

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»**

Кафедра инфокоммуникаций

Отчёт по практическому занятию №15

«Проект»

по дисциплине «Теории распознавания образов»

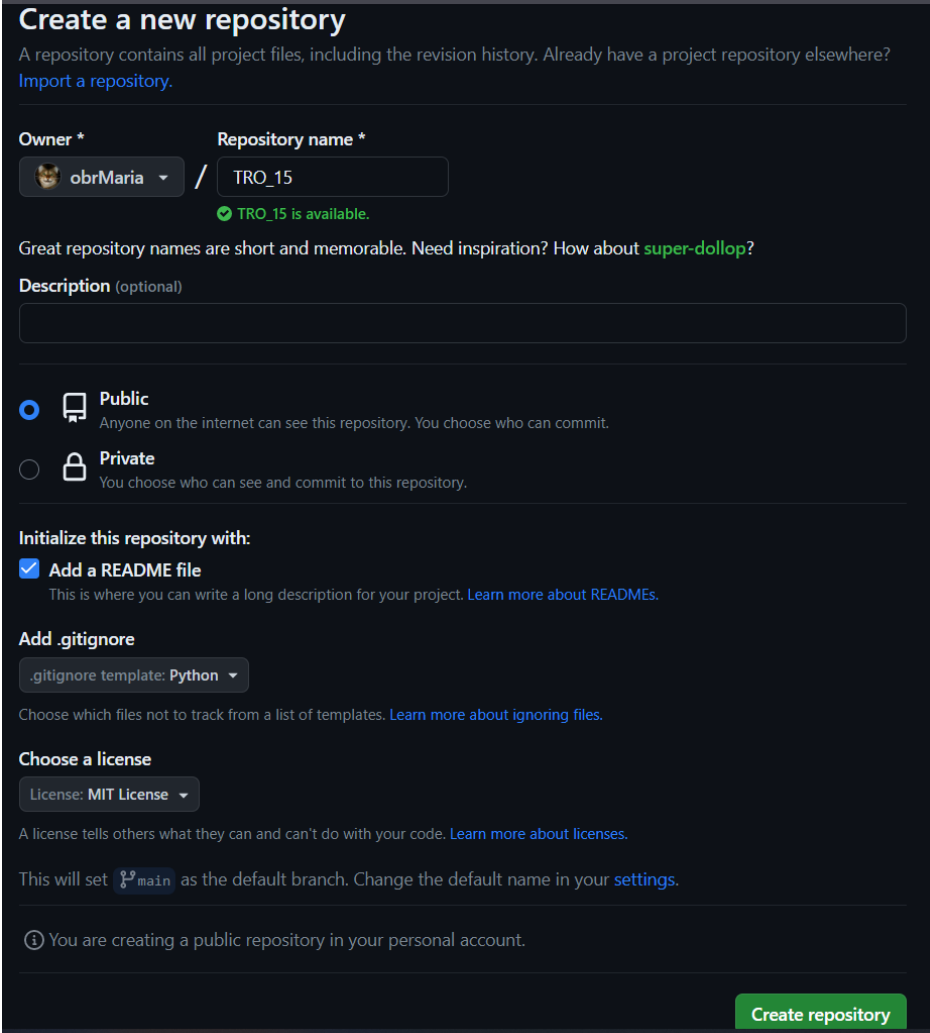
Выполнил студент группы ПИЖ-б-о-21-1
Образцова М.Д. « » _____ 20__ г.
Подпись студента _____
Работа защищена « » _____ 20__ г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель работы: решить задачу, в которой необходимо комплексное использование функций цифровой обработки изображений, которые изучены в приведенных выше заданиях. Методика и порядок выполнения работы

1. Изучить теоретический материал работы.


2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * **Repository name ***

 obrMaria / TRO_15

✔ TRO_15 is available.

Great repository names are short and memorable. Need inspiration? How about [super-dollop](#)?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **Python**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **MIT License**

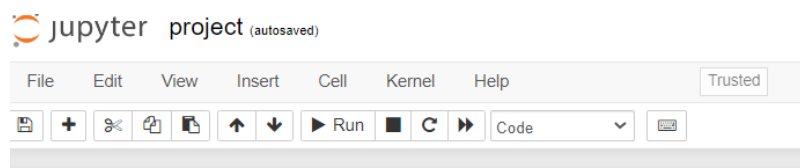
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Create repository

В этом разделе решается задача, в которой необходимо комплексное использование функций цифровой обработки изображений, которые изучены в приведенных выше заданиях. Постановка задачи. Выделить на фоне всего изображения интересные нас объекты. Создать таблицу признаков для сформированного набора объектов. Провести распознавание этих объектов с помощью нейронной сети



Лабораторная работа №15

Проект

Постановка задачи. Выделить на фоне всего изображения интересные нас объекты. Создать таблицу признаков для сформированного набора объектов. Провести распознавание этих объектов с помощью нейронной сети.

Обзор функций, которые используются в проекте

```
In [1]: import cv2
import numpy as np
import imutils
import random
from matplotlib import pyplot as plt
from collections import Counter
from sklearn.cluster import KMeans
```

На первом этапе проводится предварительная обработка изображений, которая включает в себя удаление шума, повышение резкости изображений. С помощью этой обработки выделяются характерные детали, подавляется шум, повышается быстродействие, уменьшается объем информации.

```
In [85]: image = cv2.imread('img/kot.jpg')
```

```
In [86]: # Преобразование изображения в оттенки серого
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
In [87]: # Удаление шума с помощью фильтра Гаусса
denoised = cv2.GaussianBlur(gray, (5, 5), 0)
```

Лабораторная работа №15

Проект

Постановка задачи. Выделить на фоне всего изображения интересные нас объекты. Создать таблицу признаков для сформированного набора объектов. Провести распознавание этих объектов с помощью нейронной сети.

Обзор функций, которые используются в проекте ¶

```
1]: import cv2
import numpy as np
import imutils
import random
from matplotlib import pyplot as plt
from collections import Counter
from sklearn.cluster import KMeans
```

На первом этапе проводится предварительная обработка изображений, которая включает в себя удаление шума, повышение резкости изображений. С помощью этой обработки выделяются характерные детали, подавляется шум, повышается быстродействие, уменьшается объем информации.

```
5]: image = cv2.imread('img/kot.jpg')
```

```
6]: # Преобразование изображения в оттенки серого
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
7]: # Удаление шума с помощью фильтра Гаусса
denoised = cv2.GaussianBlur(gray, (5, 5), 0)

# Повышение резкости с помощью фильтра увеличения резкости
sharpness = cv2.filter2D(denoised, -1, np.array([[ -1, -1, -1], [-1, 9, -1], [-1,
```

```
]: # Вывод результатов
plt.figure(figsize=(12, 4))

plt.subplot(131)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Исходное изображение')
plt.axis('off')

plt.subplot(132)
plt.imshow(denoised, cmap='gray')
plt.title('Изображение с шумоподавлением')
plt.axis('off')

plt.subplot(133)
plt.imshow(sharpness, cmap='gray')
plt.title('Изображение с резкостью')
plt.axis('off')

plt.tight_layout()
plt.show();
```

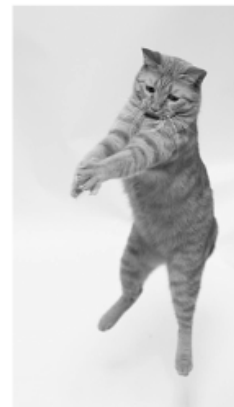
Исходное изображение



Изображение с шумоподавлением



Изображение с резкостью



Следующий шаг – удаление фона на изображении, для этого сканируется все пространство изображения и отсканированные пиксели с одинаковой интенсивностью обнуляются. В результате интересующие нас объекты будут более четко выделены на черном фоне.

Функция `def shelf ()` предназначена для удаления фона. В процессе работы этой программы

```

def shelf(image_path):
    image = cv2.imread(image_path)

    # Конвертация изображения в оттенки серого
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Бинаризация изображения
    _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_O

    # Поиск контуров объекта
    contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX

    # Создание маски для удаления фона
    mask = np.zeros_like(gray)
    cv2.drawContours(mask, contours, -1, (255), thickness=cv2.FILLED)

    # Применение маски на изображение
    result = cv2.bitwise_and(image, image, mask=mask)

    # Отображение изображений до и после удаления фона
    plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)), plt.
    plt.subplot(122), plt.imshow(cv2.cvtColor(result, cv2.COLOR_BGR2RGB)), plt.
    plt.show();

```

```

image_path = 'img/kot.jpg'

```

```

shelf(image_path)

```

Исходное изображение



Изображение без фона



```
plt.show();
```

Оригинальное изображение



```
] : # Вызов функции для сегментации изображения  
marker = segment(img)
```

Пороговое преобразование



```
plt.show();
```

Оригинальное изображение



```
] : # Вызов функции для сегментации изображения  
marker = segment(img)
```

Пороговое преобразование

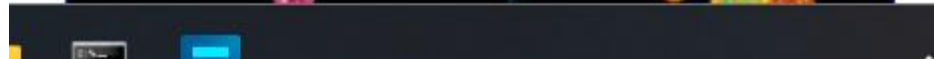




Сanny преобразование



```
# Вывод результата сегментации
plt.imshow(cv2.cvtColor(marker, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show();
```



```
plt.show();
```

```
149]: image_path = 'img/butterfly.jpg'  
      template_path = 'img/pic.jpg'
```

```
150]: match_template(image_path, template_path)
```

Шаблон



Распознавание объекта по шаблону



С помощью операции распознавания объекта по шаблону каждый объект охватывается прямоугольной рамкой.

Создание таблицы признаков

```
151]: img = cv2.imread('img/butterfly.jpg', 0)
```

```
155]: # Отображение бинарного изображения  
      plt.imshow(aa, cmap='gray')  
      plt.title('Пороговое изображение')  
      plt.axis('off')  
      plt.show()
```

Пороговое изображение



```
193]: # Применение адаптивного порогового значения
```

```

>4]: print("Площадь:", area)
      print("Периметр:", perimeter)
      print("Координаты и размеры ограничивающего прямоугольника:", x, y, w, h)
      print("Соотношение сторон:", aspect_ratio)
      print("Эквивалентный диаметр:", equi_diameter)
      print("Моменты контура:", M)

```

```

Площадь: 0.0
Периметр: 0.0
Координаты и размеры ограничивающего прямоугольника: 340 1079 1 1
Соотношение сторон: 1.0
Эквивалентный диаметр: 0.0
Моменты контура: {'m00': 0.0, 'm10': 0.0, 'm01': 0.0, 'm20': 0.0, 'm11': 0.0,
'm02': 0.0, 'm30': 0.0, 'm21': 0.0, 'm12': 0.0, 'm03': 0.0, 'mu20': 0.0, 'mu1
1': 0.0, 'mu02': 0.0, 'mu30': 0.0, 'mu21': 0.0, 'mu12': 0.0, 'mu03': 0.0, 'nu2
0': 0.0, 'nu11': 0.0, 'nu02': 0.0, 'nu30': 0.0, 'nu21': 0.0, 'nu12': 0.0, 'nu0
3': 0.0}

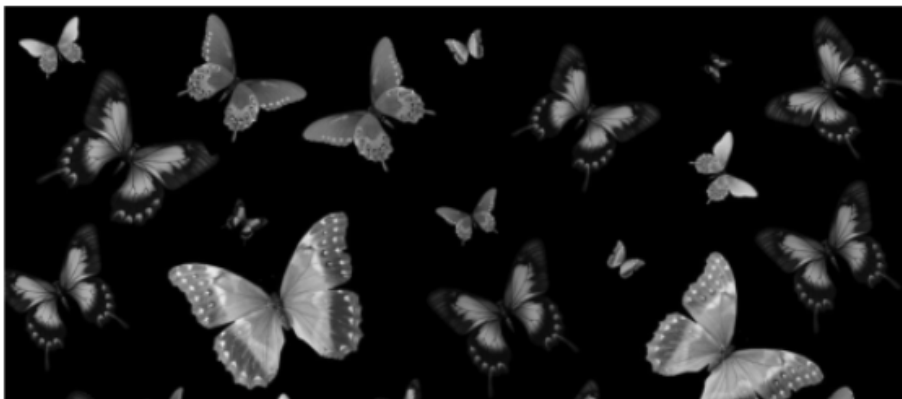
```

```

>5]: imag = cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
      plt.imshow(imag, cmap='gray')
      plt.title('Rectangle Image')
      plt.axis('off')
      plt.show();

```

Rectangle Image



Самая правая точка: (340, 1079)
Самая верхняя точка: (340, 1079)
Самая нижняя точка: (340, 1079)

Алгоритм распознавания объектов.

- По вычисленным признакам всех объектов проводится обучение нейросети, т. е. для каждого объекта по его признакам определяется значение на выходе нейросети и заносится в последнюю строку таблицы. Таким образом каждому объекту сопоставляется маркер.
- Сравнение данных из базы признаков с признаками неизвестного объекта позволяет распознать этот объект и определить его маркер.

```
s = [1249.0, 577.0, 180.5, 7795.5, 163.0, 111.0, 909.5, 29.5]
p = [157.55, 141.34, 90.476, 1262.1, 61.799, 78.064, 175.47, 48.730]
w = [31, 51, 7, 66, 15, 5, 24, 5]
h = [54, 26, 41, 158, 22, 37, 70, 22]
kw = [0.574, 1.9615, 0.1707, 0.4177, 0.6818, 0.1351, 0.3429, 0.2272]
ks = [0.7461, 0.4351, 0.6289, 0.7475, 0.4939, 0.6, 0.5414, 0.2682]
d = [39.878, 27.105, 15.160, 99.627, 14.406, 11.888, 34.029, 6.1287]
m0 = [1249.0, 577.0, 180.5, 7795.5, 163.0, 111.0, 909.5, 29.5]
m1 = [15994, 18479, 3319.8, 255475, 790.83, 174.5, 8059.0, 40.833]
m2 = [27883, 4990.2, 1440.7, 616222, 1220.3, 2033.3, 21165, 160.5]
m3 = [307788, 175877, 180.5, 19613900, 4657.8, 3300.2, 153434, 152.375]
weights = [0.3, 0.5, 0.1, 0.2, 1, 1, 1, 1, 1, 1, 1]
```

```
def sum(a, b):
    assert(len(a) == len(b))
    output = 0
    for i in range(len(a)):
        output += (a[i] * b[i])
    return output
```

```
def art_neuron(input, weights):
    pred = sum(input, weights)
    return pred
```

```
in0 = [s[0], p[0], w[0], h[0], kw[0], ks[0], d[0], m0[0], m1[0], m2[0], m3[0]]
in1 = [s[1], p[1], w[1], h[1], kw[1], ks[1], d[1], m0[1], m1[1], m2[1], m3[1]]
in2 = [s[2], p[2], w[2], h[2], kw[2], ks[2], d[2], m0[2], m1[2], m2[2], m3[2]]
in3 = [s[3], p[3], w[3], h[3], kw[3], ks[3], d[3], m0[3], m1[3], m2[3], m3[3]]
in4 = [s[4], p[4], w[4], h[4], kw[4], ks[4], d[4], m0[4], m1[4], m2[4], m3[4]]
```

```
Самая правая точка: (340, 1079)
Самая верхняя точка: (340, 1079)
Самая нижняя точка: (340, 1079)
```

Алгоритм распознавания объектов.

- По вычисленным признакам всех объектов проводится обучение нейросети, т. е. для каждого объекта по его признакам определяется значение на выходе нейросети и заносится в последнюю строку таблицы. Таким образом каждому объекту сопоставляется маркер.
- Сравнение данных из базы признаков с признаками неизвестного объекта позволяет распознать этот объект и определить его маркер.

```
s = [1249.0, 577.0, 180.5, 7795.5, 163.0, 111.0, 909.5, 29.5]
p = [157.55, 141.34, 90.476, 1262.1, 61.799, 78.064, 175.47, 48.730]
w = [31, 51, 7, 66, 15, 5, 24, 5]
h = [54, 26, 41, 158, 22, 37, 70, 22]
kw = [0.574, 1.9615, 0.1707, 0.4177, 0.6818, 0.1351, 0.3429, 0.2272]
ks = [0.7461, 0.4351, 0.6289, 0.7475, 0.4939, 0.6, 0.5414, 0.2682]
d = [39.878, 27.105, 15.160, 99.627, 14.406, 11.888, 34.029, 6.1287]
m0 = [1249.0, 577.0, 180.5, 7795.5, 163.0, 111.0, 909.5, 29.5]
m1 = [15994, 18479, 3319.8, 255475, 790.83, 174.5, 8059.0, 40.833]
m2 = [27883, 4990.2, 1440.7, 616222, 1220.3, 2033.3, 21165, 160.5]
m3 = [307788, 175877, 180.5, 19613900, 4657.8, 3300.2, 153434, 152.375]
weights = [0.3, 0.5, 0.1, 0.2, 1, 1, 1, 1, 1, 1, 1]
```

```
def sum(a, b):
    assert(len(a) == len(b))
    output = 0
    for i in range(len(a)):
        output += (a[i] * b[i])
    return output
```

```
def art_neuron(input, weights):
    pred = sum(input, weights)
    return pred
```

```
in0 = [s[0], p[0], w[0], h[0], kw[0], ks[0], d[0], m0[0], m1[0], m2[0], m3[0]]
in1 = [s[1], p[1], w[1], h[1], kw[1], ks[1], d[1], m0[1], m1[1], m2[1], m3[1]]
in2 = [s[2], p[2], w[2], h[2], kw[2], ks[2], d[2], m0[2], m1[2], m2[2], m3[2]]
in3 = [s[3], p[3], w[3], h[3], kw[3], ks[3], d[3], m0[3], m1[3], m2[3], m3[3]]
in4 = [s[4], p[4], w[4], h[4], kw[4], ks[4], d[4], m0[4], m1[4], m2[4], m3[4]]
```