МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра дискретной математики и алгоритмики

БАРСКИЙ
Антон Юрьевич

РАЗРАБОТКА СЕССИОННОГО АЛГОРИТМА СЖАТИЯ ИЗОБРАЖЕНИЙ

Дипломная работа

Научный руководитель:
кандидат физ.-мат. наук,
доцент Д.М. Васильков

Допущена к защите

«___» _____ 2021 г.

Зав. кафедрой дискретной математики и информатики
доктор физ.-мат. наук, профессор В.М. Котов

Минск, 2021

АННОТАЦИЯ

Барский А.Ю. Разработка сессионного алгоритма сжатия изображений. Дипломная работа / Минск: БГУ, 2021 – 46 с.

В дипломной работе рассмотрены основы сжатия изображений и математические модели сжатия данных. На основе обучающегося в процессе сжатия контекста, реализован алгоритм эффективный для сжатия последовательности изображений.

АНАТАЦЫЯ

Барскі А. Ю. Распрацоўка сесійнага алгарытму сціску малюнкаў. Дыпломная праца / Мінск: БДУ, 2021 – 46 с.

У дыпломнай працы разгледжаны асновы сціску малюнкаў і матэматычныя мадэлі сціску дадзеных. На аснове навучэнца ў працэсе сціску кантэксту, рэалізаваны эфектыўны алгарытм для сціску паслядоўнасці малюнкаў.

ABSTRACT

Barsky A.J. Development of a session image compression algorithm. Thesis / Minsk: BSU, 2021 – 46 p.

The thesis discusses the basics of image compression and mathematical models of data compression. Based on the learner in the process of context compression, an algorithm is implemented that is effective for compressing a sequence of images.

РЕФЕРАТ

Разработка сессионного алгоритма сжатия изображений

Дипломная работа: 46 страниц, 28 рисунков, 15 источников, 9 приложений.
Ключевые слова: СЖАТИЕ ДАННЫХ, ОБРАБОТКА ИЗОБРАЖЕНИЙ, НЕЙРОННЫЕ СЕТИ, КОЛИЧЕСТВО ИНФОРМАЦИИ, ЧИСЛО ОБУСЛОВЛЕННОСТИ, ОПЕРАТОРНАЯ НОРМА, ДИСКРЕТНОЕ КОСИНУСНОЕ ПРЕОБРАЗОВАНИЕ, ДИСКРЕТНОЕ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЕ, ГЕНЕТИЧЕСКИЙ АЛГОРИТМ.
Цель исследования: составить математическую модель для оценки эффективности алгоритмов сжатия изображений, оценить эффективность современных алгоритмов сжатия и исследовать их на предмет повышения степени сжатия. Разработать модель алгоритма сжатия, повышающую эффективность за счет использования контекста сжатия при кодировании последовательности изображений, изучить возможности применения алгоритмов машинного обучения при формировании контекста сжатия и построить полноценную клиент-серверную модель алгоритма сжатия.
Объект исследования: алгоритмы сжатия данных, изображений и видео, методы нахождения количества информации в данных.
Предмет исследования: эффективность алгоритмов сжатия, минимизация количества передаваемых и хранимых данных с минимальной потерей информации.
Методы исследования: метод анализа, метод обобщения, методы наблюдения и сравнения.
Задачи исследования: 1) Изучить математическую базу алгоритмов сжатия данных и изображений; 2) Построить алгоритм сжатия изображений, опирающийся на контекст сжатия и потенциально превосходящий по эффективности существующие алгоритмы сжатия; 3) Построить полноценную модель реализации клиент-серверного взаимодействия с использованием разработанного алгоритма.
Элементы научной новизны полученных результатов: модель алгоритма сжатия изображений, способного обучаться и подстраиваться под входные данные “на лету”.
Область возможного практического применения: передача и хранение последовательности изображений, повышение эффективности сжатия видео.

РЭФЕРАТ

Распрацоўка сесійнага алгарытму сціску малюнкаў

Дыпломная праца: 46 старонак, 28 малюнкаў, 15 крыніц, 9 прыкладанняў.
Ключавыя словы: СЦІСК ДАДЗЕНЫХ, АПРАЦОЎКА МАЛЮНКАЎ, НЕЙРОНАВЫЯ СЕТКІ, КОЛЬКАСЦЬ ІНФАРМАЦЫІ, ЛІК АБУМОЎЛЕНАСЦІ, ОПЕРАТОРНАЯ НОРМА, ДЫСКРЭТНАЕ КОСИНУСНОЕ ПЕРАЎТВАРЭННЕ, ДЫСКРЭТНАЕ ВЭЙВЛЕТАЎ-ПЕРАЎТВАРЭННЕ, ГЕНЕТЫЧНЫ АЛГАРЫТМ.
Мэта даследаванні: скласці матэматычную мадэль для ацэнкі эфектыўнасці алгарытмаў сціску малюнкаў, ацаніць эфектыўнасць сучасных алгарытмаў сціску і даследаваць іх на прадмет павышэння ступені сціску. Распрацаваць мадэль алгарытму сціску, падвышае эфектыўнасць за кошт выкарыстання кантэксту сціску пры кадаванні паслядоўнасці малюнкаў, вывучыць магчымасці прымянення алгарытмаў машыннага навучання пры фарміраванні кантэксту сціску і пабудаваць паўнавартасную кліент-серверную мадэль алгарытму сціску.
Аб'ект даследаванні: алгарытмы сціску дадзеных, малюнкаў і відэа, метады знаходжання колькасці інфармацыі ў дадзеных.
Прадмет даследавання: эфектыўнасць алгарытмаў сціску, мінімізацыя колькасці перадаюцца і захоўваемых дадзеных з мінімальнай стратай інфармацыі.
Метады даследавання: метад аналізу, метад абагульнення, метады назірання і параўнання.
Задачы даследавання: 1) Вывучыць матэматычную базу алгарытмаў сціску дадзеных і малюнкаў; 2) Пабудаваць алгарытм сціску малюнкаў, які абапіраецца на кантэкст сціску і патэнцыйна праўзыходны па эфектыўнасці існуючыя алгарытмы сціску; 3) Пабудаваць паўнавартасную мадэль рэалізацыі кліент-сервернага ўзаемадзеяння з выкарыстаннем распрацаванага алгарытму.
Элементы навуковай навізны атрыманых вынікаў: мадэль алгарытму сціску малюнкаў, здольнага навучацца і падладжвацца пад ўваходныя дадзеныя "на лета".
Вобласць магчымага практычнага прымянення: перадача і захоўванне паслядоўнасці малюнкаў, павышэнне эфектыўнасці сціску відэа.

ABSTRACT

Development of a session image compression algorithm

Diploma thesis: 46 pages, 28 figures, 15 sources, and 9 appendices.
Keywords: DATA COMPRESSION, IMAGE PROCESSING, NEURAL NETWORKS, AMOUNT OF INFORMATION, CONDITION NUMBER, OPERATOR NORM, DISCRETE COSINE TRANSFORM, DISCRETE WAVELET TRANSFORM, GENETIC ALGORITHM.
Purpose of the research: create a mathematical model for evaluating the effectiveness of image compression algorithms, evaluate the effectiveness of modern compression algorithms, and investigate them for improving the compression ratio. To develop a compression algorithm model that increases efficiency by using the compression context when encoding a sequence of images, to study the possibilities of using machine learning algorithms in forming the compression context, and to build a full-fledged client-server model of the compression algorithm.
Research object: algorithms for compressing data, images, and videos, and methods for finding the amount of information in data.
Subject of the study: efficiency of compression algorithms, minimizing the amount of transmitted and stored data with minimal information loss.
Research methods: analysis method, generalization method, observation and comparison methods.
Research objectives: 1) To study the mathematical base of data and image compression algorithms; 2) To construct an image compression algorithm based on the compression context and potentially exceeding the efficiency of existing compression algorithms; 3) To construct a full-fledged model for implementing client-server interaction using the developed algorithm.
Elements of scientific novelty of the results obtained: model of an image compression algorithm that can learn and adapt to input data on the fly.
Scope of possible practical application: transfer and store a sequence of images, improving the efficiency of video compression.

ОГЛАВЛЕНИЕ

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ, ТЕРМИНЫ

Энтропия — мера неопределенности (неупорядоченности) системы.

Кодирование серий — сжатие последовательностей подряд идущих символов.

Квантование (в терминах сжатия) — понижение энтропии информации за счет выделения уровней и округления рассматриваемых значений к этим уровням. Применяется преимущественно в lossy сжатиях, так как почти всегда приводит к потере информации.

Оптический поток — технология, использующаяся в различных областях компьютерного зрения для определения сдвигов, сегментации, выделения объектов, компрессии видео.

Дельта кодирование — преобразование, заменяющее каждое следующее число разностью между ним и предыдущим числом. Используется с целью понижения энтропии входных данных в случаях, когда входные данные представляют собой возрастающую или убывающую последовательность.

ДКП — дискретное косинусное преобразование.

ДВП — дискретное вейвлет-преобразование.

Число обусловленности — численная характеристика функции, указывающая на величину изменения функции при небольших изменениях аргумента

ВВЕДЕНИЕ

С распространением в современном мире как цифровых носителей, так и сети Интернет, всё чаще встает вопрос о хранении и передаче данных в цифровом формате. Потребности пользователей растут быстрее, чем развиваются аппаратные технологии хранения и передачи данных, поэтому всё чаще встает вопрос программной оптимизации данных процессов. Помимо механизмов балансировки нагрузки, оптимизаций I/O операций, оптимизаций в сетевой передаче на транспортном уровне, важную роль играют механизмы сжатия данных. Сжатие данных подразумевает под собой уменьшение размера рассматриваемой информации за счет использования того факта, что в большинстве своем данные не являются случайным набором бит, а подчиняются определенному закону. Другими словами, используется тот факт, что данные являются либо зависимыми случайными величинами, либо случайными величинами, подчиняющимися определенной, неравномерной функции распределения.

Большая часть данных передаваемых по сети — фото и видео файлы. Для быстрой передачи этих файлов и компактного их хранения требуются эффективные алгоритмы сжатия, способные быстро и качественно сжимать отдельные изображения и последовательности кадров видео. Проблемой при сжатии последовательности кадров видео является вычислительная сложность поиска областей корреляции изображений, которая производится с целью повышения коэффициента сжатия путем кодирования ссылки на схожую область изображения и разности между предыдущим и следующим изображением вместо кодирования полностью исходного изображения. Проблемой при сжатии набора изображений является выделение общего контекста для данных изображений с целью повышения качества сжатия.

В этой работе мы рассмотрим теоретическую базу подкрепляющую алгоритмы сжатия изображений последних десятилетий, систематизируем и обоснуем основные подходы к сжатию изображений, выясним, каков предел сжатия и как теоретически его можно достигнуть, реализуем алгоритмы позволяющие повышать эффективность сжатия последовательности видеокадров, последовательности изображений, рассмотрим применение алгоритмов машинного обучения в области сжатия изображений и реализуем модель, позволяющую повысить эффективность сжатия последовательности изображений за счет обучения на предыдущих данных и сохранении контекста. Составим модель клиент-серверного взаимодействия при использовании разработанного алгоритма сжатия для передачи потокового мультимедиа а также составим модель для “оффлайн” сжатия набора изображений.

Глава 1. ОСНОВНЫЕ ПРИНЦИПЫ СЖАТИЯ ДАННЫХ

1.1 Основные понятия

Рассмотрим основные подходы, применяющиеся в последние десятилетия для сжатия данных (в том числе изображений и видео). Для дальнейшего повествования, нам потребуются следующие термины:

- информационная энтропия — непредсказуемость появления какого-либо символа алфавита
- формула количества информации по Шеннону:

$$H(x) = -\sum_i p_i log_2 p_i$$

- сжатие без потерь (lossless) — сжатие, при котором исходные данные могут быть в точности восстановлены после сжатия
- сжатие с потерями (lossy) — сжатие, при котором восстановленные после сжатия данные могут незначительно отличаться от исходных данных

1.2 Сжатие с потерями

Сжатие с потерями применяется для тех форматов данных, для которых незначительная потеря информации не является критической. Такие форматы — это, например, изображения, аудиозаписи, видео. Сжатие с потерями достигается за счет удаления информации, к которой невосприимчивы человеческие органы чувств. Стоит заметить, что с точки зрения компьютера и стандартных математических метрик типа L_1, L_2, L_∞изображение (видео, аудио) после декомпрессии может значительно отличаться от исходного. По этой причине большое внимание в разработке алгоритмов сжатия с потерями уделяется поиску функции для измерения уровня искажений после компрессии, а также функции схожести двух изображений. Одна из самых популярных метрик, применяемых для оценки качества сжатия изображений — PSNR:

$$PSNR = 10\, log_{10}(\frac{MAX_I^2}{RMSE^2}) = 20\, log_{10}(\frac{MAX_I}{RMSE})$$

где MAX_I- это максимальное значение, принимаемое пикселем изображения (255 в случае черно-белого полутонового изображения), а $RMSE$:

$$RMSE = \sqrt{\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|I(i,j) - K(i,j)|^2}$$

где I, K — два изображения, одно из которых считается зашумленной копией другого.

1.3 Сжатие без потерь

Сжатие без потерь в общем смысле этого слова может применяться для любых данных представленных в цифровом виде. Сжатие без потерь может быть осуществлено с использованием устранения таких видов избыточности как [1]:

- Кодовая избыточность (избыточность по энтропии). Код — последовательность символов применяемых для представления информации. Последовательность кодовых символов — кодовое слово. 8, 24 и 32 битовые коды, используемые для представления значений в большинстве двумерных массивах яркостей, как правило, содержат большее число битов, чем это необходимо. Проще говоря, если в изображении есть только синий и зеленые цвета, не нужно выделять отдельный байт под красный цвет. Наиболее часто встречающиеся символы кодируются в меньшее число бит, самые редкие — в большее.
- Пространственная или временная избыточность. Поскольку значения пикселей в большинстве двумерных массивов яркостей пространственно коррелированы (т. е. значение каждого пикселя похоже на значения соседних пикселей), информация излишним образом дублируется в представлении коррелированных пикселей. В видеопоследовательностях информация дублируется также в коррелированных по времени пикселях (т. е. тех, значения которых похожи или зависят от значений пикселей в соседних кадрах).

Глава 2. ПРЕДПОСЫЛКИ К УЛУЧШЕНИЮ СУЩЕСТВУЮЩИХ АЛГОРИТМОВ

2.1 Расчет эффективности сжатия без потерь

Рассмотрим, что представляет из себя любой алгоритм сжатия без потерь с точки зрения теории множеств. Сжатие без потерь подразумевает под собой биективное отображение исходных данных в сжатые данные. Рассмотрим в качестве множества исходных данных — множество всех возможных изображений размера NxM с количеством ступеней яркости D. Пусть множество исходных данных — X, множество сжатых данных — Y. Ввиду биекции, |X| = |Y|.

Предположим теперь, что на вход для сжатия поступает случайное изображение с выбранными параметрами, причем каждое изображение поступает равновероятно. Главное отличие в рассуждениях ниже от основных принципов сжатия, описанных в первой главе, в том, что мы будем рассматривать всё изображение как единый случайный объект (случайное событие). Заметим, что тогда по формуле Шеннона при условии, что изображение поступает из равномерного распределения, количество информации, требуемое для описания этого изображения равно:

$$H(x) = -\sum_{|Y|} \frac{1}{|Y|} log_2 \frac{1}{|Y|} = log_2|Y| = log_2|X| = NMlog_2 D$$

Нетрудно заметить, что эта величина — не что иное, как размер изображения в формате BMP, без сжатия. Таким образом, вне зависимости от алгоритма сжатия, если изображение, которое подается на вход, совершенно случайное, в среднем оно будет занимать не меньше места чем в несжатом виде. Отсюда вытекают следующие следствия.

2.2 Утверждения об эффективности алгоритмов сжатия

1. Сжатие данных не существует без предположения о неравномерности распределения сжимаемых данных
2. Для любого алгоритма сжатия истинно утверждение: для любого множества входных данных X, сумма бит, затраченных на сохранение всех этих данных в сжатом виде не меньше суммы бит, затраченных на сохранение этих данных в несжатом, “сыром” виде, что равно$|X|log_2|X|$.
3. В свою очередь из пункта выше следует, что для любого алгоритма сжатия некоторые данные после “сжатия” уменьшаются в размере, а некоторые — увеличиваются, причем чем больше данных требуют меньше бит для

хранения, тем сильнее увеличивается количество бит, затрачиваемых на хранение остальных данных в сжатом виде.

4. Также из второго пункта следует, что каждому алгоритму можно для каждого множества данных присвоить **коэффициент избыточности алгоритма сжатия A на множестве X**, который равен:

$$\frac{\sum_{x \in X} A(x) - |X| log_2 |X|}{|X|}$$

где A(x) — количество бит, требуемых для хранения входных данных x, сжатых алгоритмом A. Заметим, что данный коэффициент всегда неотрицательный и чем он больше, тем хуже себя ведет рассматриваемый алгоритм на равномерно распределенных данных.

5. Нет понятия “эффективность алгоритма сжатия” или “качество алгоритма сжатия” без указания множества X, а также информации о распределении вероятностей элементов этого множества
6. Не существует и не может существовать универсального алгоритма сжатия, который лучше всех остальных алгоритмов на любых наборах входных данных и любых распределениях входных данных

Таким образом, во-первых, эффективность сжатия достигается только лишь за счет того, что на практике распределение, которому подчиняются входные данные, не является равномерным, а во-вторых, алгоритм сжатия, адаптирующийся под распределение, которому принадлежат сжимаемые им данные, теоретически может достичь значительно лучшего качества сжатия, чем статический алгоритм.

2.3 Теоретическое достижение наивысшей эффективности сжатия

Предположим теперь, что входные данные подчиняются некоторому известному нам распределению p(x). Тогда по формуле Шеннона имеем среднее количество бит для сохранения сжатых данных:

$$H(X) = - \sum_{x \in X} p(x) log_2 p(x)$$

Причем, заметим, что в данном случае, от функции вероятности напрямую зависит качество сжатия. К слову, если разработать такой алгоритм, который будет с единичной вероятностью угадывать следующие данные, количество бит, требуемых для передачи этой информации будет равно 0.

Таким образом, нам требуется найти плотность вероятности P(x) по доступной информации о природе распределения входных данных. В простейшем случае, доступная информация — это предыдущие сжимаемые изображения в данном

контексте. Другими словами, требуется найти условную вероятность P(x|X), где X — множество предыдущих сжимаемых изображений.

На этот моменте может возникнуть несколько заблуждений:

1. Можно сжимать изображение постепенно и менять функцию вероятности в зависимости от первых прочитанных частей изображения и это будет эффективнее, чем если не менять функцию вероятности и сжимать изображение целиком. Ответ: Функции вероятности считаются для всех возможных изображений, таким образом, они уже учитывают факт, что если, скажем, часто выпадают одноцветные изображения и сейчас выпал синий, то почти наверное изображение будет синим
2. Различных изображений одного размера даже при рассматривании определенной области безмерно много и функция вероятности почти всегда будет равномерной. Ответ: во-первых, будь так, не существовало бы алгоритма сжатия оптимальнее BMP, во-вторых, на самом деле различных изображений по сравнению с количеством возможных изображений безумно мало.

   Заметим, что если рассматривать только сжатие изображений хранящихся в данный момент в мире на серверах, и не учитывать генерацию и обновление этих изображений, таких изображений точно не больше, чем атомов во Вселенной, которых примерно 10^{82}. В то же время, количество различных полутоновых черно-белых изображений размера 32x32 порядка 10^{2466}. К слову, легко заметить, что при отсутствии обновления изображений, даже при условии равномерного их распределения, количество бит по Шеннону, необходимое для передачи любого из этих изображений порядка $log_2 10^{82} = 266(\text{бит}) = 33(\text{байта})$
3. Можно вычислить функцию вероятности для определенной области, в которой она будет применяться, и полученный алгоритм сжатия будет теоретически лучшим из всех существующих для данной области. Ответ: и да, и нет; теоретически это действительно верное утверждение, однако проблема в практической реализации здесь та же, по которой мы до сих пор не обмениваемся популярными изображениями сжатыми до уровня 33 байт — требуемое количество памяти для хранения полученной функции вероятности. Как было указано выше, уже для черно-белых изображений размера 32x32 количество различных изображений, для которых нужно знать вероятность выпадания — порядка 10^{2466}, такое количество информации просто невозможно сохранить на компьютере.

По итогу, имеем следующие требования к разрабатываемому алгоритму сжатия:

1. Адаптивность к контексту или обучаемость (алгоритм должен быть динамическим, обучаемым, должен учитывать предыдущие изображения при сжатии следующих)
2. Эффективное хранение вероятностей (для большого количества вариантов использование функций для вычисления по изображению его вероятности, совмещение нескольких изображений в группы с одинаковой вероятностью).

Прежде чем приступать к реализации алгоритма, рассмотрим современные подходы к сжатию изображений.

Глава 3. СОВРЕМЕННЫЕ АЛГОРИТМЫ СЖАТИЯ ДАННЫХ

3.1 Виды сжатий

Современные алгоритмы сжатия разделяются на два типа: lossless и lossy. Lossless применяется для сжатия данных любого формата, в то время как lossy используется преимущественно для сжатия аудио и медиа файлов, так как именно эти форматы данных можно незначительно изменять заметно не повредив данные. Lossy алгоритмы значительно эффективнее сжимают данные за счет того, что существенно снижают энтропию данных незначительно изменяя эти данные. Рассмотрим техники, используемые для реализации lossless и lossy алгоритмов сжатия [1].

3.2 Сжатия без потерь

Основные составляющие lossless алгоритмов сжатия — это алгоритмы кодирования серий и энтропийные алгоритмы сжатия данных. Пример алгоритмов кодирования серий — RLE, любой словарный алгоритм (в частности, все алгоритмы линейки LZxx). Основной принцип — сжатие повторяющихся подмножеств рассматриваемого множества символов. Реализуется с помощью ссылок на предыдущий такой же фрагмент, либо с помощью указания количества повторений символа. Пример энтропийных — коды Хаффмана, коды Голомб-Райса[2], арифметическое кодирование. Основной принцип — сохранение наиболее часто встречающихся символов в короткие коды и, соответственно, редко встречающихся — в длинные.

Также при сжатии изображений используют технику предсказания следующего значения на основе уже декодированных (обычно следующий пиксель предсказывается на основе левых и верхних пикселей, так как только их на момент декодирования текущего пикселя будет знать декодер). Такие предсказания используют факт, что изображение представляет собой 2D объект и соседние пиксели в большинстве своем не сильно отличаются друг от друга.[3][4]

3.3 Сжатия с потерями

Сжатие lossy достигается в основном путем квантования данных. Однако квантование исходных данных заметно повредит их и создаст видимые границы между уровнями квантования. Поэтому используют дополнительные техники, повышающие коэффициент сжатия и качество выходных данных. Одна из таких техник — применение преобразований, сосредотачивающих общую информацию в

небольшом наборе байт, оставляя в остальных байтах лишь уточнение этой информации. Такие алгоритмы — ДКП(рис. 3.1) и ДВП(рис. 3.2)[5][6]

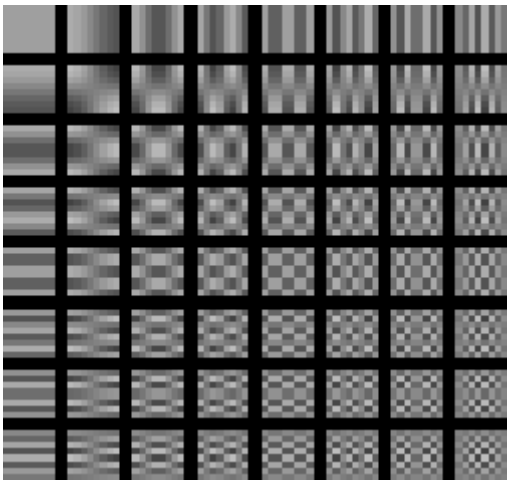


Рисунок 3.1. Иллюстрация коэффициентов ДКП-2

Рисунок 3.2. Иллюстрация одного прохода ДВП-2D

3.4 Особенности сжатия изображений

И наконец, во время сжатия изображений часто пользуются особенностями человеческого зрения, а именно тем фактом, что человеческий глаз более восприимчив к количеству света на изображении, чем к его цветности. Поэтому перед lossy кодированием изображения его обычно переводят в YCrCb формат и Y часть подвергают меньшему квантованию, чем Cr и Cb части. Также при кодировании изображений отдельное внимание уделяется переходам между цветами соседних пикселей.

Практически все из современных наиболее распространенных алгоритмов сжатия изображений базируются на описанных выше техниках, а именно —

являются всего лишь сочетанием того или иного подмножества из множества техник выше (рис. 3.3).

Рисунок 3.3. Демонстрация составляющих наиболее популярных на сегодня алгоритмов

Говоря про алгоритмы сжатия видео стоит сказать, что чаще всего они базируются на алгоритмах сжатия изображений. Единственное отличие — алгоритмы видео используют информацию о ранее сжатых кадрах и полагаются на факт, что изображения в видео потоке отличаются в большинстве лишь смещением некоторых пикселей на несколько позиций. Отсюда появляется две дополнительные техники: компенсация движения и кодирование разностей. Первая используется для нахождения перемещения областей и компенсации этого перемещения, вторая — для кодирования не самого кадра, а разности между текущим кадром и предыдущим кадром после компенсации движения (рис 3.4).

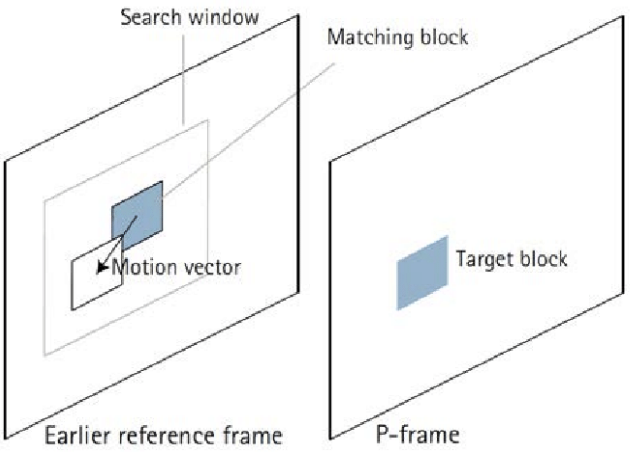


Рисунок 3.4. Схема работы компенсации движения

Помимо алгоритмов сжатия изображений и алгоритмов сжатия видео есть также алгоритмы, специализирующиеся на сжатии последовательности схожих изображений. Обычно они строятся так же, как алгоритмы сжатия видео, однако без этапа компенсации движения, так как на последовательности изображений объекты обычно сильно сдвинуты относительно друг друга и нет возможности применить существующие алгоритмы компенсации движения. Данная работа нацелена на написание алгоритма, являющегося более оптимальным для сжатия последовательности изображений, для которых нет возможности эффективно применить современные виды компенсации движения.

Глава 4. РЕАЛИЗАЦИЯ ОСНОВНЫХ АЛГОРИТМОВ СЖАТИЯ ИЗОБРАЖЕНИЙ

4.1 Используемая литература, технологии

Перед тем, как реализовывать собственный алгоритм сжатия, реализуем наиболее популярный алгоритм сжатия, чтобы углубиться в детали имплементации алгоритмов сжатия изображений. Реализуем JPEG, так как это самый популярный среди lossy алгоритмов сжатия и один из наиболее сложных алгоритмов. Подробное описание алгоритма было взято из статьи Стенфордского университета, описывающей стандарт JPEG[7][8][9]. Для реализации алгоритма будем использовать язык Python.

4.2 Алгоритм JPEG

Составляющие алгоритма:

- Перевод из RGB в YCrCb.

```
def rgb2ycbcr(img):
    a = np.array([[0.299, 0.587, 0.114],
                  [-0.1687, -0.3313, 0.5],
                  [0.5, -0.4187, -0.0813]])
    b = np.array([0, 128, 128])
    return np.array([[(b + a.dot(x)).astype(int) for x in y] for y in img])
```

Рисунок 4.1. Функция для перевода RGB в YCrCb

```
def ycbcr2rgb(img):
    a = np.array([[1, 0, 1.402],
                  [1, -0.34414, -0.71414],
                  [1, 1.77, 0]])
    b = np.array([0, 128, 128])

    return np.array([[a.dot(x - b).astype(int) for x in y] for y in img])
```

Рисунок 4.2. Функция для перевода YCrCb в RGB

Убедимся в том, что удаление данных из Y канала заметнее для человеческого глаза, нежели из каналов Cr и Cb (рис 4.1).

```
gauss = np.zeros_like(ycbcr_img)
gauss[:,:,0] = ycbcr_img[:,:,0]
gauss[:,:,1] = gaussian_filter(ycbcr_img[:,:,1], 2)
gauss[:,:,2] = gaussian_filter(ycbcr_img[:,:,2], 2)
```

Рисунок 4.3. Функция для замутнения Cr и Cb коэффициентов

```
gauss = np.zeros_like(ycbcr_img)
gauss[:,:,0] = gaussian_filter(ycbcr_img[:,:,0], 2)
gauss[:,:,1] = ycbcr_img[:,:,1]
gauss[:,:,2] = ycbcr_img[:,:,2]
```

Рисунок 4.4. Функция для замутнения Y коэффициентов

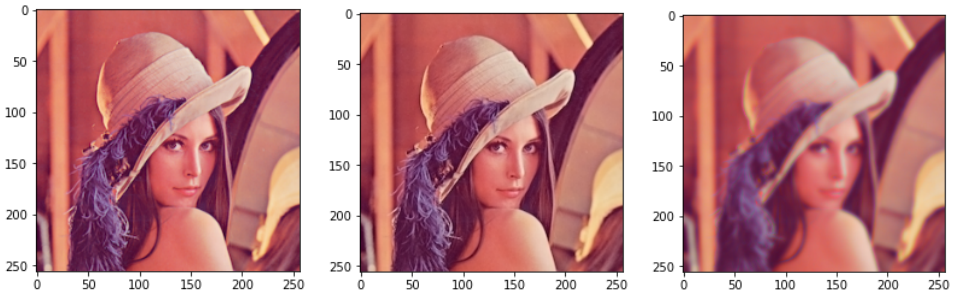


Рисунок 4.5. Демонстрация влияния одинаковой размытости разных каналов на человеческий глаз. Первое изображение — исходное, второе — с размытыми Cb и Cr каналами, третье — с размытым Y каналом

- Тайлинг — разбиваем всё изображение на квадраты 8x8 для локализации по памяти и возможности дальнейшего распараллеливания (так как каждый квадрат будем обрабатывать независимо друг от друга).

```
...
ret = [[] for i in range(3)]
for i in range(img.shape[0] // 8):
    for j in range(img.shape[1] // 8):
        block = y_component[np.ix_(range(i*8, (i+1)*8), range(j*8, (j+1)*8))]
```

Рисунок 4.6. Разбиение на тайлы на примере Y компоненты

- Дискретное косинусное преобразование

```
def dct(block):
    alpha = lambda a: 1 / np.sqrt(2) if a == 0 else 1
    cs = lambda a, b: np.cos((2 * a + 1) * b * np.pi/ 16)
    f = lambda x, y, u, v: block[x][y] * cs(x, u) * cs(y, v)
    guv = lambda u, v: 1 / 4 * alpha(u) * alpha(v) * sum([f(x, y, u, v) for y in range(8) for x in range(8)])
    g = np.array([[guv(u, v) for v in range(8)] for u in range(8)])

    return g
```

Рисунок 4.7. Реализация ДКП

- Квантование

```
y_quantization_matrix = np.array([
    [16, 11, 10, 16, 24, 40, 51, 61],
    [12, 12, 14, 19, 26, 58, 60, 55],
    [14, 13, 16, 24, 40, 57, 69, 56],
    [14, 17, 22, 29, 51, 87, 80, 62],
    [18, 22, 37, 56, 68, 109, 103, 77],
    [24, 35, 55, 64, 81, 104, 113, 92],
    [49, 64, 78, 87, 103, 121, 120, 101],
    [72, 92, 95, 98, 112, 100, 103, 99]
])
color_quantization_matrix = np.array([
    [17, 18, 24, 47, 99, 99, 99, 99],
    [18, 21, 26, 66, 99, 99, 99, 99],
    [24, 26, 56, 99, 99, 99, 99, 99],
    [47, 66, 99, 99, 99, 99, 99, 99],
    [99, 99, 99, 99, 99, 99, 99, 99],
    [99, 99, 99, 99, 99, 99, 99, 99],
    [99, 99, 99, 99, 99, 99, 99, 99],
    [99, 99, 99, 99, 99, 99, 99, 99]
])
```

Рисунок 4.8. Матрицы квантования для Y и Cr/Cb компонент соответственно

- Линеаризация — зиг-заг преобразование. Преобразуем матрицу 8x8 в строку из 64 коэффициентов. Собираем элементы зигзагообразно в виду

особенностей DCT. В терминах JPEG первый коэффициент будем называть DC, остальные — AC.

- RLE кодирование для AC. То есть кодируем последовательности нулей в AC.

```
ret = []
n0 = 0
for x in zigzag_list:
    if x == 0:
        if n0 == 0:
            ret.append(0)
        n0 += 1
        continue
    if n0 != 0:
        ret.append(n0)
        n0 = 0
    ret.append(x)

if n0 != 0:
    ret.append(n0)

return ret
```

Рисунок 4.9. Сжатие RLE после зиг-заг преобразования

- Дельта кодирование для DC
- Кодирование Хаффмана для полученных данных

4.3 Результаты

Таким образом был получен алгоритм сжатия JPEG. Аналогично был реализован алгоритм декомпрессии JPEG. Весь исходный код можно найти в репозитории. https://github.com/JustTohich/ImageCompressions/blob/main/JPEG.ipynb

Заметим, что данный алгоритм, как и большинство других алгоритмов сжатия изображений, имеет следующие составляющие:

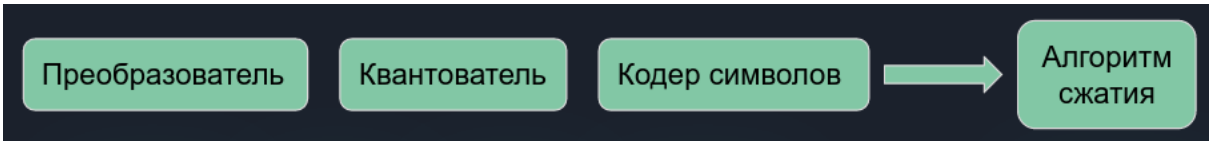


Рисунок 4.10. Общая схема алгоритма сжатия изображений

Глава 5. РАЗРАБОТКА АЛГОРИТМА СЖАТИЯ ПОСЛЕДОВАТЕЛЬНОСТИ КЛЮЧЕВЫХ КАДРОВ ВИДЕО

5.1 Исследование подходов к решению задачи нахождения схожих объектов на изображениях

5.1.1 Цель исследования

Изучив принцип работы алгоритмов сжатия и зная, что повысить уровень сжатия можно с помощью контекста предыдущих изображений, реализуем алгоритм, позволяющий на высоком уровне получать, сохранять и искать контекст изображений, а именно -- алгоритм, который будет находить схожие участки на разных изображениях. Для последовательности изображений данный алгоритм не подходит, однако он отлично подойдет для сжатия ключевых кадров видео (рис 5.1).

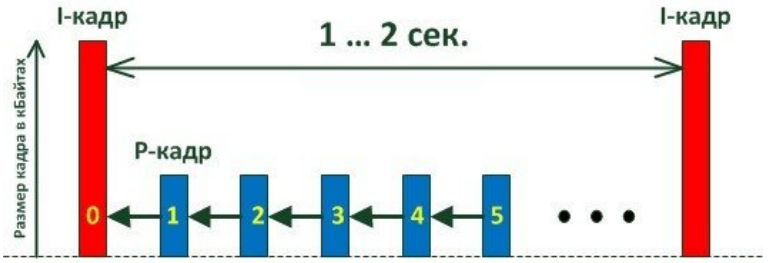


Рисунок 5.1. Ключевые(I) и разностные(P) кадры

Обычно ключевые кадры видео сжимаются при помощи обычных алгоритмов сжатия изображений, без учета контекста предыдущих сжатых ключевых кадров. Так происходит, потому что обычно на разных ключевых кадрах объекты обычно сильно меняют своё расположение и стандартные алгоритмы компенсации движения не могут справиться с такими отклонениями. Поэтому в данном случае требуется построить алгоритм позволяющий найти не точные совпадения фрагментов изображениях вне зависимости от их расположения.

5.1.2 Алгоритмы компенсации движения

Задача нахождения похожих объектов на изображениях встречается в том или ином виде во многих областях программирования: от сжатия видео до распознавания объектов. Рассмотрим самые популярные подходы к решению данной задачи и выберем наиболее подходящий для рассматриваемой задачи. Большее внимание будем уделять следующим подходам: motion estimation, optical flow, object recognition.

Для сжатия видео используют алгоритмы основанные на факте, что объекты на соседних кадрах перемещаются недалеко друг от друга. Эти алгоритмы

рассматривают близлежащую область вокруг рассматриваемой точки и подбирают наиболее подходящую область, для которой разность между рассматриваемым кадром и предыдущим по выбранной метрики будет лучшей. Мы не можем гарантировать, чтобы похожие объекты будут рядом, поэтому такой подход не подходит.

5.1.3 Алгоритмы нахождения оптического потока

Для нахождения оптического потока и распознавания объектов используется схожая методика действий [10]: сначала выделяем на изображении ключевые(особенные) точки, после строим для них дескрипторы (описание области вокруг), так делаем для обоих рассматриваемых изображений и после ищем совпадения между дескрипторами. Чем лучше совпадение между дескрипторами — тем более схожи области вокруг рассматриваемых ключевых точек. Рассмотрим несколько самых популярных алгоритмов выделения ключевых точек и нахождения дескрипторов.

Одни из самых популярных методов нахождения и описания ключевых точек — ORB, SIFT, SURF[11][12][13]. У SIFT и SURF среди особенностей — инвариант по размеру дескриптора и по поворотам. Однако мы не преследуем цель найти описание ключевых точек инвариантное к поворотам и, тем более, скейлингу. Поэтому SIFT и SURF нам сразу не подходят. Попробуем ORB на простом примере. Возьмем изображение Lenna.png, вырежем часть и проверим, насколько хорошо ORB сопоставит дескрипторы ключевым точкам. Для поиска одинаковых дескрипторов на двух фрагментах будем использовать BFMatch.

```
import cv2

def orb(file1, file2, count=50):
    img1 = cv2.imread(file1, 0)
    img2 = cv2.imread(file2, 0)

    orb = cv2.ORB_create()
    kp1, des1 = orb.detectAndCompute(img1, None)
    kp2, des2 = orb.detectAndCompute(img2, None)

    if des1 is None or des2 is None:
        return

    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
    matches = bf.match(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)
    match_img = cv2.drawMatches(img1, kp1, img2, kp2, matches[:count], None)
    cv2.imwrite('Matches.png',match_img)
    return kp1, des1, kp2, des2, matches
```

Рисунок 5.2. Получение ключевых точек и дескрипторов через ORB

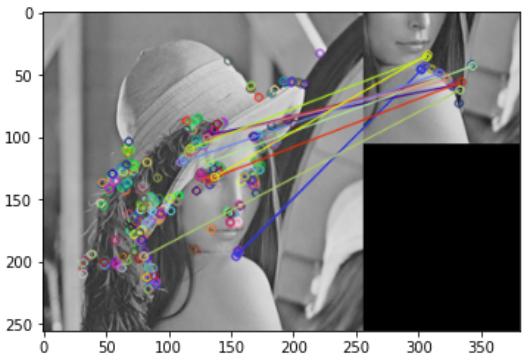


Рисунок 5.3. Результат сопоставления точек и дескрипторов ORB на тестовом примере (поиск на целом изображении его части)

Как видим, даже на таком простом примере, ORB не готов гарантировать требуемую точность. Попробуем найти отдельно алгоритм поиска ключевых точек и алгоритм их описания. Как часть ORB используется алгоритм FAST. Рассмотрим для начала его отдельно. После подбора параметров, можно прийти к следующему неплохому результату (рис 5.5).

```
def keypoints(file):
    img = cv2.imread(file,0)

    fast = cv2.FastFeatureDetector_create(threshold=15)
    kp = fast.detect(img,None)

    img2 = cv2.drawKeypoints(img, kp, None,color=(255,0,0))
    cv2.imwrite('fast_true.png',img2)
    plt.imshow(imread("fast_true.png"))
```

Рисунок 5.4. Получение ключевых точек через FAST

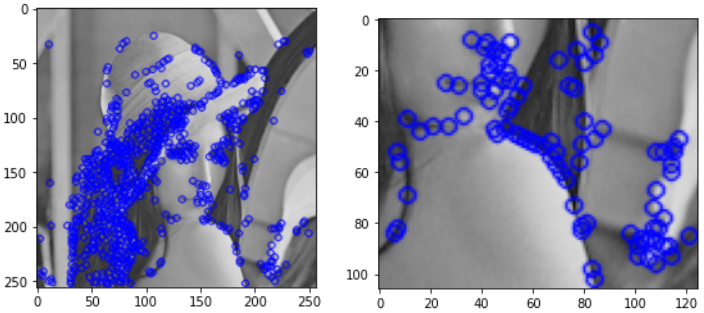


Рисунок 5.5. Получение ключевых точек через FAST на тестовых изображениях

Как видим, ключевые точки выбираются одинаково на обоих фрагментах, а значит при правильном их описании, между ними найдется соответствие.

Алгоритм BRIEF[14] не рассматриваем, так как ORB и является объединением FAST и BRIEF. Был рассмотрен алгоритм нахождения дескрипторов FREAK[15]. Однако результаты его работы вместе с FAST оказались не лучше результатов ORB. Лучшим кандидатом на роль дескриптора был найден и выбран HOG ввиду того, что

он сохраняет информацию о палитре цветов вокруг заданной области, причем учитывает расположение рассматриваемых пикселей, а потому лучше остальных будет находить соответствие между одинаковыми фрагментами (рис 5.6, 5.7).

```
def fast_hog(file1, file2, count=50, threshold=10):
    img1 = cv2.imread(file1, 0)
    img2 = cv2.imread(file2, 0)

    fast = cv2.FastFeatureDetector_create(threshold=threshold)
    winSize = (32,32)
    blockSize = (16,16)
    blockStride = (8,8)
    cellSize = (8,8)
    nbins = 9
    derivAperture = 1
    winSigma = 4.
    histogramNormType = 0
    L2HysThreshold = 2.0000000000000001e-01
    gammaCorrection = 0
    nlevels = 64
    hog = cv2.HOGDescriptor(winSize,blockSize,blockStride,cellSize,nbins,derivAperture,winSigma,
                            histogramNormType,L2HysThreshold,gammaCorrection,nlevels)

    winStride = (32,32)
    padding = (32,32)
    kp1 = fast.detect(img1, None)
    kp2 = fast.detect(img2, None)
    des1 = hog.compute(img1,winStride,padding,[k.pt for k in kp1])
    des2 = hog.compute(img2,winStride,padding,[k.pt for k in kp2])

    if des1 is None or des2 is None:
        return

    des1, des2 = np.array(des1).reshape((len(kp1),-1)), np.array(des2).reshape((len(kp2),-1))

    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
    matches = bf.match(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)
    match_img = cv2.drawMatches(img1, kp1, img2, kp2, matches[:count], None)
    cv2.imwrite('Matches.png',match_img)
    return kp1, des1, kp2, des2, matches[:count]
```

Рисунок 5.6. Получение ключевых точек через FAST и получение дескрипторов через HOG

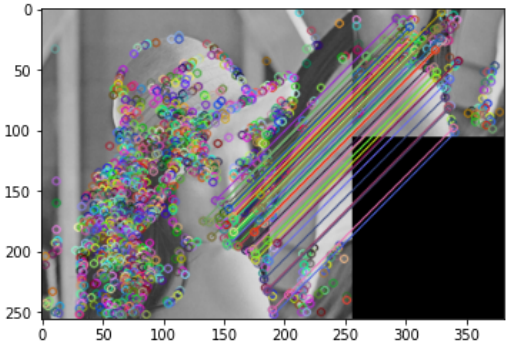


Рисунок 5.7. Получение ключевых точек через FAST и получение дескрипторов через HOG на тестовый изображениях

Как видим, количество верных соответствий близко к максимуму. Рассмотрим несколько другой пример (рис 5.8).

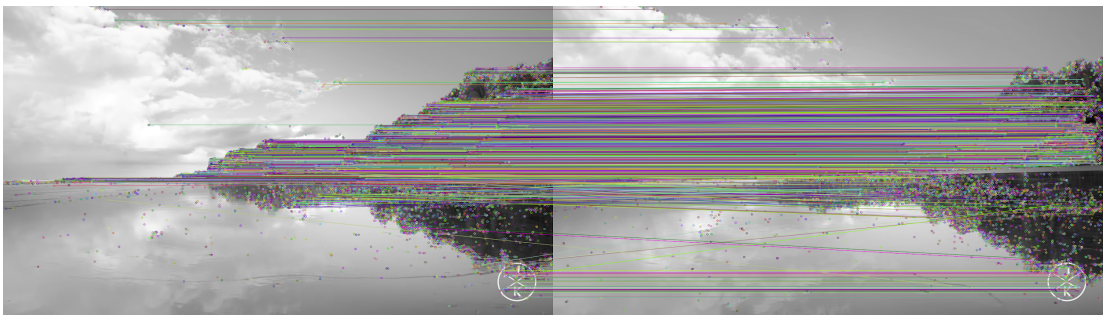


Рисунок 5.8. Результат FAST+HOG на тесте из двух ключевых кадров видеофрагмента

Практически все соответствия верные. Может показаться, что это два одинаковых кадра, поэтому приведем ниже модуль попиксельной разности между этим кадрами:

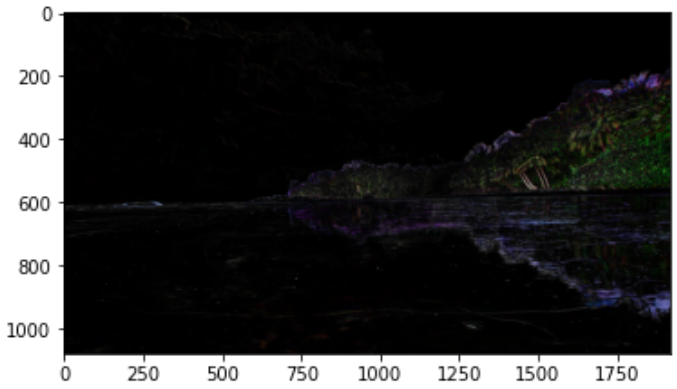


Рисунок 5.9. Модуль разности тестовых ключевых кадров видеофрагмента

Во время поиска лучшего алгоритма нахождения ключевых точек, был также рассмотрен алгоритм GoodFeaturesToTrack, позаимствованный из алгоритмов поиска оптического потока. Результаты его работы приведены ниже:

```
def good_kp(frame):
    gray = frame
    if len(frame.shape) != 2:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    corners = cv2.goodFeaturesToTrack(gray, mask=None, maxCorners=10000, qualityLevel=0.00001, minDistance=1,
    corners = np.int0(corners)
    corners = [corner.ravel() for corner in corners]
    for corner in corners:
        cv2.circle(frame, (corner[0], corner[1]), 3, [0,255,0], -1)
    cv2.imwrite('Matches.png', frame)
    return corners
```

Рисунок 5.10.Использование алгоритма поиска ключевых точек GoodFeaturesToTrack

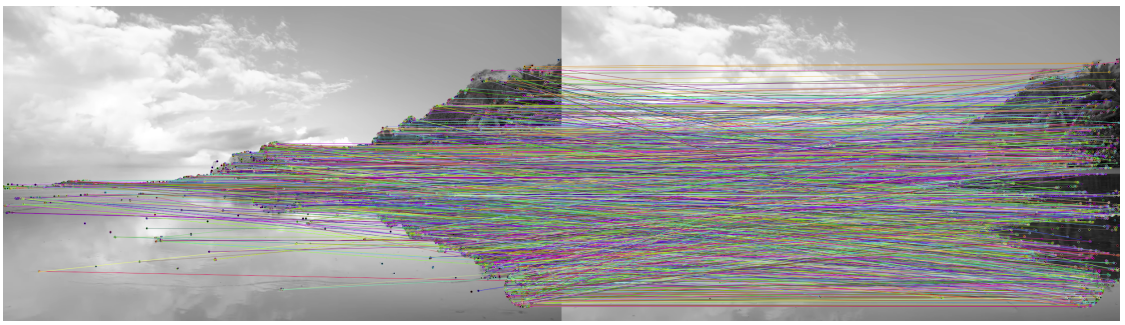


Рисунок 5.11.Использование алгоритма поиска ключевых точек GoodFeaturesToTrack на тестовых изображениях

Количество верных сопоставлений значительно меньше, чем при использовании FAST. Поэтому лучшим кандидатом для поиска ключевых точек остается FAST.

Итоговая комплектация — FAST+HOG+BFMatch.

5.2 Реализация алгоритма сжатия на основе поиска и переиспользования областей высокой энтропии

Опишем алгоритм сжатия изображения на основании полученных совпадений. Заметим, что алгоритм выбора ключевых точек положительно влияет на эффективность получаемого алгоритма, так как ключевые точки выбираются в областях с высокой энтропией, а именно эти области сложнее всего сжимаются любым алгоритмом сжатия, в том числе и JPEGом. В лучшем случае нам придется сжимать только одноцветные области и разности между кэшированными частями и исходным изображением.

Итак, выбираем лучшие совпадения дескрипторов, соответствующие ключевые точки и заменяем области вокруг этих ключевых точек на рассматриваемом изображении на разность между найденной областью и рассматриваемой. Используемые участки и разность, соответственно (размер блока вокруг ключевой точки — 64x64 пикселей):

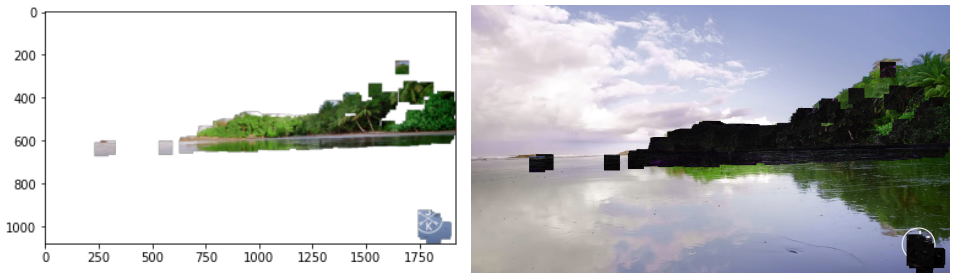


Рисунок 5.12.Выделение и удаление областей схожих с областями из предыдущего ключевого кадра

Попробуем сжать только лишь полученное изображение. Исходное изображение в формате JPEG весит 159.1 KB, разность — 147.9 KB. Лучше, но незначительно.

Вспомним, что на самом деле JPEG разбивает изображение на фрагменты по 8x8, поэтому попробуем выровнять наши получаемые части по сетке кратной 8x8. Отобразим получаемое изображение:

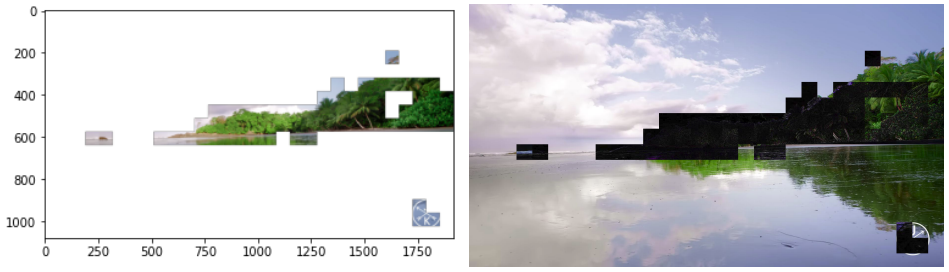


Рисунок 5.13.Выделение и удаление областей схожих с областями из предыдущего ключевого кадра с выравниванием по сетке 8x8

Полученный размер — 141.1 KB.

Теперь вспомним еще про то, что все блоки 8x8 всё же под конец обрабатываются вместе, поэтому давайте объединим разности в одно изображение, а всё остальное — в другое, тем самым расположив рядом похожие блоки 8x8. Пропорции изображения сохранить не удастся, поэтому сохранять всё будем в изображения в виде строки шириной в ширину блока, рассматриваемого вокруг ключевой точки. В текущем случае — 64.

Полученные размеры — для разностей: 37KB, для оставшихся частей изображения: 105KB. Суммарно вышло даже больше, чем было, потому что JPEG для каждого изображения сохраняет дополнительную информацию. Такую как таблицы Хаффмана, например. Однако теперь у нас есть возможность варьировать качество отдельно изображения и отдельно — разностей. Так как разности незначительны, их можно сжимать достаточно сильно без потери качества. Сохраним изображение с разностями с параметром quality=10. Получим размер разностей 7.5KB. Итого суммарный размер 7.5 KB+105 KB=112.5 KB против исходных 159.1 KB. Посмотрим на исходное и декодированное после нашего сжатия изображение соответственно:

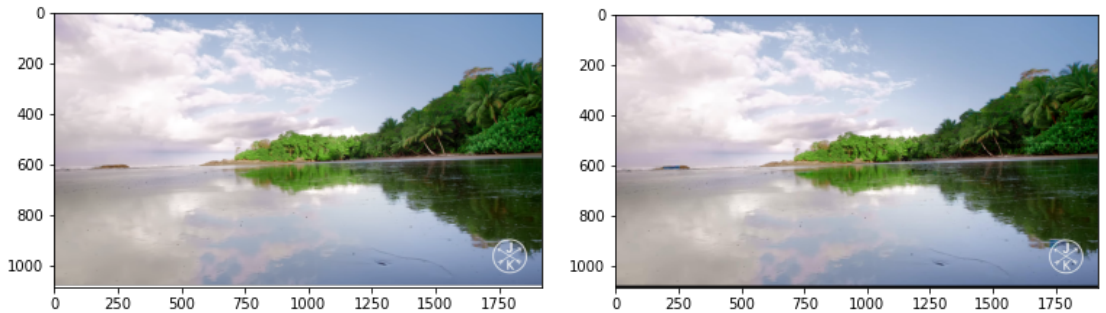


Рисунок 5.14. Слева — исходное изображение, справа — сохраненное в нашем формате

Итак, рассмотрим общую схему кодирования и декодирования с помощью приведенного выше алгоритма. Кодирование:

1. Считываем очередное изображение
2. Находим ключевые точки с помощью алгоритма FAST
3. Находим дескрипторы ключевых точек с помощью алгоритма HOG
4. Находим все соответствия между дескрипторами данного изображения и предыдущих изображений
5. Выделяем k наиболее подходящих соответствий
6. Для каждой ключевой точки, для которой нашлось соответствие, находим ближайшую точку из сетки align x align и сдвигаем рассматриваемую ключевую точку к найденной, сдвигая соответственно и точку и соответствия
7. Идем в цикле по всем блокам align x align изображения и выполняем следующее
8. Если данному блоку нашлось соответствие на предыдущих изображениях, сохраняем в массив блоков-разностей модуль разности между рассматриваемым блоком и ему соответствующим. Также отдельно сохраняем в массив знаков — знаки разностей. Это делается по причине того, что алгоритмы сжатия изображений работают с положительными числами, поэтому знак разности предпочтительно сохранять и сжимать отдельно
9. Если данному блоку не нашлось соответствие, сохраняем в массив блоков-исходников
10. Полученные массивы блоков-исходников и блоков-разностей сохраняем соответственно в два изображения, причем блоки-разности сохраняем в низком качестве.
11. Полученные знаки и словарь соответствий сохраняем в отдельный файл с lossless сжатием.

Декодирование происходит аналогично в обратном порядке.

Больше тестов можно найти в Приложении. На Рис.4 отчетливо наблюдается эффект, которого мы добивались: те изображения, которые плохо сжимаются обычным алгоритмом сжатия, то есть изображения, в которых высокий коэффициент энтропии, хорошо сжимаются разработанным алгоритмом. На Рис.5 видно, что алгоритм не справляется с изображениями, в которых нет крупных

схожих областей. Таким образом, он может быть полезен в случаях, когда имеем дело с изображениями различных размеров, в которых наблюдаются схожие фрагменты, либо с изображениями одинаковых размеров, в которых наблюдаются схожие фрагменты к примеру, видео с малым FPS.

Весь код исследований и реализацию JPEG можно найти в репозитории https://github.com/JustTohich/ImageCompressions.

Глава 6. РАЗРАБОТКА АЛГОРИТМА СЖАТИЯ НА ОСНОВЕ ЧАСТОТНЫХ ХАРАКТЕРИСТИК СЕГМЕНТОВ ИЗОБРАЖЕНИЙ

6.1 Практическая часть

В основе лежит класс, отвечающий за обучение алгоритма и сжатие изображений. При реализации клиент-серверной передачи изображений, этот класс одинаковый как для клиентской, так и для серверной стороны, причем его состояние постоянно синхронизируется (то есть если алгоритм доучился на серверной стороне, он не может начать применять новые знания, пока не узнает, что клиент тоже до них доучился и в обратную сторону так же). Процесс обучения намного дольше процесса сжатия, поэтому они запускаются параллельно. Процесс сжатия в простейшем случае происходит в два действия — взять значение вероятности по ключу изображения из словаря и закодировать его с помощью любого статистического алгоритма (арифметическое кодирование, кодирование Хаффмана). Процесс обучения заключается в обновлении вероятностей изображений. Изначально все изображения равновероятны.

Параметрами для данного класса будут:

- изначальный размер символа в байтах (во время обучения алгоритма он может меняться, так, например, может поддерживаться одновременно несколько различных размеров символов с возможностью переключаться между ними, когда происходит изменение размера коррелирующих областей изображений) (разрабатываемый алгоритм не имеет привязки к изображению в целом, он способен воспринимать как символ любую последовательность пикселей)
- доступное количество байт для хранения вероятностей
- скорость обучения (скорость адаптации)

Заметим, что при размере символа в 1 байт на черно-белых изображениях и при доступном количестве байт для хранения вероятностей достаточном для хранения 256 вероятностей, данный алгоритм превращается не во что иное, как в обычный статистический алгоритм, который используется в большинстве алгоритмов сжатия изображений. Этот факт будет использоваться для сравнения разрабатываемого алгоритма с существующими.

Также в базовой версии для сокращения количества сохраняемых вероятностей, будем сохранять только вероятности для самых вероятных изображений, а для остальных изображений будем считать вероятности одинаковыми.

Для обучения и тестирования был использован датасет CIFAR-10 из RGB изображений размера 32x32 [3]. Реализацию данного алгоритма можно найти в GoogleCollab: https://clck.ru/V4jqs

6.2 Результаты

Построенная концепция сжатия обладает следующими преимуществами:

- Универсальность с точки зрения алгоритмов сжатия: данный алгоритм построен на базе теории информации, теории множеств, формулы Шеннона и, непосредственно, определения сжатия данных без потерь. Нетрудно заметить, что из-за такого способа выведения алгоритма, он является наиболее общим алгоритмом сжатия и любой из существующих алгоритмов сжатия может быть приведен к данному алгоритму
- Теоретический потенциал качества сжатия не ниже, чем у любого из существующих алгоритмов сжатия: так как любой из алгоритмов сжатия может быть приведен к данному алгоритму. Более того, данный алгоритм претендует на наивысший возможный потенциал качества сжатия.
- Универсальность с точки зрения данных: алгоритм адаптируется под контекст и среду (формально говоря — под распределение входных данных), в которой сжимает данные, поэтому является универсальным, а именно — одинаково хорошо работает как на искусственных изображениях, так и на фотографиях природы, медицинских снимках и т.д.
- Наличие возможности манипулировать соотношением используемой памяти и качества сжатия: достигается путем повышения и понижения детализации хранения вероятностей

Также данная концепция обладает следующими потенциальными недостатками:

- Сложность реализации эффективного хранения вероятностей
- Сложность реализации адаптации алгоритма к распределению данных
- Потенциально высокая вычислительная сложность обучения

Полученная реализация алгоритма обладает следующими преимуществами:

- Высокая скорость сжатия и декомпрессии
- Эффективное хранение вероятностей на небольших (до 16 байт) размерах символов
- Динамическое изменение размера символов и, соответственно, входного алфавита

- Достижение лучших результатов, чем статистические методы без памяти

Также алгоритм в базовой версии обладает некоторыми недостатками:

- Низкая скорость обучения (высокая сложность алгоритма обучения)
- Слабая поддержка символов большого размера (при переходе на символы большого размера алгоритм оптимален только при рассмотрении небольшого количества часто повторяющихся символов)
- Как следствие пункта выше, нет возможности применить алгоритм сразу для цельного изображения, что рекомендуется в разработанной теоретической базе для достижения максимально возможного коэффициента сжатия, есть возможность лишь применить последовательно для частей изображения

Результаты, полученные с помощью данного алгоритма при варьировании параметров и тренировочных и тестовых выборок можно найти в Приложениях. Для демонстрации для обучения были выбраны 100 случайных изображений из выбранного датасета и для тестирования — другие 100 изображений. Графики показывают результаты на тестовой выборке.

Там же, в приложениях, можно заметить, что чем больше выделяется оперативной памяти для хранения модели, тем эффективнее она становится и тем дольше она обучается, и такая тенденция наблюдается до тех пор, пока размер доступной памяти не становится достаточным для сохранения вероятностей всех возможных вариантов символа.

Заметим, что при размере символа 1, мы имеем обычный статистический алгоритм. Как видно по графикам, при размере символа 2 и 4 результаты лучше, чем при размере символа 1, отсюда можно сделать закономерный вывод о том, что при достаточном количестве памяти следует брать размер символа как можно больше для достижения максимальной эффективности, а также можно сделать вывод о том, что нынешние алгоритмы сжатия можно улучшать в этом направлении.

Глава 7. УНИВЕРСАЛЬНЫЙ СЕССИОННЫЙ АЛГОРИТМ

7.1 Концепция

Как мы уже выяснили ранее, сжатие возможно только в предположении о том, что входные данные имеют не равномерное распределение. Также мы уже утвердили общую схему сжатия: преобразователь, квантователь, кодер символов.

Предположим, для простоты, что имеем дело с одномерными данными (двумерные данные несложно свести к одномерным) и преобразователь — линейная функция. В качестве квантователя рассмотрим округление до ближайшего целого. Тогда сжатие можно рассматривать как функцию вида

$$Ax = b$$
$$c = Coder([b])$$

Где x — входные данные, A — матрица преобразования, b — вектор значений, полученный после преобразования. [b] — округление каждого из значений вектора к ближайшему целому. Coder — кодировщик символов.

Свойство, которым должен обладать в целом алгоритм сжатия — возможность декодирования. А значит каждое из преобразований должно быть обратимым.

Свойство, которым должен обладать преобразователь — устойчивость к небольшим изменениям b. То есть при небольших изменениях b, x, который находится из уравнения Ax = b, также должен меняться не сильно. Эта характеристика описывается числом обусловленности матрицы, которое вычисляется по формуле:

$$\mu(A) = ||A|| * ||A^{-1}||$$

Где $||A||$- любая операторная норма матрицы. Значение числа обусловленности требуется минимизировать.

Преобразование A необходимо для обеспечения обнуления большинства последних элементов вектора, другими словами — необходимо для “уплотнения энергии”.

В качестве кодировщика символов может рассматриваться любой кодировщик, опирающийся на известные вероятности символов. В самом простом случае можно рассмотреть кодировщик, который перед записью кодируемого числа записывает, сколько бит требуется для сохранения данного числа без последних нулей. Таким образом, если имеем числа от 0 до n, количество бит, в худшем случае требуемых для сохранения одного из этих чисел:

$$[log_2(log_2(n))] + [log_2(n)]$$

Рассмотрим в общих чертах процесс декодирования:

$$b + e = Decoder(c)$$
$$x_{dec} = A^{-1}(b + e)$$

7.2 Универсальность концепции

Итак, получили общую схему для сжатия изображений. Во-первых, несложно заметить, что при правильном подборе коэффициентов A и при правильном выборе кодировщика символов, данная схема сравнима с сжатием JPEG и сжатием JPEG2000.

Для JPEG достаточно заменить матрицу A на коэффициенты дискретного косинусного преобразования:

$$A = [cos(k * (l + 1/2) * \pi/n)]_{0<=k,l<=n}$$

В качестве квантователя использовать не округление, а деление нацело на соответствующие коэффициенты из матрицы квантования.

А в качестве кодировщика символов достаточно использовать совокупность RLE и кодов Хаффмана.

В свою очередь для JPEG2000 достаточно заменить матрицу A на набор коэффициентов дискретного вейвлет-преобразования DWT-2D.

7.3 Обучение

7.3.1 Общие правила

Первым делом требуется обучить преобразование A.

Как уже писалось выше, требуется соблюдение следующих правил:

- наличие обратного преобразования к преобразованию A, то есть наличие обратной матрицы к матрице A
- небольшое значение числа обусловленности матрицы A (заметим, что при выполнении данного правила, предыдущее выполняется автоматически)
- стоимость сохранения i ненулевого коэффициента выше, чем суммарная стоимость сохранение всех 0..i-1 коэффициентов
- сохранение границ максимально возможного и минимального возможного получаемых коэффициентов в векторе b (с целью дальнейшей эффективной передачи данных коэффициентов)
- высокая скорость обучения (так как необходимо обеспечить обучение “на лету”)

7.3.2 Градиентный спуск

Для задачи обучения можно использовать градиентный спуск.

В таком случае требуется подсчитывать производную от функции оптимизации по каждому параметру. Однако производную от числа обусловленности по каждому из коэффициентов посчитать довольно проблематично.

Плюсы градиентного спуска:

- высокая скорость обучения
- низкая вычислительная сложность обучения

Минусы:

- нет возможности использовать сложные функции оптимизации

7.3.3 Генетический алгоритм

В то же время существует алгоритм обучения, который способен обучаться на любых функциях оптимизации — генетический алгоритм. Однако у генетического алгоритма также есть свои минусы.

Плюсы генетического алгоритма:

- есть возможность использовать функции оптимизации любой сложности

Минусы:

- очень низкая скорость обучения
- требуется большое число входных данных для качественного обучения
- высокая вычислительная сложность
- непредсказуемость в обучении
- слабое учитывание внутренней структуры функции оптимизации

7.3.4 Комбинация градиентного спуска и генетического алгоритма

Попробуем взять от обоих алгоритмов лучшее и реализовать алгоритм обучения, опирающийся на тот факт, что у части функции оптимизации мы можем найти производную, а у другой части — нет. Таким образом, для одной части было бы эффективно использовать градиентный спуск, а для другой — генетический алгоритм. Но так как эти части связаны через оптимизируемые нами коэффициенты, просто разделить функцию оптимизации на две нет возможности.

Вместо этого рассмотрим следующую модель:

- используем градиентный спуск для функции, для которой можно посчитать производную
- рассматриваем два набора коэффициентов (исходный и после обучения) как два генома

- между этими геномами проводим операцию скрещивания несколько раз, получая новое поколение в терминах генетических алгоритмов (таким образом, мы применяем лишь часть полученных коэффициентов, что делает данный алгоритм несколько похожим на стохастический градиентный спуск)
- используя уже полноценную функцию оптимизации выбираем из текущего поколения наборы с лучшими показателями функции оптимизации
- переходим к первому шагу с новыми входными данными

7.3.5 Результаты

Полученный алгоритм обладает следующими достоинствами:

- высокая скорость обучения
- не требуется такого же большого числа входных данных, как для генетического алгоритма
- нет главного недостатка градиентного спуска: принимаются любые функции оптимизации
- данный алгоритм учитывает внутреннюю структуру функции оптимизации везде, где это возможно

Таким образом, он полностью устраняет недостатки предыдущих алгоритмов, и обладает в наибольшей возможной степени всеми их достоинствами.

7.4 Клиент-серверная архитектура

Рассмотрим случай online обучения в контексте клиент-серверного взаимодействия. Допустим, перед нами стоит задача передать последовательность изображений (или организовать потоковую передачу изображений) от сервера клиенту.

Рассмотрим, что из себя представляет в таком случае разработанный алгоритм сжатия. Заметим, что обучение на обеих сторонах одновременно в данном случае довольно ресурсозатратно, поэтому обучаться будет только серверная сторона, а клиентская сторона будет просто получать контекст необходимый для декомпрессии — матрицу обратную к матрице A и распределение вероятностей для кодировщика символов.

Для синхронизации контекста введем понятие поколения контекста сжатия. Это численная характеристика, иллюстрирующая уровень обученности контекста сжатия. Серверная сторона при обучении повышает поколение контекста сжатия и передает контекст декомпрессии с соответствующим поколением на клиента. Клиент, в свою очередь, получив новый контекст, обновляет у себя максимально

доступное поколение и сообщает об этом серверу, который после данного сообщения может начинать присылать сообщения, закодированные с помощью контекста сжатия более нового поколения.

С целью повышения отказоустойчивости данной системы, вместе с каждым сообщением отправляет также номер поколения контекста, с помощью которого это сообщение было сжато.

7.5 Результаты

Была разработана модель обучения алгоритма сжатия на основе алгоритма градиентного спуска и генетического алгоритма, а также на основе разработанной концепции, обобщающей подходы к сжатию изображений. Была оценена эффективность данной модели по сравнению с современными популярными алгоритмами сжатия, оценен ее потенциал.

Для синхронизации контекста обучения данного алгоритма была предложена модель клиент-серверного взаимодействия, обеспечивающая корректное поведение алгоритма сжатия при параллельном обучении и обновлении контекста сжатия.

ЗАКЛЮЧЕНИЕ

Были исследованы базовые принципы сжатия данных без потерь и с потерями. На основе этих принципов была сформирована концепция универсального алгоритма сжатия данных без потерь. Данная концепция была обоснована с точки зрения теории информации и теории множеств, а также формулы Шеннона. Также было проведено глубокое исследование в области сжатия изображений, видео, в основных подходах к поиску оптического потока, компенсации движения в видео, подходах к задаче распознавания объектов. На основании полученных знаний, был проведен ряд экспериментов нацеленных на поиск наиболее подходящего метода для поиска схожих областей высокой энтропии на нескольких изображениях без поворотов и изменения размеров.

После этого был разработан алгоритм сжатия последовательности схожих изображений, оптимизирующий хранение областей высокой энтропии в изображениях, тем самым сокращая размер итогового изображения. А также базовый алгоритм, доказывающий состоятельность разработанной концепции и доказывающий наличие огромного потенциала в дальнейшей разработке алгоритмов сжатия данных.

С целью обобщения существующих подходов к сжатию изображений, был разработан универсальный алгоритм, способный обучаться входным данным, что делает его потенциально намного более эффективным, чем существующие стандартные алгоритмы сжатия. Также было доказано, что данный алгоритм способен стать сравнимым с JPEG на этапах обучения и также способен стать сравнимым с JPEG2000. Таким образом, данный алгоритм не хуже вышеперечисленных, являющихся наиболее популярными алгоритмами сжатия изображений на сегодняшний день.

Также была разработана модель клиент-серверного взаимодействия, обеспечивающая адаптирующееся под клиентские и серверные мощности обучение алгоритма сжатия.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Р. Вудс. Цифровая обработка изображений / Р. Вудс. — Москва: Техносфера, 2005;
2. Д.Сэломон. Сжатие данных, изображений и звука / Д.Сэломон. — Москва: Техносфера, 2004;
3. Сжатие на основе предсказания значений пикселей. [Электронный ресурс]. — Режим доступа: https://www.csd.uwo.ca/~melsakka/publications/journals/pdfs/2007_jvcir_Nathanael.pdf. — Дата доступа: 25.05.2021;
4. Сжатие на базе предположения схожести значений близлежащих в двумерном смысле пикселей изображения. [Электронный ресурс]. — Режим доступа: https://naun.org/main/NAUN/computers/17-679.pdf. — Дата доступа: 25.05.2021;
5. Вейвлет преобразование применительно к изображениям. [Электронный ресурс]. — Режим доступа: https://www.researchgate.net/publication/266018963_Wavelet_image_compression. — Дата доступа: 25.05.2021;
6. Стандарт JPEG. [Электронный ресурс]. — Режим доступа: https://web.stanford.edu/class/ee398a/handouts/lectures/08-JPEG.pdf. — Дата доступа: 25.05.2021;
7. Стандарт JPEG2000. [Электронный ресурс]. — Режим доступа: https://www.imaging.org/site/PDFS/Papers/2000/PICS-0-81/1645.pdf. — Дата доступа: 25.05.2021;
8. Подробное описание JPEG. [Электронный ресурс]. — Режим доступа: https://www.researchgate.net/publication/268523100_THE_JPEG_IMAGE_COMPRESSION_ALGORITHM. — Дата доступа: 25.05.2021;
9. Описание SIFT, SURF, ORB. [Электронный ресурс]. — Режим доступа: https://arxiv.org/pdf/1710.02726.pdf. — Дата доступа: 25.05.2021;
10. Описание дескриптора FREAK [Электронный ресурс]. — Режим доступа: https://www.researchgate.net/publication/258848394_FREAK_Fast_retina_keypoint . — Дата доступа: 25.05.2021;
11. База данных медицинских снимков, используемая для тестов. [Электронный ресурс]. — Режим доступа: http://www.aylward.org/notes/open-access-medical-image-repositories. — Дата доступа: 25.05.2021;

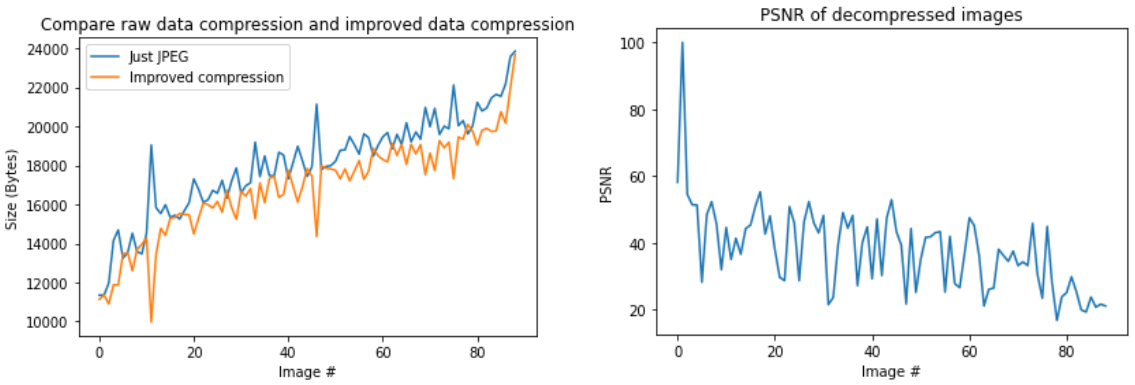
12. CIFAR-10 [Электронный ресурс]. — Режим доступа: https://www.cs.toronto.edu/~kriz/cifar.html. — Дата доступа: 25.05.2021;
13. Semantic Perceptual Image Compression using Deep Convolution Networks [Электронный ресурс]. — Режим доступа: https://arxiv.org/pdf/1612.08712v2.pdf. — Дата доступа: 25.05.2021;
14. An End-to-End Compression Framework Based on Convolutional Neural Networks [Электронный ресурс]. — Режим доступа: https://arxiv.org/pdf/1708.00838v1.pdf — Дата доступа: 25.05.2021;
15. Practical Full Resolution Learned Lossless Image Compression [Электронный ресурс]. — Режим доступа: https://openaccess.thecvf.com/content_CVPR_2019/papers/Mentzer_Practical_Full_Resolution_Learned_Lossless_Image_Compression_CVPR_2019_paper.pdf. — Дата доступа: 25.05.2021.

ПРИЛОЖЕНИЯ

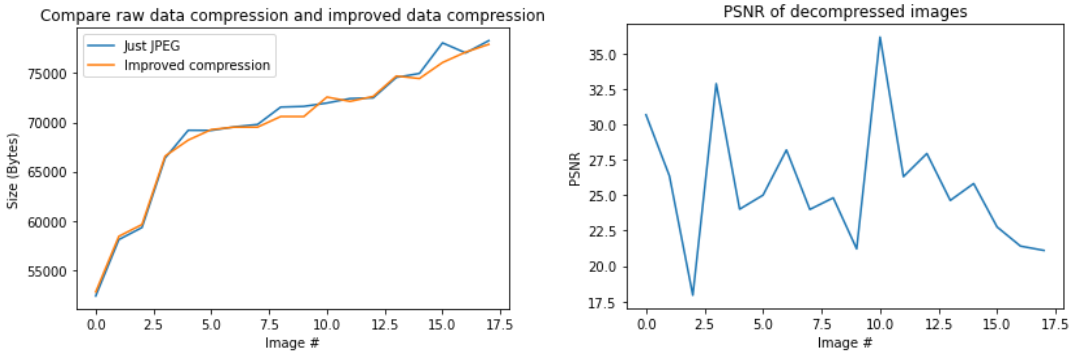
Приложение. 1 Используемое для тестов изображение Lenna.png

Приложение. 2 Кадр из видео Costa Rica 4K используемый для тестов.

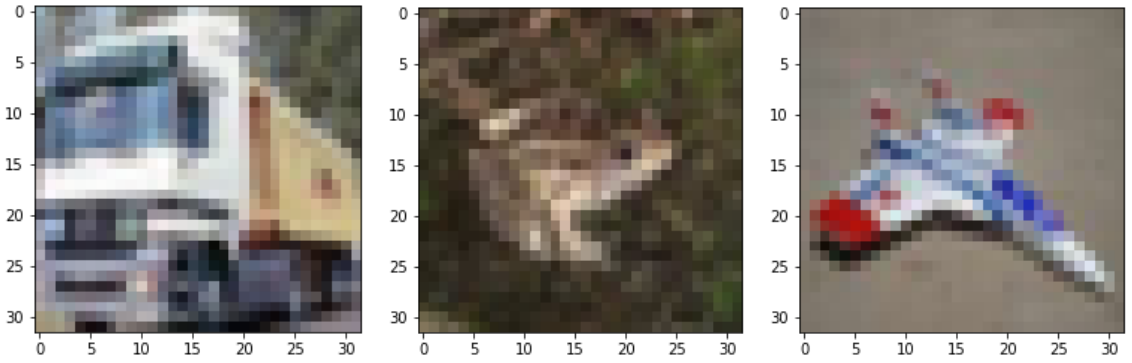
Приложение. 3 Изображения из тестового датасета медицинских снимков, используемые для тестов



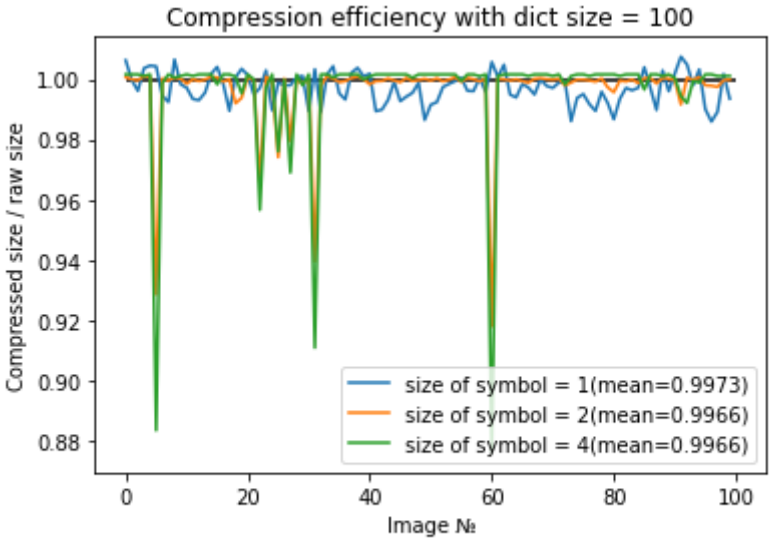
Приложение. 4 Сортированные по итоговым размерам изображений результаты тестов для операции вырезания части изображения из тестового датасета и поиска его на исходном изображении



Приложение. 5 Сортированные по итоговым размерам изображений результаты тестов для соседних изображений из тестового датасета

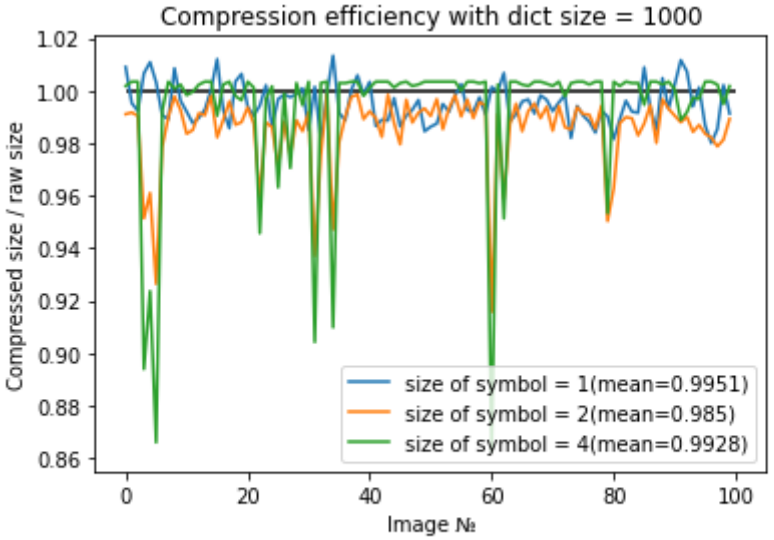


Приложение. 6 Пример изображений из датасета CIFAR-10




```
CPU times: user 16.1 s, sys: 10 ms, total: 16.1 s
Wall time: 16.1 s
```

Приложение. 7 Размер сжатого файла относительно размера исходного изображения для размера словаря 100




```
CPU times: user 1min 17s, sys: 16 ms, total: 1min 17s
Wall time: 1min 17s
```

Приложение. 8 Размер сжатого файла относительно размера исходного изображения для размера словаря 1000


```
CPU times: user 10min 6s, sys: 62 ms, total: 10min 6s
Wall time: 10min 6s
```

Приложение. 9 Размер сжатого файла относительно размера исходного изображения для размера словаря 10000