МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Информатика»

ТЕМА: Введение в архитектуру компьютера

Студент гр. 3342 _____ Легалов В. В.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2023

Цель работы

Изучить основы библиотеки для работы с изображениями: Pillow. С использованием Pillow написать функции, обрабатывающие изображения.

Задание

Вариант 4

Предстоит решить 3 подзадачи, используя библиотеку **Pillow (PIL)**. Для реализации требуемых функций студент должен использовать numpy и PIL. Аргумент image в функциях подразумевает объект типа <*class 'PIL.Image.Image'*>

1) Рисование отрезка. Отрезок определяется:

координатами начала;

координатами конца;

цветом;

толщиной.

Необходимо реализовать функцию *user_func()*, рисующую на картинке отрезок.

Функция *user_func()* принимает на вход:

изображение;

координаты начала (x0, y0);

координаты конца (x1, y1);

цвет;

толщину.

Функция должна вернуть обработанное изображение.

2) Преобразовать в Ч/Б изображение (любым простым способом).

Функционал определяется:

координатами левого верхнего угла области;

координатами правого нижнего угла области;

алгоритмом, если реализовано несколько алгоритмов преобразования (по желанию студента).

Нужно реализовать 2 функции:

check_coords(image, x0, y0, x1, y1) — проверяет координаты области (x0, y0, x1, y1) на корректность (они должны быть неотрицательными, не

превышать размеров изображения, поскольку x0, y0 — координаты левого верхнего угла, x1, y1 — координаты правого нижнего угла, то x1 должен быть больше x0, а y1 должен быть больше y0);

set_black_white(image, x0, y0, x1, y1) — преобраовывает заданную область изображения в чёрно-белый. В этой функции должна вызываться функция проверки, и, если область некорректна, то должно быть возвращено исходное изображение без изменений.

3) Найти самый большой прямоугольник заданного цвета и перекрасить его в другой цвет. Функционал определяется:

Цветом, прямоугольник которого надо найти;

Цветом, в который надо его перекрасить.

Написать функцию *find_rect_and_recolor(image, old_color, new_color)*, принимающую на вход изображение и кортежи rgb-компонент и староео и нового цветов. Она выполняет задачу и возвращает изображение. При необходимости можно писать дополнительные функции.

Выполнение работы

Для решения поставленных задач были использованы модули numpy и PIL и составлены следующие функции:

user_func(image, x0, y0, x1, y1, fill, width): принимает изображение и параметры отрезка. При помощи объекта PIL.ImageDraw.Draw функция проводит отрезок на изображении и возвращает изменённую картинку.

check_coords(image, x0, y0, x1, y1): принимает изображение и координаты левого верхнего и правого нижнего углов прямоугольника. Функция проверяет координаты области на коректность, если область можно выделить на изображении, возвращает True, иначе возвращает False.

set_black_white(image, x0, y0, x1, y1): принимает изображение и координаты левого верхнего и правого нижнего углов прямоугольника. При помощи check_coords(image, x0, y0, x1, y1) проверяется корректность полученных значений, а затем при помощи функций PIL.Image.crop() и PIL.Image.convert() выделяется указанная область картинки, и конвертируется в чёрно-белое изображение. С помощью функции PIL.Image.paste() чёрно-белый фрагмент вставляется в исходное изображение. Возвращается изменённая картинка. В случае, если были введены некорректные координаты области, изображение возвращается без изменений.

find_rect(image, color): принимает изображение и цвет. Функция ищет на изображении наибольший прямоугольник заданного цвета. Для нахождения прямоугольника используется трёхмерный массив, с количеством строк и столбцов, соответствующим ширине и высоте изображения и глубиной равной 2, изначально заполненный нулями. Функция проходит по каждому пикселю изображения, при нахожении элемента искомого цвета, в массиве по тем же координам добавляются увеличенные на 1 значения из ячеек строчкой и столбцом ранее. Так, ячейка масива с саммыми большими значениями хранит информацию о искомом прямоугольнике. Возвращаемое

значение координаты левого верхнего и правого нижнего углов прямоугольника.

find_rect_and_recolor(image, old_color, new_color): принимает изображение, старый цвет прямоугольника и новый цвет прямоугольника. Вставляет в изображение прямоугольник указанного цвета, на место старого, найденного при помощи функции find_rect(image, color). Возвращает изменённое изображение.

Выводы

Были изучены и использованы методы модуля Pillow для работы с изображенияи. Написанны функции обрабатывающие изображения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import numpy as np
from PIL import Image, ImageDraw


def user_func(image, x0, y0, x1, y1, fill, width):
    Drawing = ImageDraw.Draw(image)
    Drawing.line([(x0, y0), (x1, y1)], fill, width)
    return image

def check_coords(image, x0, y0, x1, y1):
    x, y = image.size
    return (0 <= x0 < x1 and 0 <= y0 < y1 and x1 <= x and y1 <= y)

def set_black_white(image, x0, y0, x1, y1):
    if check_coords(image, x0, y0, x1, y1):
        crop = image.crop(((x0, y0, x1, y1))).convert("1")
        image.paste(crop, (x0, y0))
    return image

def find_rect(image, color):
    im = image.load()
    sizex, sizey = image.size
    arr = np.zeros((sizex, sizey, 2), dtype=int)
    s = 0
    rect = (0, 0, sizex, sizey)
    for i in range(sizex):
        for j in range(sizey):
            if im[i, j] == color:
                arr[i][j] = [1, 1]
                if i > 0:
                    arr[i][j][0] += arr[i-1][j][0]
                if j > 0:
                    arr[i][j][1] += arr[i][j-1][1]
                if s < arr[i][j][0] * arr[i][j][1]:
                    s = arr[i][j][0] * arr[i][j][1]
                      rect = (i - arr[i][j][0] + 1, j - arr[i][j][1] +
1, i + 1, j + 1)
    return rect


def find_rect_and_recolor(image, old_color, new_color):
    rect = find_rect(image, old_color)
    size = (rect[2] - rect[0], rect[3] - rect[1])
    colored_rect = Image.new("RGB", size, new_color)
    image.paste(colored_rect, (rect[0], rect[1]))
    return image
```