**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. 3341 |  | Чинаева М.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 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.

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

Некоторые основные функции и методы:

1. Вывод изображения на экран Image.show()

2. Создание изображения Image.new(mode, size, color), где mode -- режим работы с изображением, size -- это кортеж из двух элементов (width, height) , первый width -- ширина изображения (в пикселях), второй height -- высота изображения (в пикселях), color -- цвет изображения.

3. Отрисовка одного изображения на другое Image.paste(other\_image, coordinates), где other\_image -- другое изображение класса Image, coordinates -- кортеж из двух координат x, y -- координаты на первом изображении, куда надо дорисовать второе

4. Обрезка изображения Image.crop(box), где box – кортеж из 4-х значений: (x1, y1, x2, y2)

5. Объект для рисования ImageDraw.Draw(source\_image), где source\_image -- Image, на котором нужно нарисовать.

## Выполнение работы

Импортируем библиотеки PIL (Pillow), Image, ImageDraw, ImageOps и *numpy*: *import numpy as np*

В задании требуется оформить каждую из 3 задач в виде отдельной функции согласно условиям.

1. Решение задачи 1:

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

Функция получает на вход изображение, координаты левого верхнего и правого нижнего угла квадрата, в который вписана окружность, толщину линий и окружности, цвет линий и окружности.

Сначала определяются координаты центра окружности, в которую вписана пентаграмма, в процессе вычислений значение приводится к типу *int,* т. к. координата – это пиксель, а значит может иметь только целочисленное значение. Далее используется метод *drawing = ImageDraw.Draw(img)*, чтобы сделать возможным рисование на картинке. С помощью метода *drawing.ellipse((x0, y0, x1, y1), fill = None, outline = tuple(color), width=thickness)* рисуется круг. Так как цвета подавались списком, его преобразовываем к кортежу. Далее создается пустой кортеж, в который далее будут добавляться точки, также создается список, в котором лежат значения переменной *i* в порядке, нужном для рисования звезды. Далее с помощью цикла for проходимся поэтому списку, чтобы для каждой точки вычислить координаты вершин, к кортежу прибавляются новые координаты. Точка 0 повторяется 2 раза, так как нужно, чтобы линия замкнулась. Далее с помощью метода *drawing.line(points, fill = tuple(color), width = thickness)* рисуем в кругу ломанную линию по вычисленным координатам.

Функция возвращает изображение с нарисованной на нем пентаграммой.

1. Решение задачи 2:

*def invert(img, N, vertical)*

На вход функции подается изображение, ширина полос в пикселях, признак того, вертикальные полосы или горизонтальные.

Сначала с помощью метода inv\_img = ImageOps.invert(img) инвертируем изображение полностью и сохраняем отдельно. Далее узнаем длину стороны квадрата (изображения). Если полосы вертикальные с помощью цикла for i in range(N, width, 2\*N) (начинаем именно с N, чтобы нулевая четная полоса осталась не тронутой) вырезаем из инвертированного изображения полосу по размеру аналогичную той, которую мы должны были инвертировать и отрисовываем ее на первоначальное изображение. Таким образом, мы инвертируем все нечетные вертикальные полосы. Если полосы горизонтальны, алгоритм аналогичен, но абсцисса и ордината меняются местами.

Функция возвращает изображение с инвертированными нечетными полосами.

1. Решение задачи 3:

*def mix(img, rules)*

На вход функции подается изображение и словарь с описанием того, какие части на какие менять. Сначала определяется высота и ширина изображения (в функции везде используется только высота, так как изображение квадратное). Далее вычисляем длину сторон квадратов, на которые делим изображение. Округляем это значение с помощью функции int(), так как в данном случае координаты это только целочисленные значения. Создается пустой массив, в который далее будут в заданном порядке добавлены части изображения. Далее с помощью вложенных циклов for проходимся по координатам левых верхних углов квадратов. Важно, что первый цикл – цикл с координатой y, так как нумерация квадратов идет слева на право, а только потом уже сверху вниз. С помощью метода img.crop((x, y, x + third\_part\_height, y + third\_part\_height)) вырезается нужный квадрат и добавляется в массив изображений.

Определяется счетчик number и ему присваивается значение 0, так как нумерация квадратов начинается с 0. С помощью аналогичных предыдущим вложенных циклов для каждого квадрата по его номеру узнаем из словаря, какой квадрат первоначального изображения должен быть. Далее с помощью метода img.paste(arr[need\_part], (x, y)) отрисовывается нужный квадрат в данные координаты.

Функция возвращает измененное изображение.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | pentagram(Image.new("RGB", (300, 300), ‘black’), 34, 23, 93, 82, 1, [128, 128, 0]) |  |
|  | invert(Image.new("RGB", (300, 300), ‘black’), 70, False) |  |
|  | mix(Image.open(‘krab1’),{0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8}) |  |

## Выводы

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

Таким образом, успешно достигнуты поставленные цели по разработке функций для работы с объектами типа <class 'PIL.Image.Image'> и решению 3 подзадач. Реализация данных функций позволяет удобно и эффективно обрабатывать изображения и выполнять необходимые операции с ними.

# Приложение А Исходный код программы

Название файла: main.py

import PIL

import numpy as np

from PIL import Image, ImageDraw, ImageOps

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

center\_x = x0 + int((x1 - x0) / 2)

center\_y = y0 + int((y1 - y0) / 2)

r = int((x1 - x0) / 2)

drawing = ImageDraw.Draw(img)

drawing.ellipse((x0, y0, x1, y1), fill = None, outline = tuple(color), width=thickness)

points=()

pentagram\_coordinate=[0, 2, 4, 1, 3, 0]

for i in pentagram\_coordinate:

angle = (np.pi/5)\*(2\*i+3/2)

points\_i = (int(center\_x + r \* np.cos(angle)), int(center\_y + r \* np.sin(angle)))

points = points + (points\_i,)

drawing.line(points, fill = tuple(color), width = thickness)

return img

def invert(img, N, vertical):

inv\_img = ImageOps.invert(img)

width, height = img.size

if vertical:

for i in range(N, width, 2\*N):

strip = inv\_img.crop((i, 0, i+N, height))

img.paste(strip, (i, 0))

else:

for i in range (N, height, 2\*N):

strip = inv\_img.crop((0, i, width, i+N))

img.paste(strip, (0,i))

return img

def mix(img, rules):

width, height = img.size

third\_part\_height = int(height / 3)

arr = []

for y in range (0, height, third\_part\_height):

for x in range (0, height, third\_part\_height):

part\_img = img.crop((x, y, x + third\_part\_height, y + third\_part\_height))

arr.append(part\_img)

number = 0

for y in range (0, height, third\_part\_height):

for x in range (0, height, third\_part\_height):

need\_part = rules[number]

img.paste(arr[need\_part], (x, y))

number+=1

return img