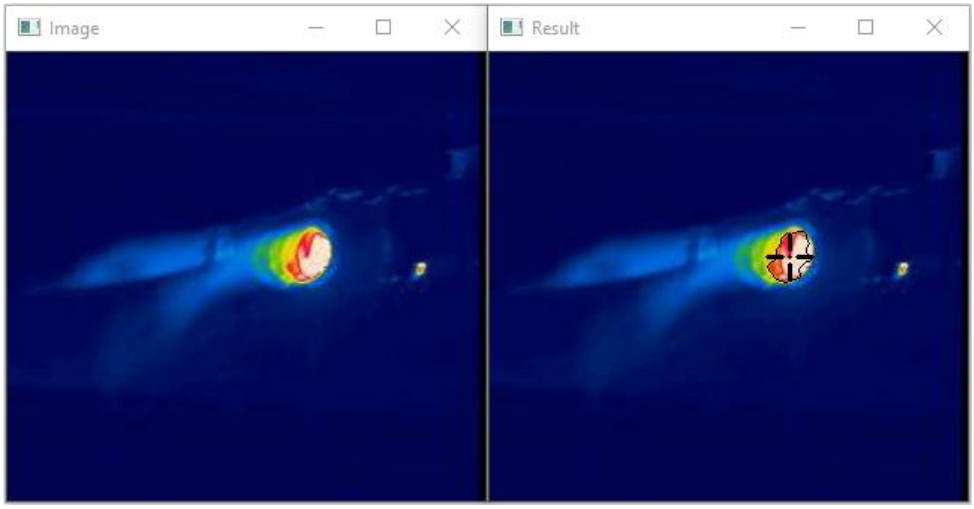

Рисунок 8 — Первое исходное изображение (слева) и результат поиска уязвимых мест на нём (справа)

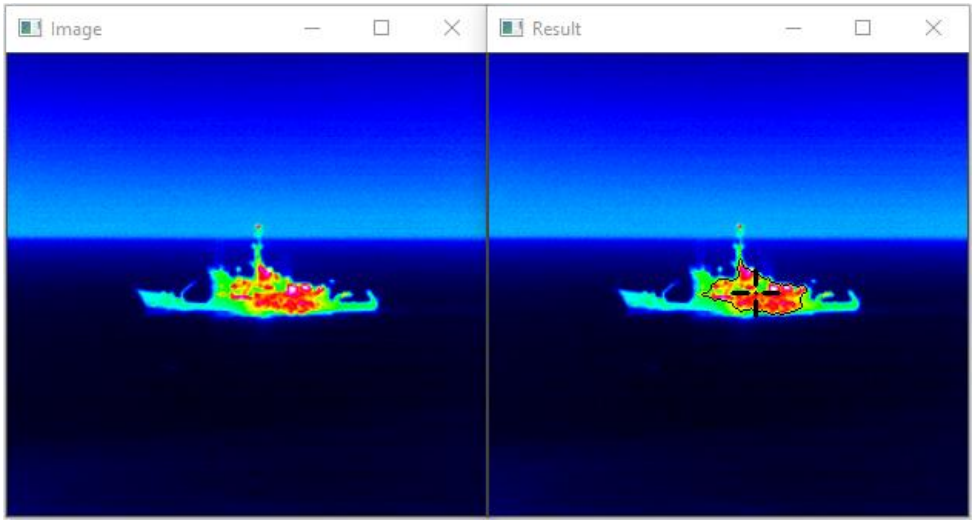

Рисунок 9 — Второе исходное изображение (слева) и результат поиска уязвимых мест на нём (справа)

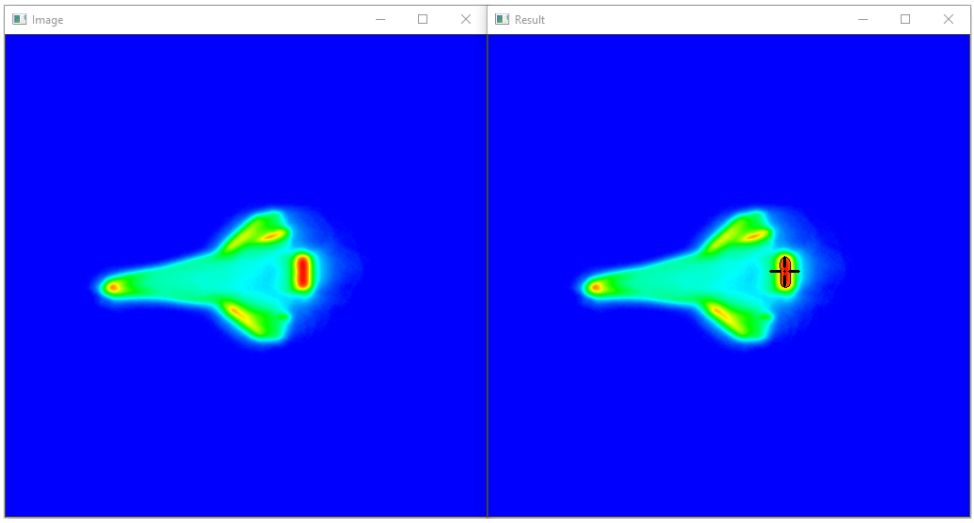

Рисунок 10 — Третье исходное изображение до (слева) и результат поиска уязвимых мест на нём (справа)

Рисунок 11 — Четвёртое исходное изображение (сверху) и результат поиска уязвимых мест на нём (снизу)

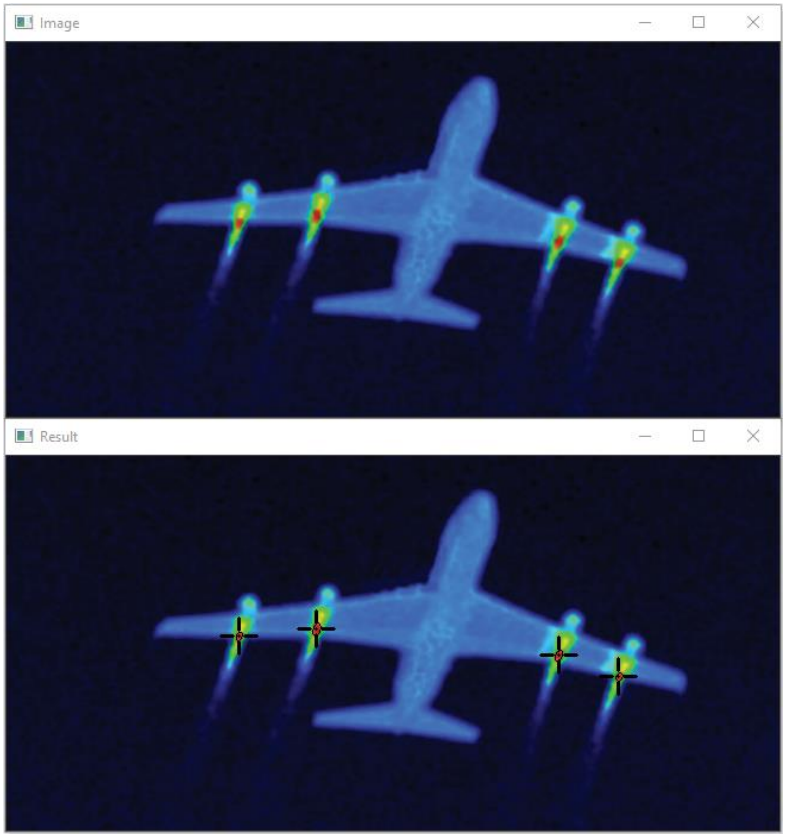

Рисунок 12 — Пятое исходное изображение (сверху) и результат поиска уязвимых мест на нём (снизу)

Задание 3. Определение команд роботов

Третье задание во многом аналогично предыдущему и заключается в поиске на изображении областей с определённой цветовой палитрой. И вновь это удобно сделать, перейдя в цветовое пространство HSV, после чего провести пороговую фильтрацию и поиск контуров.

Для выполнения определения команд роботов, а также поиска лампы и ближайших к ней роботов была разработана функция `findRobotTeams`. По аналогии с предыдущим заданием эта функция выводит изображение в окне и создаёт шесть трекбаров, положения которых записываются в аналогичные переменные. Окно с выводимым изображением представлено на рисунке 13.

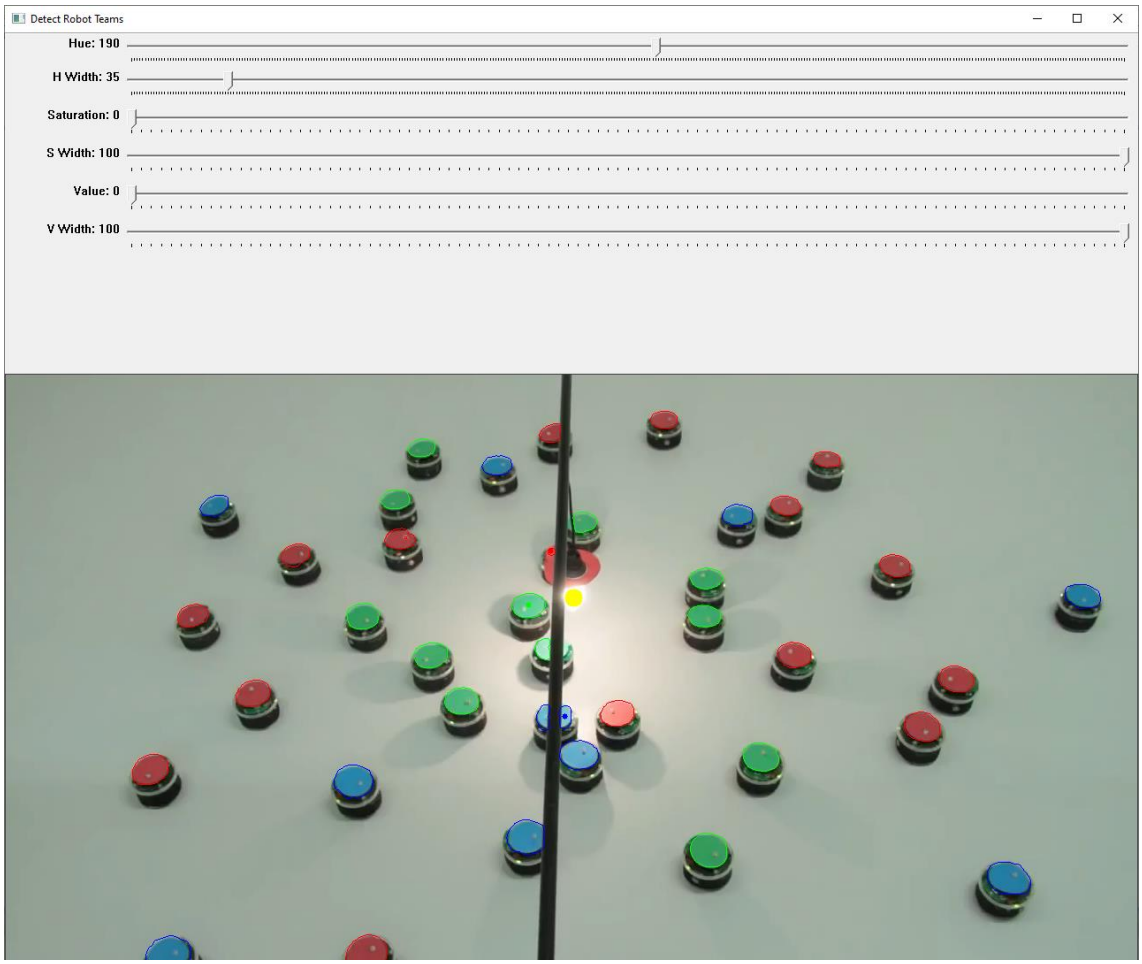

Рисунок 13 — Окно с выводимым функцией `findRobotTeams` изображением

При изменении положений трекбаров указатели на текущие значения переменных и некоторая служебная информация с помощью указателя `data` передаются во вспомогательную функцию `detectRobots`, которая выполняет основную работу.

В функции `detectTarget` определяются текущие границы диапазонов значений тонов, насыщенностей и яркостей цветов, а затем с помощью функции `inRange` выполняется пороговая фильтрация изображений и производится выделение границ с помощью функции `Canny`. Для избавления от возникающих шумов перед этапом выделения границ дополнительно выполняется фильтрация изображения открывающим фильтром с помощью морфологической операции `morphologyEx` с флагом `MorphTypes::OPEN`. Результаты проведения пороговой фильтрации и выделения границ представлены на рисунке 14.

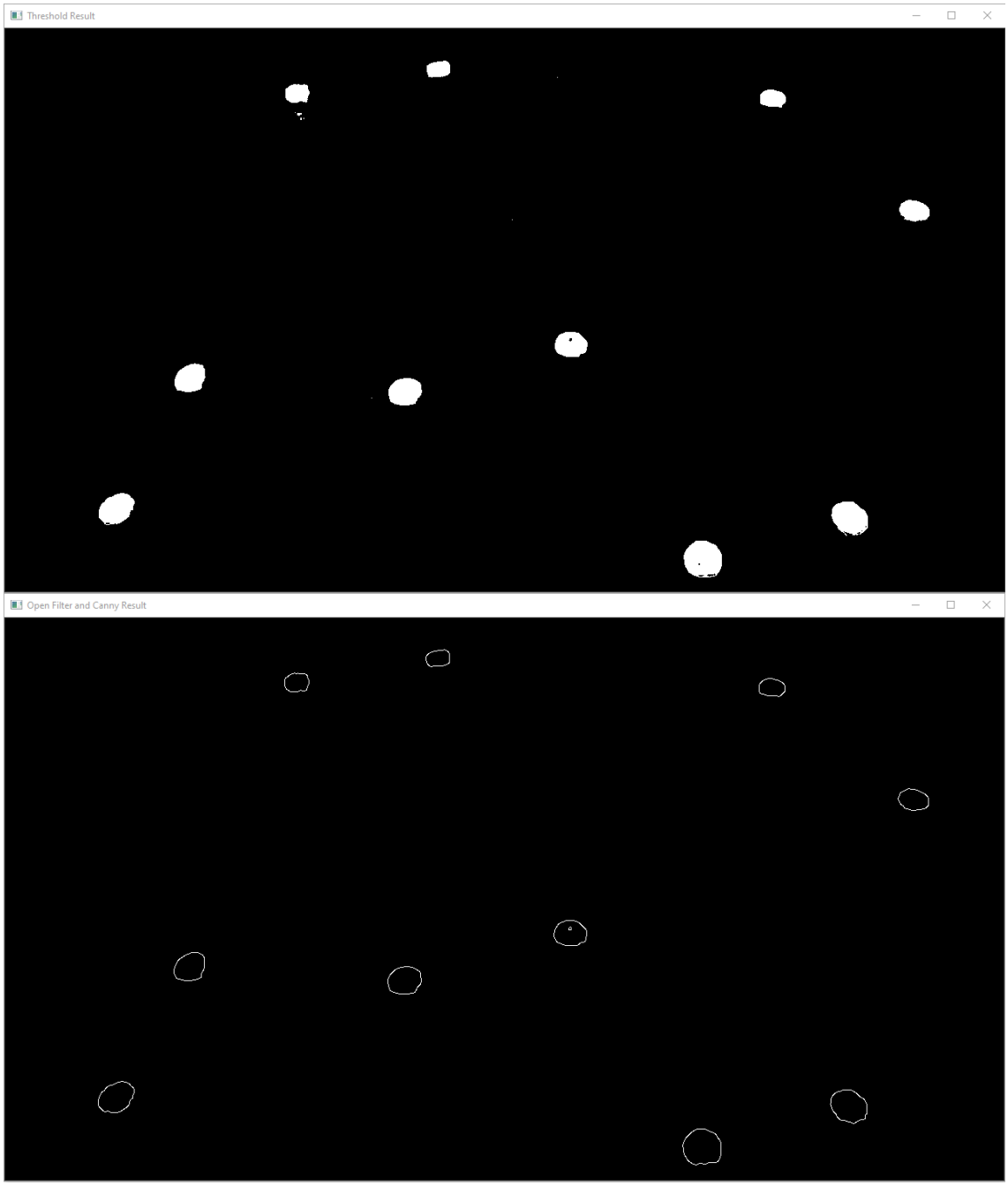

Рисунок 14 — Результаты проведения пороговой фильтрации (сверху) и выделения границ полученных областей (снизу)

Также эта функция выполняет поиск центра лампы по аналогии с первым заданием: исходное изображение переводится в полутоновое с помощью функции `cvtColor` с флагом `ColorConversionCodes::COLOR_BGR2GRAY`, а затем проводится пороговая фильтрация по уровню 0,99 от максимальной интенсивности пикселей и выделение контуров с помощью морфологической операции взятия градиента `morphologyEx` с флагом `MorphTypes::MORPH_GRADIENT`. После этого выделяются центры масс ближайших к лампе роботов из каждой команды. Результаты определения команд роботов на всех изображениях представлены на рисунках 15 и 16.

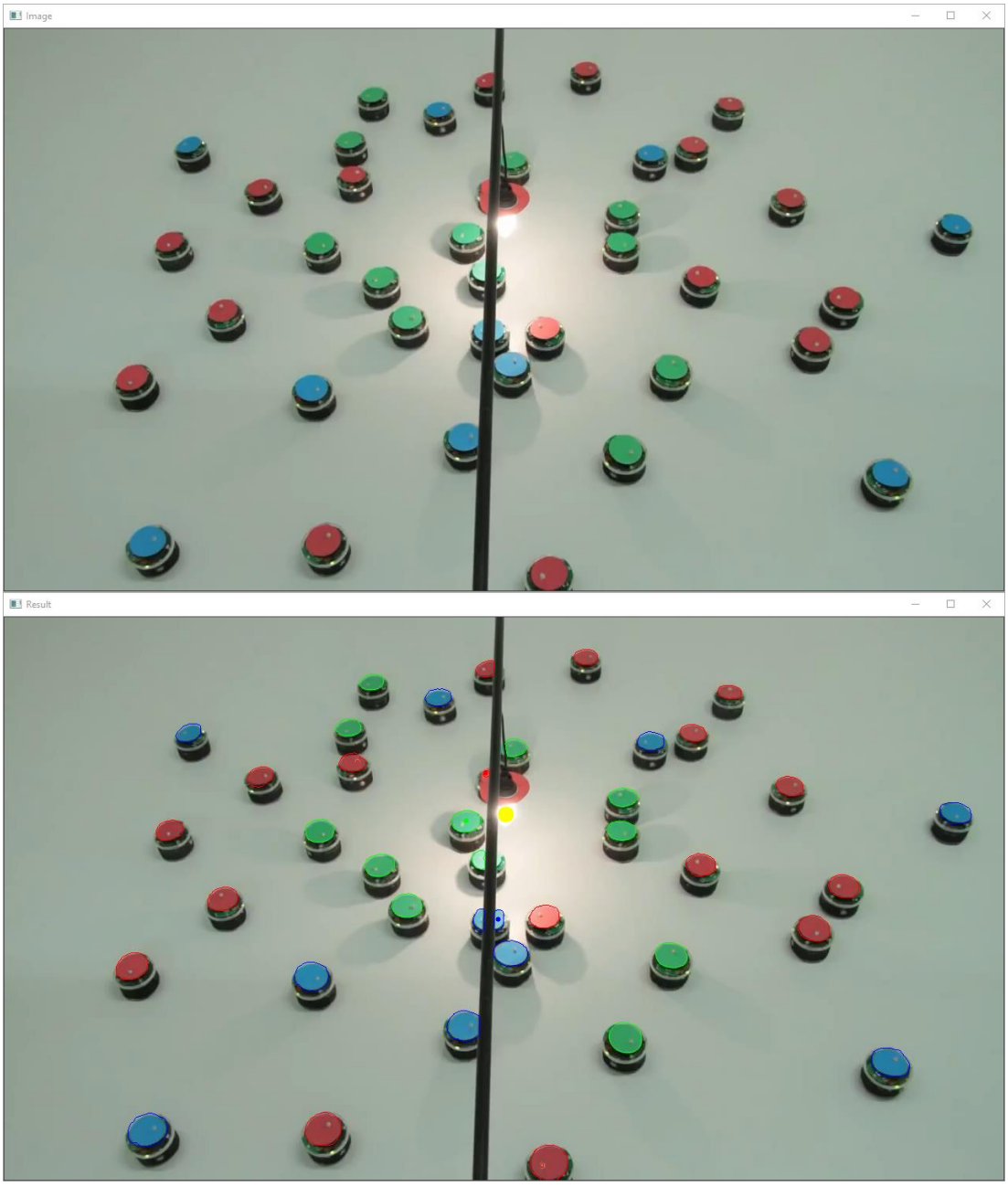

Рисунок 15 — Первое исходное изображение (снизу) и результат определения команд роботов на нём (сверху)

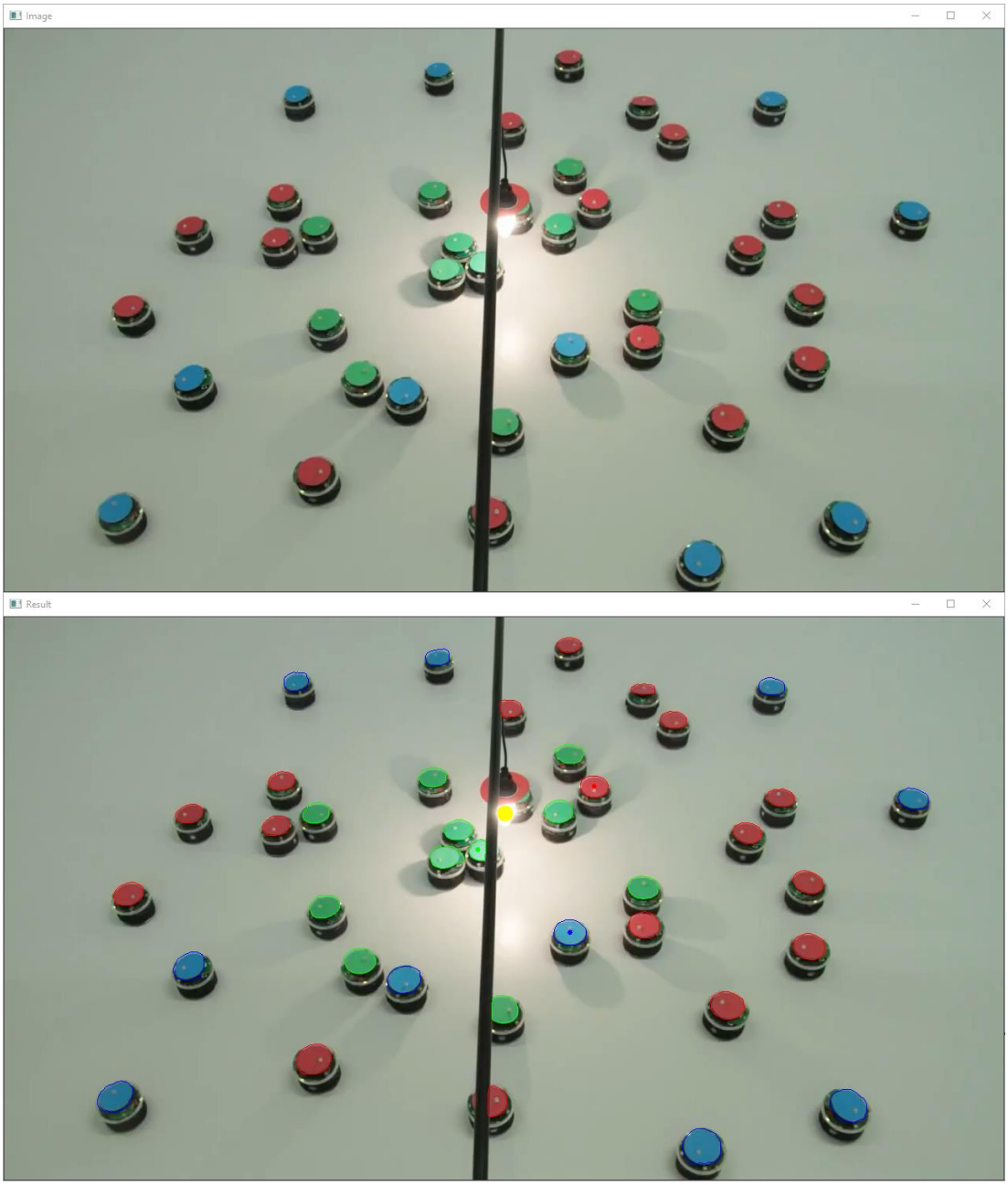

Рисунок 16 — Второе исходное изображение (слева) и
результат определения команд роботов на нём (справа)

Задание 4. Поиск по шаблону

Четвёртое задание заключается в поиске на изображении контуров объектов и последующем сравнении этих контуров с контуром шаблона. В зависимости от результатов сравнения делается выводы о соответствии (исправный гаечный ключ) или несоответствии (бракованный гаечный ключ) шаблону.

Для выполнения определения команд роботов, а также поиска лампы и ближайших к ней роботов была разработана функция `findCorrectAndWrongKeys`.

По аналогии с первым заданием эта функция выводит изображение в окне и создаёт трекбар, положение которого записывается в переменную `thresh`. Окно с выводимым изображением представлено на рисунке 17.

Рисунок 17 — Окно с выводимым функцией `findCorrectAndWrongKeys` изображением

При изменении положения трекбара текущее значение переменной `thresh` и указатель на некоторую служебную информацию `data` передаются во вспомогательную функцию `detectTemplate`, которая выполняет основную работу.

В функции `detectTemplate` сначала выполняется пороговая фильтрация изображения с помощью функции `threshold`, а затем выделение границ с помощью морфологической операции поиска градиента `morphologyEx` с флагом `MorphTypes::MORPH_GRADIENT`. Для избавления от возникающих шумов перед этапом выделения границ дополнительно выполняется фильтрация изображения открывающим фильтром с помощью морфологической операции `morphologyEx` с флагом `MorphTypes::OPEN`. Также фильтрация открывающим фильтром и взятие градиента выполняется и для шаблона с целью выделения его границ. Результаты проведения пороговой фильтрации и выделения границ представлены на рисунке 18.

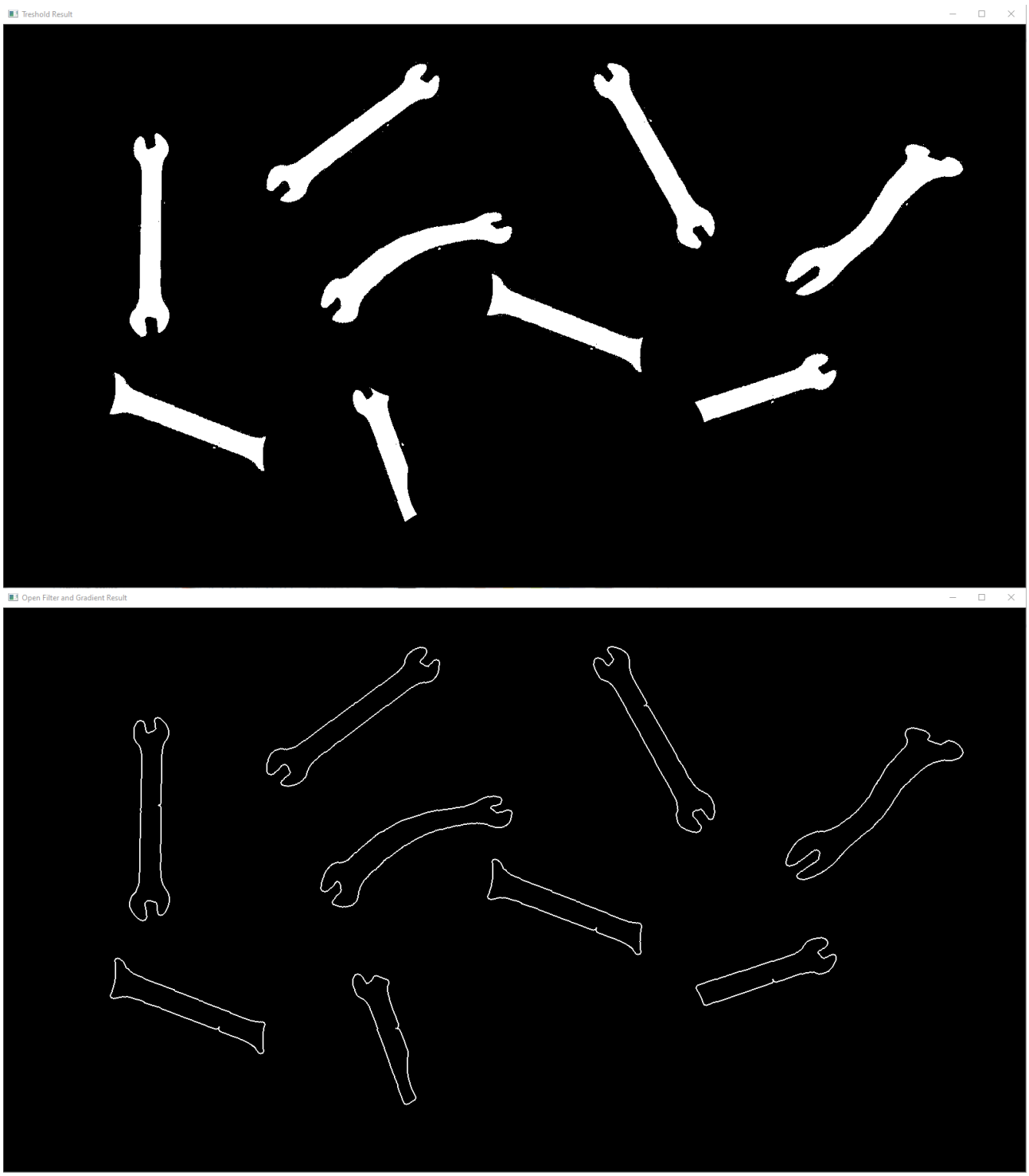

Рисунок 18 — Результаты проведения пороговой фильтрации (сверху) и выделения границ полученной области (снизу)

После выделения границ объектов на изображении и шаблона производится сравнение контуров с шаблоном с помощью функции `matchShapes`, которая сравнивает их центральные моменты. На основании результата, выводимого функцией, делается вывод о соответствии контура объекта контуру шаблона. Чем меньше результат функции, тем больше контур объекта похож на контур шаблона.

Результат сравнения контуров объектов с контуром шаблона представлен на рисунке 19.

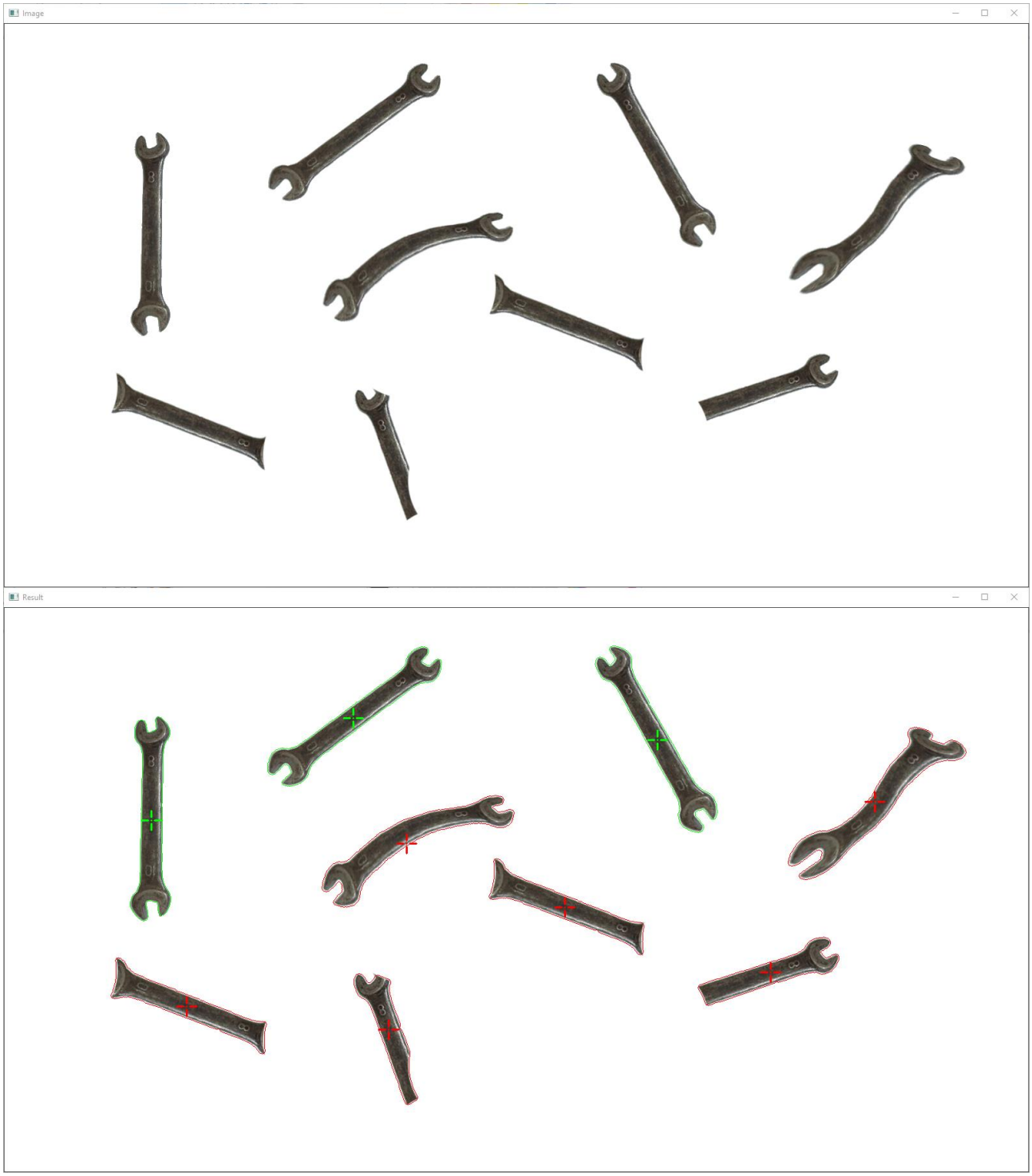

Рисунок 19 — Исходное изображение (сверху) и
результат сравнения контуров объектов с контуром шаблона (снизу)