**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. 3343 |  | Иванов П.Д. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2023

## Цель работы

Разработать 3 функции, которые обрабатывают изображение согласно заданию, и использовать в работе методы библиотеки *Pillow*, а также *numpy*.

## Задание

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).

## Выполнение работы

Была написана программа на языке Python, где каждая задача реализована в отдельной функции.

Для решения задач использовались методы библиотеки *Pillow*, а именно: *ImageDraw.polygon()* - для отрисовки на изображении полигона с заданным количеством углов, *Image.fromarray()* - для преобразования массива пикселей в изображение, *Image.new()* - для создания изображения с указанными параметрами, *Image.paste()* - для наложения изображений друг на друга.

*Numpy* использовалась для разложения на массив пикселей исходного изображения, а также получения количества строк и столбцов в массиве пикселей, для этого использовались функции *array()* и *shape()* соответственно.

Функция *triangle()* рисует на входном изображении треугольник по заданным координатам с указанными параметрами (используется *ImageDraw.polygon()*).

Функция *change\_color()* принимает на вход изображение и цвет, который нужно заменить в этом изображении. В ходе работы этой функции изображение разбивается на пиксели, количество встречаемых пикселей записывается в словарь в виде «цвет пикселя»: «кол-во этих пикселей на изображении».

Затем находится самый часто встечаемый цвет и обходом по массиву пикселей нужный цвет заменяется на указанный. После этого изображение собирается вновь(*Image.fromarray()*) и возвращается из функции.

В функции *collage()* создается коллаж из входного изображения, повторяющегося *N\*M* раз. В этой функции создается «фоновое» изображение необходимого размера, чтобы на нем возможно было разместить указанное количество входных изображений, затем с помощью цикла изображения накладываются на фоновое (*Image.paste()*). После завершения цикла возвращается созданный коллаж.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | Изображение 200\*200, 10, 10, 100, 100, 40, 90, 3, [0, 0, 255], None | Изображение, с треугольником, нарисованным синим цветом, без заливки. | Работа функции *triange().* |
|  | Изображение с черным фоном и небольшим красным квадратом, [0, 0, 255] | Изображение с синим фоном и небольшим красным квадратом (положение квадрата не изменилось) | Работа функции *change\_color().* |
|  | Изображение 100\*50, 3, 5 | Коллаж размером 500\*150 | Работа функции *collage().* |

## Выводы

Была написана программа, включающая в себя 3 функции, которые обрабатывают входящие изображения согласно заданию, а именно: рисование фигуры на изображении, замена часто встречающегося цвета на входящем изображении, создание коллажа с заданным количеством входного изображения на нем. Для решения использовались библиотеки *Pillow* и *numpy*.

# ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

import numpy as np

from PIL import Image, ImageDraw

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

drawing = ImageDraw.Draw(img)

drawing.polygon(((x0, y0), (x1, y1), (x2, y2)), fill=None if fill\_color is None else tuple(fill\_color), outline=tuple(color), width=thickness)

return img

def change\_color(img, color):

colors = dict()

pixels\_list = np.array(img)

strings, columns = np.shape(pixels\_list)[0], np.shape(pixels\_list)[1]

for i in range(strings):

for j in range(columns):

if tuple(pixels\_list[i][j]) not in colors.keys():

colors[tuple(pixels\_list[i][j])] = 0

else:

colors[tuple(pixels\_list[i][j])] += 1

most\_frec = max(colors.keys(), key=lambda x: colors[x])

for i in range(strings):

for j in range(columns):

if tuple(pixels\_list[i][j]) == most\_frec:

pixels\_list[i][j] = np.array(color)

new = Image.fromarray(pixels\_list)

return new

def collage(img, N, M):

width = img.size[0] \* M

height = img.size[1] \* N

new\_image = Image.new("RGB", (width, height), "black")

for i in range(N):

for j in range(M):

new\_image.paste(img, (img.size[0]\*j, img.size[1]\*i))

return new\_image