Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе № 4
«Выделение контуров на изображениях»

Курс: «Системы обработки изображений»

Студент: Грязнов Илья Евгеньевич
(фамилия, имя отчество)
Группы 6131-010402D

Самара 2022

Оглавление

Задание

Вариант № 8

1. Считать цветное rgb изображение. Преобразовать в градации серого.

2. Сделать выделение контуров методом простого градиента. В качестве значения модуля градиента использовать указанный в вариантах метод.

* Вход: изображение из пункта 1
* Вывод: бинарное изображение с контурами

3. Сделать выделение контуров методом по вариантам.

* Вход: изображение из пункта 1
* Вывод: бинарное изображение с контурами

4. Сделать выделение контуров методом с согласованием. Тип функции аппроксимации и размер окна указан по вариантам.

* Вход: изображение из пункта 1
* Вывод: бинарное изображение с контурами

Варианты задания

№ варианта	Задание 2	Задание 3	Задание 4
1	модуль градиента как он есть	Оператор Робертса	Аппроксимация поверхностью 2-го порядка, окно 3x3
2	модуль градиента аппроксимируется суммой модулей производных	Оператор Собеля	Аппроксимация поверхностью 2-го порядка, окно 3x3
3	модуль градиента аппроксимируется максимумом среди модулей производных	Оператор Лапласа	Аппроксимация поверхностью 1-го порядка, окно 3x3
4	модуль градиента как он есть	Оператор Лапласа	Аппроксимация поверхностью 1-го порядка, окно 3x3
5	модуль градиента аппроксимируется суммой модулей производных	Оператор Робертса	Аппроксимация поверхностью 2-го порядка, окно 3x3
6	модуль градиента аппроксимируется максимумом среди модулей производных	Оператор Робертса	Аппроксимация поверхностью 2-го порядка, окно 3x3
7	модуль градиента аппроксимируется суммой модулей производных	Оператор Собеля	Аппроксимация поверхностью 2-го порядка, окно 3x3
8	модуль градиента как он есть	Оператор Лапласа	Аппроксимация поверхностью 1-го порядка, окно 3x3
9	модуль градиента аппроксимируется максимумом среди модулей производных	Оператор Лапласа	Аппроксимация поверхностью 1-го порядка, окно 3x3
10	модуль градиента аппроксимируется суммой модулей производных	Оператор Собеля	Аппроксимация поверхностью 2-го порядка, окно 3x3

Ход выполнения работы:

1. **Считать цветное rgb изображение. Преобразовать в градации серого.**

Подгрузили цветное изображение:

In [2]:
```
plt.figure(figsize=(14, 8))
img = cv2.imread("tofly.jpeg")[..., ::-1]
plt.imshow(img, vmin=0, vmax=255)
```

Out[2]: <matplotlib.image.AxesImage at 0x1ebc2cc78b0>

Преобразовали в градации серого:

[3]:
```
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

plt.figure(figsize=(14, 8))
plt.imshow(img_gray, vmin=0, vmax=255, cmap='Greys_r')
```

[3]:
```
<matplotlib.image.AxesImage at 0x1ebc39b7d60>
```

2. Сделать выделение контуров методом простого градиента. В качестве значения модуля градиента использовать указанный в вариантах метод.

Ниже описана функция для выделения контуров методом простого градиента. Данная операция заключается в выполнении нелинейной совместной поэлементной обработке изображений S_1 и S_2 следующего вида:

$$e(m,n) = \sqrt{S_1^2 + S_2^2}$$

[5]:
```
grad_x = np.array([[-1, 1]])
grad_y = np.array([[-1], [1]])
```

[6]:
```
def convolution(mt, kernel):
    kernel = np.rot90(kernel, k = 2)
    mt = mt.ravel()
    z = [mt[i:i + len(kernel.ravel())] for i in range(0, len(mt), len(kernel.ravel()))]
    t = z * kernel.ravel()
    return t.sum()

def derivative(img, h):
    image1 = img.copy().astype(int)
    x, y = h.shape
    for a in range(0, image1.shape[0]-x+1):
        for b in range(0, image1.shape[1]-y+1):
            matrx1 = img[a:a + x, b:b + y]
            image1[a,b]=convolution(matrx1,h)
    return image1

def gradient(image, grad_x, grad_y):
    g1 = derivative(image, grad_x)
    g2 = derivative(image, grad_y)
    return np.sqrt(np.square(g1) + np.square(g2)).astype(int)
```

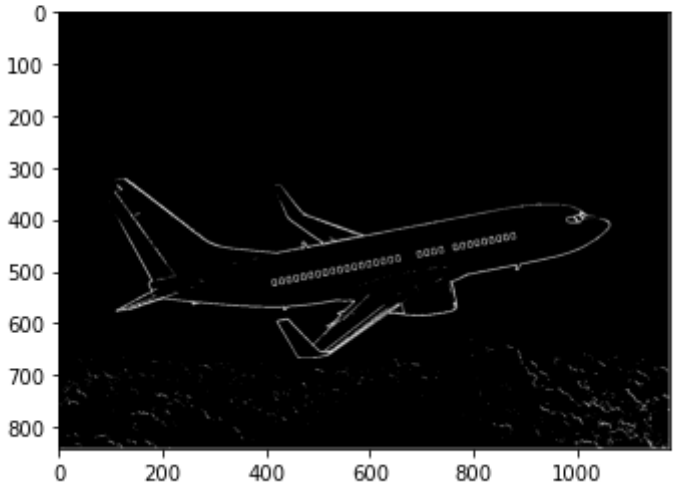
[7]:
```
grad_out = gradient(img_gray, grad_x, grad_y)
```

[8]:
```
def thresholding(image,p):
    change_image = np.where(image <= p, 0, 255)
    return change_image
```

Результат обработки изображения:

[9]:
```
thresh_out = thresholding(grad_out, 25)
plt.imshow(thresh_out, cmap='gray')
```

[9]: <matplotlib.image.AxesImage at 0x1ebc3b816d0>



3. Сделать выделение контуров методом по вариантам.

В отличие от градиентного метода выделения границ, базирующегося на использовании первой производной по направлению, метод оператора Лапласа использует производные второго порядка. Если функция яркости f(m,n) является функцией непрерывных аргументов, то лапласиан (оператор Лапласа) определяется следующим образом:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

В отличие от градиента лапласиан является скалярной функцией. Его применение для выделения границ основано на том, что лапласиан принимает максимальное (по абсолютной величине) значение на участках “перегибов” функции яркости.

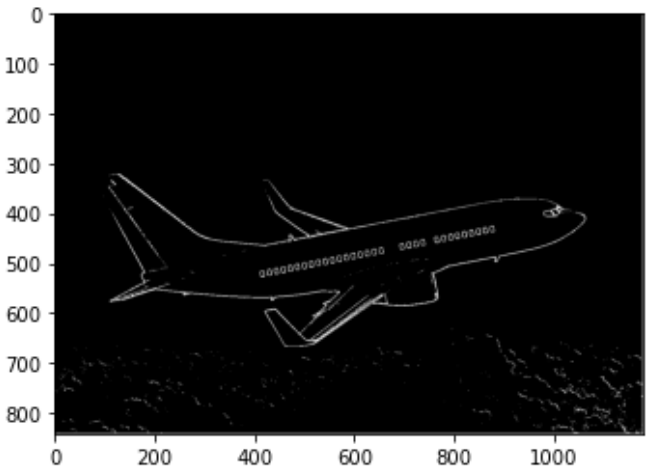
[11]:
```
laplace = np.array([[0,1,0], [1,-4,1], [0,1,0]])
```

[12]:
```
def laplace_operator(image,h):
    l = derivative(image, h)
    return np.abs(l)
```

[13]:
```
lapl_out = laplace_operator(img_gray, laplace)
```

[14]:
```
thresh_out = thresholding(grad_out, 25)
plt.imshow(thresh_out, cmap = 'gray')
```

t[14]:
```
<matplotlib.image.AxesImage at 0x1ebc3d39070>
```



4. Сделать выделение контуров методом с согласованием. Тип функции аппроксимации и размер окна указан по вариантам.

Общим недостатком рассмотренных выше методов выделения границ является высокая чувствительность к шуму. Это объясняется тем, что действие дифференциальных (разностных) операторов состоит в вычислении и комбинировании разностей отсчетов в пределах “окна” малых размеров. Из теории цифровой обработки сигналов известно, что операция дифференцирования приводит к ослаблению низкочастотных составляющих пространственной функции яркости и к усилению высокочастотных составляющих. В то же время высокочастотная составляющая функции яркости в большой степени обусловлена именно действием шума. Поэтому операторы градиента и лапласиана усиливают шум, что ведет к появлению ложных контуров. Чтобы повысить помехоустойчивость рассмотренных методов, перед выполнением операции дифференцирования производят сглаживание функции яркости в пределах “окна”, например, “подгоняют” или согласуют с функцией яркости выбранную поверхность первого или второго порядков. Такой подход приводит к методу согласования.

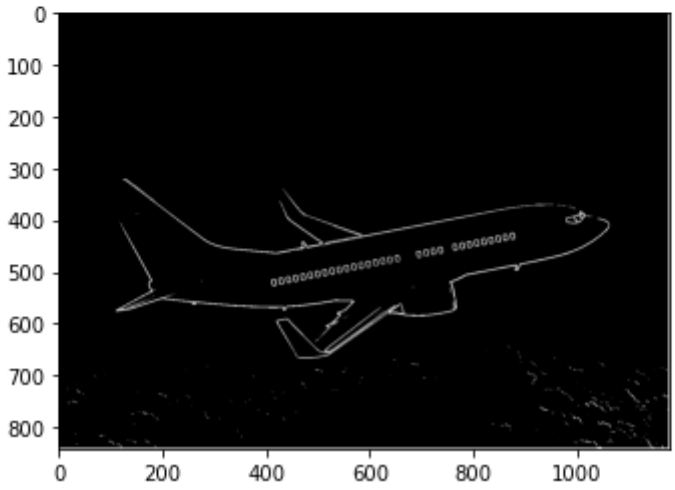
[16]:
```
first = 1/6 * np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])
second = 1/6 * np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
```

[17]:
```
def matching(image, grad_x, grad_y):
    f = derivative(image, grad_x)
    s = derivative(image, grad_y)
    return np.sqrt(np.square(f) + np.square(s)).astype(int)
```

[18]:
```
out_matching = matching(img_gray, first, second)
```

[19]:
```
fine = thresholding(out_matching, 20)
plt.imshow(fine, cmap='gray')
```

t[19]:
```
<matplotlib.image.AxesImage at 0x1ebc3d97fd0>
```



ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы было взято цветное изображение и преобразовано в градации серого. Далее были реализованы функции для обработки изображения методом простого градиента с модулем по вариантам. В процессе был изучен принцип работы данного метода, который заключается в выполнении нелинейной совместной поэлементной обработке изображений.

Следующим шагом было выделение контуров методом оператора Лапласа. Изучен метод, его определения и способы реализации.

В заключении были выделены контура методом с согласованием изучен поход к этому методу и его преимущества. Результаты работы функций наглядно отображены на выведенных изображениях.