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 Самарин А.С.

Преподаватель Варлашин В.В.

« »_____2020 г.

Санкт-Петербург

2020

Задание 1

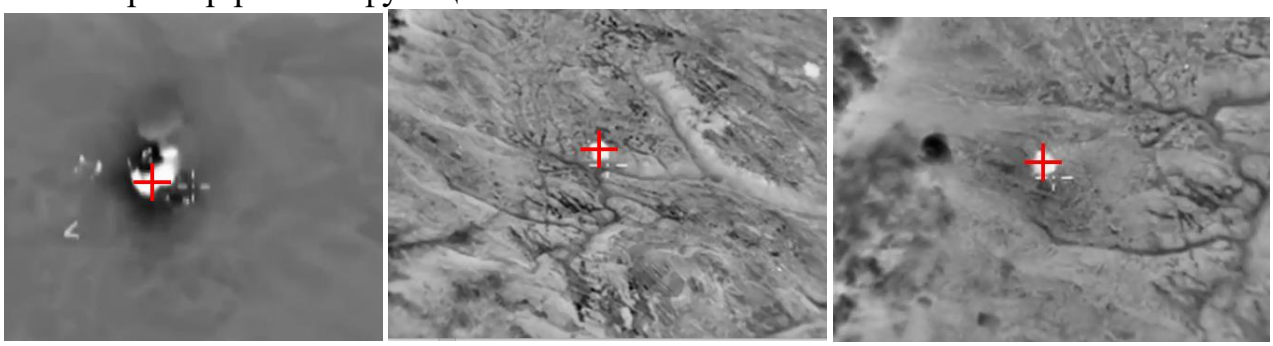
В задании 1 используются однотонные изображения, на которых нужно определить более яркий участок и найти его центр.

Для решения этой задачи была реализована функция *part1(Mat inputImage)*. На вход функция принимает одноканальное изображение.

Алгоритм работы функции:

- Пороговая фильтрация функций *threshold(inputImage, temp, 220, 255, THRESH_BINARY);*
- Создание контура из маски функцией *Canny(temp, edges, 0, 0, 3, false)*;
- Определение моментов контуров;
- Нахождение центра масс контура;
- Отрисовка «прицела».

Пример работы функции:



Задание 2

В задании 2 необходимо найти самое теплое место на снимках с тепловизора. Для решения этой задачи реализована функция *part2(Mat inputImage).*

Алгоритм работы функции:

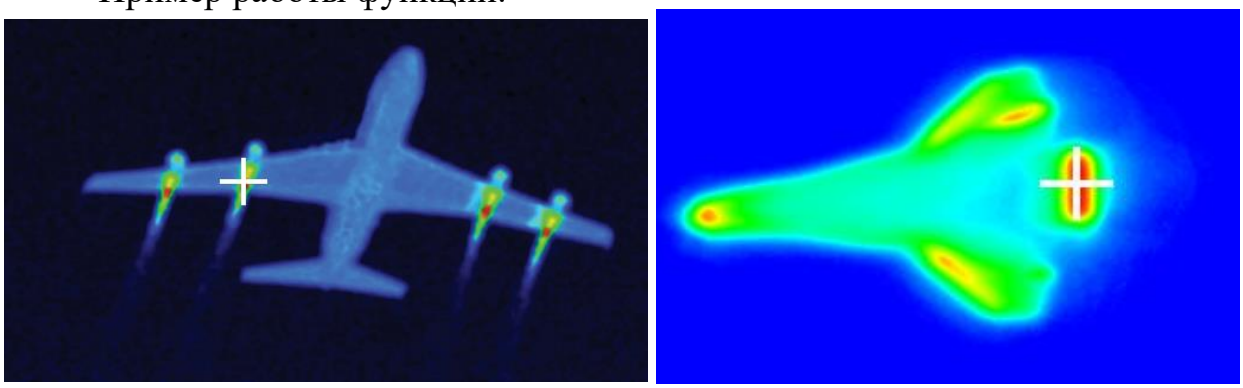
- Перевод изображение в цветовое пространство *HSV*;
- Пороговая фильтрация функций *inRange(imputImage, Scalar(34, 101, 0), Scalar(164, 255, 255), temp);*
- Создание контуров из маски функцией Canny;
- Определение моментов контуров;
- Определение контура с самым большим периметром;
- Определение центра масс для найденного контура
- Отрисовка «прицела».

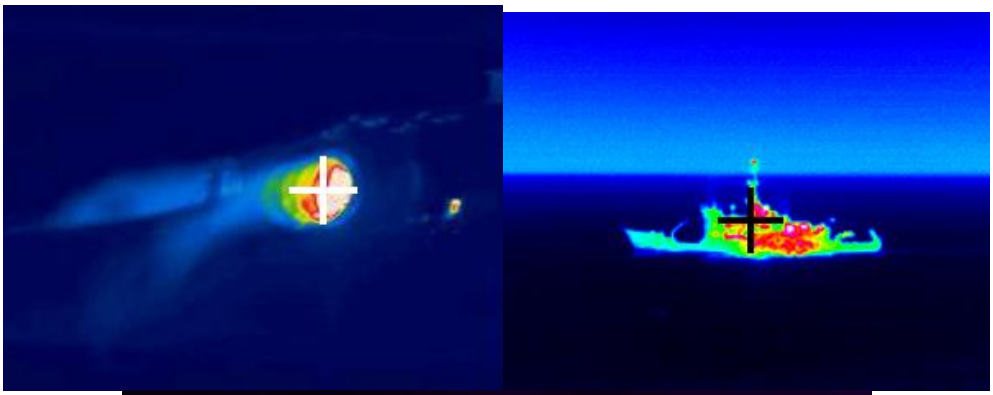
Поиск контура с наибольшим периметром:

```
//Рассчет моментов для найденных контуров
    vector<Moments> mnts;
    int i_m = cnts.size();
    for (int i = 0; i < i_m; i++)
    {
        mnts.push_back(moments(cnts[i]));
    }

    int i_max = 0; //Индекс наибольшего контура
    double tempArea = mnts[0].m00;
    for (int i = 0; i < cnts.size(); i++)
    {
        if (mnts[i].m00 >= tempArea)
        {
            i_max = i;
            tempArea = mnts[i].m00;
        }
    }
```

Пример работы функции:





Задание 3

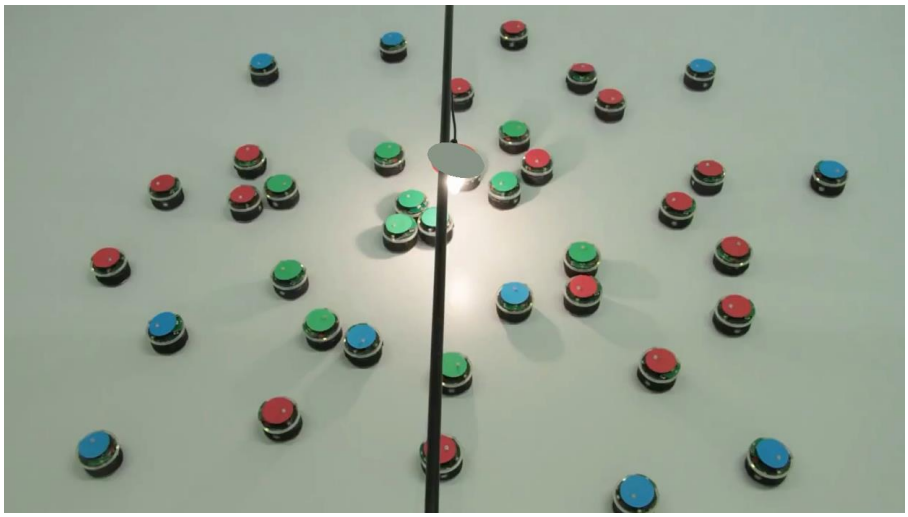
В задании 3 необходимо выделить робота каждой группы соответствующим цветом и найти роботов ближе всех расположенных к лампе. Для выполнения задания реализована функция *part3*.

В процессе выполнения возникла следующая проблема: лампа имеет корпус такого же цвета, как и роботы красной группы, поэтому при поиске робота нужного цвета находился и корпус лампы. Для решения этой проблемы решено было находить сперва ярку лампу и относительно нее накладывать маску на корпус лампы.

Алгоритм поиска лампы и наложения маски:

- Перевод изображение в цветовое пространство *HSV*;
- Нахождение маски лампы функцией *creationMask(int flagColor, Mat inputImage);*
- Создание контуров из маски функцией *Canny*;
- Определение центра масс лампы;
- Создание эллипса на месте корпуса лампы (задается относительно центра масс лампы);

Результат:



Полученное изображение позволяет без ошибок определять красных роботов.

Алгоритм работы основной функции:

- Перевод изображение в цветовое пространство *HSV*;
- Нахождение маски роботов функцией *creationMask(int flagColor, Mat inputImage)*;
- Создание контуров из маски функцией *Canny*;
- Определение центра масс роботов;
- Определение робота с минимальной дистанцией функцией *searchMinDistance(vector<Point> centerRobot, Point centerLamp)*;
- Отрисовка контуров все роботов и линии кратчайшего расстояния до лампы.

Функция *Mat creationMask(int flagColor, Mat inputImage)* принимает на вход флаг цвета (0 – лампа, 1 – голубой и тд) и входное изображение в пространстве цветов HSV. Возвращает маску с роботами.

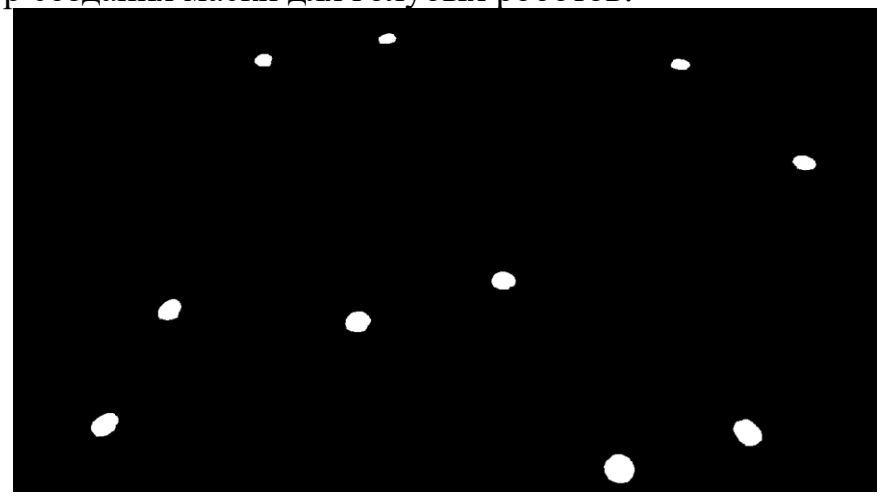
```
Mat creationMask(int flagColor, Mat inputImage)
{
    Scalar low, high;
    int dilateSize = 1, erodeSize = 1;
    string color = "mask";

    if (flagColor == 0) //Лампа
    {
        Scalar lampLow(107, 0, 255);
        low = lampLow;
        Scalar lampHigh(180, 255, 255);
        high = lampHigh;
        dilateSize = 10;
        erodeSize = 1;
        color = "Lamp";
    }

    Mat tempMask;
    inRange(inputImage, low, high, tempMask);
    dilate(tempMask, tempMask, getStructuringElement(MORPH_RECT, Size(dilateSize, di-
lateSize)));
    erode(tempMask, tempMask, getStructuringElement(MORPH_RECT, Size(erodeSize,
erodeSize)));
    imshow(color, tempMask);
    waitKey();
    //mask = tempMask.clone();
    return tempMask;

}
```

Пример создания маски для голубых роботов:

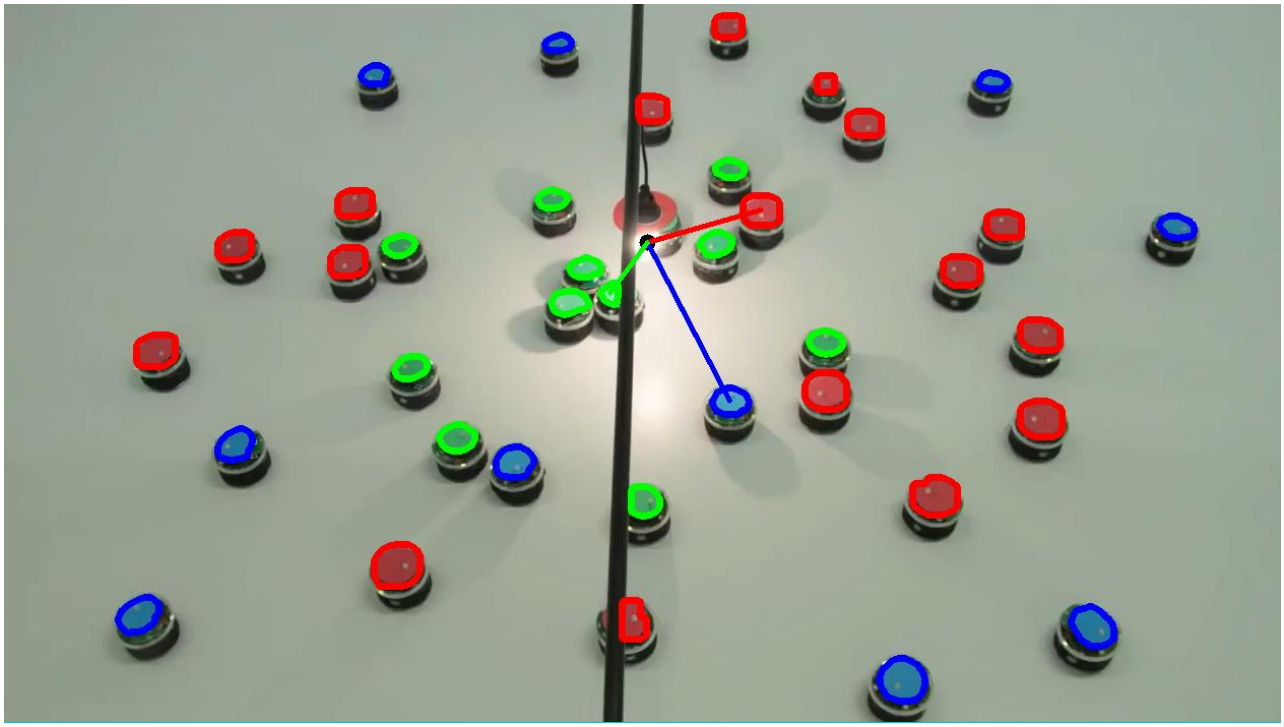


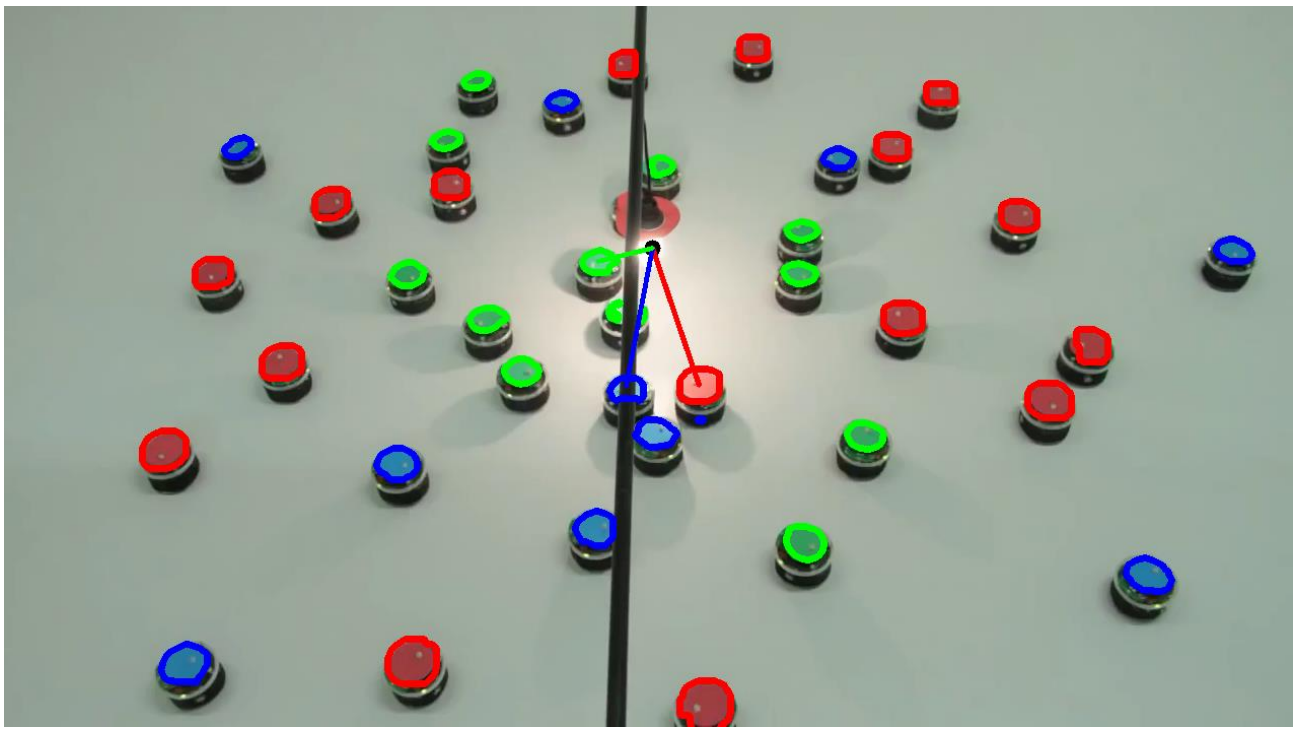
Функция *searchMinDistance(vector<Point> centerRobot, Point centerLamp)* принимает на вход вектор точек центров масс роботов и центр лампы. Возвращает индекс робота с минимальной дистанцией до лампы.

```
int searchMinDistance(vector<Point> centerRobot, Point centerLamp)
{
    vector<double> distance;
    for (int i = 0; i < centerRobot.size(); i++)
    {
        double a1 = abs(centerRobot[i].x - centerLamp.x);
        double a2 = abs(centerRobot[i].y - centerLamp.y);
        double temp = sqrt(a1 * a1 + a2 * a2);
        distance.push_back(temp);
    }

    double minDistance = distance[0];
    int indMin = 0;
    for (int i = 0; i < distance.size(); i++)
    {
        if (distance[i] <= minDistance)
        {
            minDistance = distance[i];
            indMin = i;
        }
    }
    return indMin;
}
```

Пример работы функции part3:





Задание 4

В задании 4 необходимо найти на изображении ключи верной формы и не верной и отметить их. Для выполнения задания реализована функция *part4(Mat inputImage1, Mat inputImage2)*. Функция принимает на вход картинку ключей и эталон ключа в оттенках серого.

Алгоритм работы функции:

- Нахождение маски ключей функцией *threshold(inputImage1, mask1, 229, 255, THRESH_BINARY);*
- Нахождение маски эталона функцией *threshold(inputImage1, mask1, 100, 255, THRESH_BINARY);*
- Нахождение контуров на полученных масках;
- Сравнение контуров функцией *matchShapes*.
- Отрисовка контуров красным или зеленым цветом в зависимости от значений сравнения.

Сравнение и отрисовка:

```
    vector<double> likeness;
    cvtColor(inputImage1, inputImage1, COLOR_GRAY2BGR);
    for (int i = 0; i < cnts1.size(); i++)
    {
        likeness.push_back(matchShapes(cnts2[0], cnts1[i], ShapeMatchModes::CON-
TOURS_MATCH_I2, 0));
        cout << likeness[i] << endl;
        if (likeness[i] <= 1)
        {
            polylines(inputImage1, cnts1[i], true, Scalar(0, 255, 0), 4, 8);
        }
        if (likeness[i] > 1)
        {
            polylines(inputImage1, cnts1[i], true, Scalar(0, 0, 255), 4, 8);
        }
     }
```

Контуры, имеющие результат сравнения больше 1 являются браком. Результат работы: