**Министерство образования и науки Российской Федерации**

**федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет программной инженерии и компьютерной техники

# ЛАБОРАТОРНАЯ РАБОТА №2

**по дисциплине «Компьютерное зрение»**

**на тему «Алгоритмы детектирования объектов на изображении»**

Выполнила

студентка группы P4140

очной формы обучения

ПиРСИИ

Строкова Анастасия Владиславовна

Преподаватель:

Денисов Алексей Константинович

Санкт-Петербург – 2022

**Содержание**

**Постановка задачи**

Цель работы: исследовать простейшие алгоритмы детектирования объектов на изображении.

Базовый алгоритм: прямой поиск одного изображения на другом, поиск ключевых точек эталона на входном изображении.

Задание: на вход поступает два изображения: эталон и изображение, на котором будет производиться поиск. На выходе программа должна строить рамку в виде четырехугольника в области, где с наибольшей вероятностью находится искомый объект.

Задачи:

- реализовать алгоритм прямого поиска одного изображения на другом (template matching);
- реализовать алгоритм поиска ключевых точек эталона на входном изображении (ORB).

**Теоретическая база**

Задача нахождения объектов на изображении – задача машинного обучения, в рамках которой выполняется определение наличия или отсутствия объекта определенного домена на изображении, нахождение границ этого объекта в системе координат пикселей исходного изображения [3].

В данной работе будет реализован алгоритм детекции объектов. Программный код написан в Google Collaboratory на языке Python.

В первой половине работы использовался алгоритм прямого поиска изображения с помощью библиотеки OpenCV. Для этого бралось изображение объекта на нейтральном фоне, вырезался фрагмент и сохранялся под другим именем. На вход подавалось два изображения, которые алгоритм сравнивал и выводил рамку вокруг детектированного объекта.

Существует несколько методов matching template, в данной работе использовался cv.TM_CCOEFF (находятся глобальные максимумы с помощью функции minMaxLoc). Отмечу, что на вход может подаваться как черно-белое, так и цветное изображение, однако на выходе оно все равно преобразуется в одноканальное.

Во второй половине работы опробован алгоритм ORB (Oriented FAST and Rotated BRIEF). Алгоритм работает следующим образом [4]:

1) Особые точки обнаруживаются при помощи быстрого древовидного FAST (сравнивается яркость пикселей вокруг выбранной точки) на исходном изображении и на уменьшенном.

2) Для обнаруженных точек вычисляется мера Харриса, кандидаты с низким значением меры Харриса отбрасываются.

3) Вычисляется угол ориентации особой точки, на основе которого последовательность точек для бинарных сравнений в дескрипторе BRIEF поворачивается.

4) По полученным точкам вычисляется бинарный дескриптор BRIEF (набор признаков, который характеризует окрестность каждой особой точки).

Для выполнения этой задачи мною использовались библиотеки, которые показаны на рисунке 1.

```
# импорт библиотек
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
from google.colab.patches import cv2_imshow
```

**Рис. 1** Библиотеки, используемые для выполнения поставленной задачи

**Прямой поиск одного изображения на другом (Template Matching)**

Для осуществления Template Matching считаем исходное изображение и вырезанный фрагмент, найдем объект, затем выведем итоговые изображения. Программный код представлен на рисунке 2.

```
# считаем файл, на котором будет детектирован объект
img = cv.imread('Groot.jpg',0)
img2 = img.copy()

# считаем объект, который будем искать на другом изображении
template = cv.imread('GrootGroot.jpg',0)
w, h = template.shape[::-1]

# найдем объект
img = img2.copy()
method = eval('cv.TM_CCOEFF')
res = cv.matchTemplate(img,template,method)
min_val, max_val, min_loc, max_loc = cv.minMaxLoc(res)
top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)
cv.rectangle(img,top_left, bottom_right, 255, 2)

# выводим изображения на экран
plt.subplot(121),
plt.imshow(res,cmap = 'gray')
plt.title('Matching Result'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img,cmap = 'gray')
plt.title('Detected Point'), plt.xticks([]), plt.yticks([])
plt.show()
```

**Рис. 2** Программный код для Template Matching

Исходное изображение и фрагмент из него представлены на рисунке 2.

**Рис. 3** Исходные изображения

Далее приведу итоговые результаты работы алгоритма прямого поиска одного изображения на другом. Фотографии были сделаны с разных ракурсов.

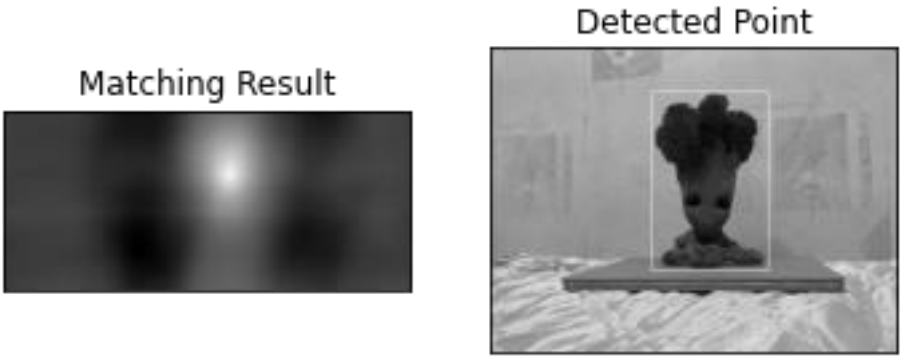


**Рис. 4** Вид спереди

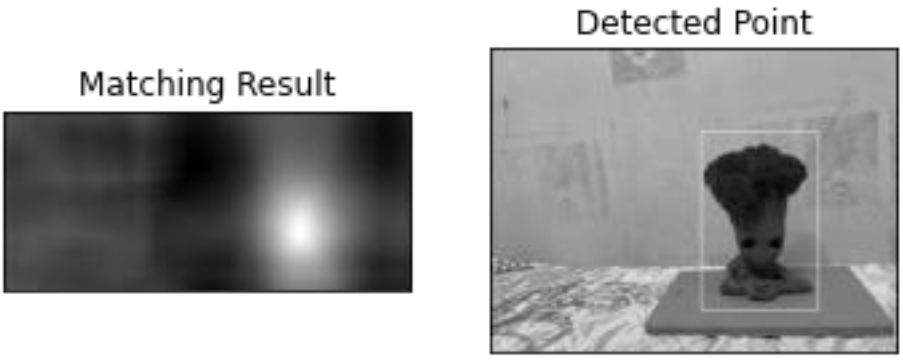


**Рис. 5** Объект снизу справа

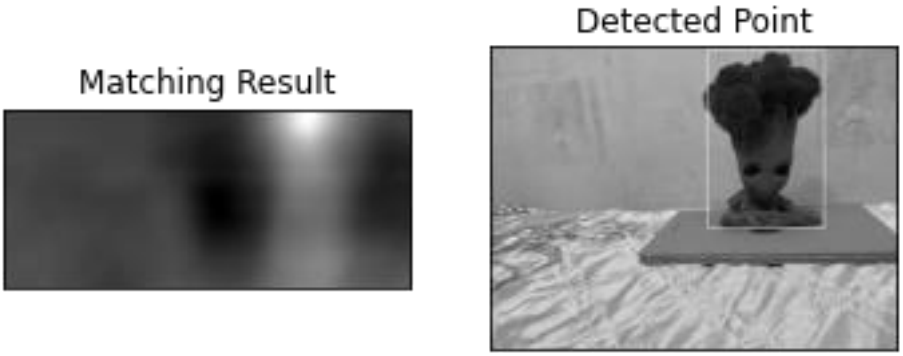


**Рис. 6** Объект сверху справа

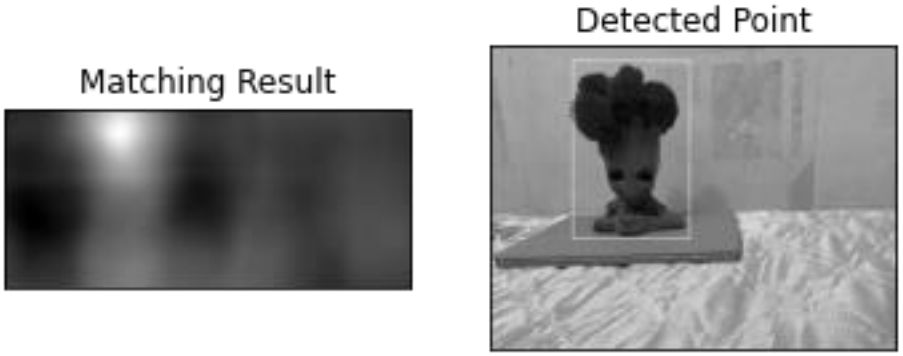


**Рис. 7** Объект сверху слева

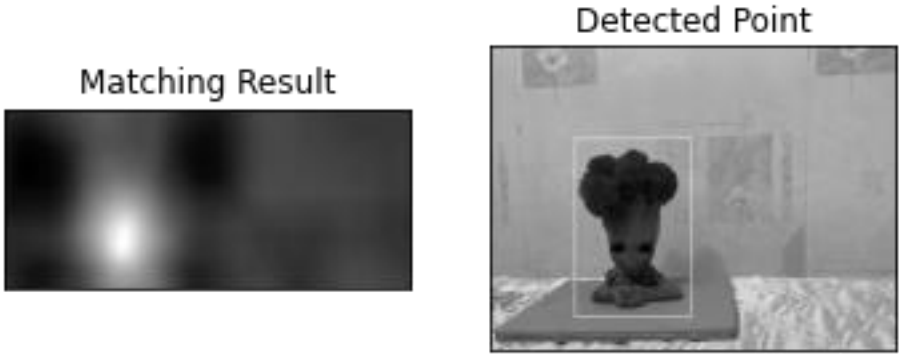


**Рис. 8** Объект снизу слева

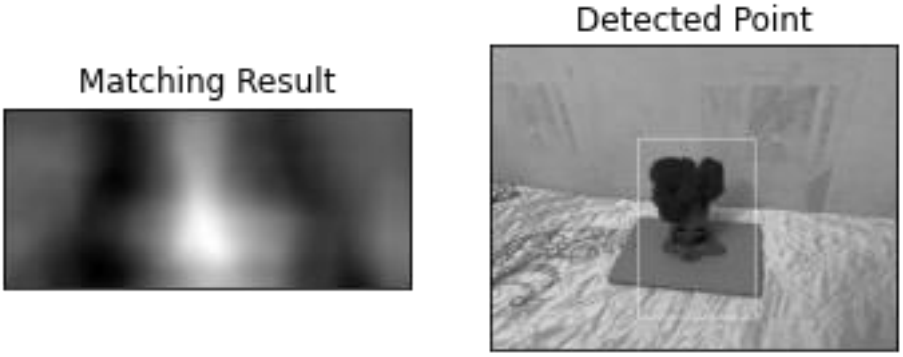


**Рис. 9** Вид сверху

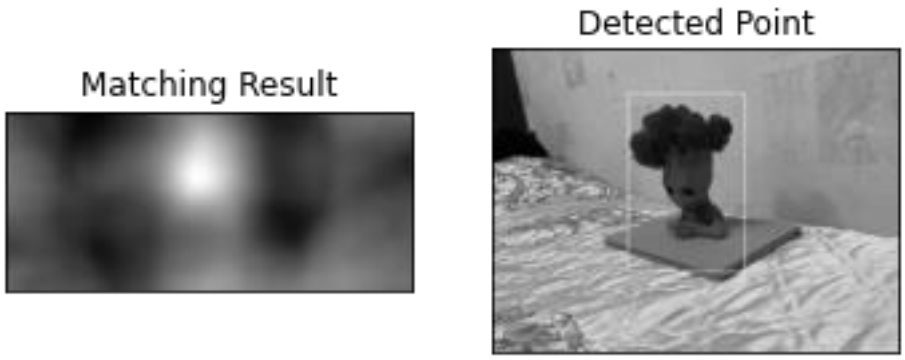


**Рис. 10** Вид с правого бока

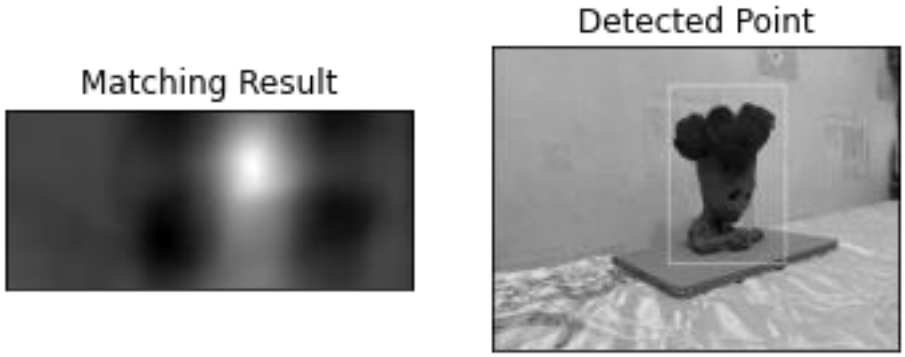


**Рис. 11** Вид с левого бока

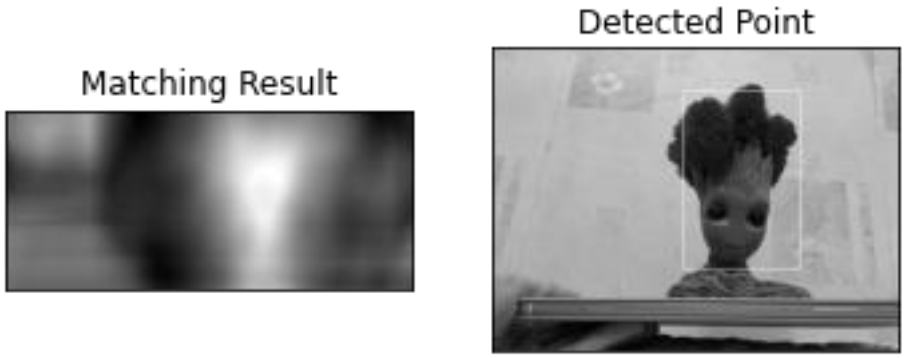


**Рис. 12** Вид снизу

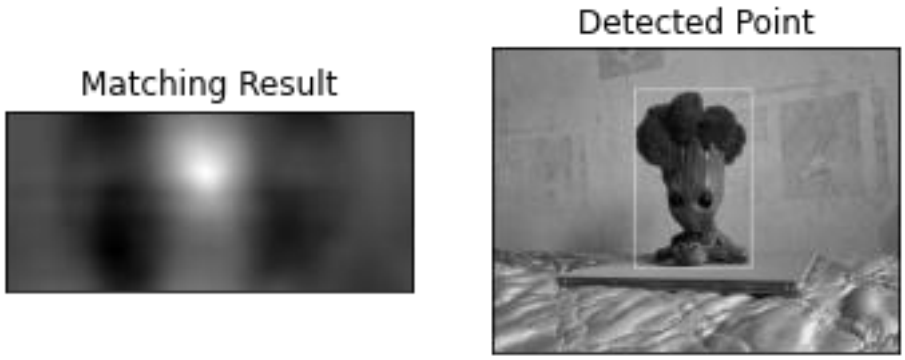


**Рис. 13** Фотография объекта с естественным освещением в вечернее время (все фотографии до были сделаны с искусственным светом)

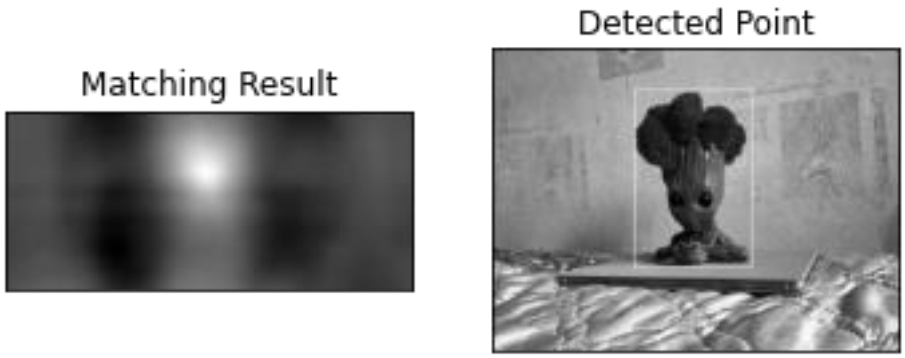


**Рис. 14** Фотография объекта с измененным вручную контрастом, яркостью и насыщенностью цвета

**Поиск ключевых точек эталона**

Для алгоритма поиска ключевых точек эталона воспользуемся методом ORB от библиотеки cv2. Алгоритм будет состоять из нескольких этапов: считывание изображения, поиск ключевых точек, их сравнение, наложение рамки на детектируемый объект и вывод результатов на экран. Программный код представлен на рисунках 15.

```
def get_bbox_points(train_kp, matches):
    #Получить ограничивающую рамку из результатов ORB
    points = [trainKP[m.trainIdx].pt for m in matches]
    xs = [pt[0] for pt in points]
    ys = [pt[1] for pt in points]
    x0 = int(min(xs))
    x1 = int(max(xs))
    y0 = int(min(ys))
    y1 = int(max(ys))
    return (x0, y0), (x1, y1)
```

```
# считываем изображения
query_img = cv.imread('GrootGroot.jpg')
original_img = cv.imread('Groot.jpg')
query_img_bw = cv.cvtColor(query_img, cv.IMREAD_GRAYSCALE)
original_img_bw = cv.cvtColor(original_img, cv.IMREAD_GRAYSCALE)
```

```
# поиск ключевых точек
orb = cv.ORB_create()
queryKP, queryDes = orb.detectAndCompute(query_img_bw,None)
trainKP, trainDes = orb.detectAndCompute(original_img_bw,None)

# сравнение ключевых точек и сортировка результатов
matcher = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
matches = matcher.match(queryDes,trainDes)
matches = sorted(matches, key = lambda x:x.distance)

# результат сравнения по первым 100 ключевым точкам
final_img = cv.drawMatches(query_img, queryKP, original_img, trainKP, matches[:100],None)
# размер выходного изображения
final_img = cv.resize(final_img, (800,450))
```

```
# получение точек и изображения с рамкой
point1, point2 = get_bbox_points(trainKP, matches)
bbox_img = bbox_img = original_img_bw.copy()
cv.rectangle(bbox_img, point1, point2, (0, 225, 255), 2)

# Вывод изображения
plt.figure(figsize=(15,10))
plt.subplot(121)
plt.imshow(cv.cvtColor(final_img, cv.COLOR_BGR2RGB))
plt.title('Match keypoints'), plt.xticks([]), plt.yticks([])
plt.subplot(122),
plt.imshow(cv.cvtColor(bbox_img, cv.COLOR_BGR2RGB))
plt.title('Detected object'), plt.xticks([]), plt.yticks([])
plt.show()
```

**Рис. 15** Программный код алгоритма с использованием ORB

Результаты нахождения эталонных точек и сопоставление их с исходным изображением представлены на рисунках ниже.

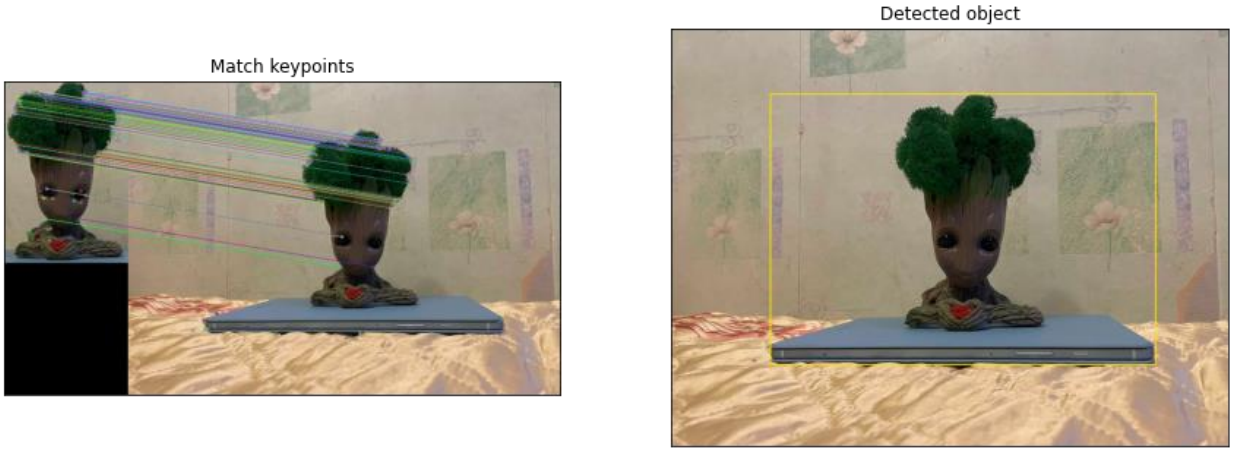


**Рис. 16** Вид спереди


**Рис. 17** Объект снизу справа

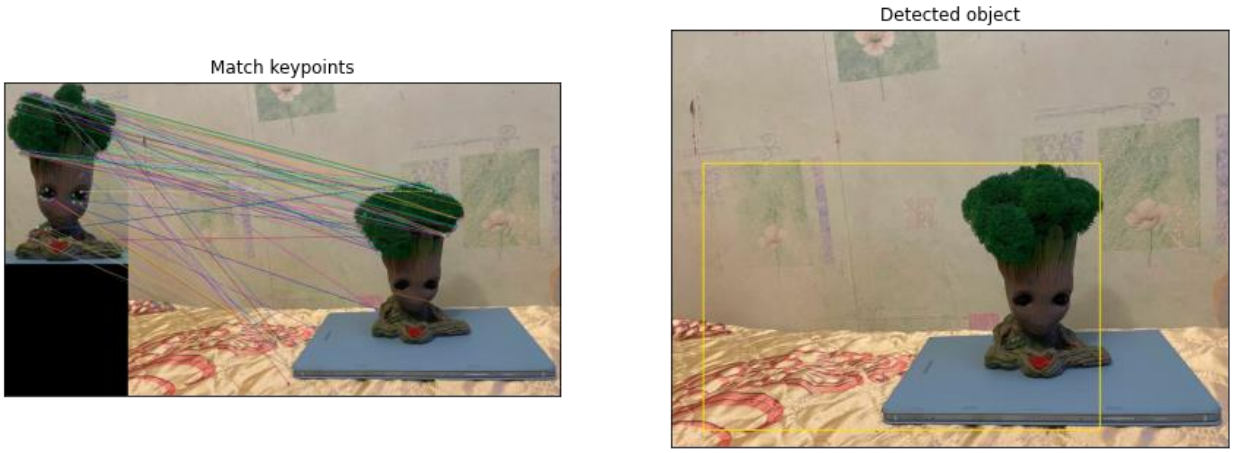


**Рис. 18** Объект сверху справа


**Рис. 19** Объект сверху слева


**Рис. 20** Объект снизу слева

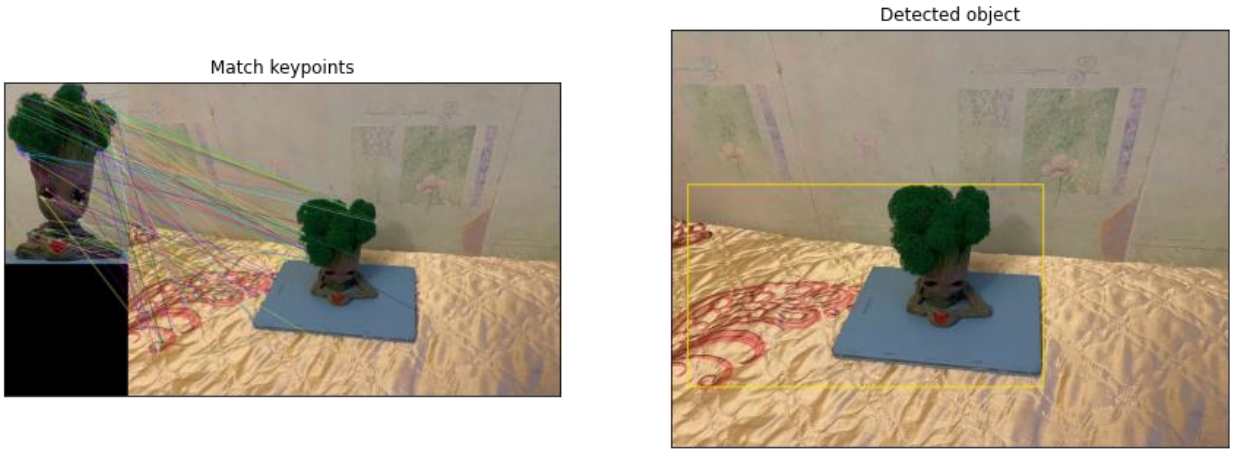


**Рис. 21** Вид сверху

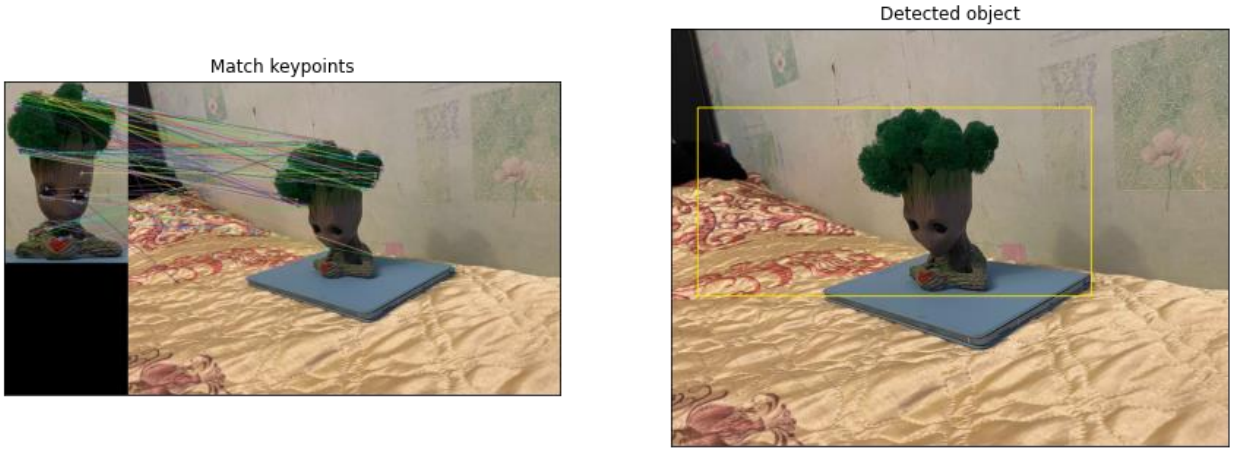


**Рис. 22** Вид с правого бока

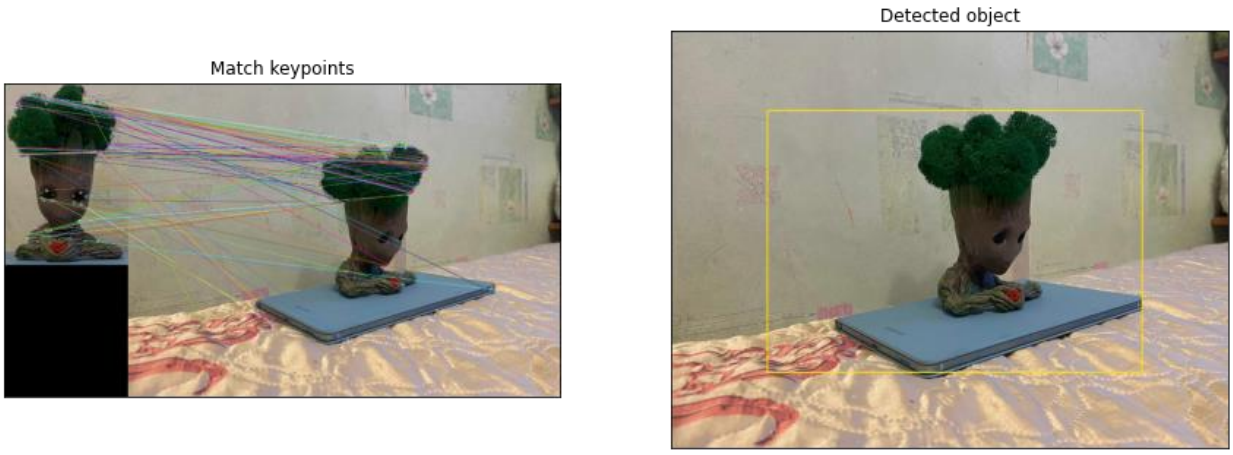


**Рис. 23** Объект с левого бока


**Рис. 24** Вид снизу

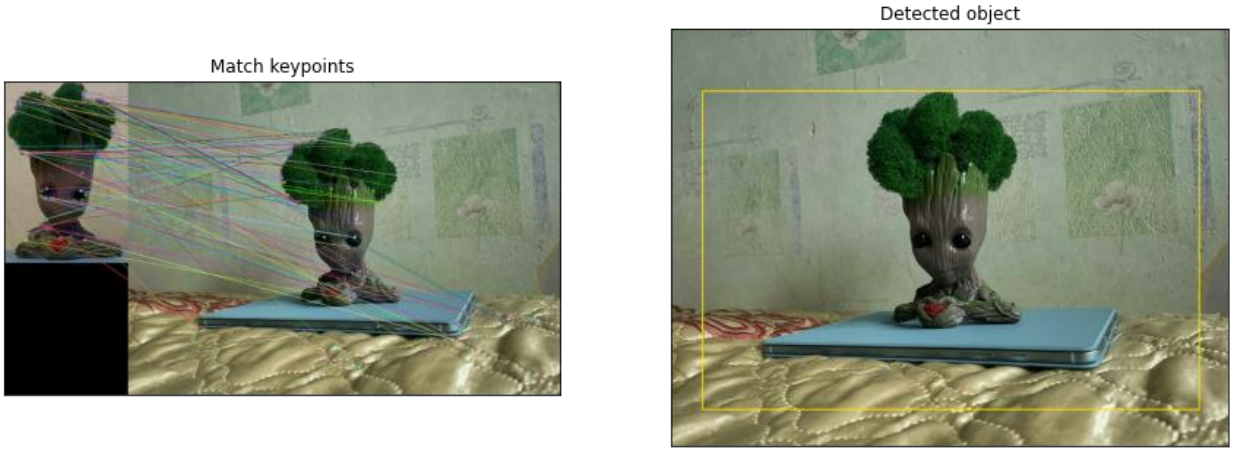


**Рис. 25** Фотография объекта с естественным освещением в вечернее время (все фотографии до были сделаны с искусственным светом)

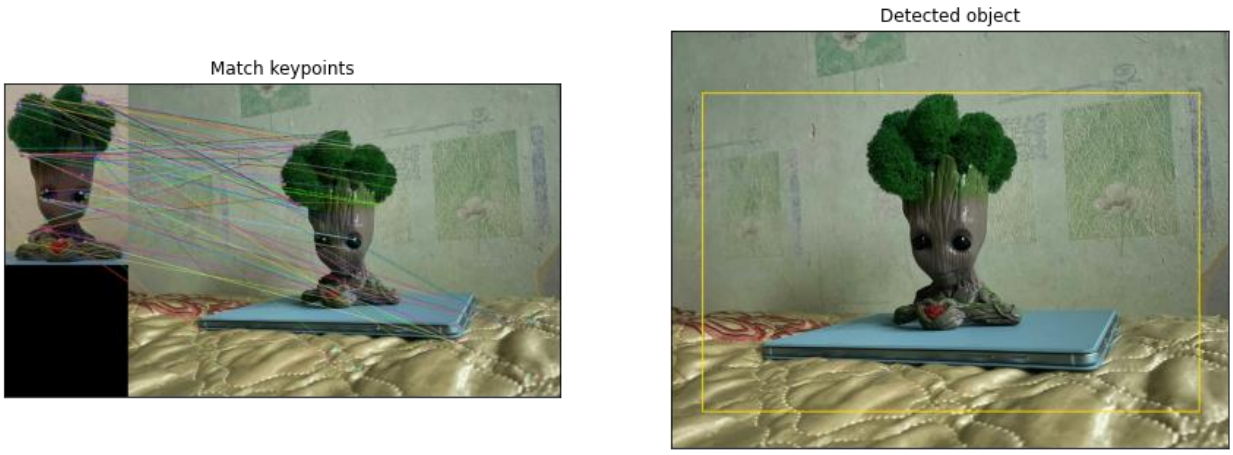


**Рис. 26** Фотография объекта с измененным вручную контрастом, яркостью и насыщенностью цвета

**Выводы по работе**

Проведено детектирование объекта на изображении двумя разными методами: методом прямого поиска (matching template) и методом поиска ключевых точек эталона (с помощью метода ORB).

На вход подавались 10 + 1 фотографий, сделанных с разных ракурсов. Также в предпоследнем изображении был выключен искусственный свет, а в последнем – вручную сделана обработка изображения (изменен контраст, насыщенность, яркость цвета). На выходе алгоритм выдавал рамку вокруг детектированного изображения. Можно заметить, что прямой поиск изображения дает лучшие результаты.

Составим сравнительную таблицу, чтобы подтвердить гипотезу о лучшей работе прямого поиска. Будем использовать трех бальную шкалу, где 2 – объект полностью помещен в рамку, 1 – объект частично помещен в рамку (обрезана часть детектируемого изображения, очень большие «лишние» поля вокруг), 0 – изображение не найдено или очень большие границы рамки.

**Таблица 1**. Оценка качества методов

| № | Положение объекта | Template Matching | ORB |
|---|---|---|---|
| 0 | Исходное изображение | 2 | 1 |
| 1 | Снизу справа | 2 | 0 |
| 2 | Сверху справа | 2 | 0 |
| 3 | Сверху слева | 2 | 1 |
| 4 | Снизу слева | 2 | 1 |
| 5 | Вид сверху | 1 | 0 |
| 6 | Вид с правого бока | 2 | 1 |
| 7 | Вид с левого бока | 2 | 1 |
| 8 | Вид снизу | 1 | 2 |
| 9 | Вечернее освещение | 2 | 1 |
| 10 | Ручное изменение цветов | 2 | 1 |
| | **Сумма** | **20** | **9** |

Таким образом, мы видим, что прямой поиск дает результат лучше, чем метод эталонных точек через ORB. Освещение и ручное изменение цветов на фотографиях на результат не повлияли.

**Список использованных источников**


1. Template Matching [Электронный ресурс] // URL: https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html (дата обращения: 03.11.2022).
2. Поиск изображений по фрагменту с помощью ORB [Электронный ресурс] // URL: https://vc.ru/dev/249864-poisk-izobrazheniy-po-fragmentu-s-pomoshchyu-orb (дата обращения: 05.11.2022).
3. Задача нахождения объектов на изображении [Электронный ресурс] // URL: https://neerc.ifmo.ru/wiki/index.php?title (дата обращения: 12.11.2022).
4. Детекторы и дескрипторы особых точек FAST, BRIEF, ORB [Электронный ресурс] // URL: https://habr.com/ru/post/414459/ (дата обращения: 12.11.2022).