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 Шкабара Я. А.

Преподаватель Варлашин В. В.

« »_____2020 г.

Санкт-Петербург

2020

Цели

Ознакомиться с некоторыми методами распознавания образов на изображении при помощи контурного анализа с использованием встроенных функций библиотеки *OpenCV*.

Задачи

1. Обозначить примерный центр выделяющегося объекта на изображении (пример изображения на рисунке 1);

Рисунок 1 – Пример изображения для задания 1

2. Обозначить наиболее теплые места на снимках, полученных с помощью тепловизора (пример изображения на рисунке 2);

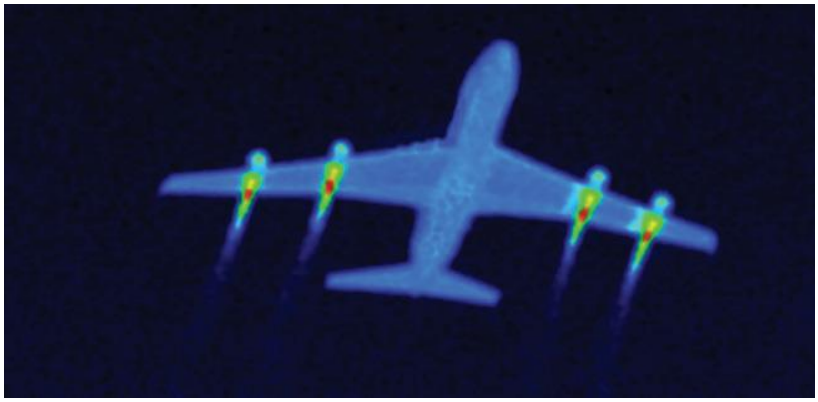

Рисунок 2 – Пример изображения для задания 2

3. На изображении с множеством роботов (рисунок 3):

3.1 Найти на каждом роботе цветную крышку и обвести контуром цвета его команды;

3.2 Найти лампу и обозначить ее;

3.3 Для каждой команды нарисовать центр масс ближайшего к лампе робота;

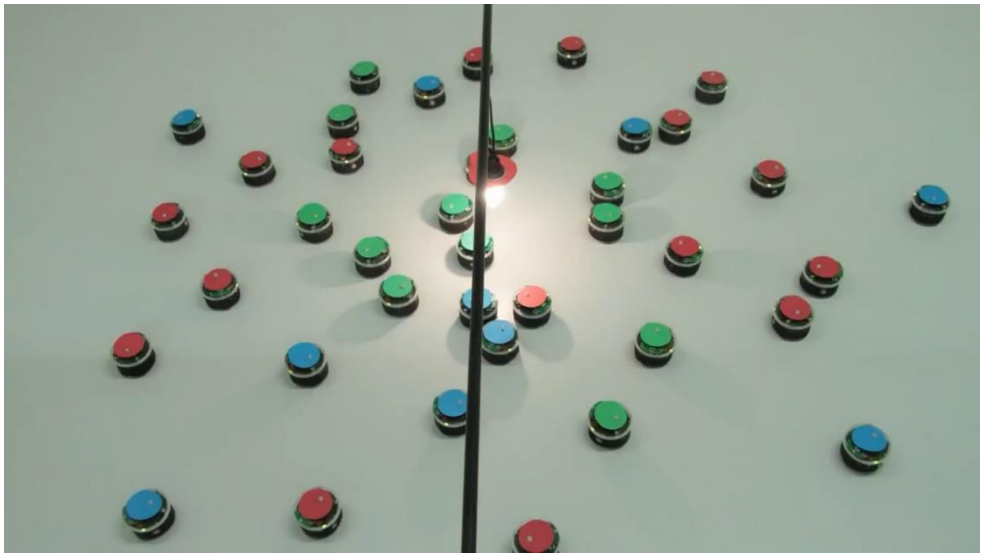

Рисунок 3 – Пример изображения для задания 3

4. Обозначить бракованные (не соответствующие шаблону) и пригодные гаечные ключи разными метками (рисунок 4).

Рисунок 4 – Изображения для задания 4

Задание 1

Алгоритм выполнения задания 1:

1. Использовать функцию *threshold* для пороговой фильтрации, чтобы выделить объекты, которые резко выделяются относительно фона;
2. Использовать функцию *erode* для удаления шумов;
3. Использовать функцию *dilate* для восстановления размеров участков изображения после эрозии;
4. Использовать функцию Canny для выделения границ;
5. При помощи функции *findContours* найти контуры на изображении;
6. Определить центры масс контуров с помощью функции *getCenterOfMass* (представлена на странице 6) и пометить их.

Результат выполнения алгоритма представлен на рисунке 5 справа (слева находится исходное изображение):

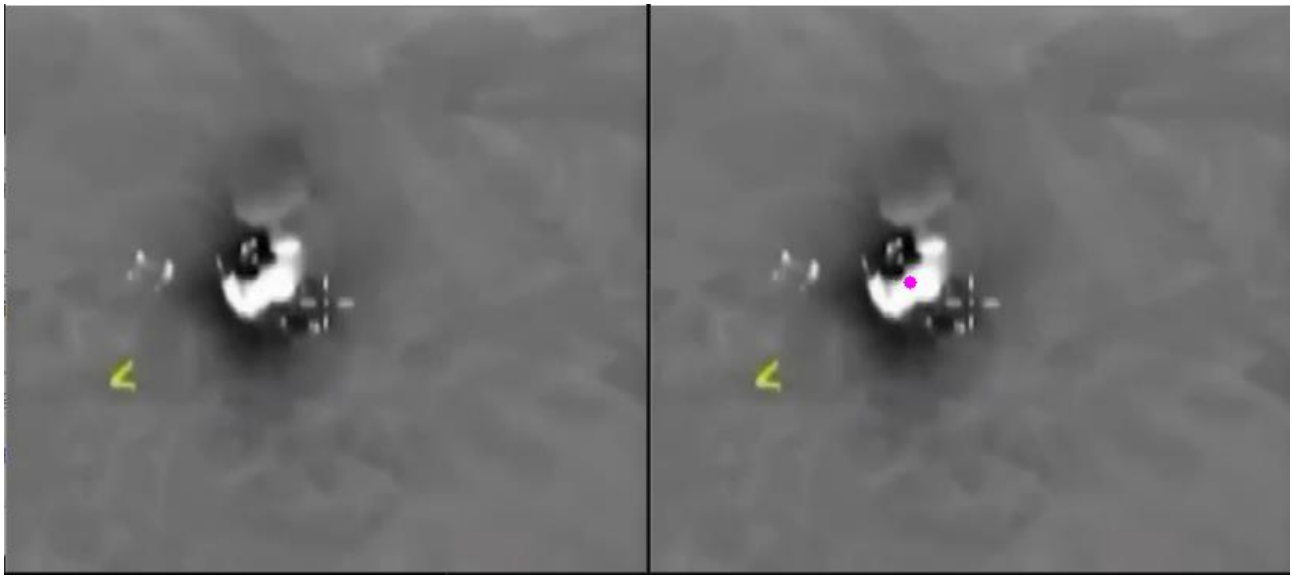

Рисунок 5 – Результат выполнения задания 1

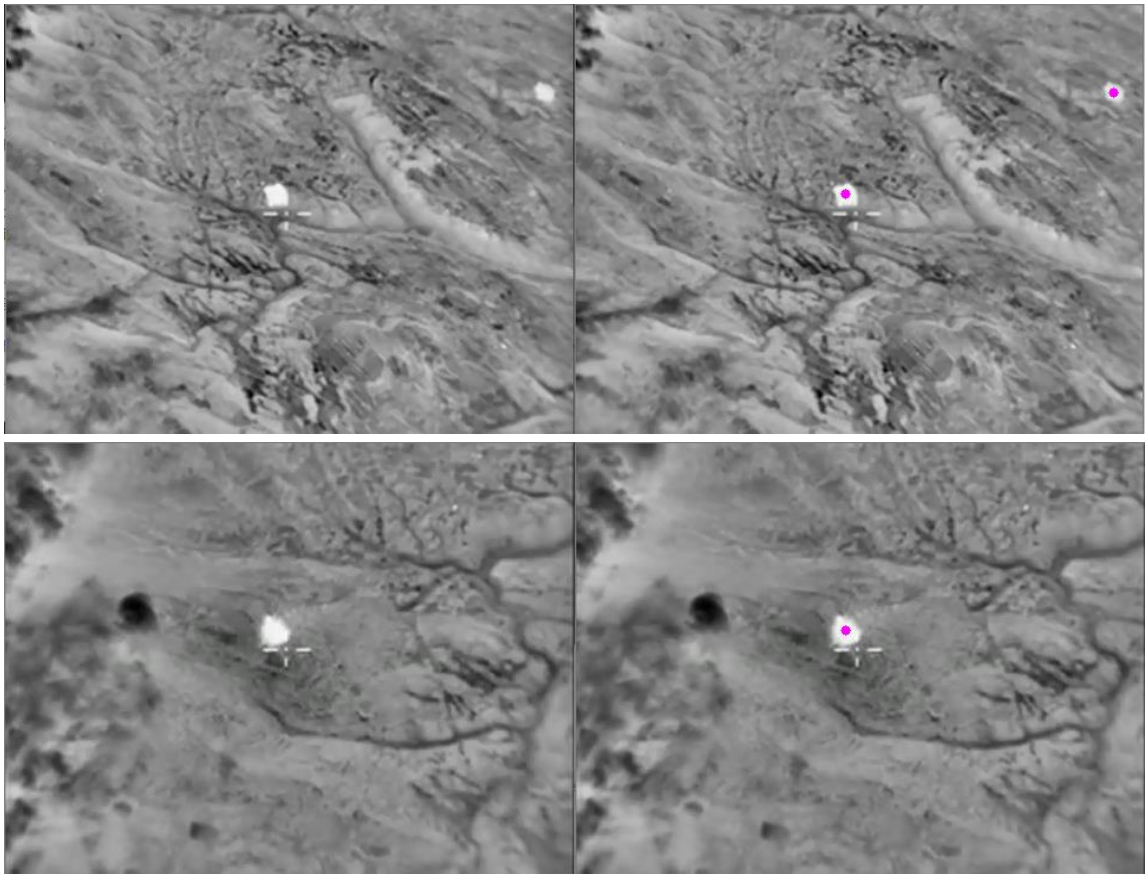

Рисунок 5 – Результат выполнения задания 1 (продолжение)

Задание 2

Алгоритм выполнения задания 2:

1. Перевести изображение из шкалы BGR в шкалу HSV при помощи функции *cvtColor*;
2. Использовать функцию *inRange* для пороговой фильтрации.
3. Использовать функцию *erode* для удаления шумов;
4. Использовать функцию *dilate* для восстановления размеров участков изображения после эрозии;
5. Использовать функцию Canny для выделения границ;
6. При помощи функции *findContours* найти контуры на изображении;
7. Определить центры масс контуров с помощью функции *getCenterOfMass* и пометить их.

```
void getCenterOfMass(vector<vector<Point>> contours, vector<Point>
&centers)
{
	vector<Point> ctrs;
	for (int i = 0; i < contours.size(); i++)
	{
		Moments mom = moments(contours[i]);
		Point center;
		center.x = mom.m10 / mom.m00;
		center.y = mom.m01 / mom.m00;
		ctrs.push_back(center);
	}

	centers = ctrs;
	return;
}
```

Результат выполнения алгоритма представлен на рисунке 6.

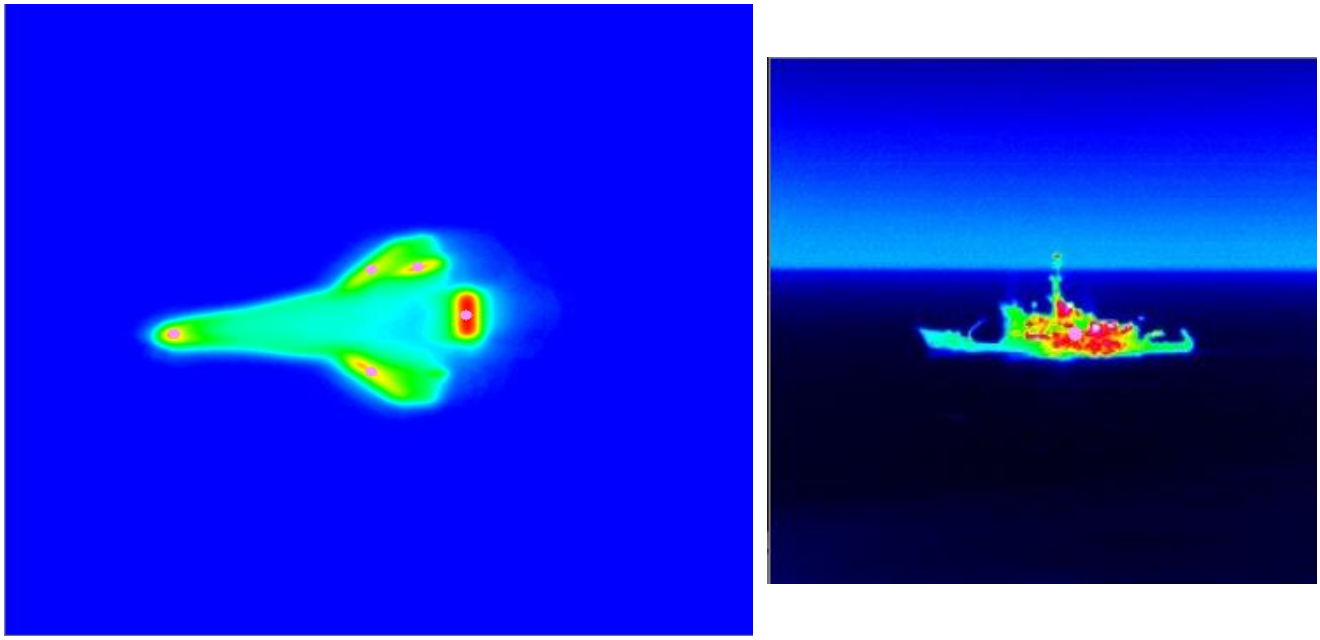

Рисунок 6 – Результат выполнения задания 2

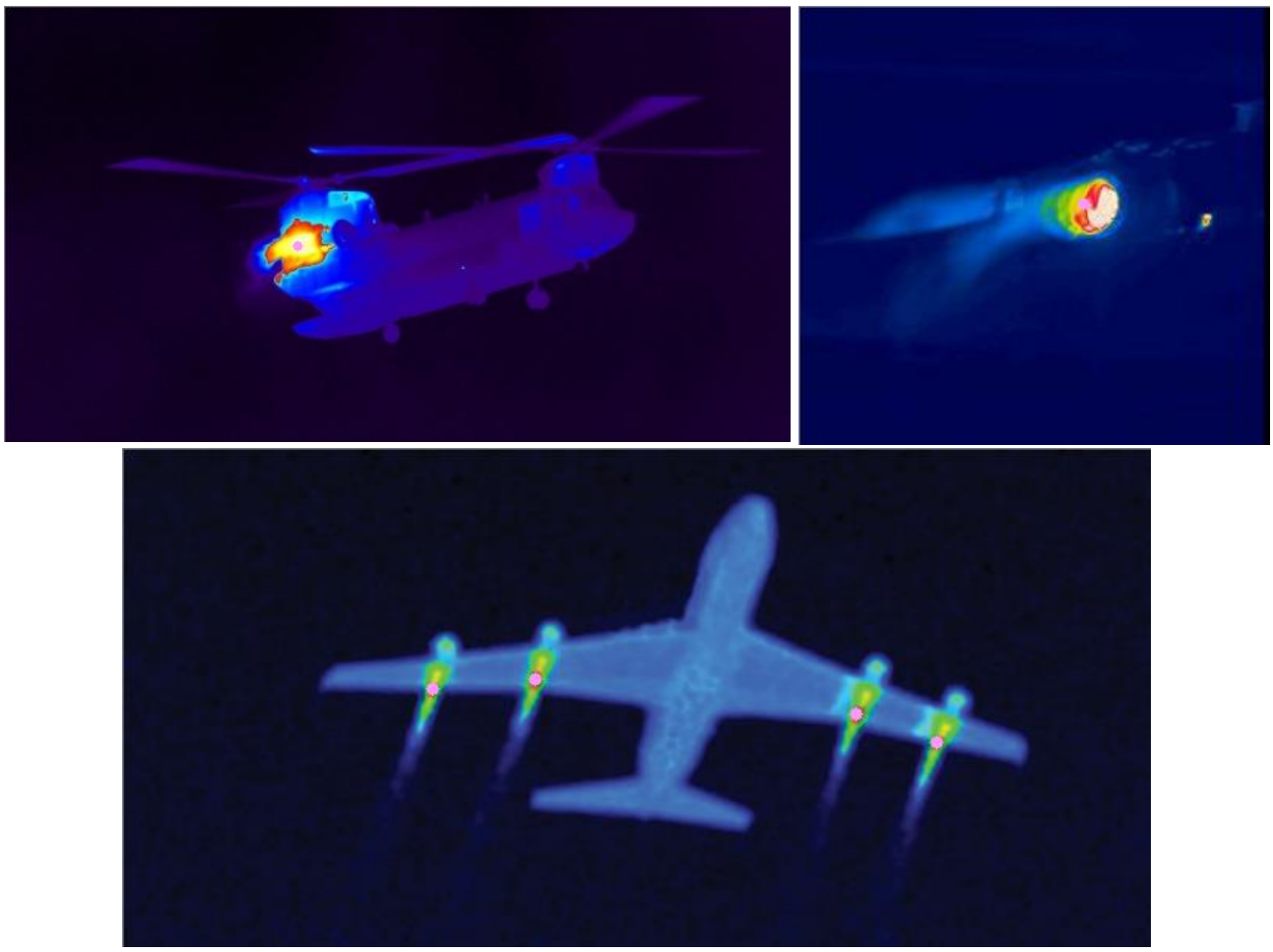

Рисунок 6 – Результат выполнения задания 2 (продолжение)

Задание 3

Для нахождения контуров роботов и лампы используется тот же подход, что и в задании 2.

Алгоритм выполнения задания 3:

1. Поиск лампы;
2. Поиск красных роботов;
   - ❖ Так как положение камеры и лампы неизменно от снимка к снимку, то во избежание ложного срабатывания детектора на красный абажур лампы, он закрывается эллипсом аналогичного размера.
3. Поиск зеленых и синих роботов;
4. Поиск ближайшего к лампе робота каждого цвета;

- В данной части за точку расположения лампы принимается ее проекция на поверхность пола, которая находится примерно в центре изображения.
- Для обводки крышки ближайшего робота используется функция *drawNeares*:

```
void drawNearest(vector<vector<Point>> contours, Point source, Mat
&forDrawing, Scalar color)
{
	vector<Point> ctrs;
	getCenterOfMass(contours, ctrs);

	int min_value = 1000, min_index = 0;

	for (int i = 0; i < ctrs.size(); i++)
	{
		float dist = sqrt(pow((ctrs[i].x - source.x), 2) +
pow((ctrs[i].y - source.y), 2));
		if (dist < min_value)
		{
			min_index = i;
			min_value = dist;
		}
	}

	circle(forDrawing, ctrs[min_index], 4, color, FILLED);
	line(forDrawing, ctrs[min_index], source, color, 1);

	return;
	}
```

Результат выполнения задания 3 представлен на рисунке 7.

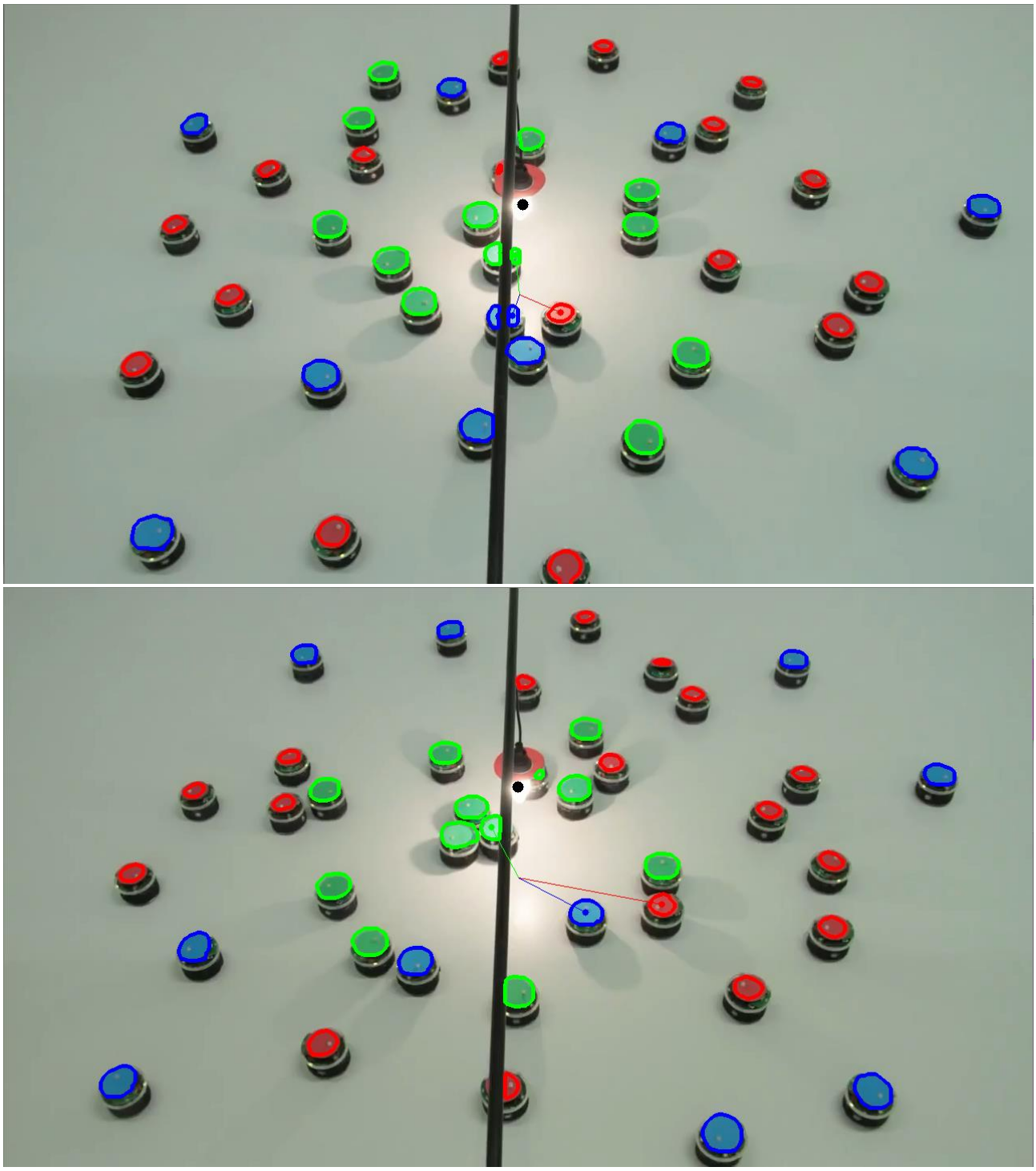

Рисунок 7 – Результат выполнения задания 3

Задание 4

Алгоритм выполнения задания 4:

1. Определение контуров ключей на исходном изображении и изображении шаблона как в задании 1;
2. Сравнение контуров ключей с контуром шаблона с помощью функции *matchShapes*;
3. Выделение контуров пригодных ключей зеленым и непригодных ключей красным цветом;

2 и 3 пункты реализованы в функции *highlight*:

```
void highlight(vector<vector<Point>> contours, vector<vector<Point>>
templat_contour, Mat &forDrawing)
{
	for (int i = 0; i < contours.size(); i++)
	{
		double diff = matchShapes(contours[i], templat_contour[0],
CONTOURS_MATCH_I2, 0);
		if (diff < 0.1) polylines(forDrawing, contours[i], true, Scalar(50,
255, 50), 2);
		else polylines(forDrawing, contours[i], true, Scalar(50, 50, 255),
2);
	}
	return;
	}
```

Результат выполнения задания 4 представлен на рисунке 8.

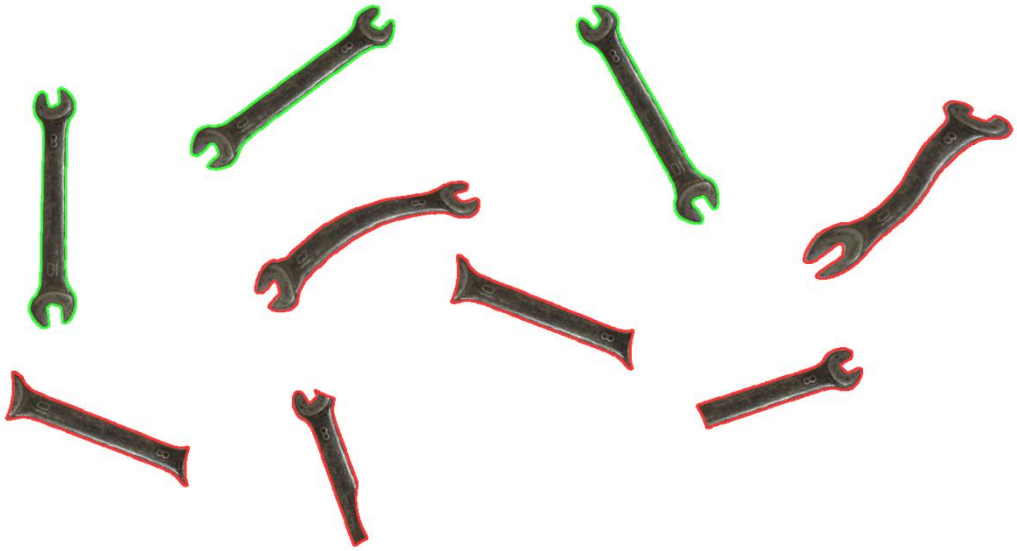

Рисунок 8 – Результат выполнения задания 4