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 <*подпись*> А.А. Ларионов

Преподаватель <*подпись*> В.В. Варлашин

«__»_____2020 г.

Санкт-Петербург
2020

# ЗАДАНИЕ

Определить место попадания снаряда на тепловизионных изображениях с воздуха и отметить произвольным знаком его центр.

Определить местоположение моторного отделения транспортных средств на тепловизионных изображениях и отметить произвольным знаком его центр.

На изображениях с роботами разных цветов и лампой отметить каждого робота контуром соответствующего цвета, обозначить лампу и найти ближайшего робота к лампе, выделив его центр масс.

На изображении с бракованными и качественными гаечными ключами классифицировать их путем сравнения с шаблоном и отметить разными метками.

Для выполнения задания использовать библиотеку *OpenCV 4.0*.

# РЕШЕНИЕ ПЕРВОГО ЗАДАНИЯ

Алгоритм решения первого задания следующий:

1. нахождение первоначального значения порога, как среднее арифметическое интенсивностей всех пикселей;
2. пороговая фильтрация с помощью встроенной функции *threshold()*;
3. эрозия и последующая дилатация с ядром 9×9 и якорем в центре с помощью встроенных функций *erode()* и *dilate()*;
4. нахождение границ встроенной функцией *Canny()* с нижним порогом 10 и верхним 255;
5. поиск и выделение контуров встроенными функциями *findContours()* и *drawContours()*;
6. если количество найденных контуров равно 1, то происходит определение центра контура через результат встроенной функции *moments()* и выход из программы;
7. в противном случае значение порога увеличивается на единицу, и происходит переход в пункт 2.

Результат работы алгоритма приведен на рисунке 1.


Рисунок 1 – Решение первого задания

## РЕШЕНИЕ ВТОРОГО ЗАДАНИЯ

Алгоритм решения второго задания следующий:

1. пороговая фильтрация изображения в формате *HSV* по красному цвету с помощью встроенной функции *inRange()*;
2. дилатация с ядром 15×15 и якорем в центре с помощью встроенной функции *dilate()*;
3. нахождение границ встроенной функцией *Canny()* с нижним порогом 10 и верхним 255;
4. поиск контуров встроенной функцией *findContours()* и нахождение среди них наибольшего;
5. определение и выделение центра контура через результат встроенной функции *moments()*;
6. выход из программы.

Результат работы алгоритма приведен на рисунке 2.

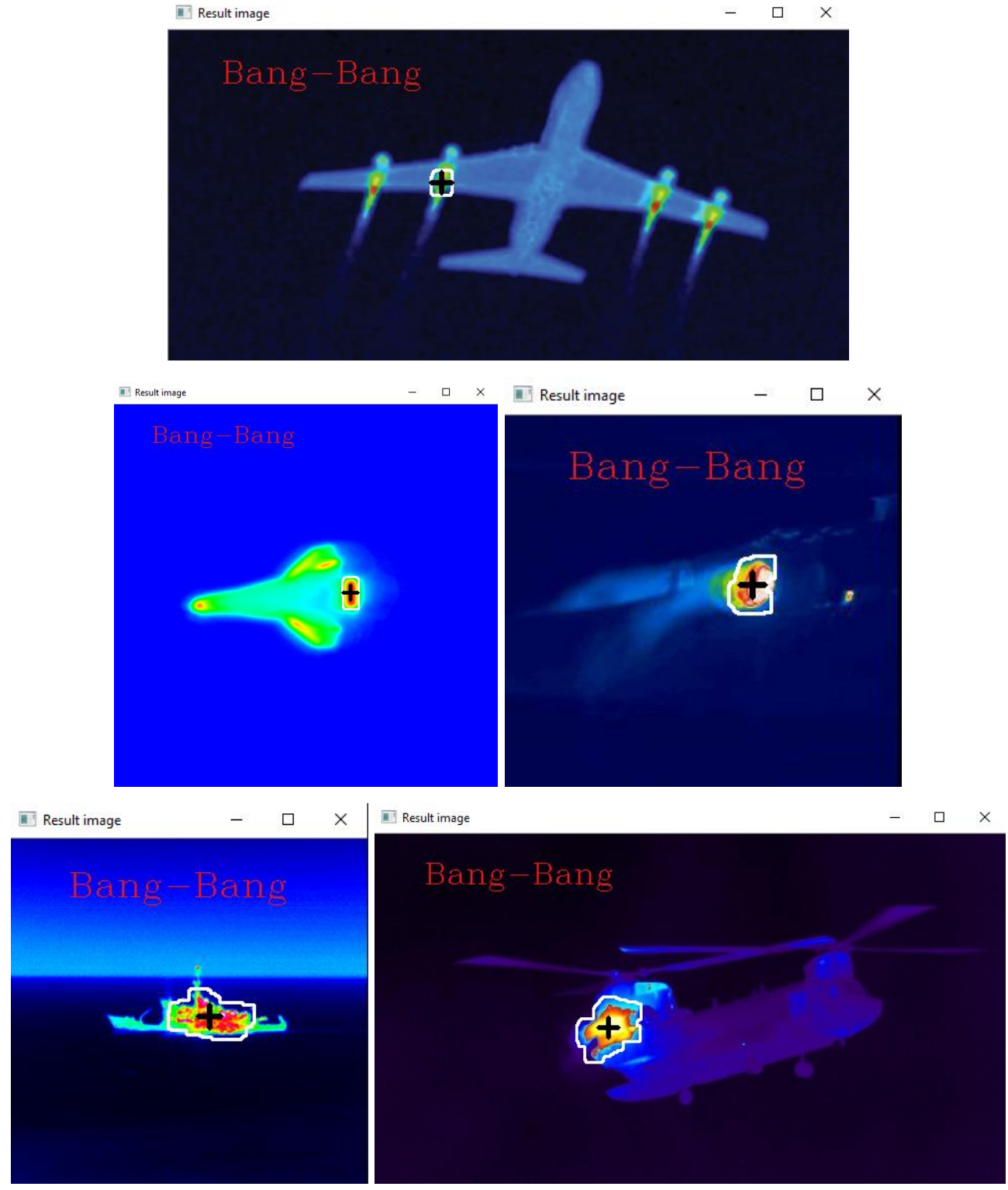


Рисунок 2 – Решение второго задания

## РЕШЕНИЕ ТРЕТЬЕГО ЗАДАНИЯ

Алгоритм решения третьего задания следующий:

1. пороговая фильтрация изображения в формате *HSV* по цвету каждого из искомых объектов с помощью встроенной функции *inRange()*;
2. эрозия и последующая дилатация с ядром 9×9 и якорем в центре с помощью встроенных функций *erode()* и *dilate()*;
3. нахождение границ встроенной функцией *Canny()* с нижним порогом 10 и верхним 255;
4. поиск контуров встроенной функцией *findContours()*;
5. определение центра каждого контура через результат встроенной функции *moments()*;
6. рисование в центре каждого контура фигуры соответствующего цвета с помощью встроенной функции *ellipse()*;
7. определение проекции лампы на пол как точки пересечения вертикальной оси лампы и центральной строки изображения;
8. нахождение ближайшего к лампе объекта из каждой группы и выделение его центра;
9. выход из программы.

Картинки были предварительно обработаны в программе *Paint*, чтобы исключить влияние цвета ободка лампы на конечный результат.

Результат работы алгоритма приведен на рисунке 3.

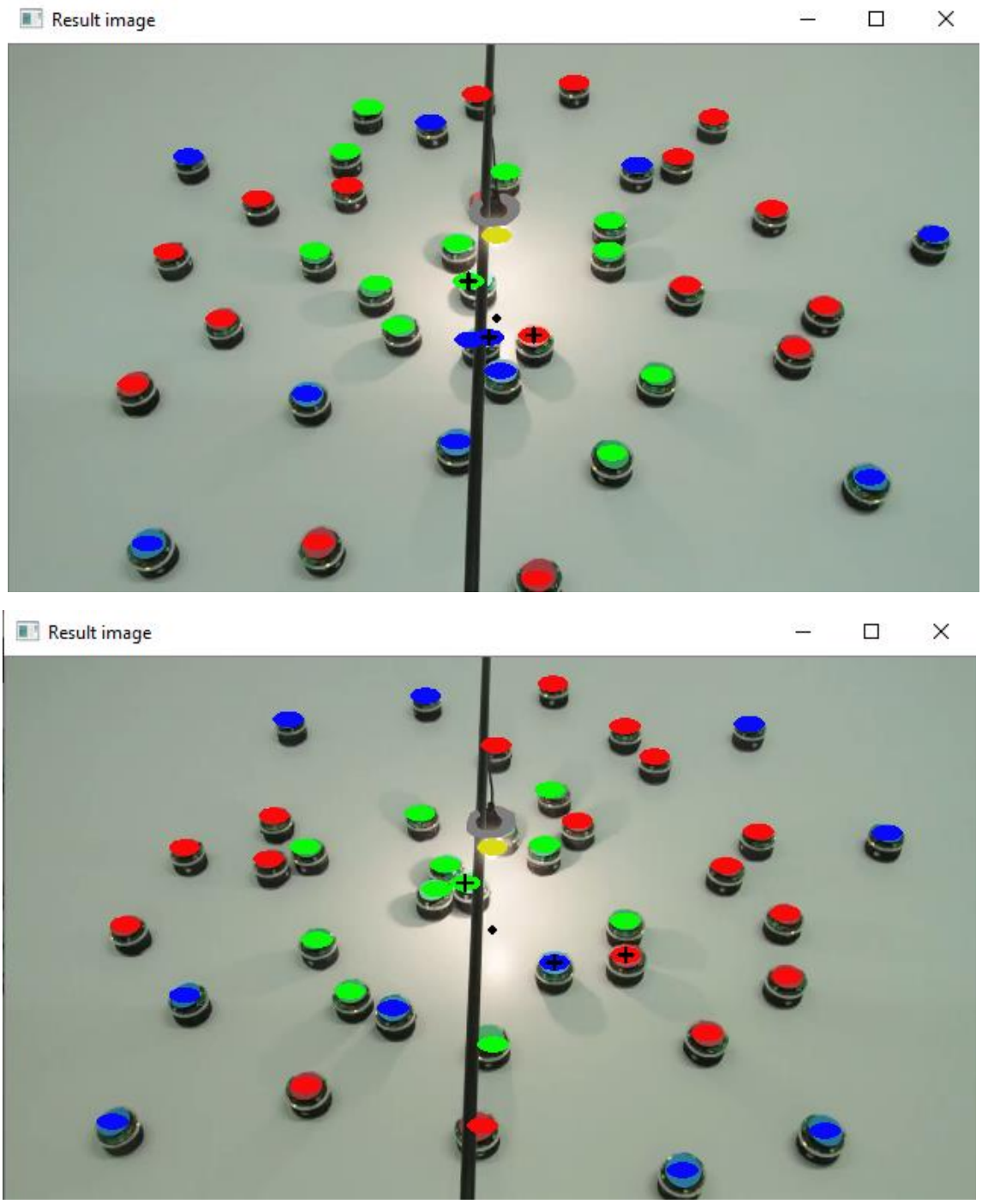


Рисунок 3 – Решение третьего задания

## РЕШЕНИЕ ЧЕТВЕРТОГО ЗАДАНИЯ

Алгоритм решения четвертого задания следующий:

1. фильтрация изображения с гаечными ключами ядром Гаусса 5×5 и якорем в центре с помощью встроенной функции *GaussianBlur()*;
2. пороговая фильтрация изображения с гаечными ключами с помощью встроенной функции *threshold()*;
3. эрозия и последующая дилатация с ядром 7×7 и якорем в центре с помощью встроенных функций *erode()* и *dilate()*;
4. нахождение границ встроенной функцией *Canny()* с нижним порогом 10 и верхним 255;
5. поиск контуров встроенной функцией *findContours()* на изображении с гаечными ключами и на изображении шаблона;
6. сравнение контуров на изображении с гаечными ключами и контура шаблона с помощью встроенной функции *matchShapes*();
7. определение центра каждого контура на изображении с гаечными ключами через результат встроенной функции *moments()*;
8. выделение центра каждого контура соответствующим символом в зависимости от результата функции *matchShapes*();
9. выход из программы.

Результат работы алгоритма приведен на рисунке 4.

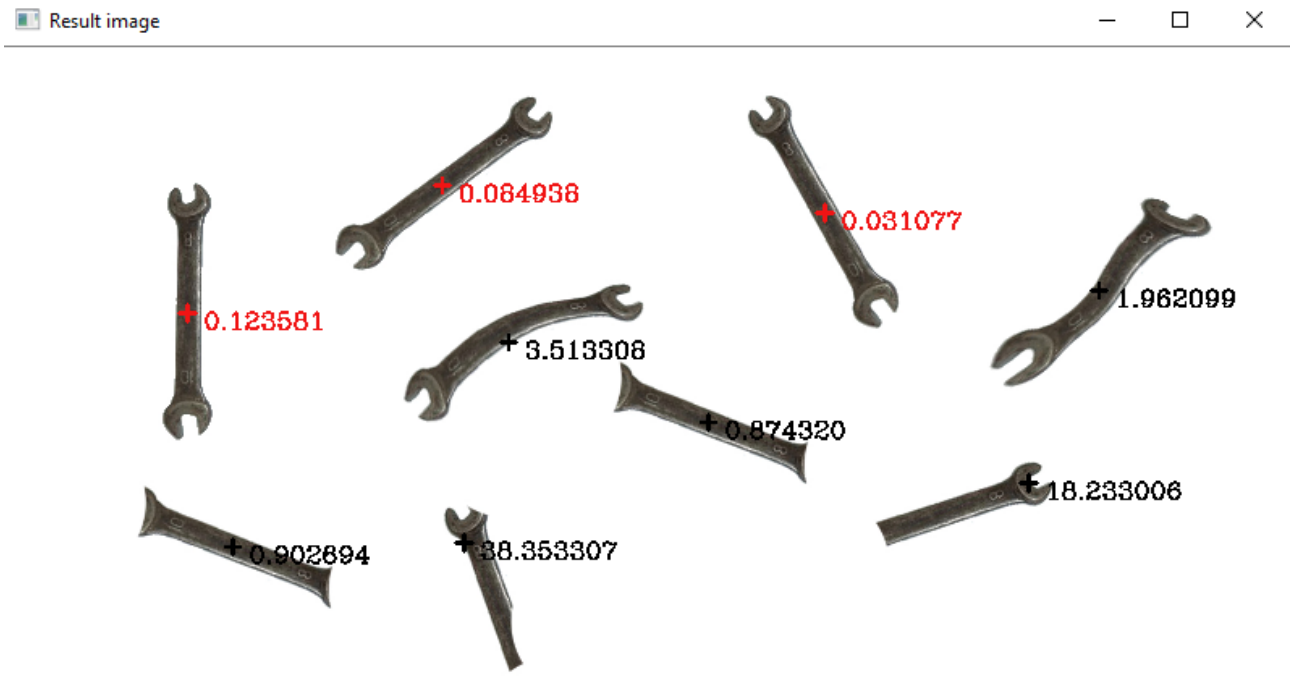


Рисунок 4 – Решение четвертого задания

## ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ

Из-за особенностей исходных изображений в третьем задании определение количества контуров, т.е. предполагаемых роботов того или иного цвета, происходит неверно. Подобная ситуация показана на рисунке 5. Контуры 6 и 7, 4 и 5, 1 и 2 на изображениях сверху вниз прежний алгоритм посчитает за разных роботов.

Для решения этой задачи дополним алгоритм решения третьего задания:

1. пороговая фильтрация изображения в формате *HSV* по цвету искомого объекта и заранее подобранным параметрам насыщенности и яркости с помощью встроенной функции *inRange()*;
2. применение закрывающего фильтра (дилатация и последующая эрозия) с ядром 9×9 и якорем в центре с помощью встроенных функций *dilate()*и *erode()*;
3. нахождение границ встроенной функцией *Canny()* с нижним порогом 10 и верхним 255;
4. поиск контуров встроенной функцией *findContours()*;
5. определение центра каждого контура и поиск максимального радиуса среди всех контуров как окружностей через результат встроенной функции *moments()*;
6. расчет расстояния между центрами контуров по координате X, если центры контуров имеют разницу по координате Y не более чем максимальный радиус;
7. сравнение со значением выражения "2,1 · максимальный радиус";
8. если значение меньше, то центры двух контуров заменяются одним, координаты которого вычисляются как среднее арифметическое координат центров;
9. рисование в каждом центре фигуры с помощью встроенной функции *ellipse()*;
10. выход из программы.

Результат работы обновленного алгоритма приведен на рисунке 6.

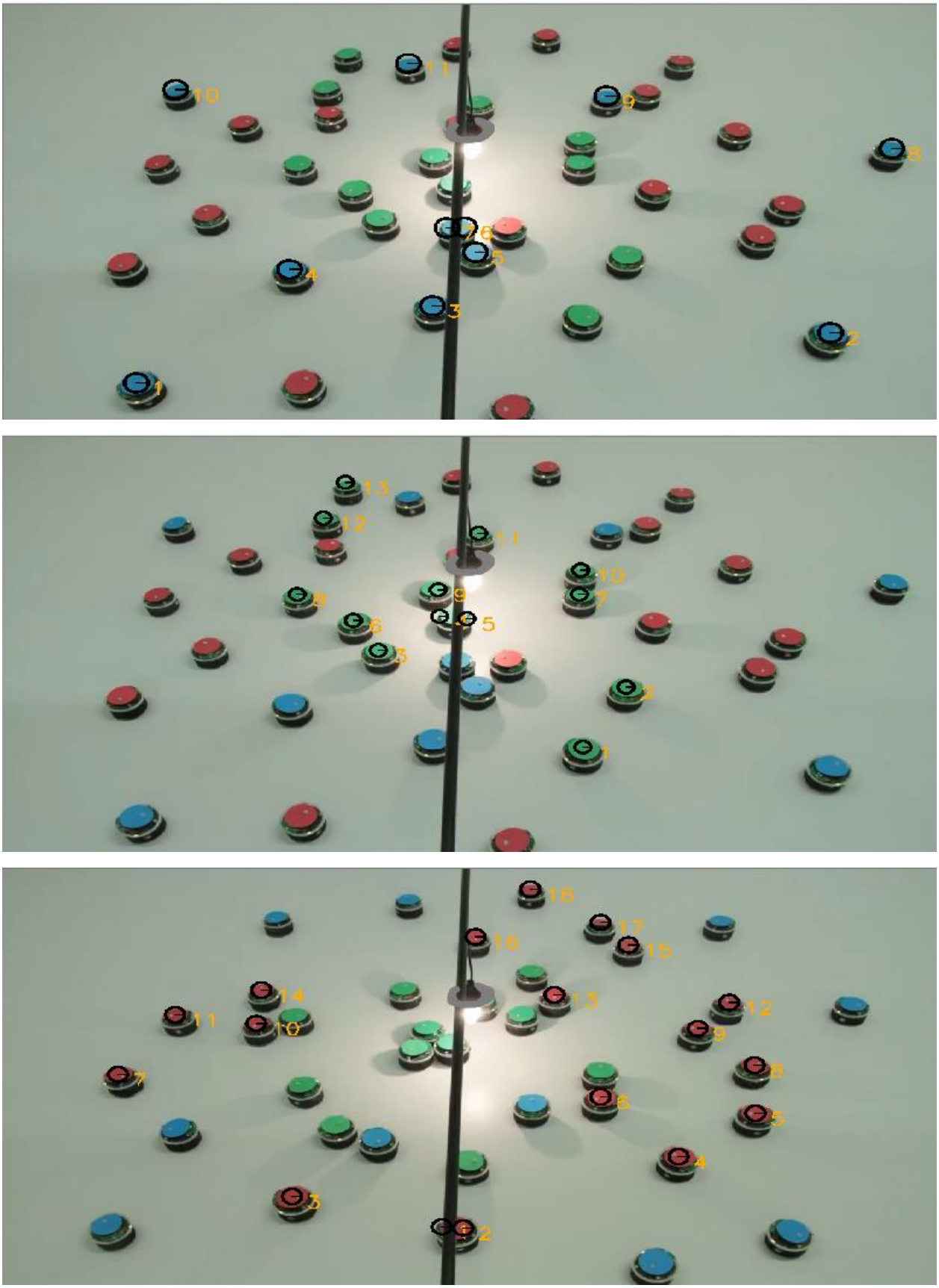

Рисунок 5 – Проблема алгоритма

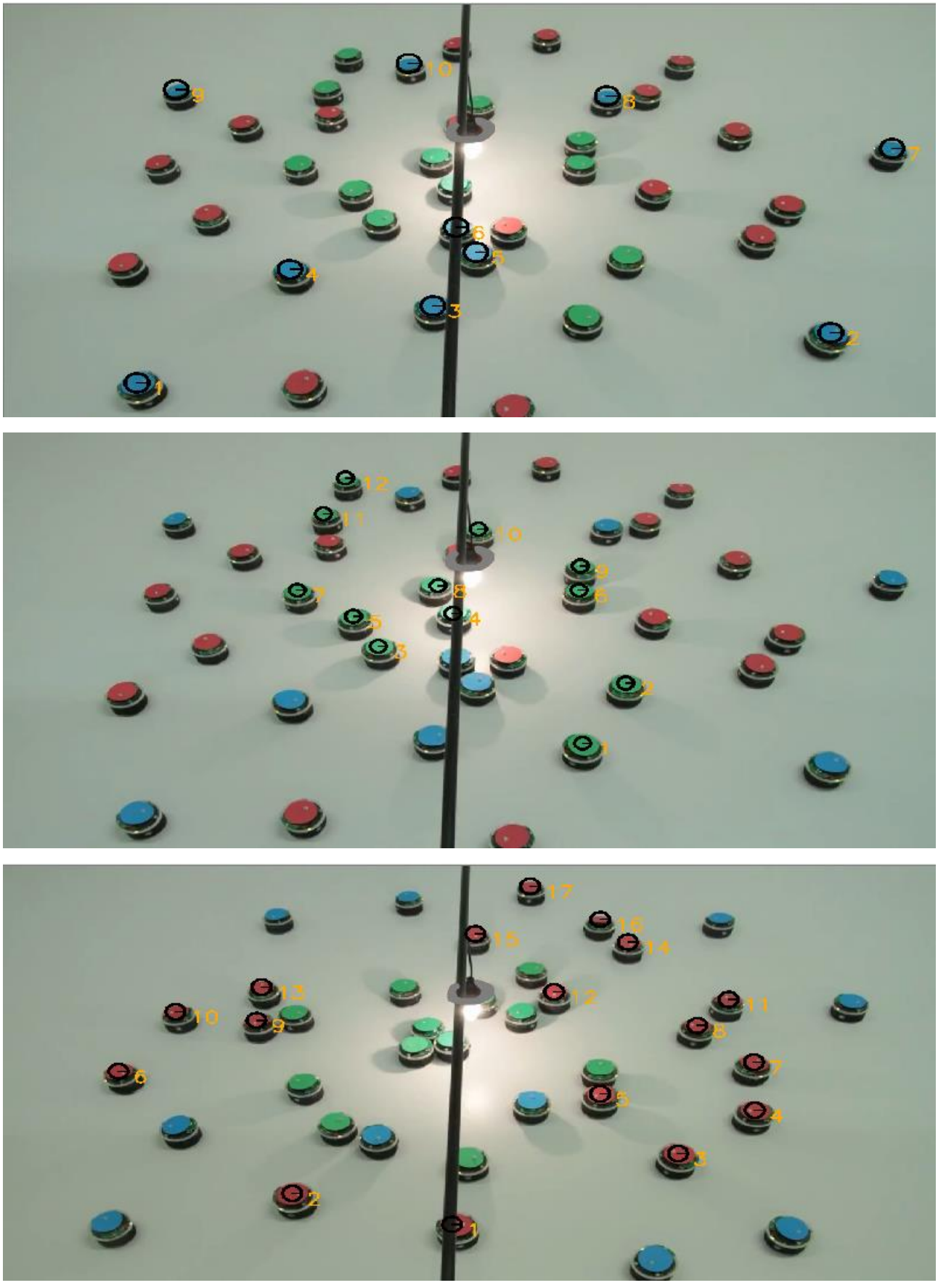


Рисунок 6 – Обновленный алгоритм