Задание №3.2. Метод k-средних

В этом задании мы применим алгоритм K-средних к задаче сжатия изображений (с потерей качества) путем уменьшения количества цветов в используемой палитре.

Для работы мы будем использовать два файла: `peppers-small.tiff` и `peppers-large.tiff`. Файл `peppers-large.tiff` содержит изображение размером 512×512 пикселей в 24-битном цвете. Это означает, что каждый из 262144 пикселей задан тремя 8-битными числами (в диапазоне от 0 до 255), которые представляют собой интенсивность красного, зеленого и синего цветов в этом пикселе. Для хранения этого изображения без какого-либо сжатия нам понадобится $262144 \times 3 = 786432$ байта.

Суть нашего подхода очень простая – с помощью метода K-средних мы сократим всю палитру до $k = 16$ цветов. Более точно: каждый пиксель на изображении будем считать точкой в трехмерном (r, g, b)-пространстве. Мы сгруппируем все эти точки в 16 кластеров и заменим каждый пиксель в изображении цветом центроида того кластера, к которому он принадлежит.

Следуйте инструкциям, представленным ниже. Имейте в виду, что некоторые из операций могут занимать достаточно продолжительное время (вплоть до нескольких минут даже на быстродействующих компьютерах)!

Вопрос №1. Сжатие методом K-средних [6 баллов]

Прежде всего, давайте *посмотрим* на наши данные. Запустите интерактивную оболочку питона из каталога, в котором у вас находятся изображения, и наберите команды:

```
from matplotlib.image import imread; import matplotlib.pyplot as plt;
```

После этого запустите `A = imread(`peppers-large.tiff')`. Теперь `A` представляет собой трехмерную матрицу, в которой измерения `A[:,:,0]`, `A[:,:,1]` и `A[:,:,2]` хранят в себе информацию о красном, зеленом и синем компонентах каждого пикселя. Выполните команды `plt.imshow(A); plt.show()`, чтобы изображения отрисовались на холсте.

Так как большая версия картинки содержит 262144 пикселя и процесс кластеризации будет занимать значительное время, мы вместо нее будем использовать маленькую версию того же изображения для нахождения 16 доминирующих цветов. Повторите вышеуказанные шаги для `peppers-small.tiff`.

Завершите код, написанный в файле `k_means.py`. Трактуя каждый пиксель как точку в пространстве $\mathbb{R}^3$, реализуйте и выполните алгоритм K-средних для нахождения 16 кластеров данных точек. Выполните столько итераций, сколько потребуется для схождения алгоритма, но не менее 30. В целях инициализации установите центроиды кластеров равными случайным 16 пикселям на изображении.

Затем возьмите изображение `peppers-large.tiff` и замените в нем каждый пиксель на значение ближайшего из 16 найденных на маленьком изображении центроидов. Визуально сравните результат с исходной картинкой, чтобы убедиться, что алгоритм работает корректно. Добавьте файл отчета исходное и результирующее изображения.

Вопрос №2. Коэффициент сжатия [1 балл]

Если мы теперь сохраним сжатое изображение, используя всего 16 цветов, то каков (примерно) получится коэффициент сжатия изображения? Напомним, что коэффициентом сжатия называется отношение исходного объема к сжатому объему, т.е. если было 10 байт, а стало 5 байт, то коэффициент сжатия равен двум.