МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Нахождение и обработка контуров»

Отчет по лабораторной работе № 13 (7)

по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Кучеренко С. Ю. « » 2023 г.

Подпись студента_____

Работа защищена « »_____2023 г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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

Выполнение работы:

Примеры лабораторной работы

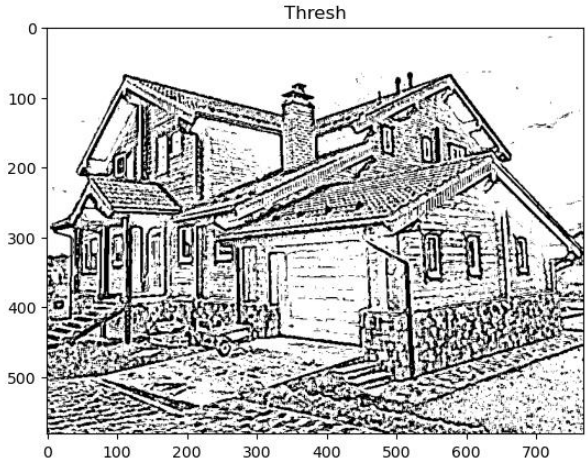
Задание 7.1.

С помощью функции cv2.findContours найти все контуры изображения.

In [4]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

In [5]:
```
img = cv2.imread('img/house.jpg', 0)
img = cv2.medianBlur(img, 5)
```

In [6]:
```
thresh = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                               cv2.THRESH_BINARY,11,2)
plt.imshow(thresh,cmap = 'gray'),plt.title("Thresh");
```



In [7]:
```
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)
```

In [12]:
```
plt.figure(figsize=(15,15))
plt.subplot(121),
plt.imshow(img,cmap = 'gray'),plt.title("Contours"),
plt.axis('off')

plt.subplot(122),
plt.imshow(thresh,cmap = 'gray'),plt.title("Thresh"),
plt.axis('off');
```

Задание 7.2.

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

In [26]:
```
img = cv2.imread('img/smile.jpg', 0)
img = cv2.resize(img, (900, 600))
image = cv2.medianBlur(img, 5)
```

In [27]:
```
thresh = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                               cv2.THRESH_BINARY_INV, 3,10)
```

In [28]:
```
contours, hierarchy = cv2.findContours(thresh.copy(), cv2.RETR_LIST,
                                       cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(image, contours, -1, (255,255,255),3);
```

In [30]:
```
plt.figure(figsize=(15,15))
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.

In [31]:
```
img = cv2.imread('img/smile.jpg', 0)
img = cv2.resize(img, (900, 600))
edges = cv2.Canny(img, 700, 100, apertureSize = 3)
```

In [32]:
```
plt.figure(figsize=(15,15))
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 s edges')
plt.axis('off')
plt.show()
```

Самостоятельное задание

Сравнените 4 метода обнаружения контуров, проанализируйте и сравните результаты:

- алгоритм Кэнни
- оператор Собеля
- оператор Робертса
- оператор Лапласа

In [66]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

In [67]:
```
image = cv2.imread('img/ind1.jpg', 0)
```

Применение алгоритма Кэнни для обнаружения контуров

In [68]:
```
canny_edges = cv2.Canny(image, 100, 200)
```

Применение оператора Собеля для обнаружения контуров

In [69]:
```
sobelx = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel_edges = cv2.magnitude(sobelx, sobely)
```

Применение оператора Робертса для обнаружения контуров

In [70]:
```
ImageCopy = image.copy()
```

In [71]:
```
kernel_x = np.array([[1, 0], [0, -1]])
kernel_y = np.array([[0, 1], [-1, 0]])
gradient_x = cv2.filter2D(ImageCopy, cv2.CV_64F, kernel_x)
gradient_y = cv2.filter2D(ImageCopy, cv2.CV_64F, kernel_y)
```

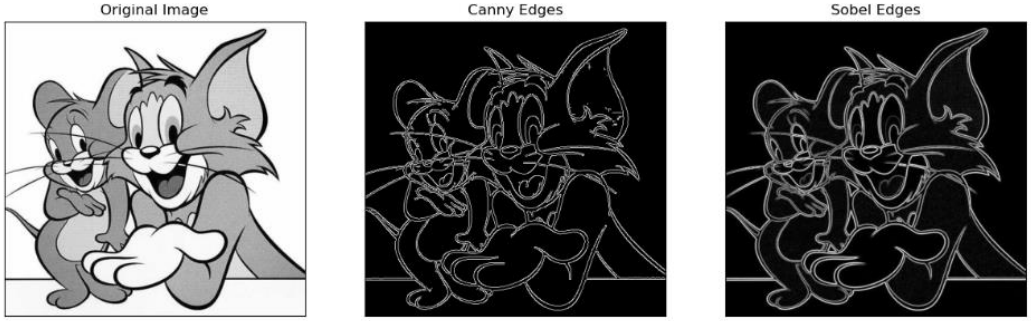
In [72]:
```
# Вычисление абсолютного значения градиента и его преобразование в uint8
gradient_abs = np.abs(gradient_x) + np.abs(gradient_y)
gradient_abs = np.uint8(gradient_abs)
```

In [73]:
```
threshold = 50
ret, roberts = cv2.threshold(gradient_abs, threshold, 255, cv2.THRESH_BINARY)
```

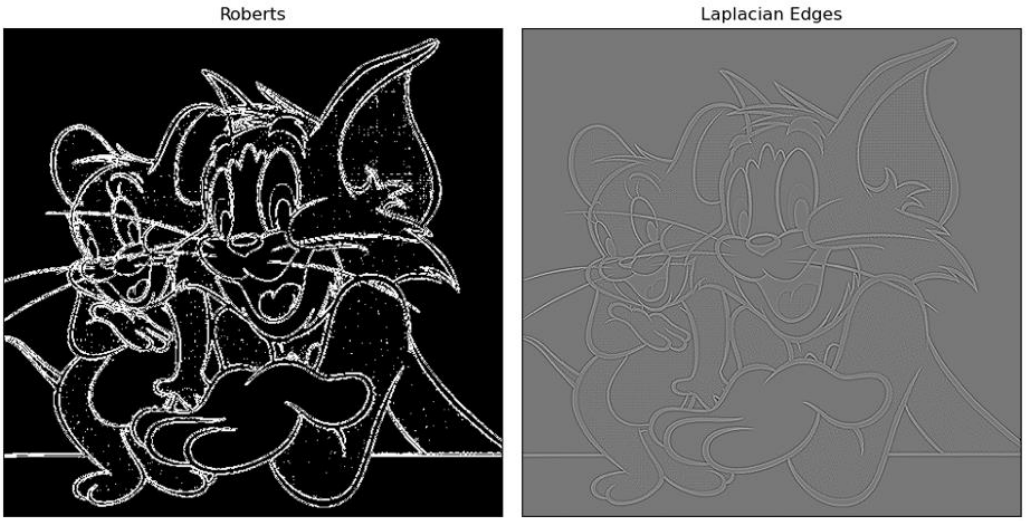
Применение оператора Лапласа для обнаружения контуров

In [74]:
```
laplacian_edges = cv2.Laplacian(image, cv2.CV_64F)
```

In [75]:
```
plt.figure(figsize=(15,15))
plt.subplot(131), plt.imshow(image, cmap='gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(132), plt.imshow(canny_edges, cmap='gray')
plt.title('Canny Edges'), plt.xticks([]), plt.yticks([])

plt.subplot(133), plt.imshow(sobel_edges, cmap='gray')
plt.title('Sobel Edges'), plt.xticks([]), plt.yticks([]);
```



In [76]:
```
plt.figure(figsize=(15,15))
plt.subplot(132), plt.imshow(roberts, cmap='gray')
plt.title('Roberts'), plt.xticks([]), plt.yticks([])

plt.subplot(133), plt.imshow(laplacian_edges, cmap='gray')
plt.title('Laplacian Edges'), plt.xticks([]), plt.yticks([]);

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



Анализ и сравнение результатов по количеству обнаруженных контуров

In [77]:
```
contour_counts = [np.count_nonzero(canny_edges), np.count_nonzero(sobel_edges),
                  np.count_nonzero(roberts), np.count_nonzero(laplacian_edges)]
method_names = ['Canny', 'Sobel', 'Roberts', 'Laplacian']

plt.bar(method_names, contour_counts)
plt.title('Number of Contours Detected')
plt.xlabel('Method')
plt.ylabel('Count')
plt.show()
```