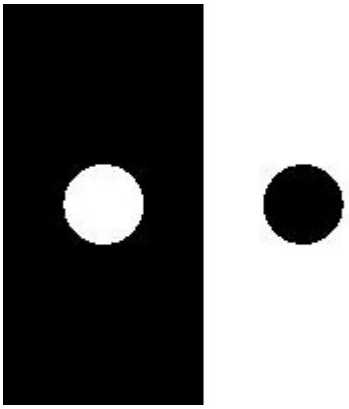
Лабораторная работа (анализ изображений/восстановление изображений);



Теоретическая информация.

I IMAGE.PY

1. Основные понятия/термины.

Изображение представляет собой визуальное представление данных, состоящее из множества пикселей, каждый из которых содержит информацию о цвете или яркости. Различные аспекты работы с изображениями включают в себя импортирование необходимых библиотек, чтение и запись изображений, а также понимание форматов цветового представления.

a) Импортирование библиотеки Pillow

Библиотека Pillow является мощным инструментом для работы с изображениями в языке программирования Python. Чтобы использовать ее, необходимо импортировать ее в свой проект:

```
from PIL import Image
```

b) Открытие и чтение изображения

Для открытия и чтения изображения с использованием библиотеки Pillow, используйте метод **open**():

```
image = Image.open("img.jpeg")
```

Этот код открывает изображение с именем "img.jpeg" в переменной image.

c) Сохранение нового изображения с изменениями

Для сохранения нового изображения, которое было изменено, используйте метод **save**():

```
new_image.save("modified.jpeg")
```

Этот код сохраняет новое изображение с изменениями под именем "modified.jpeg".

d) Получение размеров изображения

```
width, height = image.size
```

Пиксели являются основными строительными блоками изображения. Они представляют собой маленькие точки на экране, каждая из которых содержит информацию о цвете или яркости.
Каждое изображение имеет **размер** который измеряется в пкс (пикселях).

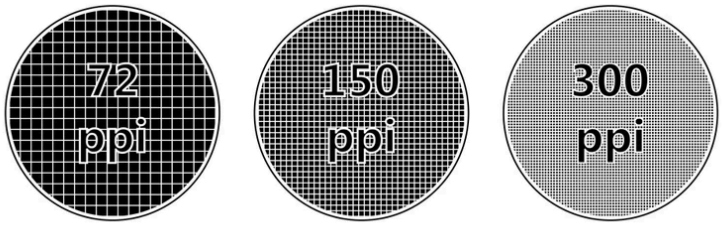
- 100x100 - изображение размера 100 на 100 пикселей;
- 50x40 - изображение размера 50 на 40 пикселей;

Можно вычислить площадь пикселей для этого достаточно просто перемножить значения по вертикали и горизонтали;

```
S = height * width
```

например, если размер 10x10 -> **S** = 10 * 10 = 100

Плотность пикселей обычно выражается в пикселях на дюйм (PPI - pixels per inch) или пикселях на сантиметр (PPC - pixels per centimeter). Для вычисления плотности пикселей вам нужно знать не только размеры изображения в пикселях, но и его размеры в единицах длины (дюймы или сантиметры).



PPI (pixels per inch): Плотность пикселей на дюйм вычисляется как отношение количества пикселей к длине изображения в дюймах. Например, если изображение имеет размер 800x600 пикселей и его фактический размер 8x6 дюймов, то плотность пикселей будет:

$\frac{800}{8} = 100ppi$ по ширине,

$\frac{600}{6} = 100ppi$ по высоте,

PPC (pixels per centimeter): Плотность пикселей на сантиметр вычисляется аналогичным образом, но используются сантиметры вместо дюймов.

Чем выше плотность пикселей (DPI, PPI/PPC), тем более детализированным может быть изображение на экране или распечатанное на бумаге.

2. Цветовой анализ изображения

2.2.1 BW-анализ

Преобразование изображения в черно-белый формат и получение новой матрицы MxN:

Черно-белый анализ изображения предполагает преобразование цветного изображения в градации серого, где каждый пиксель представлен одним значением яркости, не зависящим от цвета. *Пример кода:* `Image.convert("L")`

2.2.2 RGB/RGBA-анализ

Преобразование изображения в RGB/RGBA формат и получение новой матрицы MxN:

Таблица цветов RGB

Красный	Зеленый	Синий	Цвет
0	0	0	Черный
255	0	0	Красный
0	255	0	Зеленый
0	0	255	Синий
0	255	255	Голубой
255	255	0	Желтый
255	0	255	Пурпурный
255	255	255	Белый

Преобразование изображения в формат RGB (Red, Green, Blue) или RGBA (Red, Green, Blue, Alpha) позволяет работать с цветными изображениями, сохраняя информацию о каждом канале цвета.

Для преобразования изображения с дефектом в формат RGB/RGBA и получения новой матрицы 100x100 можно использовать ту же библиотеку Pillow. Пример кода аналогичен предыдущему, но с использованием метода **convert**() с соответствующим параметром.

2.3 Анализ изменения составляющей изображения;

провести анализ изменения составляющей изображения, означает получить изображение с типами blur и crop, а затем сравнить полученные новые изображения со старыми (исходными) версиями;

2.4 Анализ границ изображения;

Анализ границ изображения является важной задачей обработки изображений, которая направлена на выделение контуров объектов на изображении. Для этой цели часто применяются различные фильтры, которые помогают выявить различия в интенсивности пикселей и выделить контуры объектов.

- **ImageFilter.FIND_EDGES:** Этот фильтр выделяет границы объектов на изображении путем выделения перепадов интенсивности пикселей. Границы выделяются как области с большим изменением яркости между соседними пикселями.
- **ImageFilter.EDGE_ENHANCE:** Данный фильтр улучшает контраст границ объектов на изображении, делая их более четкими и выразительными.

```
image.filter(parametr);
```

2.ZERO Восставновление изображения;

Для восстановления изображения с дефектом на основе исходной (NORMAL) матрицы пикселей можно применить различные методы обработки изображений. Один из подходов может заключаться в замене дефектных областей изображения на соответствующие пиксели из нормальной матрицы.

алгоритм восстановления изображения:

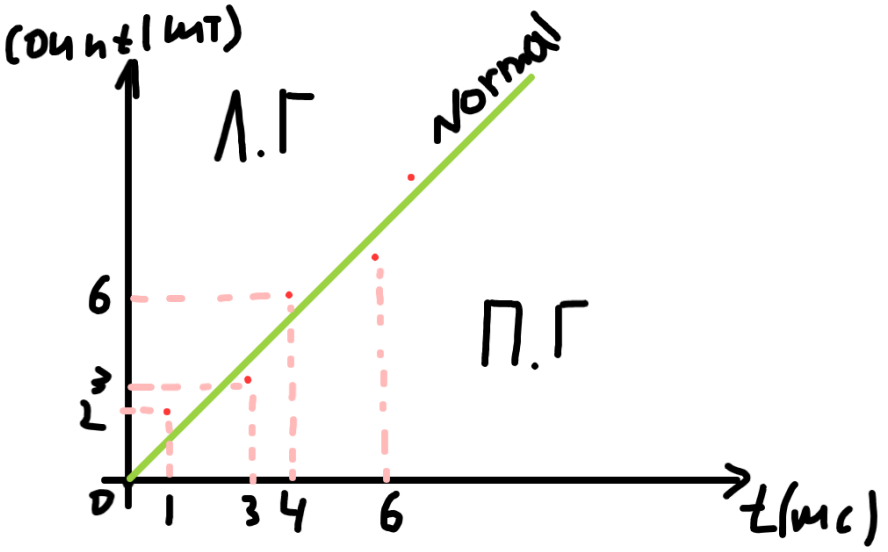
1. Загрузка исходной матрицы пикселей:
2. Замена дефектных областей:
    a. Идентифицируйте дефектные области в матрице пикселей с дефектом.
    b. Замените пиксели в дефектных областях на соответствующие пиксели из нормальной матрицы.
3. Сохранение восстановленного изображения:
    a. Сохраните восстановленную матрицу пикселей в файл изображения.

II GRAPH.PY

1) График - это визуальное представление данных, которое помогает анализировать и интерпретировать их в контексте их взаимосвязи. Он состоит из точек данных, которые соединены линиями или другими геометрическими формами.

Оси графика X, Y:

- **Ось X:** Горизонтальная ось графика, которая обычно отображает независимую переменную или параметр. На оси X откладываются значения переменной, изменяющейся в процессе эксперимента или исследования.
- **Ось Y:** Вертикальная ось графика, которая обычно отображает зависимую переменную или результаты эксперимента. На оси Y откладываются значения переменной, зависящей от переменной на оси X.



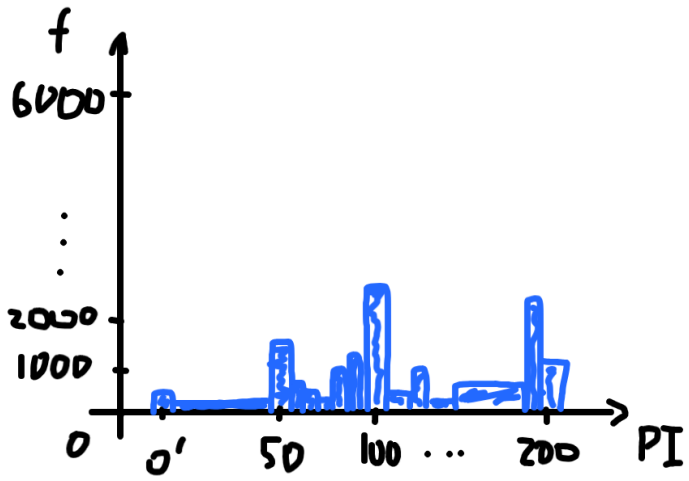
Л.Г - левая граница от нормали (count_left);

П.Г - правая граница от нормали (count_right);

2) Гистограмма частот изображения - это графическое представление распределения яркости или цветов на изображении. Она показывает, сколько пикселей имеют определенную яркость или значение цвета. Гистограмма частот является важным инструментом в анализе изображений и может помочь в понимании его характеристик, таких как контрастность, экспозиция, цветовой баланс и динамический диапазон.

```
# Преобразование в массив numpy
image_array = np.array(image_gray)

# Вычисление частот пикселей
histogram = np.histogram(image_array.flatten(), bins=256, range=(0, 255))
```



f - частоты пикселей;
PI - интенсивность пикселей;

OY: f, OX: PI;

Ход работы.

Основные требования:

- каждая функция должна иметь docstring - множ-й комментарий (по типу что делает данная функция)
- написание функций должно быть компактным (не в 100 строчек кода)
- именования функций должны быть нормальными и четко отражать смысл самой функции
- код должен соответствовать стандарту языка Python (PEP).
- классы и объекты должны соответствовать стандартам языка программирования Python.

1. Импорт необходимых библиотек и модулей.

2. Файл image.py

2.1 Получение матрицы со значениями цветов пикселей;

У вас есть изображение с **дефектом** (https://disk.yandex.ru/i/OEPqQ7L_7wDtrA) размера **100x100пкс**. Необходимо получить матрицу **100x100** со значениями цветов каждого из пикселей.

```
[
    [(1, 0, 1), (33, 33, 0), (12, 12, 12)...],
    [(2, 2, 2), (1, 0, 1), (33, 45, 1), ...],
    ...
]
```

Вычислить площадь изображения с дефектом и плотность пикселей;

2.2 Цветовой анализ изображения;

2.2.1 BW-анализ;

a) преобразовать изображение с дефектом в черно-белый формат и получить новую матрицу 100x100;

b) построить график (зависимость времени от количества преобразованных в черно-белый формат пикселей);

c) построить гистограмму частот;

2.2.2 RGB/RGBA-анализ;

a) преобразовать изображение с дефектом в RGB/RGBA формат и получить новую матрицу 100x100;

b) построить график (зависимость времени от количества преобразованных в RGB/RGBA формат пикселей);

c) построить гистограмму частот;

d) ответить на главный вопрос в чем разница между BW, RGB и RGBA?;

**сохранить картинки для отчетности;*

2.3 Анализ изменения составляющей изображения;

2.3.1 BLUR-анализ;

а) применить разные типы blur - фильтра к изображению с дефектом и сделать вывод, что изменилось? к чему привело?;

a-1) BoxBlur()

a-2) GaussianBlur

2.3.2 CROP-анализ;

а) применить crop (обрезку по пикселям с дефектами) к изображению с дефектом и сделать вывод, что изменилось? к чему привело?;

**сохранить картинки для отчетности;*

2.4 Анализ границ изображения;

провести два анализа границ изображения (ImageFilter.**FIND_EDGES**/ImageFilter.**EDGE_ENHANCE**)

**сохранить картинки для отчетности;*

(самый важный пункт) 2.ZERO Восстановление изображения с дефектом;

Вам необходимо восстановить изображение с дефектом на базе **NORMAL** матрицы;

(первоначально получите NORMAL матрицу из изображения без дефекта) https://disk.yandex.ru/i/Mk_xNknOmzDkVg

3. Файл graph.py

3.1 построение графиков всех анализов переносим в этот файл для удобства**;**

3.2 строим график при восстановлении изображения (зависимости количества обработанных пикселей с дефектом в ед.времени t)**;**

- достраиваем к графику **NORMAL** линию и подсчитываем программно количество левых и правых точек графика **count_left**, **count_right** соответственно;
- если **count_left** равно **count_right** -> изображение восстановлено с значением 1 близким к высокой нормальности.
  в противных случаях нужно искать какое то число 0.55 <= value < 1 средней нормальности и 0 <= value < 0.55 - низкой нормальности;

**сохранить графики для отчетности;*

3. Файл main.py

- основной файл программы;
- тестовый запуск всех написанных функций согласно пунктам лабораторной работы;