Афанасов Артем Константинович, M4105

Лабораторная работа 2

Улучшение процента сжатия JPG-файла без потерь

16.12.2021

Запуск

В папке **ExecutableFiles** лежат сборки приложения. Для запуска улучшенного компрессора и декомпрессора JPEG вводить следующие команды:

- **JPEGEnhancer**.exe cat30.jpg girl30.jpg
    - Замечу, что интерфейс пользователя можно было бы улучшить. Но сейчас нужно класть файлы filenameQuality.jpg в папку с исполняемым файлом **JPEGEnhancer**.exe.
    - В качестве аргумента передаются имея/имена файла/файлов jpg, который/которые надо сжимать.
    - Вывод **JPEGEnhancer**.exe в директорию: pathToJPEGEnhancer.exe\OutPutPatokaExtension\
- **JPEGEnhancerDecoder**.exe fileToPath1.patoka fileToPath2.patoka
    - В качестве аргумента передаются путь/пути до файла/файлов, сжатого/сжатых моим компрессором, улучшающим процент сжатия JPG’a.
    - Вывод **JPEGEnhancerDecoder**.exe в директорию: pathToJPEGEnhancerDecoder.exe\OutPutUnzippedPatokaExtension\

Описание работы

Реинжиниринг cpp кодера/декодера JPG

В cpp-декодере был встроен код, достающий DCT коэффициенты поблочно. В одном блоке один DC коэффициент и 63 AC. Эти коэффициенты и были сжаты лучшим образом.
В cpp-кодер был встроен код, читающий DCT декодированные коэффициенты, которые были закодированы лучшим образом.

Работа кодера

На вход подается JPG-файл.
Выходной файл декодера называется `fileNameOfJpg`EnhancedJPG.patoka.

1. С помощью cpp-декодера извлекаются DCT-коэффициенты. Коэффициенты записываются в два файла: AC и DC. Также cpp-декодер извлекает хедеры: ширину, высоту, цветность (цветное или ч/б) – записывается в файл headers.
    a. DecodedJPG - это выходной файл, который продуцирует cpp-декодер в своей начальной версии
2. Улучшенный декодер считывает таблицы из хедера и формирует по ним коэффициент качества.
3. Внутри кодера формируются байтовые/битовые потоки для обработки AC и DC коэффициентов.
4. Идея кодирования AC/DC частично взята из стандарта JPG (два скриншота снизу), но вместо хаффмана использован PPMA, а также знак не передаётся отдельным битом (он вшивается в бинарное представление, о котором будет сказано чуть позже)

$$n \rightarrow \underbrace{\texttt{huff(DC bits)}}_{\texttt{DC bit size}} \ \underbrace{\texttt{bin(DC)}}_{\texttt{DC}} \ \underbrace{\texttt{one bit}}_{\textit{sign}}.$$

$$[run, level] \rightarrow \underbrace{\texttt{Huffman}}_{[run, \texttt{level bit size}]} \ \underbrace{\texttt{bin(|level|)}}_{\texttt{level}} \ \underbrace{\texttt{one bit}}_{\textit{sign}}.$$

5. Знак вшивается в бинарное представление коэффициента
    a. По переданному size можно понять: отрицательное число или нет. У отрицательного числа самый старший бит в реализации данной работы является нулем. У положительного числа самый старший бит является единицей. Для перевода в отрицательное число используется инверсия всех битов.
    b. Пример: -9 запишется как 0110 в битовый поток. Размер бинарного представления будет 4. Декодер считает 0110 и поймет, что число, на самом деле, отрицательное. В итоге, получаем 1001, что являтся “-9” и что мы и передавали.

Декодер формирует следующую сигнатуру файла:
Ширина (2байт) **||** Высота(2байт) **||** Цветность(2байт) **||** Качество(2байт) **||** Длина закодированного PPMA’ом DC bit size(8 байт) **||** Закодированные PPMA’ом DC bit size **||** Бинарное представление DC с швитым знаком **||** Длина закодированных PPMA’ом пар [run, level size] (8 байт) **||** Закодированные PPMA’ом пары [run, level size] **||** Бинарное представление level с вшитым знаком.

Байтовые/битовые потоки объединяются и формируется выходной файл.

Работа декодера

На вход подается .patoka-файл

1. Считываются хедеры Ширина (2байт) **||** Высота(2байт) **||** Цветность(2байт) **||** Качество(2байт). Они используются для формирования хедера JPG’а cpp-кодером.
2. Считываются в байтовые/битовые потоки: PPMA-закодированные DC bit size. Затем бинарное представление DC с швитым знаком. Затем PPMA-закодированные пары [run, level size]. Бинарное представление level с вшитым знаком.
3. Декодируются PPMA-закодировнные данные.
4. Используя знание о размере и вшитых знаках получаем DC, AC коэффициенты
5. DC и AC коэффициенты, хедеры **[**Ширина (2байт) **||** Высота(2байт) **||** Цветность(2байт) **||** Качество(2байт)**]**. Это передаётся в cpp-кодер, который формирует JPG-файл, равный исходному JPG-файл бит в бит

Таблица размеров ниже.

Таблица результатов

Исходный размер в байтах для QF 80: 1388409
Улучшенный в байтах для QF 80 : 1327604
Процент улучшения: 4,37947

Исходный размер в байтах для QF 30: 606923
Улучшенный в байтах для QF 30 : 583481
Процент улучшения: 3,862

Название\размер, байт	QF=30		QF=80	
	Исходный	Улучшенный	Исходный	Улучшенный
airplane	19207	18321	44079	43219
arctichare	12080	11622	26569	26727
baboon	36113	35362	88465	85525
cat	35902	36032	81974	79457
fruits	17646	17506	45303	45037
frymire	200860	189941	440857	408714
girl	24310	23321	59979	59167
lena	17578	17382	43872	43110
monarch	28501	27758	64055	62145
peppers	18746	18346	47929	46685
pool	7467	6283	14492	13825
sails	45512	44128	105830	102494
serrano	58288	55862	138167	129618
tulips	37972	36969	85764	82361
watch30	46741	44648	101074	99520