МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

ЛАБОРАТОРНАЯ РАБОТА

Алгоритм RLE

студента 4 курса 431 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Серебрякова Алексея Владимировича

Научный руководитель

доцент, к. п. н. _____ А. С. Гераськин

подпись, дата

Саратов 2022

Сжатие изображений

Цифровое изображение при хранении занимает большие объемы памяти. Так растровое изображение размером 1024 на 1024 пикселов с глубиной цвета 24 бит занимает 3 Мб. Понятно, что хранение и передача изображений в таком виде является весьма трудоёмкой задачей. Поэтому задача представления изображений в компактной форме (сжатие данных) является весьма актуальной. При этом должны быть разработаны алгоритмы как для кодирования, так и для декодирования (восстановления) изображений.

Алгоритмы сжатия изображений делятся на два больших класса: без потерь и с потерями. В первом случае в ходе компрессии информация об изображении сохраняется в полном объеме, а во втором – частично утрачивается. Первая группа методов сжатия обеспечивает восстановление исходного изображения без потерь и искажений. Для хранения изображений, предназначенных для дальнейшей обработки, следует применять форматы, использующие именно такие методы сжатия. Однако, если изображение предназначено для визуального восприятия, это не всегда необходимо. В ряде случаев исходный сигнал уже содержит такие искажения и шумы, что небольшие потери информации при кодировании (в пользу высокой степени сжатия) не испортят качества изображения в целом.

Одна из серьезных проблем компьютерной графики заключается в том, что до сих пор не найден адекватный и однозначный критерий оценки потерь качества изображения. Для изображений, наблюдаемых визуально, основным является неотличимость глазом исходного и компрессированного изображения.

Рассмотрим некоторые из используемых методов сжатия изображений.

Груповое сжатие

Одним из простейших методов сжатия изображений является *алгоритм RLE* (Run Length Encoding – кодирование с переменной длиной строки). Основной идеей этого метода является поиск одинаковых пикселов в одной строке. Найденные цепочки одинаковых элементов заменяются на пары (счетчик повторений, значение), что в определенных случаях существенно уменьшает избыточность данных.

Алгоритм в первую очередь рассчитан на изображения с большими областями повторяющегося цвета (деловая графика, схемы, рисунки и т.п.). Недостатком такого подхода является то, что в определенных ситуациях он может вместо уменьшения приводить к увеличению размера файла (например, в некоторых случаях при сохранении цветных фотографий).

Существует много схем группового сжатия, одну из которых можно проиллюстрировать следующим образом:

Входной поток данных:
17 8 54 0 0 0 97 5 16 0 45 23 0 0 0 0 0 3 67 0 0 8

Поток данных после кодирования:
17 8 54 0 3 97 5 16 0 1 45 23 0 5 3 67 0 2 8

Чаще всего для кодирования используется схема, которая называется PackBits. По аналогии с хранением отрицательных чисел, каждые 7 бит исходных данных заменяются в результате на 8 бит. Дополнительный девятый бит интерпретируется как флаг сжатия. Например:

Входные данные: 1,2,3,4,2,2,2,2,4

Данные после кодирования: 1,2,3,4,2,&3,4.

Принцип: Последовательности повторяющихся значений цвета заменяются его значением и количеством повторений.

Форматы: BMP, TIFF, GIF

Коэффициент сжатия: 2

Код программы

```
from PIL import Image,ImageDraw
import numpy as np

def image_to_grey(original_im, height, width):
    array_gray = np.zeros([height, width])
    for row in range (height):
        for col in range (width):
            r = original_im[col, row][0]
            g = original_im[col, row][1]
            b = original_im[col, row][2]
            gray = (r + g + b) // 3 #перевод в оттенки серого
            array_gray[row, col] = gray
    return array_gray

def encode(array,width,height):
    count = 1
```

```
    prev = ''
    lst = []
    print("width = ", width, "height =", height)
    for row in range(height):
        for col in range(width):
            corepixel = array[row, col]
            if corepixel != prev:
                if prev:
                    entry = (prev, count)
                    lst.append(entry)
            # print lst
                count = 1
                prev = corepixel
            else:
                count += 1
        else:
            try:
                entry = (corepixel, count)
                lst.append(entry)
                return (lst, 0)
            except Exception as e:
                print("Exception encountered {e}".format(e=e))
                return (e, 1)

def decode(lst,height, width):
    row=0
    col=0
    for pair in lst:
        bright=pair.count(1)
        count=pair.count(2)
        for x in range(0,count):
            if (row+1<width):
                row+=1
            else:
                row=0
                col=col+1
                all[row,col]=bright
        return all



#рисуем
def draw_image_to_grey(array_gray, draw, width, height):
    for row in range(0, height):
        for col in range(0, width):
            grey = array_gray[row,col]
            draw.point((col, row), (int(grey), int(grey), int(grey)))

def main():
    image = Image.open("jiraf"".jpg")  # Открываем изображение.
    draw = ImageDraw.Draw(image)  # Создаем инструмент для рисования.
    height = image.size[1]  # Определяем ширину.
    width = image.size[0]  # Определяем высоту.
    pix = image.load()  # Выгружаем значения пикселей.

    array_gray1 = image_to_grey(pix, height, width)
    array_gray2=encode(array_gray1, height, width)
    array_gray3=decode(array_gray2,height, width)
    draw_image_to_grey(array_gray3, draw, width, height)

    image.save("compression.jpg", "JPEG")
    del draw
main()
```