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

Студентка гр. 3343 _____ Ермолаева В. А.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2023

Цель работы

Изучить и научиться применять функции библиотеки Pillow (PIL) для решения задач, связанных с обработкой изображений.

Задание

Вариант 3.

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 пентаграммы в круге

Необходимо написать функцию solve(), которая рисует на изображении пентаграмму в окружности.

Функция solve() принимает на вход:

- Изображение (img)
- координаты центра окружность (x,y)
- радиус окружности
- Толщину линий и окружности (thickness)
- Цвет линий и окружности (color) - представляет собой список (list) из 3-х целых чисел

Функция должна изменить исходное изображение и вернуть его изображение.

Примечание:

Вершины пентаграммы высчитывать по формуле:

phi = (pi/5)*(2*i+3/2)

node_i = (int(x0+r*cos(phi)),int(y0+r*sin(phi)))

x0,y0 - координаты центра окружности, в который вписана пентаграмма

r - радиус окружности

i - номер вершины от 0 до 4

2) Поменять местами участки изображения и поворот

Необходимо реализовать функцию solve, которая меняет местами два квадратных, одинаковых по размеру, участка изображений и поворачивает

эти участки на 90 градусов по часовой стрелке, а затем поворачивает изображение на 90 градусов по часовой стрелке.

Функция solve() принимает на вход:

- Квадратное изображение (img)
- Координаты левого верхнего угла первого квадратного участка(x0,y0)
- Координаты левого верхнего угла второго квадратного участка(x1,y1)
- Длину стороны квадратных участков (width)

Функция должна сначала поменять местами переданные участки изображений. Затем повернуть каждый участок на 90 градусов по часовой стрелке. Затем повернуть всё изображение на 90 градусов по часовой стрелке.

Функция должна вернуть обработанное изображение, не изменяя исходное.

3) Средний цвет

Необходимо реализовать функцию solve, которая заменяет цвет каждого пикселя в области на средний цвет пикселей вокруг (не считая сам этот пиксель).

Функция solve() принимает на вход:

- Изображение (img)
- Координаты левого верхнего угла области (x0,y0)
- Координаты правого нижнего угла области (x1,y1)

Функция должна заменить цвета каждого пикселя в этой области на средний цвет пикселей вокруг.

Пиксели вокруг :

- 8 самых близких пикселей, если пиксель находится в центре изображения
- 5 самых близких пикселей, если пиксель находится у стенки
- 3 самых близких пикселя, если пиксель находится в угле

Функция должна вернуть обработанное изображение, не изменяя исходное. Средний цвет - берется целая часть от среднего каждой компоненты из rgb. (int(sum(r)/count),int(sum(g)/count),int(sum(b)/count)).

Выполнение работы

Функция swap(img, x0, y0, x1, y1, width) принимает на вход квадратное изображение (img), координаты левого верхнего угла первого квадратного участка(x0,y0), координаты левого верхнего угла второго квадратного участка(x1,y1) и длину стороны квадратных участков (width). Меняет местами два квадратных, одинаковых по размеру, участка изображений и поворачивает эти участки на 90 градусов по часовой стрелке, а затем поворачивает изображение на 90 градусов по часовой стрелке. Возвращает обработанное изображение.

Функция pentagram(img, x, y, r, thickness, color) принимает на вход изображение (img), координаты центра окружности (x,y), радиус окружности, толщину линий и окружности (thickness), цвет линий и окружности (color) - представляет собой список (list) из 3-х целых чисел. Рисует на изображении пентаграмму в окружности. Возращает обработанное изображение.

Функция avg_color(img, x0, y0, x1, y1) принимает на вход изображение (img), координаты левого верхнего угла области (x0,y0), координаты правого нижнего угла области (x1,y1). Заменяет цвет каждого пикселя в области на средний цвет пикселей вокруг и вызывает функцию get_pixel(img, x, y), которая меняет цвет пикселя. Возвращает обработанное изображение, не изменяя исходное.

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
1.	img.size=(300, 300); x = 28; y = 124; r = 5; thickness = 1; color = [5, 134, 172]	Обработанное изображение.	Тестирование функции pentagram(). Выходные данные соответствуют ожиданиям.
2.	img.size = (300, 300); x0 = 0; y = 0; x1 = 150; y1 = 150; width =100	Обработанное изображение.	Тестирование функции swap(). Выходные данные соответствуют ожиданиям.
3.	img.size = (300, 300); x0 = 9; y0 = 69; x1 = 243; y1 = 229	Обработанное изображение.	Тестирование функции avg_color(). Выходные данные соответствуют ожиданиям.

Выводы

В ходе выполнения лабораторной работы была изучена библиотека pillow (PIL), с помощью которой были созданы функции для обработки изображений.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from PIL import ImageDraw
import numpy as np

def swap(img, x0, y0, x1, y1, width):
img1 = img.copy()
square1 = img1.crop((x0, y0, x0 + width, y0 + width))
square2 = img1.crop((x1, y1, x1 + width, y1 + width))

square1 = square1.rotate(270)
square2 = square2.rotate(270)

img1.paste(square2, (x0, y0))
img1.paste(square1, (x1, y1))

img1 = img1.rotate(270)
return img1

def get_pixel(img, x, y):
     neighbours = [img.getpixel((x-1, y)), img.getpixel((x-1, y-
1)),     img.getpixel((x,     y-1)),     img.getpixel((x+1,     y-1)),
img.getpixel((x+1,  y)),  img.getpixel((x+1,  y+1)),  img.getpixel((x,
y+1)), img.getpixel((x-1, y+1))]

avg_color = (
int(sum(color[0] for color in neighbours)/8),
int(sum(color[1] for color in neighbours)/8),
int(sum(color[2] for color in neighbours)/8),
)

return avg_color

def avg_color(img, x0, y0, x1, y1):
img_copy = img.copy()

for x in range(x0, x1+1):
for y in range(y0, y1+1):
img_copy.putpixel((x,y), get_pixel(img, x, y))

return img_copy

def pentagram(img, x, y, r, thickness, color):
draw = ImageDraw.Draw(img)
color = tuple(color)

draw.ellipse([x-r, y-r, x+r, y+r], None, color, thickness)
points = []

for i in range(5):
phi = (np.pi/5)*(2*i+3/2)
points.append((int(x+r*np.cos(phi)), int(y+r*np.sin(phi))))
```

```
     points[1],   points[2],   points[3],   points[4]   =   points[2],
points[4], points[1], points[3]

     for i in range(4):
     draw.line((points[i], points[i+1]), color, thickness)
     draw.line((points[4], points[0]), color, thickness)

     return img
```