

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 3.13
по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Трушева В. О. .«__»__ 2023г.

Подпись студента_____

Работа защищена « » 20 г.

Проверила Воронкин Р.А. _____
(подпись)

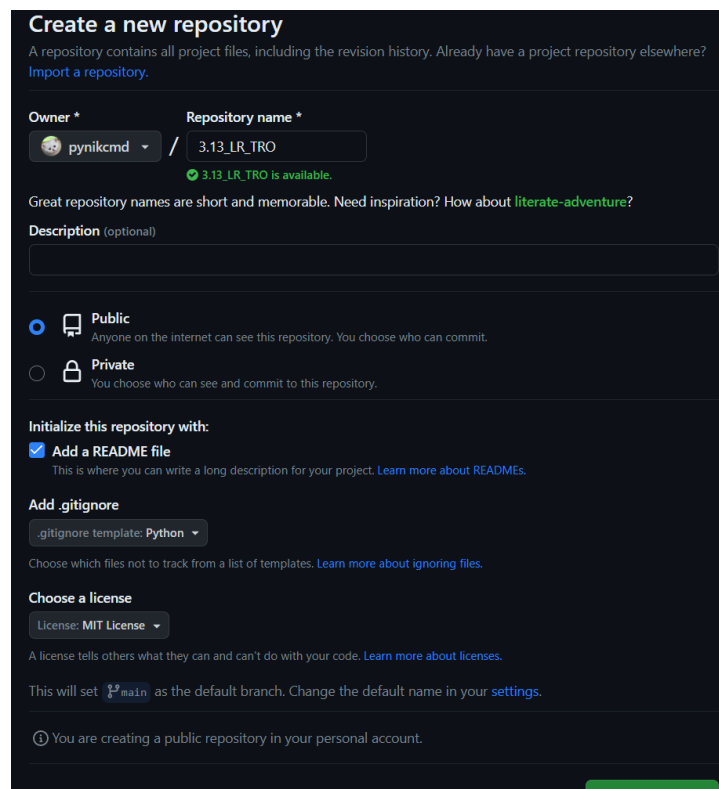
Ставрополь 2023

Цель работы: обнаружение и выделение контуров на изображении, анализ контуров. Изучение функций `cv2.findContours()`, `cv2.drawContours()`.

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).



The screenshot shows the GitHub 'Create a new repository' page. The 'Owner' is 'pynikcmd' and the 'Repository name' is '3.13_LR_TRO'. A green checkmark indicates '3.13_LR_TRO is available'. The 'Description' field is empty. The 'Public' option is selected under 'Initialize this repository with:'. The 'Add a README file' checkbox is checked. The '.gitignore' template is set to 'Python'. The 'License' is set to 'MIT License'. A note at the bottom states 'You are creating a public repository in your personal account.' A green 'Create repository' button is at the bottom right.

3. Выполните клонирование созданного репозитория на рабочий компьютер.

```
D:\fgit>git clone https://github.com/pynikcmd/3.13_LR_TRO.git
Cloning into '3.13_LR_TRO'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

4. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
D:\fgit\3.13_LR_TRO>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/fgit/3.13_LR_TRO/.git/hooks]
```

5. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования, интерактивной оболочки Jupyter notebook и интегрированной среды разработки.

6. Проработать примеры лабораторной работы в отдельном ноутбуке.

```
# Превращаем входное изображения в бинарноинвертированное изображение
thresh = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)
plt.imshow(thresh,cmap = 'gray'),plt.title("Thresh"), plt.axis('off');
```

Thresh



```
# Модуль контуров
contours, hierarchy = cv2.findContours(thresh.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cnt = contours[4]
img = cv2.drawContours(img, [cnt], 0, (0,255,0), 3)
```

```
plt.imshow(img,cmap = 'gray'),plt.title("Contours"), plt.axis('off');
```

Contours



Задание 7.2

Протестировать функцию поиска контура `cv2.findContours` с аргументом `cv2.CHAIN_APPROX_SIMPLE`, который экономит память.

```
img = cv2.imread('img/belka.png', 0)
img = cv2.resize(img, (900, 600))
image = cv2.medianBlur(img, 5) # Функция размытия (сглаживания) изображения медианным фильтром

# Превращаем входное изображения в бинарноинвертированное изображение
thresh = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 3, 10)

# При использовании cv2.CHAIN_APPROX_SIMPLE экономим память
contours, hierarchy = cv2.findContours(thresh.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
# Функция рисует контур (на основе другого контура)
cv2.drawContours(image, contours, -1, (255, 255, 255), 3);

plt.figure(figsize=(10, 10))
plt.subplot(121), plt.imshow(thresh, cmap = 'gray'), plt.title('Thresh')
plt.axis('off')
plt.subplot(122), plt.imshow(image, cmap = 'gray'), plt.title('Contours')
plt.axis('off')
plt.show();
```



Задание 7.3

Выделить границу методом Канни.

```
img = cv2.imread('img/belka.png', 0)
img = cv2.resize(img, (900, 600))
edges = cv2.Canny(img, 700, 100, apertureSize = 3)

plt.figure(figsize=(10, 10))
plt.subplot(121), plt.imshow(img, cmap = 'gray'), plt.title('Original')
plt.axis('off')
plt.subplot(122), plt.imshow(edges, cmap = 'gray'), plt.title('Cannys edges')
plt.axis('off')
plt.show();
```



7. Создать ноутбук, в котором выполнить решение

вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.).

Условие. Выделить контуры с помощью функции `cv2.findContours` и с использованием различных аргументов: `cv2.CHAIN_APPROX_NONE`, `cv2.CHAIN_APPROX_SIMPLE`, `cv2.CHAIN_APPROX_TC89_KCOS`, `cv2.CHAIN_APPROX_TC89_L1`. Вывести изображение с применением метода Канни.

```
image = cv2.imread('img/img.jpg', 0) # Замените 'your_image.jpg' на путь к вашему изображению
# Бинаризация изображения
ret, thresh = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
```

`cv2.CHAIN_APPROX_NONE`: Этот метод не выполняет аппроксимацию контуров и сохраняет все точки контура. Таким образом, каждый контур будет содержать все точки границы, что может быть полезно, если вам нужна точная информация о контуре. Однако количество точек может быть большим, что может привести к повышенному использованию памяти и вычислительным затратам.

```
# Поиск контуров с использованием cv2.CHAIN_APPROX_NONE
contours_none, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
image_copy_none = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
# Рисование контуров с использованием cv2.CHAIN_APPROX_NONE (контур красного цвета)
cv2.drawContours(image_copy_none, contours_none, -1, (255, 0, 0), 2, cv2.LINE_AA);
```

```
plt.imshow(image_copy_none)
plt.title('APPROX: cv2.CHAIN_APPROX_NONE')
plt.axis('off');
```

APPROX: cv2.CHAIN_APPROX_NONE



cv2.CHAIN_APPROX_SIMPLE: Этот метод выполняет простую аппроксимацию контуров путем удаления лишних точек. Он сжимает горизонтальные, вертикальные и диагональные сегменты и оставляет только их конечные точки. Это позволяет представить контур с меньшим количеством точек, что обычно является достаточным для большинства приложений и снижает использование памяти.

```
# Поиск контуров с использованием cv2.CHAIN_APPROX_SIMPLE
contours_simple, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
image_copy_simple = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
# Рисование контуров с использованием cv2.CHAIN_APPROX_SIMPLE (контур зеленого цвета)
cv2.drawContours(image_copy_simple, contours_simple, -1, (0, 255, 0), 2, cv2.LINE_AA);
```

```
plt.imshow(image_copy_simple)
plt.title('APPROX: cv2.CHAIN_APPROX_SIMPLE')
plt.axis('off');
```

APPROX: cv2.CHAIN_APPROX_SIMPLE

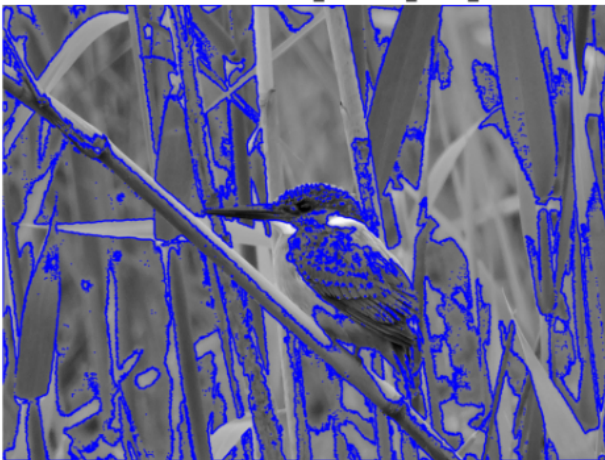


cv2.CHAIN_APPROX_TC89_L1: Этот метод использует аппроксимацию Тайи-Чжонга (Teh-Chin) для сжатия контуров. Он позволяет представить контур с ещё меньшим количеством точек, чем cv2.CHAIN_APPROX_SIMPLE, но с сохранением допустимой точности формы контура.

```
] # Поиск контуров с использованием cv2.CHAIN_APPROX_TC89_L1
contours_tc89_l1, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_TC89_L1)
image_copy_tc89_l1 = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
# Рисование контуров с использованием cv2.CHAIN_APPROX_TC89_L1 (контур синего цвета)
cv2.drawContours(image_copy_tc89_l1, contours_tc89_l1, -1, (0, 0, 255), 2, cv2.LINE_AA);
```

```
] plt.imshow(image_copy_tc89_l1)
plt.title('APPROX: cv2.CHAIN_APPROX_TC89_L1')
plt.axis('off');
```

APPROX: cv2.CHAIN_APPROX_TC89_L1



cv2.CHAIN_APPROX_TC89_KCOS: Этот метод также использует аппроксимацию Тайи-Чжонга (Teh-Chin), но с использованием метода k-косинусов для оценки качества аппроксимации. Это позволяет сохранить ещё больше деталей контура, чем cv2.CHAIN_APPROX_TC89_L1, но может потребовать больше вычислительных ресурсов.

```
[40]: # Поиск контуров с использованием cv2.CHAIN_APPROX_TC89_KCOS
contours_tc89_kcos, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_TC89_KCOS)
image_copy_tc89_kcos = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
# Рисование контуров с использованием cv2.CHAIN_APPROX_TC89_KCOS (контур желтого цвета)
cv2.drawContours(image_copy_tc89_kcos, contours_tc89_kcos, -1, (255, 255, 0), 2, cv2.LINE_AA);
```

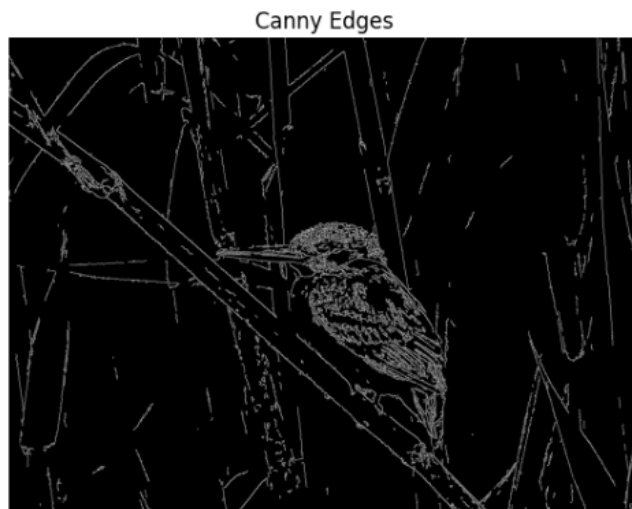
```
[41]: plt.imshow(image_copy_tc89_kcos)
plt.title('APPROX: cv2.CHAIN_APPROX_TC89_KCOS')
plt.axis('off');
```



Применение метода Канни для выделения границ

```
: edges = cv2.Canny(image, 100, 200) # Параметры порогов для детектирования границ
```

```
: plt.figure()
plt.imshow(edges, cmap='gray')
plt.title('Canny Edges')
plt.axis('off');
```



8. Зафиксируйте сделанные изменения в репозитории.

9. Выполните слияние ветки для разработки с веткой main(master).

10. Отправьте сделанные изменения на сервер GitHub.