**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. 3341 |  | Ягудин Д.Р. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Решить 3 подзадачи, используя библиотеку Pillow (PIL) и numpy. Необходимо разработать функции, которые работают с объектами типа <class 'PIL.Image.Image'>.

## Задание

Вариант 1

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 треугольника

Необходимо написать функцию triangle(), которая рисует на изображении треугольник

Функция triangle() принимает на вход:

* Изображение (img)
* Координаты вершин (x0,y0,x1,y1,x2,y2)
* Толщину линий (thickness)
* Цвет линий (color) - представляет собой список (list) из 3-х целых чисел
* Цвет, которым залит (fill\_color - если значение None, значит треугольник не залит) - представляет собой список (list) из 3-х целых чисел

Функция должна вернуть исходное обработанное изображение.

2) Замена наиболее часто встречаемого цвета.

Необходимо написать функцию change\_color(), которая заменяет наиболее часто встречаемый цвет на переданный.

Функция change\_color() принимает на вход:

* Изображение (img)
* Цвет (color - представляет собой список из трех целых чисел)

Функция должна найти в изображении самый частый цвет и заменить его на переданный, затем вернуть новое изображение (исходное изображение не должно меняться).

3)Коллаж

Необходимо написать функцию collage().

Функция collage() принимает на вход:

* Изображение (img)
* Количество изображений по "оси" Y (N - натуральное)
* Количество изображений по "оси" X (M - натуральное)

Функция должна создать коллаж изображений (это же изображение, повторяющееся NxM раз. (N раз по высоте, M раз по ширине) и вернуть его (новое изображение).

## Основные теоретические положения

*PIL* — это библиотека, предназначенная для работы с изображениями. Она предоставляет функции для открытия, создания, изменения, обработки и сохранения изображений.

Модуль *Image* — это класс, предоставляющий различные методы для работы с изображениями: изменение размера, поворот, фильтрация и многое другое.

Модуль *ImageDraw* — это класс, который предоставляет методы для рисования на изображениях. Он использован для рисования фигур и линий на изображении.

Для импортирования модулей PIL используем “from PIL import Image, ImageDraw”

Библиотека *numpy* — это библиотека для выполнения математических операций, включая многомерные массивы и функции для работы с ними.

Библиотека импортирована numpy используем "*import numpy as np*".

## Выполнение работы

- Импортируем библиотеки PIL (Pillow), Image, ImageDraw и библиотеку numpy.

- Объявляем функцию triangl с входными параметрами img, x0, y0, x1, y1, x2, y2, thickness, color, fill\_color.

- Создаем объект ImageDraw для рисования на изображении img.

- Создаем переменную points, в которую записываем координаты вершин треугольника

- Изменяем переменную color для представления цвета в формате кортежа.

- Определяем нужно ли заливать треугольник указанным цветом. Если треугольник нужно залить, то представляем цвет заливки как переменную color, после чего по заданным параметрам рисуем треугольник при помощи метода polygon. В ином случае сразу же рисуем треугольник при помощи того же метода

- Возвращаем измененное изображение

- Объявляем функцию change\_color с входными параметрами img, color.

- Преобразуем изображение в массив numpy для работы с пикселями.

- При помощи метода shape получаем высоту, ширину и количество каналов цвета.

- Переформатируем массив изображения в одномерный, представляя каждый пиксель как одномерный из ‘c’ элементов при помощи reshape.

- Модуль unique находит уникальные строки в одномерном массиве и подсчитывает их количество, сохраняя их количество в unique а соответствующие частоты в k.

- argmax находит индекс самого часто встречающегося цвета в массиве unique.

- Дальше мы заменяем все пиксели, имеющие самый часто встречаемый цвет.

- Преобразуем массив numpy обратно в изображение и возвращаем его.

- Объявляем функцию collage с входными параметрами img, N, M.

- Получаем ширину и высоту изображения.

- Создаем новое изображение для размещения на немколажжа.

- Вложенными циклами проходимся от левого нижнего угла до правого верхнего и вставляем заданное на изображение.

- Возвращаем новое изображение

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | triangle (Image.new("RGB", (100, 100), “red”), 10, 10, 20, 80, 80, 40, 4, (0, 0, 250), (0, 0, 0)) |  |
|  | Change\_color(Image.open(“C:\projects\py\test1.jpg”), (100, 100, 100)) |  |
|  | Collage(Image.open(“C:\projects\py\sd.jpg”), 3, 4) |  |

## Выводы

Выполняя данную работу мы научились пользовать библиотекой PIL, а так же разработали 3 функции для обработки изображений.

# Приложение А Исходный код программы

Название файла: main\_lb.py

from PIL import Image, ImageDraw

import numpy as np

def triangle(img, x0, y0, x1, y1, x2, y2, thickness, color, fill\_color):

cp = ImageDraw.Draw(img)

points = [(x0, y0), (x1, y1), (x2, y2)]

color = (color[0], color[1], color[2])

if fill\_color != None:

fill\_color = (fill\_color[0], fill\_color[1], fill\_color[2])

cp.polygon(points, outline = color, fill = fill\_color, width = thickness)

else:

cp.polygon(points, outline = color, width = thickness)

return img

def change\_color(img, color):

img\_arr = np.array(img)

h, w, c = img\_arr.shape

imgr = img\_arr.reshape((h \* w, c))

unique, k = np.unique(imgr, axis=0, return\_counts = True)

common = np.argmax(k)

img\_arr[(img\_arr == unique[common]).all(axis =-1)] = color

result\_img = Image.fromarray(img\_arr)

return result\_img

def collage(img, N, M):

img\_size = img.size

result = Image.new("RGB", (img\_size[0] \* M, img\_size[1] \* N), "red")

for i in range(N):

for j in range(M):

x = j \* img\_size[0]

y = i \* img\_size[1]

result.paste(img, (x, y))

return result