**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 |  | Перевалов П.И. |
| 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 2

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 pentagram(), которая рисует на изображении пентаграмму в круге.

Функция pentagram() принимает на вход:

Изображение (img)

координаты левого верхнего и правого нижнего угла квадрата, в который вписана окружность (x0,y0,x1,y1)

Толщину линий и окружности (thickness)

Цвет линий и окружности (color) - представляет собой список (list) из 3-х целых чисел

Функция должна вернуть обработанное 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

Подсказка: Округляйте все вычисляемые вами значения (кроме значений углов) до целых чисел.

2) Инвертирование полос

Необходимо реализовать функцию invert, которая делит изображение на "полосы" и инвертирует цвет нечетных полос (счёт с нуля).

Функция invert() принимает на вход:

Изображение (img)

Ширину полос в пикселах (N)

Признак того, вертикальные или горизонтальные полосы(vertical - если True, то вертикальные)

Функция должна разделить изображение на вертикальные или горизонтальные полосы шириной N пикселей. И инвертировать цвет в нечетных полосах (счет с нуля). Последняя полоса может быть меньшей ширины, чем N.

3) Поменять местами 9 частей изображения

Необходимо реализовать функцию mix, которая делит квадратное изображение на 9 равных частей (сторона изображения делится на 3), и по правилам, записанным в словаре, меняет их местами.

Функция mix() принимает на вход:

Изображение (img)

Словарь с описанием того, какие части на какие менять (rules)

Пример словаря rules:

{0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}

Элементы нумеруются слева-направо, сверху-вниз.

В данном случае нулевой элемент заменяется на первый, первый на второй, второй на четвертый, третий на четвертый и так далее.

Функция должна вернуть обработанное изображение.

Пример входной картинки и словаря:

Картинка



{0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8}

Результат:



Можно реализовывать дополнительные функции.

## Основные теоретические положения

Библиотека *PIL (Python Imaging Library)* - это библиотека для работы с изображениями. Она предоставляет функции для открытия, изменения, сохранения и обработки изображений, а также для создания новых изображений. Для доступа к функциям библиотеки мы импортировали ее используя "*import PIL*".

Модуль *Image* из библиотеки *PIL* - это класс, предоставляющий различные методы для работы с изображениями, такие как открытие, сохранение, изменение размера, поворот, фильтрация и многое другое. Модуль импортирован из библиотеки *PIL* с помощью "*from PIL import Image*".

Модуль *ImageDraw* из библиотеки *PIL* - это класс, который предоставляет методы для рисования на изображениях. Он использован для рисования фигур и линий на изображении. Модуль импортирован из библиотеки *PIL* с помощью "*from PIL import ImageDraw*".

Модуль *ImageOps* из библиотеки *PIL* - это класс, предоставляющий различные методы для обработки изображений, такие как изменение контраста, наложение эффектов и другие операции. Использован в функции *invert* для инвертирования цветов изображения. Модуль импортирован из библиотеки *PIL* с помощью "*from PIL import ImageOps*".

Библиотека *numpy* - это библиотека для выполнения математических операций, включая многомерные массивы и функции для работы с ними. Библиотека импортирована с помощью "*import numpy as np*".

## Выполнение работы

- Импортируем библиотеки PIL (Pillow), Image, ImageDraw, ImageOps и cos, sin, pi из модуля match.

- Объявляем функцию pentagram с входными параметрами img, x0, y0, x1, y1, thickness, color.

- Создаем объект ImageDraw для рисования на изображении img.

- Вычисляем радиус и центр окружности.

- Создаем пустой список posts для хранения координат вершин пентаграммы.

- В цикле проходим по числам от 0 до 4 и вычисляем координаты вершин пентаграммы с помощью формулы.

- Сохраняем вершины пентаграммы в списке posts.

- Устанавливаем последней вершине координаты первой вершины, чтобы пентаграмма была замкнутой.

- Рисуем окружность на изображении с заданными координатами, цветом и толщиной контура.

- Рисуем линию, соединяющую вершины пентаграммы, с заданным цветом и толщиной.

- Возвращаем исходное изображение с нарисованной пентаграммой.

- Объявляем функцию invert с входными параметрами img, N, vertical.

- Получаем ширину и высоту изображения.

- Если vertical равно True, выполняем следующие операции для каждой нечетной части ширины.

- Выбираем часть изображения с помощью crop.

- Инвертируем цвета выбранной части с помощью функции invert из библиотеки ImageOps.

- Вставляем инвертированную часть обратно в изображение с помощью paste.

- Если vertical равно False, выполняем аналогичные операции для каждой нечетной части высоты.

- Возвращаем исходное изображение с инвертированными частями.

- Объявляем функцию mix с входными параметрами img и rules.

- Получаем ширину и высоту изображения.

- Создаем пустой список bob для хранения частей изображения.

- Вложенными циклами проходим по 9 частям изображения (3 по горизонтали и 3 по вертикали).

- Выбираем часть изображения с помощью crop.

- Добавляем выбранный фрагмент и его координаты в список bob.

- Проходим по всем элементам словаря rules.

- Извлекаем значение элемента словаря, используя значение как индекс в списке bob.

- Извлекаем фрагмент изображения и его координаты, используя значение ключа как индекс в списке bob

- Вставляем фрагмент изображения обратно в изображение, используя его координаты.

- Возвращаем исходное изображение с перемешанными частями.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | pentagram(Image.new("RGB", (300, 300), 0), 20, 40, 150, 170, 3, [128, 128, 0]) |  | - |
|  | invert(Image.new("RGB", (300, 300), 0), 15, True) |  | - |
|  | mix(Image.open(‘krab1’),{0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8}) |  | - |

## Выводы

Созданы функции, которые могут работать с объектами типа <class 'PIL.Image.Image'>, а также решены 3 задачи, используя библиотеки Pillow (PIL) и match.

# **ПРИЛОЖЕНИЕ А** **ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: main.py

import PIL

from PIL import Image, ImageDraw, ImageOps

from math import cos, sin, pi

def pentagram(img, x0, y0, x1, y1, thickness, color):

drawing = ImageDraw.Draw(img)

r = abs(x1 - x0) // 2

x\_c = x1 - (abs(x1 - x0) // 2)

y\_c = y1 - (abs(y1 - y0) // 2)

posts = []

for i in range(5):

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

post\_i = (int(x\_c + r \* cos(phi)), int(y\_c + r \* sin(phi)))

posts.append(post\_i)

drawing.ellipse((x0, y0, x1, y1), outline=tuple(color), width=thickness)

for i in range(5):

drawing.line([posts[i], posts[(i + 2) % 5]], fill=tuple(color), width=thickness)

return img

def invert(img, N, vertical):

width, height = img.size

w = width//N + 1

h = height//N + 1

if (vertical):

for i in range(1, w, 2):

inv\_color = img.crop((i\*N, 0, (i+1)\*N, height))

inv\_color = ImageOps.invert(inv\_color)

img.paste(inv\_color, (i\*N, 0))

else:

for f in range(1, h, 2):

inv\_color = img.crop((0, f\*N, width, (f+1)\*N))

inv\_color = ImageOps.invert(inv\_color)

img.paste(inv\_color, (0, f\*N))

return img

def mix(img, rules):

width, height = img.size

w = width // 3

h = height // 3

bob = []

for f in range(3):

for i in range(3):

a = ((i \* h, f \* w, (i + 1) \* h, (f + 1) \* w))

piece = img.crop(a)

bob.append([piece, (i \* h, f \* w)])

for i in rules:

img.paste(bob[rules[i]][0], bob[i][1])

return img