

Отчёт по задаче №13

«Невидимая подпись изображения»

1. Задача

Программа должна загрузить графические изображения из файлов `InputFile1` и `InputFile2`, заменить в первом изображении все пиксели, не равные соответствующим пикセルам второго изображения, на пиксели первого изображения с ближайшим значением и вывести полученный результат в файл `OutputFile` (с помощью данной процедуры можно создавать невидимую подпись изображения). Размер полученного изображения равен размеру первого изображения.

2. Создание изображений с подписью

Создадим два изображения в формате BMP:



ball.bmp



face.bmp

Рис. 1: Исходные изображения.

Кроме этого, с помощью программы GIMP подпишем созданные изображения [рис. 1](#):



ball_sign.bmp



face_sign.bmp

Рис. 2: Подписанные изображения.

3. Программа, скрывающая подпись

Напишем программу `make_invisible_sign.py`, создающую невидимую подпись. Подключим необходимые библиотеки и загрузим изображения из файлов с учётом ошибок. Напечатаем ширину и высоту изображения:

```
1  from imageio import imread, imwrite
2  import sys
3  import numpy as np
4
5  global im_sign, im_unsign, h, w #global for using in all functions
6  if len(sys.argv) == 4: #if not enough args in command line
7      im_unsign = imread(sys.argv[1]) #unsigned image
8      im_sign = imread(sys.argv[2]) #signed image
9
10 else:
11     print("err")
12     exit()
13 h = len(im_sign)
14 w = len(im_sign[0])
15 print('width = ', w, 'height = ', h)
```

Сделаем функцию, возвращающую ближайший пиксель к заданной точке, но не совпадающий с ней. Для этого в окрестности заданной точки будем искать пиксель, расстояние (в смысле линейного пространства RGB) от которого до заданной точки будет минимальным. Если такой пиксель нашёлся, то его и будем считать ближай-

шим, а если такого нет (например, всё изображение одного цвета), то в качестве ближайшего пикселя выберем исходный с изменённой компонентой Blue на единицу. Минимальное расстояние задается из условия, что ближайшие цвета должны всё-таки отличаться не сильно (можно регулировать):

```
1 def nearest(pix, pix_i, pix_j):
2     nearest_pix = [0,0,0]
3     min_dist = 100
4     for i in range(5):
5         for j in range(5):
6             next_pix = im_unsign[pix_j + j][pix_i + i]
7             if np.linalg.norm(next_pix - pix) < min_dist \
8                 and list(next_pix) != list(pix):
9                 #list() is used to compare three RGB components
10                min_dist = np.linalg.norm(next_pix - pix)
11                nearest_pix = next_pix
12
13    if list(nearest_pix) == [0,0,0]:
14        nearest_pix = pix
15        if nearest_pix[2] > 0:
16            nearest_pix[2] -= 1
17        else:
18            nearest_pix[2] += 1
19
20 return nearest_pix
```

Пройдёмся по всём изображению. Если найдётся пикセル, который не совпадает в исходном и подписанном изображениях, то такой пиксел меняем на ближайший:

```
1 for i in range(w):
2     for j in range(h):
3         if list(im_sign[j][i]) != list(im_unsign[j][i]):
4             im_sign[j][i] = nearest(im_unsign[j][i], i, j)
5
6 im_sign = imwrite(sys.argv[3], im_sign)
7 print("Unvisible sign is done!")
```

Вызываем программу из командной строки с параметрами InputFile1 InputFile2 OutputFile:

```
1 python make_invisible_sign.py face.bmp face_sign.bmp out_face.bmp
2 python make_invisible_sign.py ball.bmp ball_sign.bmp out_ball.bmp
```

Получим изображения с невидимой подписью: [рис. 3](#).

4. Программа, проявляющая невидимую подпись

Как можно видеть на [рис. 3](#), невидимую подпись различить глазом практически невозможно. Поэтому создадим [программу](#) `is_signed.py`, которая по данному и исходному изображениям выясняет, было ли подписано данное изображение, в том числе, невидимой подписью, и проявляет эту подпись чёрным цветом.



out_ball.bmp



out_face.bmp

Рис. 3: Изображения с невидимой подписью.

Программа `is_signed.py`:

```
1 from imageio import imread, imwrite
2 import sys
3
4 if len(sys.argv) == 4:
5     im0 = imread(sys.argv[1])
6     im = imread(sys.argv[2])
7 else:
8     print("err")
9     exit()
10 h = len(im0)
11 w = len(im0[0])
12 print('width = ', w, 'height = ', h)
13 fl = False
14 for i in range(w):
15     for j in range(h):
16         if list(im[j][i]) != list(im0[j][i]):
17             #list() is used to compare three RGB components
18             fl = True
19             im[j][i] = (0,0,0)
20 if fl:
21     print("Image", sys.argv[2], "is signed")
22     print("See the sign in file", sys.argv[3])
23     im_sign = imwrite(sys.argv[3], im)
24 else:
25     print("Image", sys.argv[2], "is UNSIGNED")
```

Эту программу можно вызвать из командной строки следующим образом:

```
1 python3 is_signed.py face.bmp face.bmp out_face_sign.bmp  
2 python3 is_signed.py ball.bmp ball.bmp out_ball_sign.bmp
```

Её работу демонстрируют полученные изображения:



out_ball_sign.bmp



out_face_sign.bmp

Рис. 4: Изображения с проявленной невидимой подписью.

5. Оценка результатов и ссылки

Итак, программа `make_invisible_sign.py` получает на вход изображение с подписью и без подписи и создает по ним изображение с невидимой подписью. Программа `is_signed.py` позволяет выяснить, было ли изображение подписано, и проявить подпись, если она была. В случае однородного по цветам изображения невидимая подпись немного видна, но если же подпись поставлена на изображении, где много различных близких цветов (например, фотография), то тогда невидимую подпись заметить глазом невозможно.

Файлы программ и картинки можно найти по ссылке:

https://github.com/Nekrasov-VV/homework/tree/master/4th_semester/4