

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего
образования «Санкт–Петербургский государственный университет
аэрокосмического приборостроения»
Кафедра №43 «Компьютерных технологий и программной инженерии»

ОТЧЁТ ПО ПРАКТИКЕ

ЗАЩИЩЁН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

Ст. преподаватель

С.А. Рогачев

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ОТЧЁТ ПО ПРАКТИКЕ

вид практики производственная

тип практики технологическая (проектно-технологическая)

на тему индивидуального задания Алгоритм детектирования объекта

выполнен Сидиропуло Хетагом Владимировичем

фамилия, имя, отчество обучающегося в творительном падеже

по направлению подготовки

09.03.04

Программная инженерия

код

наименование направления

наименование направления

направленности

код

наименование направленности

наименование направленности

Обучающийся группы №

4033

Х.В.Сидиропуло

номер

подпись, дата

инициалы, фамилия

Санкт–Петербург 2022

Оглавление

Задание в соответствии с вариантом3	
Теоретический раздел3	
Результаты работы4	
Демонстрация5	
Листинг программы6	
Выводы.....	10
Список литературы	11

Задание в соответствии с вариантом

Реализация алгоритма детектирования объекта на изображении.

Теоретический раздел

Изображение представляет собой двумерную матрицу значений от 0 до 255.

В зависимости от представления изображения (RGB (HSV в OpenCV) и т. д.) каждый пиксель будет представляться разной размерности вектор.

Контурный анализ — это один из важных и очень полезных методов описания, хранения, распознавания, сравнения и поиска графических образов/объектов.

Контур — это внешние очертания (обвод) предмета/объекта.

При проведении контурного анализа:

* предполагается, что контур содержит достаточную информацию о форме объекта;

Результат работы

Оригинальные изображения:



Демонстрация



Листинг программы

Ball.cpp

```
#include "Ball.h"

Ball::Ball(){
}

Ball::Ball(Color color, cv::Rect rect, int x, int y){
    this->color = color;
    this->rect = rect;
    this->position = cv::Point(x, y);
}
```

Ball.h

```
#pragma once
#include "Color.h"
#include <opencv2/core/types.hpp>

class Ball
{
public:
    Color color;
    cv::Point position;
    cv::Rect rect;

    Ball();
    Ball(Color color, cv::Rect rect, int x, int y);
};
```

Color.h

```
#pragma once
enum Color { BLUE, RED, GREEN, YELLOW, PURPLE };
```

Main.cpp

```
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>

#include "Ball.h"
#include <iostream>

using namespace std;
using namespace cv;

//вектор, используется для представления значения каждого цвета

//Hue - цветовой тон, т.е.оттенок цвета.
//Saturation - насыщенность.Чем выше этот параметр, тем "чище" будет цвет, а чем ниже,
тем ближе он будет к серому.
//Value(Brightness) - значение(яркость) цвета.(0 % -черный)|

// P.S значения меняются в зависимости от картинки (разные версии игры имеют различные
оттенки)
Scalar yellowLow = Scalar(25, 130, 180);
Scalar yellowHigh = Scalar(45, 255, 255);

Scalar greenLow = Scalar(50, 35, 30);
Scalar greenHigh = Scalar(70, 255, 255);

Scalar blueLow = Scalar(100, 150, 150);
```

```

Scalar blueHigh    = Scalar(140, 255, 255);

Scalar purpleLow   = Scalar(148, 117, 89);
Scalar purpleHigh  = Scalar(152, 255, 255);

Scalar redLow      = Scalar(150, 120, 140);
Scalar redHigh     = Scalar(180, 255, 255);
////////////////////////////////////

std::vector<Ball> balls;

void GetBalls(Mat img, Scalar low, Scalar high, Color color) {
    Mat mask;
    inRange(img, low, high, mask);
    vector<vector<Point>> contours;
    findContours(mask, contours, cv::RETR_EXTERNAL, cv::CHAIN_APPROX_SIMPLE);

    for (size_t i = 0; i < contours.size(); i++)
    {
        // поиск контуров
        cv::Rect boundRect = boundingRect(contours[i]);
        if (boundRect.area() > 350 && (boundRect.width < 65 || boundRect.height < 65)) {
            balls.emplace_back(color, boundRect, boundRect.x + boundRect.width / 2,
boundRect.y + boundRect.height / 2);
        }
    }
}

void drawBalls(Mat background) {
    for (size_t i = 0; i < balls.size(); i++) {
        switch (balls[i].color) {

            // rectangle используется для рисования прямоугольника на любом изображении
            case RED:
                                                                    //цвет обводки
                rectangle(background, balls[i].rect.tl(), balls[i].rect.br(), CV_RGB(255, 0,
0), 2);
                break;

            case BLUE:
                rectangle(background, balls[i].rect.tl(), balls[i].rect.br(), CV_RGB(0, 0,
255), 2);
                break;

            case GREEN:
                rectangle(background, balls[i].rect.tl(), balls[i].rect.br(), CV_RGB(0, 255,
0), 2);
                break;

            case YELLOW:
                rectangle(background, balls[i].rect.tl(), balls[i].rect.br(), CV_RGB(255,
255, 0), 2);
                break;

            case PURPLE:
                rectangle(background, balls[i].rect.tl(), balls[i].rect.br(), CV_RGB(128, 0,
128), 2);
                break;
        }
    }
}

int main() {

```

```

setlocale(LC_ALL, "rus");
Mat target = imread("frog2.jpg");

if (target.empty() )
{
    cout << "Лягушка спряталась в болоте, не можем найти" << endl;
    cin.get();
    return -1;
}

// Размер матрицы, метод, используемый для хранения, адрес хранения матрицы и т. Д
Mat background;

//копирование матрицы в другую матрицу.
// используется как копия слоя в фотошопе
target.copyTo(background);

// перевод RGB в HSV для лучшей работы
cvtColor(target, target, cv::COLOR_BGR2HSV);

rectangle(target, cv::Point(0, 0), cv::Point(640, 30), (0, 0, 0), cv::FILLED);

GetBalls(target, yellowLow, yellowHigh, Color::YELLOW); // желтый шар
GetBalls(target, blueLow, blueHigh, Color::BLUE); // синий шар
GetBalls(target, redLow, redHigh, Color::RED); // красный шар
GetBalls(target, greenLow, greenHigh, Color::GREEN); // зеленый шар
GetBalls(target, purpleLow, purpleHigh, Color::PURPLE); //фиолетовый шар
drawBalls(background);

// имя консоли/фото
imshow("FROG", background);
waitKey(0);

```


}

Выводы

В ходе проделанной работы были освоены базовые навыки использования библиотеки `opencv`, на его основе было разработано приложение по поиску контуров – детектирование шариков.

Список литературы

1. <https://robocraft.ru/opencv>
2. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library 1st Edition, Kindle Edition
3. <https://russianblogs.com/article/5416285417/>
4. <https://docs.opencv.org/>
5. <https://www.youtube.com/c/MurtazasWorkshopRoboticsandAI/playlists>