

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ  
УНИВЕРСИТЕТ «МИСиС»

Институт ИТАСУ  
Кафедра инженерной кибернетики  
Направление подготовки: «01.03.04 Прикладная математика»  
Квалификация: бакалавр  
Группа: БПМ-16-1

# ОТЧЕТ

**ПО КУРСОВОЙ РАБОТЕ ПО ПРЕДМЕТУ  
«МЕТОДЫ И СРЕДСТВА ОБРАБОТКИ ИЗОБРАЖЕНИЙ»**

## на тему: Распознавание элементов пользовательского интерфейса приложения Instagram

**Студент**

**Троцюк В.**

## Руководитель

\_\_\_\_\_/\_\_\_\_\_**Полевой Д.В.**\_\_\_\_\_/\_\_\_\_\_  
подпись должность, уч. степ. Фамилия И.О.

**Оценка:** \_\_\_\_\_

**Дата защиты:** \_\_\_\_\_

**Москва 2020**

|                                                               |                  |
|---------------------------------------------------------------|------------------|
| <b><u>Введение</u></b>                                        | <b><u>3</u></b>  |
| <b><u>Актуальность</u></b>                                    | <b><u>3</u></b>  |
| <b><u>Цель работы</u></b>                                     | <b><u>4</u></b>  |
| <b><u>Краткое описание работы фильтра частиц</u></b>          | <b><u>4</u></b>  |
| <b><u>Исходные данные</u></b>                                 | <b><u>5</u></b>  |
| <b><u>Используемый инструментарий</u></b>                     | <b><u>6</u></b>  |
| <b><u>Описание используемых функций библиотеки OpenCV</u></b> | <b><u>6</u></b>  |
| <b><u>Интерфейс программы, описание работы</u></b>            | <b><u>6</u></b>  |
| <b><u>Алгоритм работы</u></b>                                 | <b><u>7</u></b>  |
| <b><u>Результат работы</u></b>                                | <b><u>8</u></b>  |
| <b><u>Источники</u></b>                                       | <b><u>11</u></b> |
| <b><u>Приложение</u></b>                                      | <b><u>12</u></b> |

## **Введение**

В рамках текущей курсовой работы была выбрана тема «Распознавание элементов пользовательского интерфейса приложения Instagram». В данной работе я использовал метод multi-scale template matching. Эта тема позволяет реализовать возможности базовых структур библиотеки для обработки изображений OpenCV.

## **Актуальность**

Данная работа полезна при наличии большого числа картинок и поиска тех, где есть искомый шаблон.

## Цель работы

Цель работы – распознать элементы пользовательского приложения Instagram и выделить их.

### Краткое описание работы метода **template matching**

Сопоставление с шаблоном (template matching) - это метод поиска и поиска местоположения шаблона изображения на большом изображении. OpenCV имеет функцию `cv.matchTemplate ()` для этой цели. Шаблонное изображение скользит по входному изображению (как в 2D-свертке) и сравнивает шаблон и патч входного изображения под шаблонным изображением. В OpenCV реализовано несколько методов сравнения. Он возвращает изображение в градациях серого, где каждый пиксель обозначает, насколько окрестность этого пикселя соответствует шаблону.

Если входное изображение имеет размер  $(W \times H)$ , а шаблонное изображение имеет размер  $(w \times h)$ , выходное изображение будет иметь размер  $(W - w + 1, H - h + 1)$ . Получив результат, можно использовать функцию `cv.minMaxLoc ()`, чтобы найти, где находится максимальное / минимальное значение. Возьмите верхний левый угол прямоугольника и за  $(w, h)$  за ширину и высоту прямоугольника. Этот прямоугольник является искомой областью шаблона.

В своей работе я использую метод multi-scale template matching. Соответственно, я буду изменять размер своего изображения для более точного нахождения шаблона на нём. После каждого изменения размера я применяю template matching к изображению.

## Исходные данные

Исходными данные – это скриншоты приложения Instagram на разных устройствах с разными операционными системами.

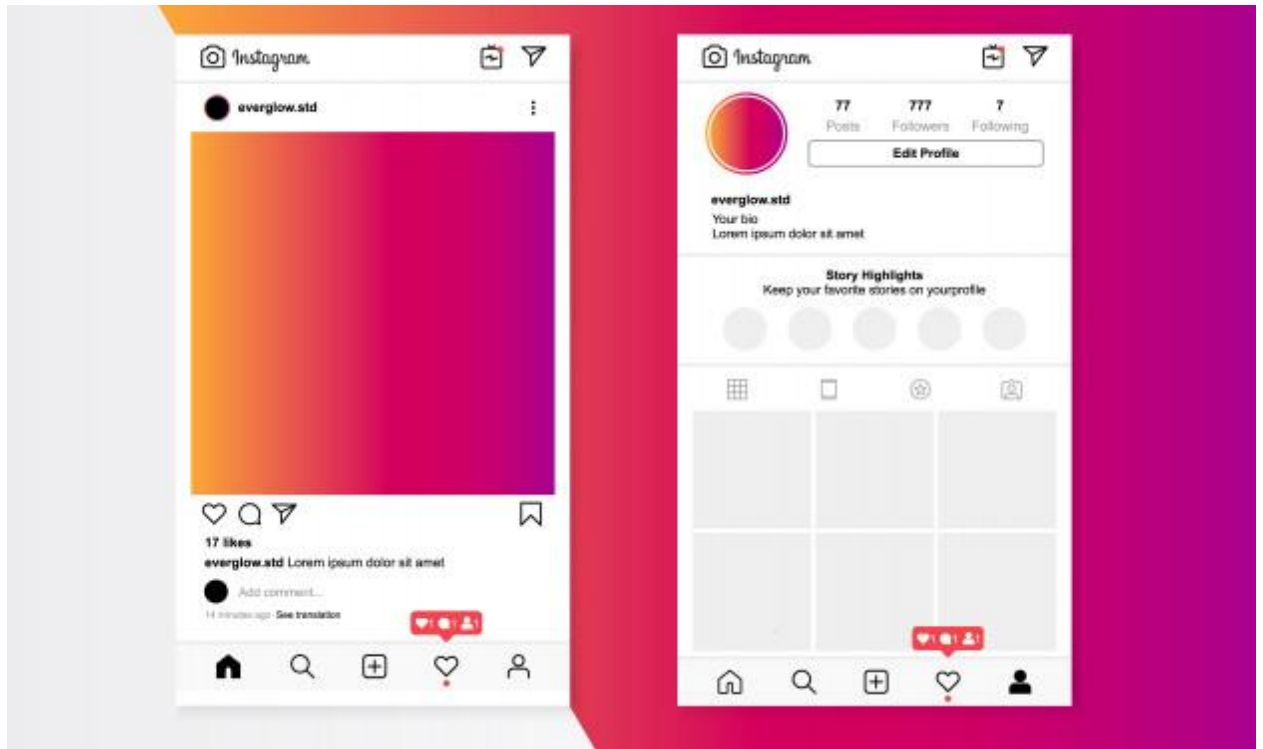


Рис. 1. Шаблон приложения инстаграм

## **Используемый инструментарий**

Операционная система: Microsoft Windows 10

Язык программирования: C++ 11

Среды разработки: Visual Studio 2019

Используемые библиотеки: OpenCV

Сторонние программы: Cmake 3.8

### **Описание используемых функций библиотеки OpenCV**

`cv::Mat()` — основной контейнер библиотеки OpenCV. Представляет собой матрицу. В работе используется для хранения изображения.

`cv::cvtColor()` — функция преобразует входное изображение из одного цветового пространства в другое.

`cv::Canny()` — это краевой детектор, разработанный как проблема обработки сигналов. В OpenCV он выводит двоичное изображение, обозначающее обнаруженные края.

`cv::Point()` — структура точки.

`cv::minMaxLoc()` — функция, ищущая глобальный минимум и максимум в массиве.

## Алгоритм работы.

Программа реализована в виде класса с набором всех необходимых для работы функций и данных.

После считывания изображения и шаблона, переводим их в серые тона, а также ищем края с помощью метода Canny.

Далее запускаем цикл, в котором будем менять размер изображения и с помощью метода `matchTemplate` ищем координаты места, которое больше всего похоже на шаблон, а также с помощью различных методов(в работе автор использует метод `TM_CCOEFF`) получаем значение, насколько шаблон и картинка совпадают. Сравнивая полученные результаты, выбираем лучшее совпадение и берём координаты. Далее строим прямоугольник по координатам, который выделяет элемент интерфейса, который был подан на вход, как шаблон.

## Результаты работы

Входными данными для приложения помимо шаблона элемента интерфейса и изображения является номер элемента интерфейса, который искал пользователь. 1 – это элемент интерфейса, отвечающий за переход на главную страницу, 2 – за поиск, 3 – за добавление фотографий, 4 – за переход на страницу уведомлений, 5 – за переход на страницу своего профиля. Выходными данными будут являться изображение с выделенным элементом интерфейса, который больше всего сходится с шаблоном, который подавался на вход, а также вывод сообщения «pattern found», если программа нашла элемент интерфейса в зоне, указанной пользователем и «pattern not found» - иначе. Разметка зон показана на рисунке 2. Если наиболее подходящий вариант имеет диаметр меньший, чем  $1/20$  ширины страницы, то он будет считаться неверным.



Рис. 2. Разметка элементов интерфейса.



Пример работы программы №1. На вход подаётся элемент интерфейса, отвечающий за переход на страницу, с понравившимися записями(рис. 3)



Рис.3 Элемент интерфейса 1

Выводом программы должно служить изображение, на котором этот элемент отмечен прямоугольником. Соответственно, вывод программы корректен(рис. 4)

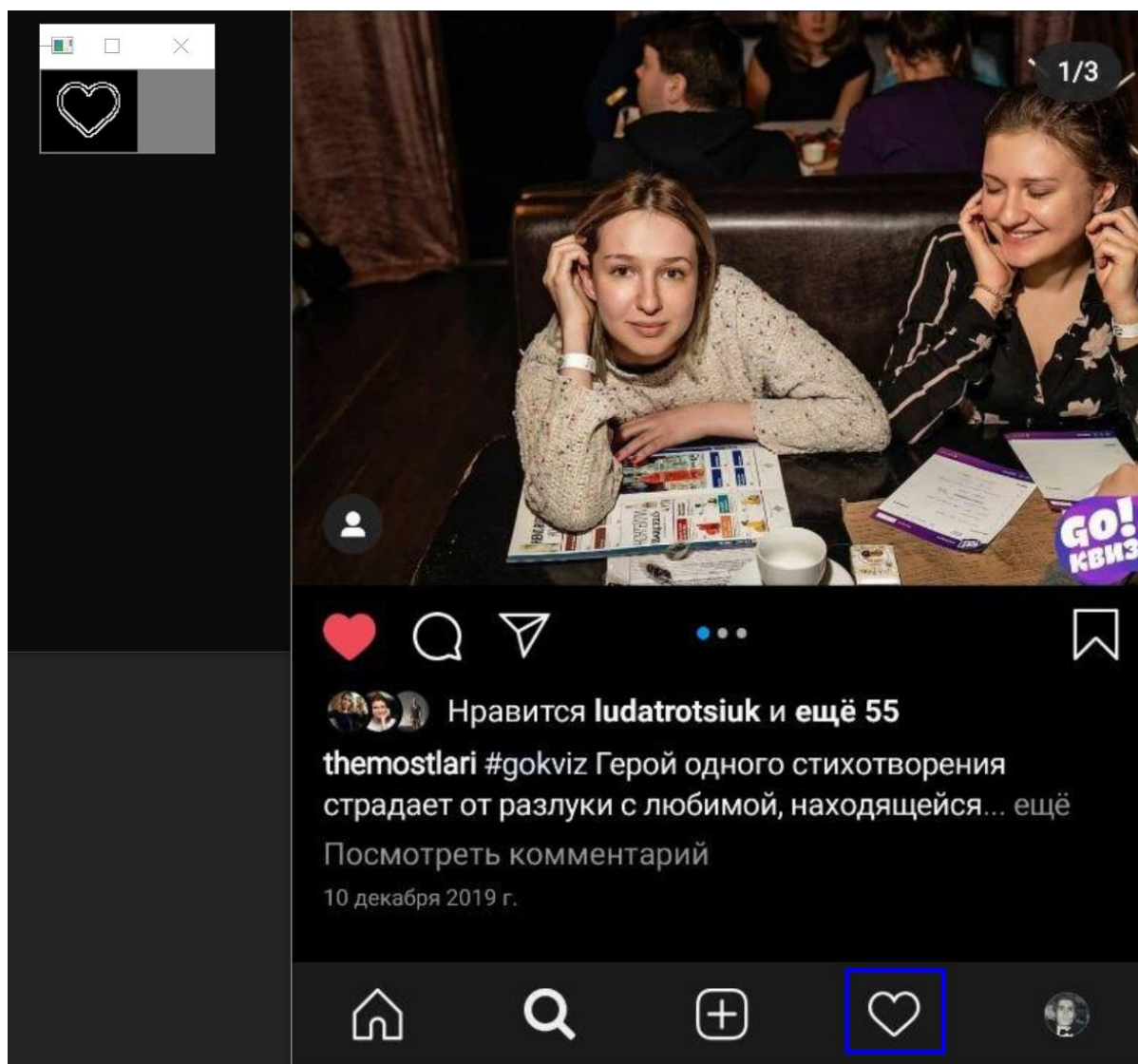


Рис. 4 Результат работы 1

Также получаем положительный результат, где на вход был подан элемент интерфейса, отвечающий за переход на главную страницу(рис.5.)



Рис. 5 Элемент интерфейса 2

Вывод программы показан на рисунке 6.

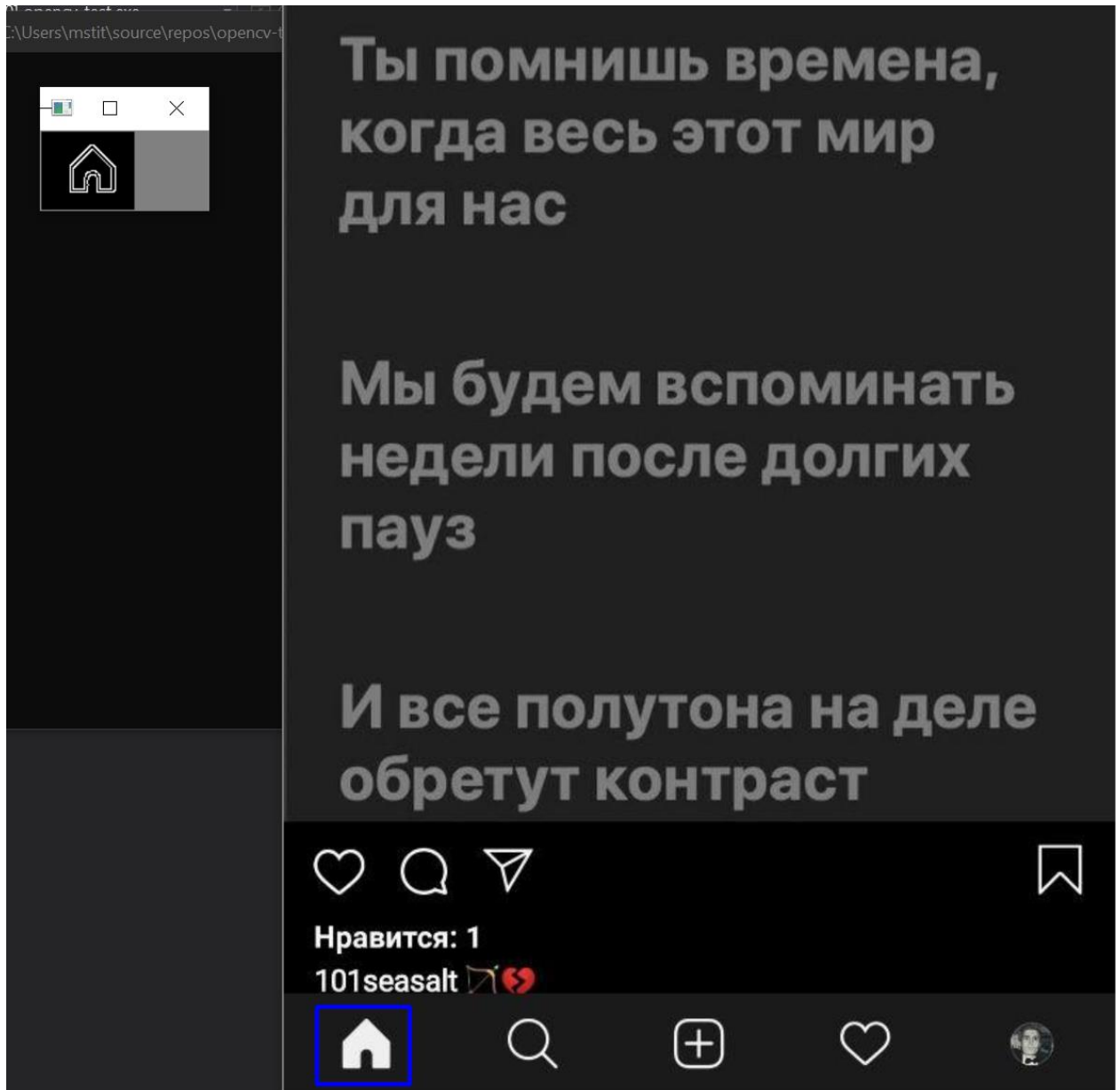


Рис.6 Результат работы 2

На данном изображении видно, что элемент интерфейса выделен синим прямоугольником.

Программа работает корректно независимо от темы приложения(светлая или тёмная) и ОС(Android/iOS).(Рис. 7)

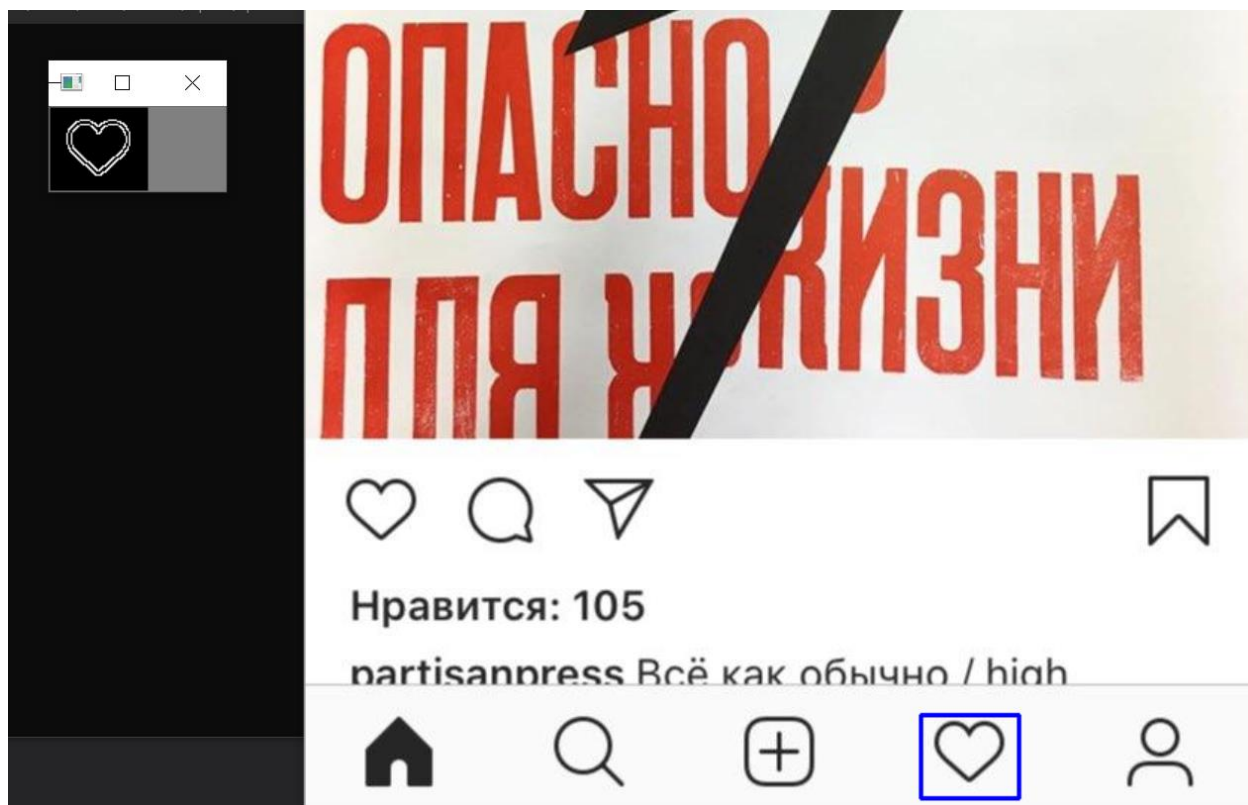


Рис. 7 Результат работы 3

## Источники

1. [https://docs.opencv.org/master/d4/dc6/tutorial\\_py\\_template\\_matching.html](https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html) - OpenCV template matching
2. <https://www.pyimagesearch.com/2015/01/26/multi-scale-template-matching-using-python-opencv/> - Multi-scale Template Matching
3. Bjarne Stroustrup - The C++ Programming Language
4. Coursera - Искусство разработки на современном C++
5. <https://www.machinelearningmastery.ru/using-opencv-python-and-template-matching-to-play-wheres-waldo/> - Использование OpenCV и Template Matching

## Приложение

Ссылка на репозиторий: [https://github.com/vladtrotsiuk/2020-misis-spring-image-processing/tree/master/course\\_work/course\\_work](https://github.com/vladtrotsiuk/2020-misis-spring-image-processing/tree/master/course_work/course_work)

```
//импортируем необходимые библиотеки
#include <opencv2\highgui.hpp>
#include <iostream>
#include <opencv2\core.hpp>
#include <opencv2\imgproc\imgproc.hpp>

using namespace cv;
using namespace std;

int main(int argc, char* argv[])
{
    //считываем изображения и шаблон, переводим их в серый цвет
    int flags = IMREAD_GRAYSCALE;
    Mat image = imread("imglike.jpg");
    Mat gray;

    if (image.data == NULL)
    {
        printf("file cannot be loaded\n");
        return 1;
    }

    cvtColor(image, gray, cv::COLOR_BGR2GRAY);

    namedWindow("My");

    Mat result;
    //используем краевой детектор для выделения границ
    Canny(gray, gray, 50, 200);
    int h = gray.rows;
    int w = gray.cols;

    Mat tmplt = imread("tmplt1.png", flags);
    int tw = tmplt.rows;
    int th = tmplt.cols;
    Canny(tmplt, tmplt, 50, 200);
    imshow("Template", tmplt);

    double found0 = 0;
    Point found1(0.0, 0.0);
    float found2 = 0.0;
    Mat resized;
    Mat edged;

    //запускаем цикл с различными размерами изображения
    for (long double i = 1; i > 0.1; i -= 0.2) {

        resize(image, resized, Size(w * i, h * i));
        float r = gray.cols / resized.cols;
        if ((resized.rows < th) || (resized.cols < tw)) {
            break;
        }
        Canny(resized, edged, 50, 200);
        matchTemplate(edged, tmplt, result, TM_CCORR);

        double minVal;
```

```

        double maxVal;
        Point minIdx;
        Point maxIdx;
        //смотрим, нашли ли хороший шаблон, если да, то
        minMaxLoc(result, &minVal, &maxVal, &minIdx, &maxIdx);
        if (found0 < maxVal) {
            found0 = maxVal;
            found1 = maxIdx;
            found2 = r;
        }
    }

    int startX = found1.x*found2;
    int startY = found1.y * found2;

    int endX = (found1.x + th)* found2;
    int endY = (found1.y + tw) * found2;
    Point pt1(startX, startY);
    Point pt2(endX, endY);
    //строим прямоугольник вокруг шаблона
    rectangle(image, pt1, pt2, (0, 0, 255), 2);
    imshow("My", image);
    waitKey(0);

    return 0;
}

```