ЛАБОРАТОРНАЯ РАБОТА №1. Базовые операции с изображениями в Python

ЦЕЛЬ РАБОТЫ – научиться читать, отображать и записывать изображения в Python.

Содержание

1 Теоретические сведения

1.1 Базовые понятия

Изображение можно определить как двухмерную функцию $f(x, y)$, где x и y – пространственные координаты, а амплитуда f для каждой пары (x, y) называется интенсивностью или яркостью изображения в точке (x, y).

Для обозначения яркости монохромных изображений используется термин – ***уровень серого***. Пример монохромного изображения приведен на следующем рисунке.

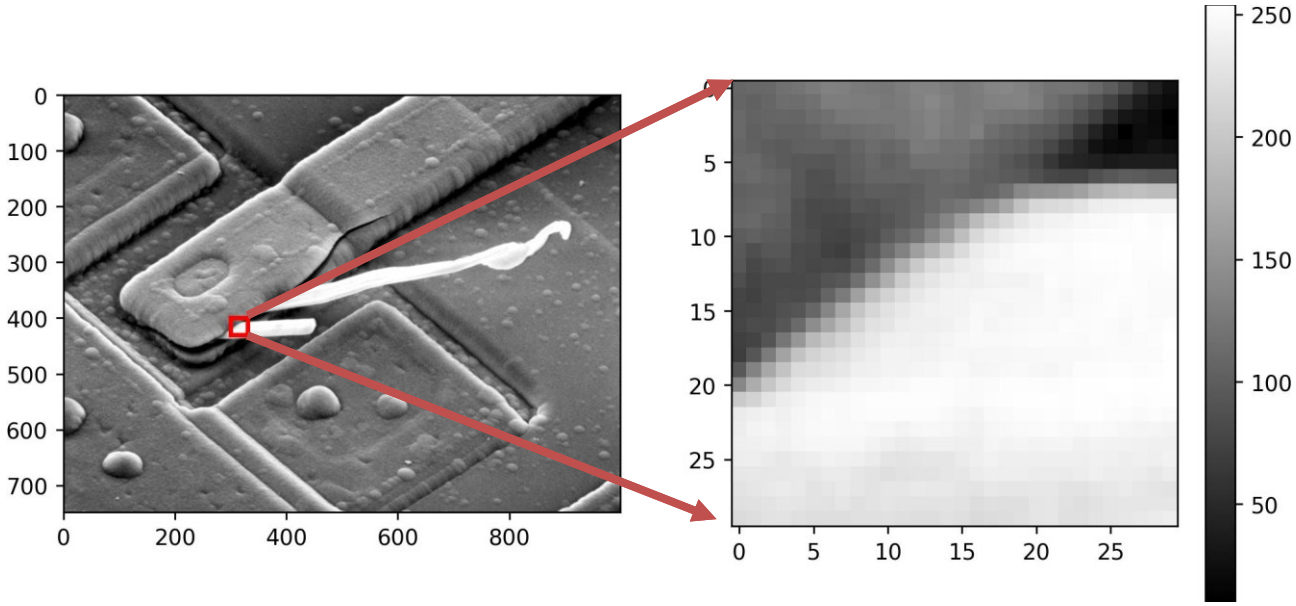

Рисунок 1 – Пример монохромного изображения

Если в результате дискретизации изображения $f(x, y)$ получена матрица у которой M строк и N столбцов, то говорят, что изображение имеет размер $M \times N$.

В литературе по обработке изображений за начало координат принимают верхний левый угол изображения, координатами которого служит пара $(0,0)$. Следующая точка в первой строке изображения имеет координаты $(x, y) = (0,1)$. На рисунке 2 дана иллюстрация координатного соглашения.

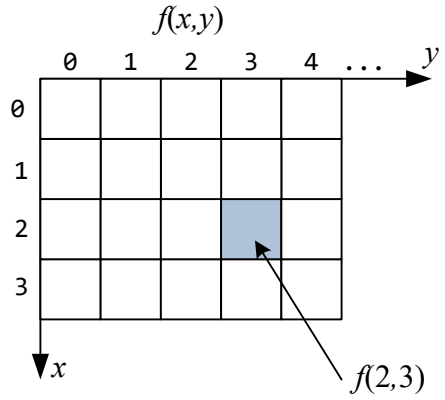

Рисунок 2 – Координатное соглашение в книгах по обработке изображений

1.2 Чтение изображения в Python

Для чтения изображения в Python можно воспользоваться библиотекой `matplotlib`. Для этого её необходимо в начале импортировать в пространство имен:

```
>>> import matplotlib.pyplot as plt
```

Для загрузки изображения применим команду `imread`:

```
>>> my_img = plt.imread("img\\blown_ic_hr.tif")
```

В данном случае изображение имеет расширение tif и располагается в директории img. Чтобы узнать размер прочитанного изображения необходимо прочитать поле `shape` объекта `my_img`:

```
>>> print ('Shape = ', my_img.shape)
```

В результате на экране выведется следующая строка:

```
Shape = (748, 1000)
```

Мы узнали, что прочитанное изображение имеет 748 строк и 1000 столбцов. Дополнительно мы может установить какой формат данных используется для хранения пикселей изображения:

```
>>> print ('Type = ', my_img.dtype)

Type = uint8
```

Это означает, что изображение хранится в формате беззнаковых 8-битных чисел. 8-битные изображения являются очень распространенными на практике.

1.3 Вывод изображения на экран

Для вывода изображения на экран нужно воспользоваться функцией `imshow` из библиотеки `matplotlib`.

Графики в matplotlib «живут» в объекте `figure`. Создать новый рисунок можно методом `plt.figure`:

```
>>> fig = plt.figure()
```

У команды plt.figure есть ряд параметров. Наиболее важным является fig-size, который гарантирует, что при сохранении рисунка на диск у него будет определенный размер и отношение сторон. Например, чтобы создать окно вывод с соотношением сторон 10 к 5 нужно выполнить команду

```
>>> fig = plt.figure(figsize=(10,5))
```

Часто требуется выводить в одном окне сразу несколько графиков (или изображений). Для этой задачи лучше подойдет команда

```
>>> fig = plt.subplots(figsize=(10,5))
```

Для разделения области окна вывода на отдельный сегменты и перехода к выводу в отдельную область используется метод

```
>>> plt.subplot(r, c, num)
```

Здесь `r` и `c` – это число строк и столбцов, на которые разбивается область вывода, а `num` – номер текущей области.

Воспользуемся изученными командами для вывода изображения `my_img` и его «негатива» на одном графике:

```
[1]: my_img_neg = 255 - my_img
[2]: fig, ax = plt.subplots(figsize = (10,5))
[3]: plt.subplot(1,2,1)
[4]: plt.imshow(my_img, cmap='gray')
[5]: plt.subplot(1,2,2)
[6]: plt.imshow(my_img_neg, cmap='gray')
```

В результате выполнения команд на экране должно появится следующее изображение.

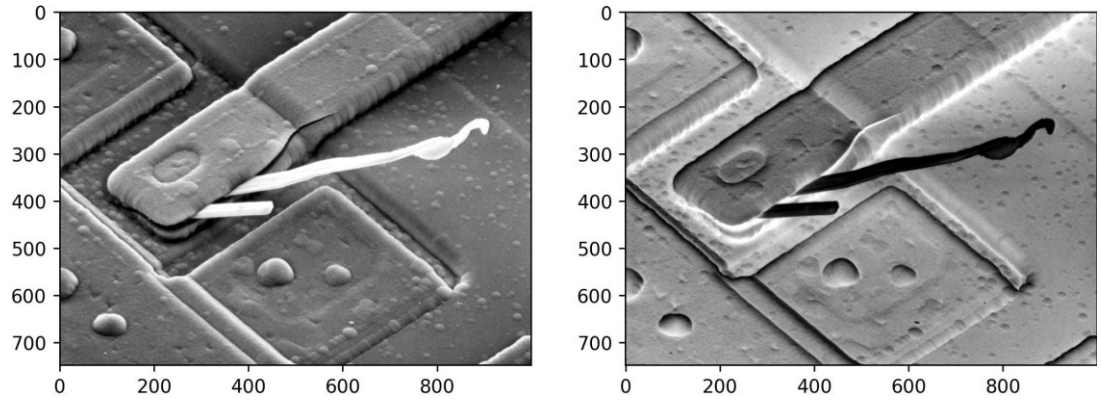

Рисунок 3 – Изображение и его «негатив»

В первой строке мы создали «негатив» исходного изображения используя формулу

$$g(x,y) = 255 - f(x,y), \tag{1.1}$$

где $f(x,y)$ – исходное изображение, а $g(x,y)$ – результирующее изображение.

Третья и четвертая строки кода используются для отображения исходного изображения в первой строке и первом столбце области вывода. Обратите внимание на то, что для отображения монохромного изображения мы выбрали цветовую палитру (`cmap`) с названием `'gray'` для отображения *в градациях серого* цвета. Существуют и другие цветовые палитры.

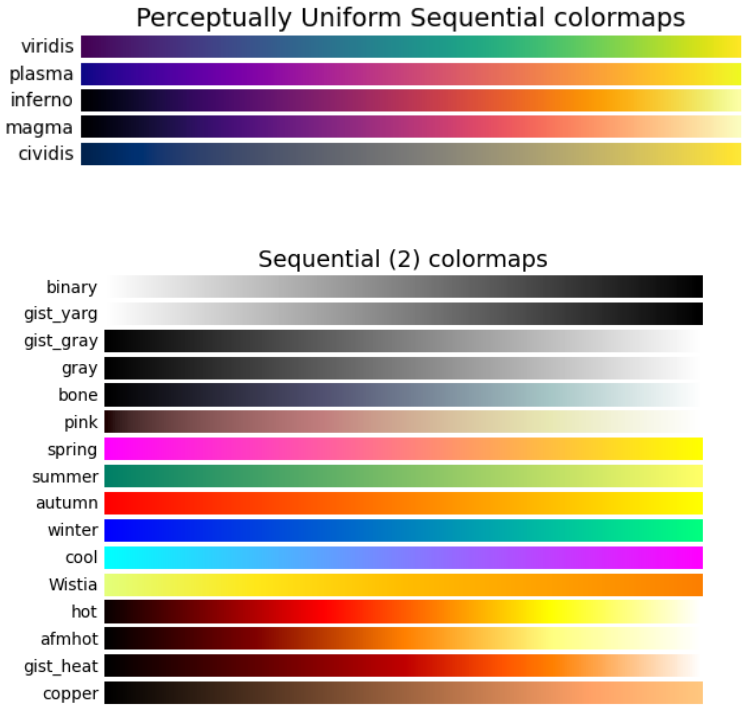

Рисунок 4 – Цветовые палитры (`cmap`)

1.4 Сохранение изображения на диск

Для сохранения сформированного графика на диск используется следующие команды:

```
[7]: image_name = 'figures\\gray_image_neg_example.jpg'
[8]: fig.savefig(image_name,         format='jpg',         dpi=250,
bbox_inches='tight', pad_inches = 0)
```

В 7-й строке задается имя файла, в который будет записан график. В следующей строке с использованием метода `fig.savefig()` выполняется запись полученного результата. Функция `fig.savefig()` имеет много параметров. Первый обязательный (позиционный) параметр – это имя файла. Далее следуют именованные параметры `format` – задает формат изображения, `dpi` – число точек (пикселов) на дюйм (*dot per inch*), `bbox_inches` – ограничивающая рамка в дюймах: сохраняется только заданная часть рисунка, `pad_inches` –отступы вокруг фигуры (используется, когда `bbox_inches` имеет значение `'tight'`.

1.5 Интерполяция цифровых изображений

Интерполяция – процесс использования данных для оценки неизвестных значений. Интерполяция используется при увеличении (*zooming*), поворотах, геометрической коррекции изображений.

Интерполяцию также называют ресемплингом (*resampling*), поскольку она представляет собой метод, который позволяет увеличить (или уменьшить) число пикселей в цифровом изображении. Некоторые цифровые камеры используют интерполяцию для получения изображения с разрешением выше, чем размер фоточувствительной матрицы (*digital zoom*).

Пусть изначально изображение задано, как двумерная функция целочисленного аргумента (рис. 5).

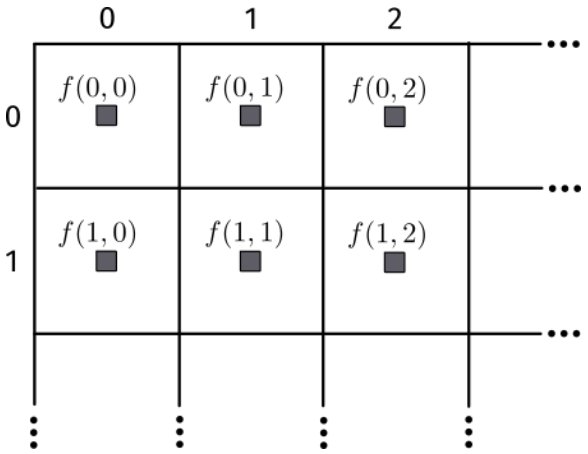

Рисунок 5 – Цифровое изображение

Допустим необходимо выполнить интерполяцию изображения с шагом α. Если $0 < \alpha < 1$, то мы получим увеличения изображения, если $\alpha > 1$, то изображения получится меньшего размера.

На рис. 6 показана новая координатная сетка с шагом $\alpha = 0{,}3$.

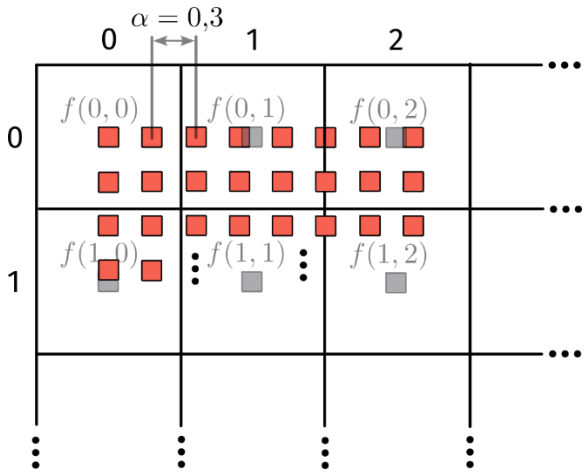

Рисунок 6 – Новый шаг сентки цифрового изображения

При выполнении интерполяции необходимо определить значение интенсивности изображения в новой сетке координат. Эта задача решается в каждом конкретном методе интерполяции по-своему.

Процесс интерполяции разбивается на два этапа:

1) Формирование новой координатной сетки $\mathbf{n}$ для получения нового изображения:

$$g(\mathbf{n}) = f(\mathbf{n}') = f[\mathbf{a}(\mathbf{n})], \tag{1.2}$$

где координаты $\mathbf{a}(\mathbf{n}) = [a_1(n_1, n_2),\ a_2(n_1, n_2)]$ обычно имеют не целые значения.

2) Вычисление (интерполяция) значений g в нецелых координатах $a_1(n_1, n_2),\ a_2(n_1, n_2)$.

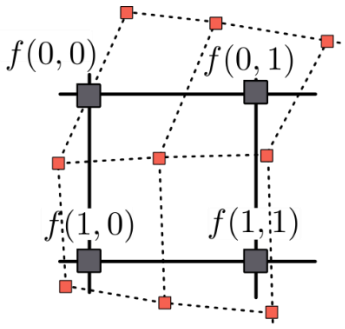

Рисунок 7 – Координатная сетка

Интерполяция по ближайшему соседу

В данном методе, чтобы присвоить значение яркости любому пиксел нового изображения g, найдем ближайший к нему пиксель исходного изображения

и припишем его яркость данному элементу сетки. Это утверждение имеет следующую интерпретацию

$$g(x,y) = f(floor\{x + 0{,}5\}, floor\{y + 0{,}5\}), \tag{1.3}$$

где $floor\{r\}$ – означает ближайшее целое число меньшее, либо равное r.

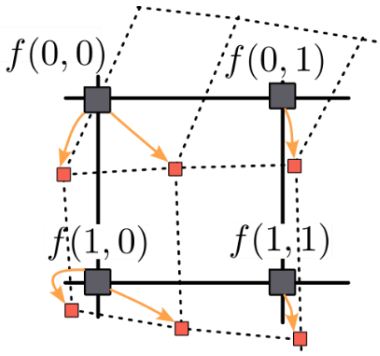

Рисунок 8 – Интерполяция по ближайшему соседу

Билинейная интерполяция

При билинейной интерполяции значение пикселя нового изображения g ищется, как линейная комбинация четырех ближайших пикселей исходного изображения. Математически это записывается следующим образом:

$$\begin{aligned} g(x,y) = (1-a)\cdot(1-b)\cdot f(l,k) + a\cdot(1-b)\cdot f(l+1,k) + \\ +(1-a)\cdot b\cdot f(l,k+1) + a\cdot b\cdot f(l+1,k+1), \end{aligned} \tag{1.4}$$

где $l = floor(x), k = floor(y), a = x - l, b = y - k.$

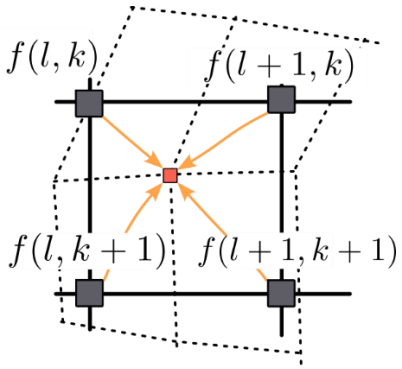

Рисунок 9 – Билинейная интерполяция

Бикубическая интерполяция

В отличие от билинейной интерполяции, которая учитывает только 4 пикселя (2×2), бикубическая интерполяция учитывает 16 пикселей (4×4). Изображения, передискретизированные с помощью бикубической интерполяции, могут иметь различные артефакты. Математическая запись бикубической интерполяции следующая:

$$g(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j. \quad (1.5)$$

Для выполнения бикубической интерполяции можно использовать функцию `resize` (из библиотеки `cv2`) с значением `interpolation=cv2.INTER_CUBIC`, как показано в следующем примере.

```
import cv2

img = plt.imread('img/zebra_32x32.png')
img_resized = cv2.resize(img, (0,0), fx=.5, fy=.5, interpolation=cv2.IN-
TER_CUBIC)
```

2 Порядок выполнения работы

1) Найти в сети Интернет монохромное изображение размером не более 1000x1000 пикселей.

2) Загрузить изображение с помощью команды `plt.imread`.

3) Вывести на экран информацию о изображении (`shape`, `dtype`).

4) Отобразить изображение с помощью команды `imshow`. Используйте в качестве цветовой палитры один из вариантов, приведенных на рисунке 4.

5) Выполнить обработку изображения, согласно варианту

Вариант	Задание
1,4,7	Получить «негатив» исходного изображения и отобразить его экране. Записать результирующее изображение в файл.
2,5,8	Поэтапно понизить пространственное разрешение изображения, пока одна из сторон не станет меньше 32 пикселей. Отобразить полученные изображения на экране.
3,6	Изменить яркостное разрешение изображения путем «зануления» части битов. Для 3-го варианта сохранить 2 старших бита, а для 6-го варианта сохранить 3 старших бита.

6) Найти в сети Интернет монохромное изображение размером 32x32 пикселя и выполнить его интерполяцию согласно варианту.

Вариант	Тип интерполяции
1,2,3	По ближайшему соседу
4,6,7	Билинейная
5,8	Бикубическая

7) Оформить отчет в соответствии с СТП 01-2017.

3 Дополнительные задания и вопросы

1) Разделите изображение по вертикали на 3 равные части и поменяйте местами 1-ю и 3-ю части. Полученный результат отобразите на экране.

2) Что такое координатное соотношение?

3)