Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Высшая школа программной инженерии

Работа допущена к защите
Директор ВШПИ
_____П.Д.Дробинцев
«___»_____2025 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

работа бакалавра

РЕАЛИЗАЦИЯ ИНСТРУМЕНТА СЖАТИЯ ФАЙЛОВ НА ОСНОВЕ ОСОБЕННОСТЕЙ ЦВЕТОВОСПРИЯТИЯ

по направлению подготовки (специальности)

09.03.04 Программная инженерия

Направленность (профиль)

09.03.04_01 Технология разработки и сопровождения качественного программного продукта

Выполнил студент гр. 5130904/10102 — Д.Ф.Набиуллин

Руководитель, старший преподаватель — О.В.Александрова

Консультант по нормоконтролю — Е.Г.Локшина

Санкт-Петербург
2025

Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

УТВЕРЖДАЮ

Директор ВШПИ

П. Д. Дробинцев

«__» _____ 2025 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студенту Набиуллину Данису Фирдусович, группа 5130904/10102

1. Тема работы: Реализация инструмента сжатия файлов на основе особенностей цветовосприятия

2. Срок сдачи студентом законченной работы: 17.05.2025

3. Исходные данные по работе:
   - Техническое задание
   - Документация на язык Java

4. Содержание работы (перечень подлежащих разработке вопросов):
   - Обзор существующих программ для сжатия файлов.
   - Представление сравнительного анализа найденных программ для работы с изображениями.
   - Разработка инструмента для сжатия файлов.
   - Применение инструмента для демонстрации эффективности.

5. Перечень графического материала (с указанием обязательных чертежей):

6. Перечень используемых информационных технологий, в том числе программное обеспечение, облачные сервисы, базы данных и прочие сквозные цифровые технологии (при наличии): Microsoft Windows, Microsoft Word, JetBrains IntelliJ IDEA, Git, Java

7. Консультанты по работе: отсутствуют

8. Дата выдачи задания: 14.04.2025

Руководитель ВКР _____ Александрова О.В.
(подпись)

Задание принял к исполнению 14.04.2025

Студент _____ Набиуллин Д.Ф.
(подпись)

РЕФЕРАТ

На 44 с., 10 рисунков, 2 таблицы, 0 приложений.

СЖАТИЕ ФАЙЛОВ, ДИХРОМАЗИЯ, ЦВЕТОВОСПРИЯТИЕ, ДАЛЬТОНИЗМ, АЛГОРИТМЫ СЖАТИЯ, ОПТИМИЗАЦИЯ ИЗОБРАЖЕНИЙ, ЭФФЕКТИВНОСТЬ СЖАТИЯ

Тема выпускной квалификационной работы — «Реализация инструмента сжатия файлов на основе особенностей цветовосприятия».

Данная работа посвящена разработке метода сжатия изображений с потерями, который основан на особенностях цветовосприятия людей с дихромазией.

Задачи, которые были решены в процессе работы: анализ существующих решений; разработка подхода к оптимизации изображений на основе моделирования дихроматического зрения; реализация программного инструмента на языке Java; проверка его эффективности.

В результате была разработана консольная утилита на языке Java, которая выполняет сжатие изображений. Инструмент удаляет избыточные цветовые данные, неразличимые для людей с дихромазией, что приводит к уменьшению итогового размера файла.

ABSTRACT

44 pages, 10 figures, 2 tables, 0 appendices.

FILE COMPRESSION, DICHROMACY, COLOR PERCEPTION, COLOR BLINDNESS, COMPRESSION ALGORITHMS, IMAGE OPTIMIZATION, COMPRESSION EFFICIENCY

The topic of the graduate qualification work is "Implementation of a file compression tool based on the specifics of color perception."

This work is dedicated to the development of a lossy image compression method based on the specifics of color perception in people with dichromacy.

Tasks solved during the work: analysis of existing solutions; development of an image optimization approach based on dichromatic vision simulation; implementation of a software tool in Java; verification of its effectiveness.

As a result, a command-line utility in Java was developed that performs image compression. The tool removes redundant color data, imperceptible to people with dichromacy, which leads to a reduction in the final file size.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

7Z – формат сжатия файлов в архив, поддерживающий несколько различных алгоритмов сжатия, шифрования и предварительной обработки данных

AES-256 – симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES

BT.709 – это стандарт, разработанный для кодирования изображения и характеристик сигнала телевидения высокой чёткости

Brettel 1997 – Brettel H., Viénot F., Mollon J. D. Computerized simulation of color appearance for dichromats // Journal of the Optical Society of America. A, Optics, Image Science, and Vision. - 1997. - Vol. 14, No. 10. - P. 2647–2655

CIE XYZ – одно из первых математически определённых цветовых пространств, созданное Международной комиссией по освещению (CIE) в 1931 году

GZ – утилита сжатия и восстановления (декомпрессии) файлов, использующая алгоритм Deflate

HDR – технологии работы с изображениями и видео, диапазон яркости которых превышает возможности стандартных технологий

ISO – образ оптического диска, содержащего файловую систему (обычно стандарта ISO 9660 с его расширениями или UDF)

JPEG - один из популярных растровых графических форматов, применяемый для хранения фотографий и подобных им изображений

LMS – цветовое пространство, представляющее собой отклики трёх типов колбочек

RAR – проприетарный формат сжатых данных и условно-бесплатная программа-архиватор

RGB – аддитивная цветовая модель, описывающая способ кодирования цвета для цветовоспроизведения с помощью трёх цветов, которые принято называть основными

SFX – файл, компьютерная программа, объединяющая в себе архив и исполняемый код для его распаковки

sRGB – является стандартом представления цветового спектра с использованием модели RGB

TAR – это команда командной оболочки для объединения нескольких компьютерных файлов в один архивный файл

UX – восприятие и ответные действия пользователя, возникающие в результате использования и/или предстоящего использования продукции, системы или услуги

Vienot 1999 – Viénot F., Brettel H., Mollon J. D. Digital video colourmaps for checking the legibility of displays by dichromats // Color Research & Application. - 1999. - Vol. 24, No. 4. - P. 243–252

ZIP – формат архивации файлов и сжатия данных без потерь

Дейтеранопия – отсутствие зрительного пигмента, ответственного за распознавание зелёного цвета

Мб - мегабайт

Протанопия – отсутствие зрительного пигмента, ответственного за распознавание красного цвета

Тританопия – отсутствие зрительного пигмента, ответственного за распознавание синего цвета

ЦОД – центр обработки данных

ВВЕДЕНИЕ

Выпускная квалификационная работа посвящена разработке инструмента для сжатия файлов, который будет обеспечивать эффективное уменьшение размеров данных с потерей информации. В условиях быстро развивающихся технологий и увеличения объёмов цифровых данных, проблема оптимизации хранения и передачи информации становится особенно актуальной. Это и является основной причиной данного проекта.

В процессе работы следует рассмотреть существующие программы, такие как 7-Zip, PeaZip и WinRAR.

Ожидается, что результаты работы могут быть использованы для выполнения различных задач, таких как архивирование данных, уменьшение времени загрузки и передачи файлов, а также в системах резервного копирования.

В первую очередь на основании необходимости экономии трафика, передаваемого пользователю, например, при передаче фото в социальной сети, стало необходимо разработать решение.

Таким образом, целью данной работы является реализация инструментального средства, которое способно снизить размер изображений за счёт создания подхода оптимизации цветов взяв за основу восприятие цвета людей с дихромазией.

Для достижения данной цели были поставлены следующие задачи:

1. Обзор существующих программ по сжатию файлов.
2. Предложение подхода сжатия файлов для оптимизации цвета изображений.
3. Реализация данного подхода в рамках обработки рисунков, путём трансформаций цветовой модели, использования матриц преобразований и других методов.
4. Исследование и анализ процесса сжатия файлов по сравнению с архиваторами - аналогами.

Каждый день человечество генерирует порядка 330 млн терабайт данных. Хотя по оценкам экспертов Google всего 10% из них являются свежими и оригинальными, даже копии копий нужно где-то хранить. И эта задача имеет ряд нюансов. Здесь уместно провести аналогию с известным транспортным парадоксом: чем больше дорог строится, тем больше образуется автомобилей, чтобы заполнить их (постулат Льюиса — Могриджа) [1].

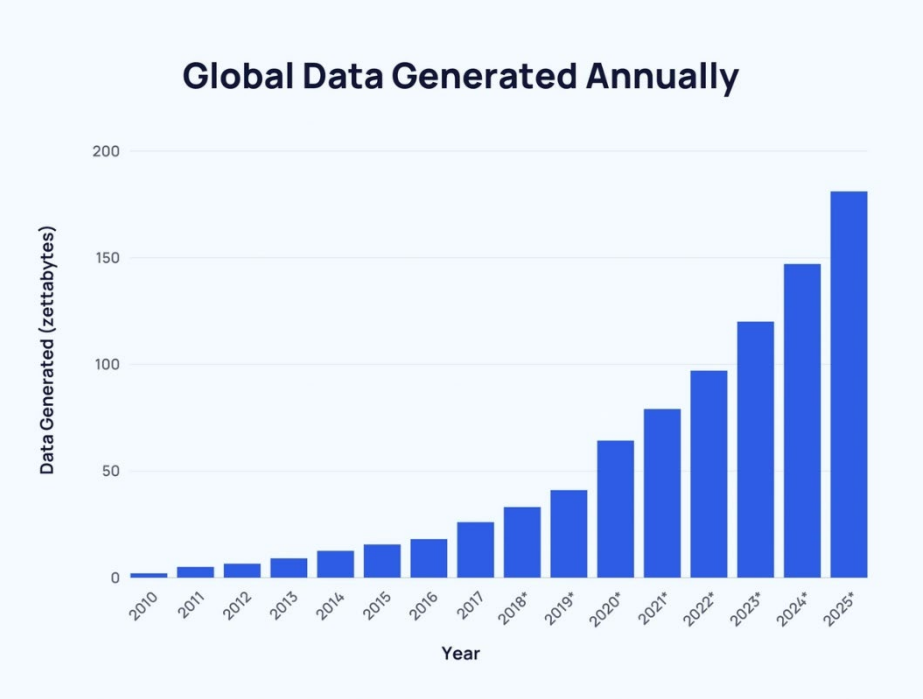

Рисунок 1. Объём сгенерированных данных по годам

Несмотря на то, что стоимость накопителей данных постоянно снижается, к примеру 1 терабайт жёсткого диска в 2013-ом году стоил 36 долларов, а 1 терабайт жёсткого диска в 2023-ом году уже 13 долларов [2], количество данных растёт гораздо быстрее, чем снижается цена на хранилища для них. Если в 2013-ом году было создано 9 зеттабайт (10^{21} байт) информации, то в 2023-ом уже 120 зеттабайт, рост более чем в 13 раз [3].

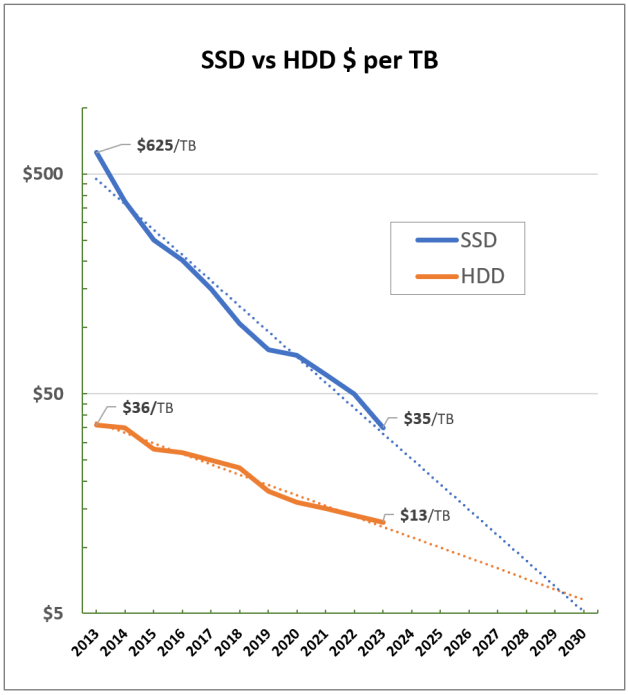

Рисунок 2. Стоимость терабайта SSD и HDD накопителей

Согласно отчёту Комиссии по международной торговле США, в мире насчитывается более 8 000 центров обработки данных. В Европе абсолютным лидером по количеству дата-центров являются Нидерланды. ЦОДов в стране настолько много, что в прошлом году правительство ввело мораторий на запуск новых площадок. Дело в том, что плохо контролируемое строительство становится яблоком раздора — местные жители были крайне обеспокоены тем, что для охлаждения серверов использовалась питьевая вода. Более того, многие активисты подняли вопрос о чрезмерном потреблении электроэнергии центрами обработки данных.

Дата-центры потребляют порядка 4% мировой электроэнергии. В попытках сократить влияние на окружающую среду и сэкономить на счетах за электричество операторы ЦОД внедряют новые технологии охлаждения, пересматривают архитектуру машинных залов. Однако оптимизировать инфраструктуру можно не только на аппаратном, но и программном уровне, изменяя подходы к хранению и работе с данными [1].

Актуальность проблемы сжатия файлов в условиях современного цифрового мира сложно переоценить, в частности, можно выделить эти пункты:

1. Неуклонный рост объёмов визуального контента. Ежедневно генерируются петабайты изображений и видео, что создаёт колоссальную нагрузку на сетевую инфраструктуру и системы хранения данных, требуя новых, более интеллектуальных подходов к оптимизации, особенно в популярных онлайн-сервисах и социальных сетях.
2. Ограниченность универсальных алгоритмов сжатия. Стандартные архиваторы (7-Zip, WinRAR и др.) и их алгоритмы (DEFLATE, LZMA) эффективно работают с повторяющимися последовательностями байт, но не способны распознать и устранить перцептуальную избыточность в изображениях — ту часть цветовой информации, которая неразличима для человеческого глаза.
3. Потенциал использования особенностей цветовосприятия. Существование различных форм цветовосприятия, в частности дихромазии, открывает возможность для разработки нишевых методов сжатия. Удаление цветовой информации, невидимой для определённой группы пользователей, является незадействованным резервом для уменьшения размера файлов без субъективной потери качества для этой аудитории.
4. Повышение пользовательского опыта (UX). В условиях ограниченных сетевых ресурсов, например, при использовании мобильного интернета, уменьшение размера медиафайлов напрямую влияет на скорость их загрузки. Оптимизация изображений с учётом особенностей зрения ведёт к ускорению работы приложений и веб-сайтов, что является важным фактором удовлетворённости конечного пользователя.

В современном мире, где объёмы генерируемых и хранимых данных растут экспоненциально, а онлайн-сервисы и социальные сети стали неотъемлемой частью жизни, потребность в эффективных методах хранения и

передачи информации становится критически важной задачей. Оптимизация рабочих процессов, особенно связанных с медиаконтентом, приобретает особую значимость как для крупных организаций, так и для рядовых пользователей.

Сжатие данных традиционно подразделяется на две основные категории: сжатие без потерь и сжатие с потерями. Если первая категория гарантирует полное восстановление исходных данных и незаменима для архивации документов или программного кода, то вторая позволяет достичь гораздо большей степени сжатия за счёт отбрасывания избыточной, малозаметной для человека информации [4].

Именно на принципах сжатия с потерями и сфокусирована данная работа. Подобно тому, как популярный формат MP3 отбрасывает звуковые частоты, которые человеческое ухо не воспринимает, предлагаемый подход использует особенности человеческого зрения для оптимизации изображений. Ключевая идея заключается в использовании феномена дихромазии (одной из форм дальтонизма), при котором человек не различает определённые цветовые оттенки. Эта «перцептуальная избыточность» — цветовая информация, невидимая для целевой аудитории — может быть безболезненно удалена, что приведёт к существенному уменьшению размера файла без видимой потери качества для дихроматов.

Таким образом, цель данной работы — это не просто теоретическое исследование, а разработка и реализация конкретного программного инструмента. Этот инструмент должен выполнять функцию интеллектуального препроцессора для изображений, который, основываясь на модели цветовосприятия людей с дихромазией, оптимизирует их для дальнейшего хранения или передачи.

Для последовательного достижения поставленной цели структура данной выпускной квалификационной работы выстроена следующим образом.

В первой главе будет проведён анализ существующих универсальных решений в области сжатия данных и архивации. Мы рассмотрим их алгоритмы

и оценим их эффективность на нашем тестовом наборе изображений, чтобы создать отправную точку для сравнения.

Во второй главе будет подробно описана архитектура предлагаемого инструмента. Мы обоснуем выбор алгоритмов, лежащих в основе симуляции цветовой слепоты, рассмотрим математический аппарат преобразования цветовых пространств (из sRGB в LMS и обратно) и выбранные технологии для реализации.

Третья глава будет полностью посвящена деталям программной реализации. В ней будут описаны ключевые классы, структура проекта и логика работы основных модулей программы, реализованной на языке Java.

В четвертой главе мы представим и проанализируем результаты работы нашего инструмента. Будет продемонстрировано снижение размера изображений и проведено повторное сравнение с универсальными архиваторами, чтобы оценить эффективность предложенного метода на практике.

В заключении будут подведены итоги проделанной работы, сформулированы выводы и намечены возможные пути для дальнейшего развития проекта.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

1.1. Сжатие данных

В современном цифровом мире проблема хранения и передачи информации стоит как никогда остро. Ежедневно генерируются колоссальные объёмы данных, и львиную долю этого потока составляет визуальный контент — фотографии и видео. Для социальных сетей, облачных хранилищ и любых онлайн-сервисов эффективное управление этими данными становится ключевой задачей. Именно здесь на сцену выходит сжатие данных — фундаментальная технология, без которой невозможно представить функционирование современного интернета.

В целом, все подходы к сжатию можно разделить на две большие категории. Первая — сжатие без потерь (lossless). Её главный принцип — возможность восстановить исходный файл бит в бит. Это критически важно для программного кода, текстовых документов или баз данных, где любая ошибка недопустима. Вторая категория — сжатие с потерями (lossy). Здесь мы сознательно жертвуем частью информации, которая считается избыточной или малозаметной для человека, чтобы добиться гораздо большей степени сжатия. Классический пример — аудиоформат MP3, который «вырезает» звуковые частоты, неразличимые для человеческого уха.

Именно второй подход, основанный на особенностях восприятия, лёг в основу данной работы. Стандартные архиваторы, которые мы рассмотрим далее, великолепно справляются с поиском математических закономерностей и повторяющихся последовательностей байтов, но они не способны анализировать данные с точки зрения человеческой физиологии. Они не знают, какую часть цветовой информации в изображении человек с определённым типом цветовосприятия просто не увидит.

Поэтому, прежде чем перейти к детальному описанию нашего собственного метода, необходимо создать отправную точку для сравнения. В этой главе мы рассмотрим популярные универсальные инструменты для сжатия файлов — 7-Zip, PeaZip и WinRAR. Наша цель — не просто описать их

функциональность, а оценить, насколько эффективно их стандартные алгоритмы (такие как LZMA и Deflate) справляются со сжатием изображений, которые и являются объектом нашего исследования. Это позволит нам продемонстрировать, какую дополнительную выгоду может принести специализированный подход, учитывающий особенности цветовосприятия.

1.2. 7-Zip

Рассматривая существующие решения в области архивации данных, нельзя обойти стороной 7-Zip. Эта программа, разработанная российским программистом Игорем Павловым, стала своего рода отраслевым стандартом в мире бесплатного и открытого программного обеспечения для сжатия файлов. С момента своего первого релиза в 1999 году 7-Zip завоевал популярность благодаря своей эффективности и простоте, что делает его отличной отправной точкой для нашего сравнительного анализа.

Ключевым фактором успеха 7-Zip является его собственный формат 7z. В его основе лежат высокоэффективные алгоритмы сжатия LZMA и его усовершенствованная версия LZMA2. Именно благодаря им формат 7z часто демонстрирует более высокую степень сжатия по сравнению с традиционными форматами, такими как ZIP. Важно подчеркнуть, что эти алгоритмы реализуют сжатие без потерь (lossless), что является критичным для сохранения целостности программного кода или документов, но, как мы покажем далее, не является оптимальным подходом для медиафайлов, где можно использовать перцептуальную избыточность.

С точки зрения инженера, 7-Zip представляет интерес не только за счёт высокой степени сжатия. Программа универсальна: она поддерживает множество форматов архивов, включая ZIP, RAR, TAR, GZ, ISO и другие. Это позволяет работать с различными архивами без необходимости установки дополнительных утилит.

Важным аспектом является кроссплатформенность — наличие официальных сборок для Windows и неофициальных для Linux и macOS, а также наличие интерфейса командной строки (CLI). Именно CLI открывает

возможности для автоматизации процессов архивации и распаковки в скриптах, что является неотъемлемой частью многих инженерных задач. Кроме того, программа поддерживает надёжное шифрование по алгоритму AES-256, обеспечивая безопасность конфиденциальных данных.

Таким образом, 7-Zip представляет собой мощный и универсальный инструмент для работы с архивами общего назначения. Его алгоритмы эффективно справляются с поиском математических закономерностей в последовательностях байт, однако они не анализируют семантическое содержание файлов. В контексте нашей работы это означает, что 7-Zip обрабатывает изображение как произвольный набор данных, не имея возможности учесть особенности человеческого цветовосприятия [5].

1.3. PeaZip

В качестве ещё одного значимого представителя универсальных архиваторов, продолжающего наш обзор, рассмотрим PeaZip. Эта бесплатная программа с открытым исходным кодом, разработанная Габриэле Пагано, заслужила признание в сообществе благодаря своей гибкости и впечатляющей поддержке более 180 форматов архивов. Это делает её одним из наиболее универсальных решений на рынке.

С инженерной точки зрения, важным преимуществом PeaZip является наличие как графического интерфейса для рядовых пользователей, так и мощного интерфейса командной строки (CLI). Последний, как и в случае с 7-Zip, критически важен для автоматизации процессов архивации в скриптах и сложных системах обработки данных, что является неотъемлемой частью многих инженерных задач.

Однако, при всех своих достоинствах, PeaZip разделяет фундаментальное ограничение с другими программами этого класса. Его алгоритмы сжатия ориентированы на поиск математических закономерностей и повторяющихся последовательностей байтов в файле, но они абсолютно агностичны к содержимому. Они обрабатывают изображение как абстрактный набор данных, не имея возможности анализировать его семантическое

содержание. Для PeaZip, как и для 7-Zip, нет разницы между фотографией, текстовым документом или исполняемым файлом — всё это лишь поток байтов.

Именно эта «семантическая слепота» не позволяет PeaZip учитывать особенности человеческого цветовосприятия. Инструмент не способен выявить и устранить перцептуальную избыточность в изображениях, что и является отправной точкой для нашего исследования и разработки специализированного подхода, описанного в последующих главах. Таким образом, PeaZip, будучи мощным универсальным архиватором, также включается в наш сравнительный анализ как эталонное решение, работающее на принципах, отличных от предлагаемого нами метода [6].

1.4. WinRAR

Завершая обзор универсальных архиваторов, нельзя не упомянуть WinRAR — один из старейших и наиболее известных инструментов в этой области, разработанный компанией RARLAB. Появившись в 1995 году, он стал для многих пользователей синонимом сжатия данных благодаря своему проприетарному формату RAR, который зачастую обеспечивает высокую степень компрессии.

С точки зрения нашего исследования, ключевой особенностью WinRAR, как и у рассмотренных ранее 7-Zip и PeaZip, является его ориентация на сжатие без потерь. Его алгоритмы направлены на поиск и устранение математической избыточности в последовательности байтов. Это делает его эффективным для архивации документов или программного кода, но одновременно демонстрирует его «семантическую слепоту» при работе с медиафайлами. Для WinRAR изображение — это лишь произвольный поток данных, а не структура, обладающая перцептуальными характеристиками, которые можно оптимизировать. Инструмент не способен анализировать визуальную информацию и удалять те цветовые данные, которые неразличимы для определённой группы пользователей, что является ядром нашего подхода.

Как и другие архиваторы, WinRAR предоставляет широкий набор функций: поддержка множества форматов, шифрование AES-256 и создание самораспаковывающихся архивов (SFX). Для инженера особый интерес представляет наличие интерфейса командной строки (CLI), позволяющего автоматизировать процессы архивации в скриптах.

Таким образом, мы включаем WinRAR в наш сравнительный анализ в качестве третьего эталонного решения. Он представляет собой мощный инструмент общего назначения, работающий на принципах, кардинально отличных от предлагаемого нами метода. Сравнение с ним в последующих главах позволит продемонстрировать, какую дополнительную выгоду даёт специализированный подход, учитывающий особенности физиологии человеческого зрения [7].

1.5. Сравнение архиваторов

Для сравнения представленных выше инструментов сжатия файлов необходимо подготовить данные, при помощи которых мы могли бы сравнить их между собой, а также, после реализации своего собственного инструмента, и с ним самим. Поскольку только основываясь на одних и тех же файлах мы можем показать эффективность работы своего собственного продукта.

В рамках поиска фотографий было решено обратиться на официальный сайт Санкт-Петербургского политехнического университета Петра Великого [8], где в разделе фотогалерея из папки «Кампус и студенты» были выбраны случайные 30 фотографий – по 10 для тестирования каждого вида симуляции, протанопии, дейтеранопии и тританопии суммарным объёмом 51,7 мегабайт.

Для чистоты эксперимента на один и тот же компьютер под управлением операционной системы Windows 11 64-bit (Версия 10.0.26100 Сборка 26100) [9] были установлены все 3 программы сжатия – 7-Zip, PeaZip и WinRAR. Далее все фотографии были единожды пересохранены без каких-либо изменений. Это было сделано потому, что формат JPEG предусматривает собой сжатие с потерями, тем самым для объективности необходимо было единожды его совершить, чтобы была возможность в дальнейшем сравнение с

предложенным инструментом, поскольку тот подразумевает пересохранение файла в процессе редактирования. В результате получилось 18,1 мегабайта.

Далее при помощи каждой из программ сжатия 30 фотографий были сжаты и далее был измерен размер архива с ними. Настройки сжатия были оставлены по умолчанию. 7-Zip показал результат в 17,4 мегабайт. PeaZip – 17,6 мегабайт. WinRAR также достиг результата предыдущего архиватора – 17,5 мегабайт.

Таблица 1. Результаты сравнений программ сжатия

Программа	Размер архива с файлами после сжатия, Мб	Процент сжатия файлов после архиватора, %
7-Zip	17,4	96,1
PeaZip	17,6	97,2
WinRAR	17,5	96,7

1.6. Выводы

Проведённый в этой главе анализ существующих архиваторов позволил нам установить базовый уровень эффективности для стандартных методов сжатия. Результаты, полученные с помощью 7-Zip, PeaZip и WinRAR, оказались достаточно скромными: на нашем наборе тестовых JPEG-файлов удалось добиться сокращения объёма данных в среднем лишь на 3–4%.

Этот результат не является критикой данных программ, но демонстрирует фундаментальное ограничение их подхода. Универсальные архиваторы используют lossless-алгоритмы, которые ищут математические закономерности в потоке байт. Они «слепы» к содержимому файла и не способны анализировать изображение с точки зрения человеческой физиологии. Их задача — гарантировать побитовое восстановление данных, а не оптимизировать их с учётом особенностей восприятия.

Именно здесь и кроется ключевая возможность для нашего исследования. Существует пласт перцептуальной избыточности — цветовой информации, которая неразличима для определённой группы людей, — который стандартные инструменты просто не могут обнаружить и устранить.

Таким образом, мы видим, что существует потенциал для разработки инструмента, работающего на совершенно ином принципе. Цель данной работы — не конкурировать с универсальными архиваторами, а создать и исследовать специализированный метод сжатия с потерями, который нацелен именно на устранение перцептуальной избыточности. Проведённый анализ лишь подтверждает, что эта ниша не занята, и мотивирует разработку нашего собственного решения, которому и будут посвящены последующие главы.

ГЛАВА 2. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО ИНСТРУМЕНТА ДЛЯ СЖАТИЯ ФАЙЛОВ

2.1. Алгоритм сжатия

Восприятие цвета человеком достигается с помощью колбочек в сетчатке. У людей с нормальным зрением есть 3 типа клеток, чувствительных к разным длинам волн света. Колбочки L улавливают длинные волны (~красный), колбочки M улавливают средние волны (~зелёный), а колбочки S улавливают короткие волны (~синий).

Большинство нарушений цветового зрения можно объяснить тем, что один тип колбочек ведёт себя неправильно. У протанопов, дейтеранопов и тританопов соответственно отсутствуют или неисправны колбочки L, M или S.

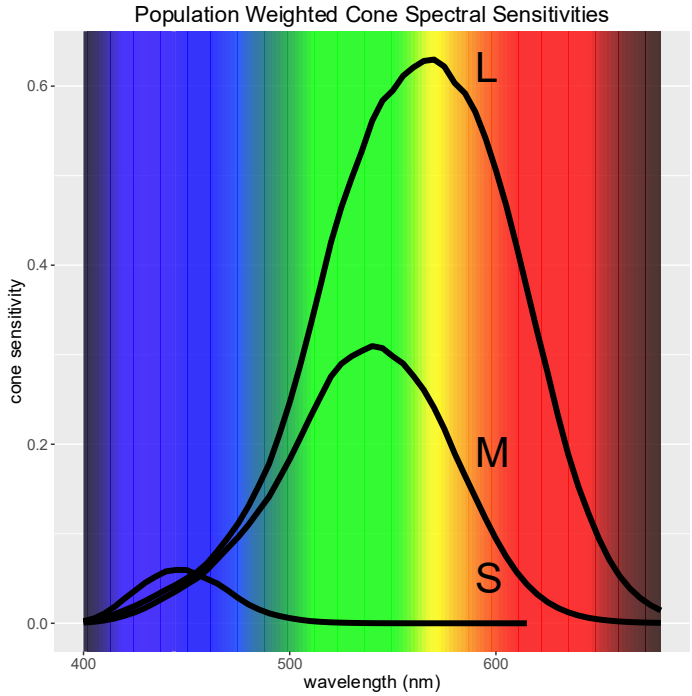

Рисунок 3. График чувствительности колбочек к разным длинам волн света

Моделирование цветовой слепоты на основе физиологических экспериментов обычно заключается в преобразовании изображения в цветовое пространство, где влияние каждого вида колбочек явно и может быть легко уменьшено или удалено. Цветовое пространство LMS было разработано специально для соответствия реакциям человеческих колбочек и, таким образом, является выбором подавляющего большинства методов [10]. Таким

образом, типичный алгоритм состоит в преобразовании изображения RGB в LMS, применении там моделирования и возврате к RGB.

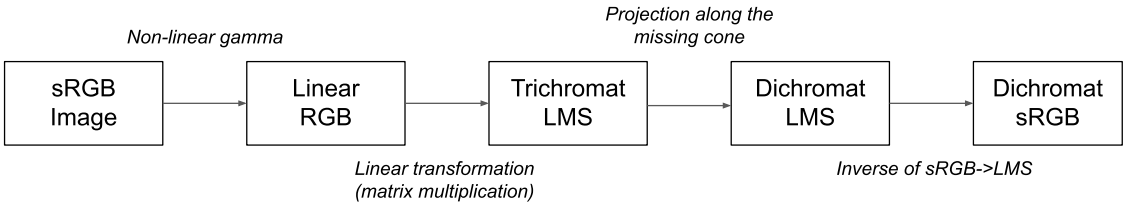

Рисунок 4. Архитектура решения

Большинство цифровых изображений кодируются тремя значениями на пиксель: красным, зелёным и синим. Но часто забывают о том, что этот "RGB" обычно соответствует стандарту sRGB, который точно определяет, как эти значения следует интерпретировать. sRGB не является линейным пространством, поскольку он пытается компенсировать нелинейную гамма-функцию, применяемый мониторами, так, что изображение, полученное цифровой камерой и затем отображённое на мониторе, будет выглядеть как исходник.

Гамма-функция также обусловлена особенностями человеческого зрения. Оно воспринимает цвета линейно, но при этом яркость нелинейно. Изменения темных оттенков более заметно, чем светлых. В связи с этим цифровые изображения хранятся в нелинейном виде, представляя больший объём памяти на тёмные участки, чем на светлые. То есть изначально записанная на камеру линейная яркость преобразуется посредством гамма-коррекции в файл с нелинейной яркостью. Но мониторы в свою очередь показывают яркость линейно, в связи с этим применяется процедура, обратная гамма-коррекции [11].

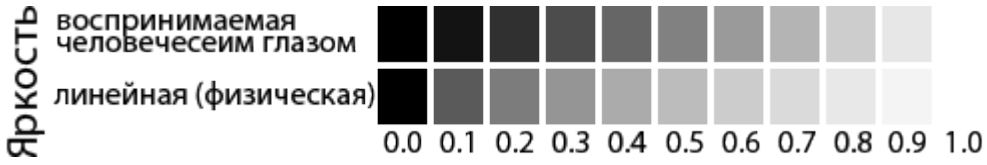

Рисунок 5. Различие линейной и нелинейной яркости

После устранения этой нелинейности мы будем называть цветовое пространство RGB цветовым пространством linearRGB. Преобразование примерно соответствует применению показателя гаммы, равного 2,4, к

каждому каналу, со специальным линейным наклоном для малых значений. Таким образом, в отличие от всех других преобразований цвета в рамках этой работы, оно нелинейное и не может быть выражено в виде матрицы. В рамках реализации мы используем функцию, указанную в стандарте sRGB [12].

$$R = \begin{cases} R'/12.92, & R' \leq 0.04045 \\ \left(\frac{R' + 0.055}{1.055}\right)^{2.4}, & R' > 0.04045 \end{cases}$$

Рисунок 6. Функция преобразования из sRGB в linearRGB

$$R' = \begin{cases} 12.92R, & R \leq 0.0031308 \\ (1.055)R^{1/2.4} - 0.055, & R > 0.0031308 \end{cases}$$

Рисунок 7. Функция преобразования из linearRGB в sRGB

Поскольку и Brettel 1997 [13], и Vienot 1999 [14] были опубликованы до того, как стандарт sRGB стал общепринятым, в своей статье они используют другую модель. Brettel 1997 откалибровали электронно-лучевой монитор со спектрорадиометром для оценки спектрального распределения первичных сигналов, в то время как Vienot 1999 использовали постоянную гамму 2,2 для преобразования в линейную RGB. К счастью, в этом случае они использовали ту же цветность BT.709, что и стандарт sRGB, поэтому его не нужно адаптировать.

До сих пор мы упоминали только цветовое пространство LMS, говоря, что мы просто перейдём от linearRGB к LMS. Но чтобы разобраться как именно будет происходит преобразование, необходимо рассказать о промежуточном шаге - цветовое пространство CIE XYZ. Причина этого историческая: это цветовое пространство было стандартизировано в 1931 году и с тех пор используется в качестве основного пространства для научных исследований в области цвета. Основная причина заключается в том, что он может отображать практически весь спектр цветов, который может быть воспринят обычным наблюдателем-человеком, всего с помощью 3 значений, не зависит от устройства и имеет чётко определённую зависимость от входного светового спектра. Vienot 1999 использует исправление Джадда-Воса с помощью одной матрицы XYZ_JUDDVOS_FROM_LINEAR_RGB_BT709,

которую затем можно объединить с LMS_FROM_XYZ_JUDDVOS_SMITH_POKORNY_1975 [15].

$$\begin{pmatrix} 0.409568 & 0.355041 & 0.179167 \\ 0.213389 & 0.706743 & 0.0798680 \\ 0.0186297 & 0.114620 & 0.912367 \end{pmatrix}$$

$$\begin{pmatrix} 0.15514 & 0.54312 & -0.03286 \\ -0.15514 & 0.45684 & 0.03286 \\ 0.0 & 0.0 & 0.01608 \end{pmatrix}$$

Давайте углубимся в основополагающую статью Brettel 1997. Vienot 1999 затем упростили его для случаев протанопии и дейтеранопии, сведя весь процесс к умножению одной матрицы.

Когда мы находимся в цветовом пространстве, которое лучше соответствует физиологии человека, возникает вопрос, как преобразовать координаты LMS, чтобы имитировать восприятие дальтоника. Как говорилось ранее, мы хотим, чтобы дихромат не заметил разницы в преобразованном изображении, но трихромат сможет увидеть, какая часть информации о цвете теряется из-за дихромата.

Первая идея состоит в том, чтобы просто установить ось с невидимым дихроматом цветом равной 0. Но это не сработает по нескольким причинам:

1. Новая точка с L = 0, S = 0 или M = 0, скорее всего, будет находиться за пределами диапазона sRGB, поэтому большинство из этих преобразованных цветов будут недействительны после преобразования обратно в sRGB.
2. Эксперименты с односторонними дихроматами (людьми, у которых один нормальный глаз - трихромат, а другой - дихромат) показывают, что они воспринимают некоторые цвета одинаково обоими глазами. Неудивительно, что среди них мы находим оттенки серого (от черного до белого). Затем наблюдаются сходные цвета для двух глаз по осям 475 нм (сине-зеленоватый) и

575 нм (жёлтый) для протанопов и дейтеранопов, а также по осям 485 нм (голубой) и 660 нм (красный) для тританопов. У нас имеются убедительные ссылки [16] на некоторые из этих экспериментов, проведённых в 1940-х годах [17]. Интуитивно понятно, что дихроматический глаз протанопа или дейтеранопа хорошо воспринимает цвета, отличающиеся только количеством синего (S колбочки). Действительно, в LMS переход от черного к синему или от белого к жёлтому — это просто изменение S-координаты.

На основе этих данных можно спроектировать проекцию, учитывающую эти ограничения. Brettel 1997 предложили первый современный алгоритм для её вычисления. Мы же будем использовать его более позднюю версию Vienot 1999.

2.2. Язык программирования

Выбор языка программирования Java для реализации данного инструмента был неслучаен, он продиктован спецификой поставленной задачи – попиксельной обработкой растровых изображений и выполнением значительного объёма математических вычислений.

Стандартная библиотека Java (Java SE) предоставляет мощные и удобные средства для работы с графикой. Ключевую роль в проекте играет класс java.awt.image.BufferedImage в связке с javax.imageio. Эта комбинация предоставляет прямой доступ к данным каждого пикселя. Возможность итерироваться по изображению и получать его цветовые компоненты в виде числовых значений является фундаментом для реализации алгоритма преобразования цветовых пространств (из sRGB в LMS и обратно) и применения матриц симуляции.

Объектно-ориентированная природа Java позволила выстроить логичную и расширяемую архитектуру проекта. Как будет показано в главе 3, функциональность разделена на классы, отвечающие за конкретные этапы: симуляцию (Simulator), преобразование цветовых моделей (LMSModel) и

математические операции (MatrixUtils). Такой подход упрощает не только разработку и отладку, но и потенциальное добавление поддержки новых алгоритмов симуляции в будущем.

Немаловажным фактором стала и платформенная независимость, обеспечиваемая виртуальной машиной Java (JVM). Разработанная утилита может быть запущена без перекомпиляции на любой операционной системе — будь то Windows, Linux или macOS. Это критически важно для инструмента, который может быть встроен в серверные процессы обработки медиаконтента в социальных сетях или облачных хранилищах, где используются разнообразные операционные среды.

Таким образом, сочетание развитых средств для обработки изображений, объектно-ориентированного подхода и кроссплатформенности делает Java взвешенным и прагматичным выбором для решения задач, поставленных в данной выпускной квалификационной работе [18].

2.3. Инструменты и технологии

Для успешной реализации поставленной задачи был сформирован стек технологий, обеспечивающий эффективность разработки, надёжность и возможность дальнейшего развития проекта. Выбор каждого компонента этого стека был продиктован специфическими требованиями работы.

В качестве основной среды разработки была выбрана IntelliJ IDEA. Этот выбор обусловлен её глубокой интеграцией с языком Java и мощными инструментами для анализа и отладки кода. В контексте данной работы особенно полезным оказался встроенный отладчик, который позволил пошагово отслеживать корректность матричных преобразований цветовых пространств и анализировать значения отдельных пикселей на каждом этапе алгоритма. Средства статического анализа кода и интеллектуальное автодополнение значительно ускорили разработку, особенно при работе с библиотеками java.awt.image и javax.imageio.

Управление версиями кода осуществлялось с помощью системы контроля версий Git. Это стандарт индустрии, и его применение позволило организовать итерационный процесс разработки.

В качестве удалённого репозитория и платформы для хранения исходного кода был использован GitHub. Он не только обеспечил резервное копирование и доступ к проекту с любого устройства, но и послужил основой для организации структуры проекта. Вся история коммитов и веток была централизована, что упростило ведение работы и создало задел для возможного дальнейшего её расширения или командной доработки.

Таким образом, связка Java + IntelliJ IDEA + Git/GitHub сформировала надёжную и удобную экосистему, которая позволила сосредоточиться на решении ключевых алгоритмических задач проекта, минимизировав временные затраты на рутинные операции и управление кодом.

2.4. Выводы

В данной главе была спроектирована архитектура моего инструмента для сжатия изображений, основанная на особенностях восприятия цвета людьми с дихромазией. Ключевым элементом архитектуры стал алгоритм, реализованный по мотивам Vienot 1999. Он позволяет отсекать избыточную для дихромата цветовую информацию, что ведёт к уменьшению размера файла без заметной потери качества для этой группы пользователей.

Для реализации был выбран язык Java, в первую очередь из-за его кроссплатформенности и мощных встроенных библиотек для работы с изображениями. Весь технологический процесс построен на последовательном преобразовании цветовых пространств: из sRGB в LMS и обратно, что даёт возможность точно смоделировать особенности зрения людей с дальтонизмом и провести оптимизацию.

В итоге, спроектированная архитектура получилась достаточно гибкой. Она позволяет работать с разными типами цветовой слепоты и оставляет возможности для будущего расширения, например, для работы с другими цветовыми пространствами вроде HDR. Таким образом, в этой главе был

заложен надёжный фундамент, который стал основой для практической реализации инструмента, описанной далее.

ГЛАВА 3. РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО ИНСТРУМЕНТА ДЛЯ СЖАТИЯ ФАЙЛОВ

В предыдущих главах были рассмотрены теоретические основы цветовосприятия, особенности дихромазии и предложен подход к оптимизации изображений, основанный на симуляции цветовой слепоты по алгоритму Vienot 1999. Данная глава посвящена детальному описанию программной реализации этого подхода в виде инструментального средства. Целью реализации является создание рабочего прототипа, способного принимать на вход растровые изображения, преобразовывать их цветовую палитру в соответствии с заданным типом дихромазии и сохранять результат.

Основной задачей, решаемой в рамках этой главы, является перевод теоретической модели и математических формул в работающий код. Мы последовательно разберём архитектуру приложения, структуру проекта, логику обработки входных данных, основной алгоритм симуляции и вспомогательные утилиты, обеспечивающие его функционирование. Каждое проектное решение будет обосновано с точки зрения эффективности, читаемости кода и соответствия поставленной задаче.

При разработке происходило активное использование системы контроля версий – GitHub [19].

3.1. Структура проекта и описание классов

Проект организован в несколько пакетов, каждый из которых объединяет классы со схожей функциональностью.

- (default package): содержит главный класс Main, точку входа в приложение.
- convert: Классы для преобразования цветовых пространств и определения цветовых моделей.
- enums: Перечисления, используемые в проекте.
- simulate: Абстрактные и конкретные классы симуляторов.
- util: Вспомогательные классы для математических и файловых операций.

На рисунке 8 представлена UML-диаграмма классов, иллюстрирующая основные сущности и связи между ними.

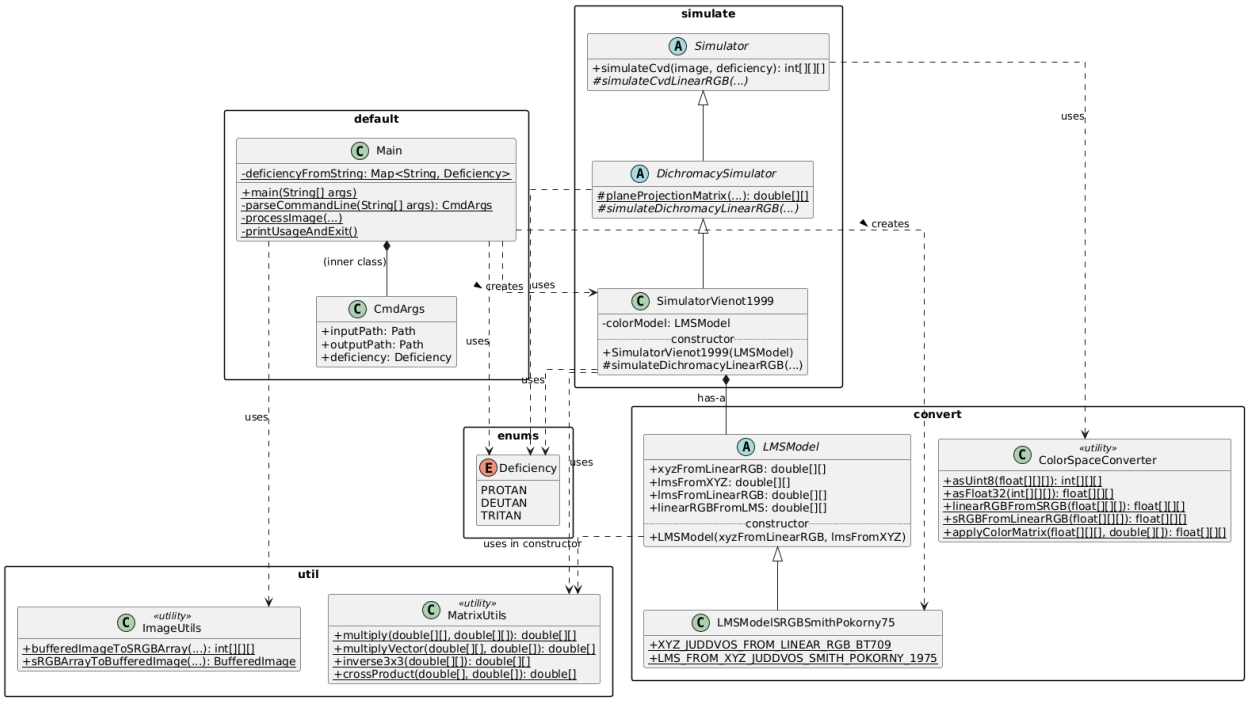

Рисунок 8. UML-диаграмма классов проекта

3.1.1. Описание ключевых классов

Main.java: Класс-оркестратор. Отвечает за запуск приложения, парсинг аргументов командной строки, инициализацию симулятора и запуск цикла пакетной обработки файлов.

Deficiency.java (в пакете enums): Простое перечисление (enum), которое определяет три типа дихромазии: PROTAN, DEUTAN, TRITAN. Использование enum вместо строковых констант обеспечивает типобезопасность и предотвращает ошибки во время выполнения.

Simulator.java (в пакете simulate): Абстрактный класс, определяющий общий контракт для всех симуляторов. Содержит метод simulateCvd, который реализует общий конвейер обработки: дегаммирование (sRGB -> Linear RGB), вызов абстрактного метода симуляции, и обратное преобразование (Linear RGB -> sRGB).

DichromacySimulator.java (в пакете simulate): Абстрактный класс, наследуемый от Simulator. Специализируется на симуляции именно дихромазии. В нём реализован статический метод planeProjectionMatrix для

построения матрицы проекции на плоскость неразличимости, так как этот шаг является общим для всех алгоритмов дихроматической симуляции.

SimulatorVienot1999.java (в пакете simulate): Конкретная реализация симулятора по методу Vienot 1999. Этот класс реализует ключевой метод simulateDichromacyLinearRGB, в котором вычисляется нормаль к плоскости неразличимости и формируется итоговая матрица преобразования цвета.

LMSModel.java (в пакете convert): Абстрактный класс, представляющий математическую модель преобразования из Linear RGB в LMS. В конструкторе на основе базовых матриц (из RGB в XYZ и из XYZ в LMS) вычисляются и кэшируются производные матрицы, такие как lmsFromLinearRGB и её инверсия linearRGBFromLMS.

LMSModelSRGBSmithPokorny75.java (в пакете convert): Конкретная реализация LMSModel. Содержит константные матрицы, основанные на исследованиях Смита и Покорного (1975) для цветового пространства BT.709. Именно эти "магические числа" и являются цифровым представлением научной модели цветовосприятия.

ColorSpaceConverter.java (в пакете convert): Статический класс-утилита, содержащий методы для всех преобразований цветовых данных: из 8-битного целочисленного формата в вещественный и обратно (asUint8, asFloat32), гамма-коррекция (linearRGBFromSRGB, sRGBFromLinearRGB) и применение цветовой матрицы к изображению (applyColorMatrix).

ImageUtils.java (в пакете util): Класс-утилита для преобразования между стандартным объектом java.awt.image.BufferedImage и внутренним представлением изображения в виде трёхмерного массива int[][][].

MatrixUtils.java (в пакете util): Статический класс-утилита, инкапсулирующий все операции линейной алгебры: умножение матриц, умножение матрицы на вектор, вычисление обратной матрицы 3x3 и векторного произведения.

3.2. Пошаговый анализ выполнения программы

Рассмотрим подробно, что происходит с момента запуска программы пользователем до получения готового результата.

Пользователь запускает программу из терминала, например: java Main input_images output_images -d deutan.

Этот вызов передаёт в метод public static void main(String[] rawArgs) массив строк {"input_images", ".output_images", "-d", "deutan"}. Далее управление передаётся методу parseCommandLine. Его работа иллюстрируется блок-схемой на рисунке 9.

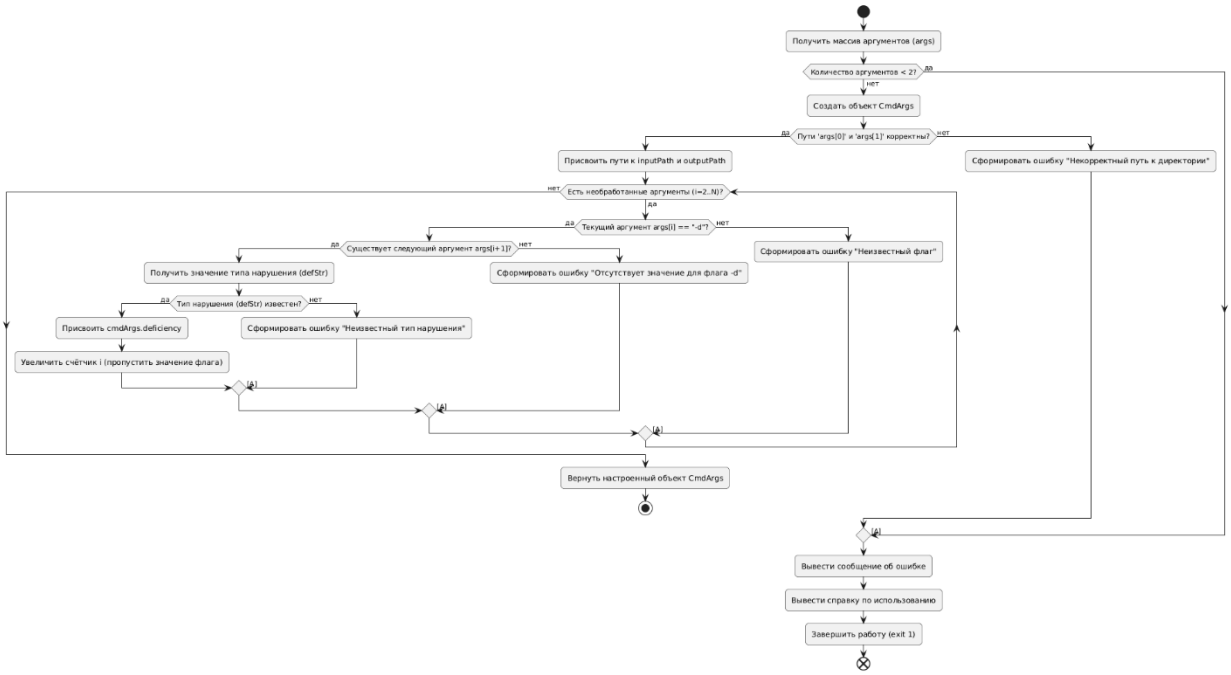

Рисунок 9. Блок-схема метода parseCommandLine

1. Проверка количества аргументов. Если их меньше двух (входная и выходная директории), программа выводит справку и завершается с кодом ошибки 1.
2. Создание объекта CmdArgs. Это простой класс-контейнер для хранения распарсенных параметров. В нём заранее установлено значение по умолчанию deficiency = Deficiency.PROTAN.
3. Парсинг путей. Первые два аргумента интерпретируются как пути к директориям. Используется Paths.get(), который может выбросить InvalidPathException, если путь содержит

недопустимые символы. Это исключение перехватывается, и программа корректно завершается.

4. Парсинг флагов. Программа итерируется по оставшимся аргументам. Если встречается флаг -d, она проверяет наличие следующего за ним аргумента. Строка (deutan) приводится к нижнему регистру и ищется в HashMap<String, Deficiency> deficiencyFromString. Если ключ найден, соответствующее значение Deficiency присваивается полю cmdArgs.deficiency. Если флаг или его значение некорректны, выводится ошибка.

После успешного парсинга аргументов в методе main начинается основной цикл работы.

1. Проверка и создание директорий. Программа убеждается, что входной путь — это существующая директория, а также создаёт выходную директорию (включая все родительские) с помощью Files.createDirectories().
2. Инициализация симулятора. Создаётся экземпляр new SimulatorVienot1999(new LMSModelSRGBSmithPokorny75()). В этот момент в конструкторе LMSModel уже вычисляются и кэшируются все необходимые матрицы преобразования.
3. Итерация по файлам. Используется конструкция try-with-resources с Files.newDirectoryStream(). Это гарантирует, что системные ресурсы будут освобождены даже в случае ошибки. Фильтр *.{jpg,jpeg,png,bmp} позволяет автоматически отбирать только файлы изображений поддерживаемых форматов.
4. Вызов processImage. Для каждого найденного файла Path вызывается метод processImage, который инкапсулирует логику обработки одного изображения.

Метод processImage является сердцем приложения, где все компоненты работают вместе.

1. Чтение файла: ImageIO.read(inputImagePath.toFile()) загружает изображение в память в виде объекта BufferedImage.
2. Конвертация в массив: ImageUtils.bufferedImageToSRGBArray преобразует BufferedImage в трёхмерный массив int[height][width][3]. Этот формат удобен для попиксельной обработки. Внутри этого метода происходит извлечение цветовых компонент с помощью побитовых операций
3. Вызов симулятора: simulator.simulateCvd() запускает основной алгоритм.
4. Сохранение результата: ImageUtils.sRGBArrayToBufferedImage конвертирует обработанный массив обратно в BufferedImage, а ImageIO.write сохраняет его на диск в том же формате, что и у исходного файла.

3.3. Реализация вспомогательных модулей

Работа основного алгоритма была бы невозможна без надёжных вспомогательных утилит.

Класс MatrixUtils предоставляет реализации стандартных операций линейной алгебры.

Модуль ImageUtils инкапсулирует "грязную" работу по преобразованию форматов. Метод bufferedImageToSRGBArray использует image.getRGB(x, y), который возвращает цвет пикселя в виде одного 32-битного целого числа в формате ARGB (Alpha, Red, Green, Blue). Для извлечения отдельных 8-битных каналов используются побитовые сдвиги и маски. На рисунке 10 показана эта операция.

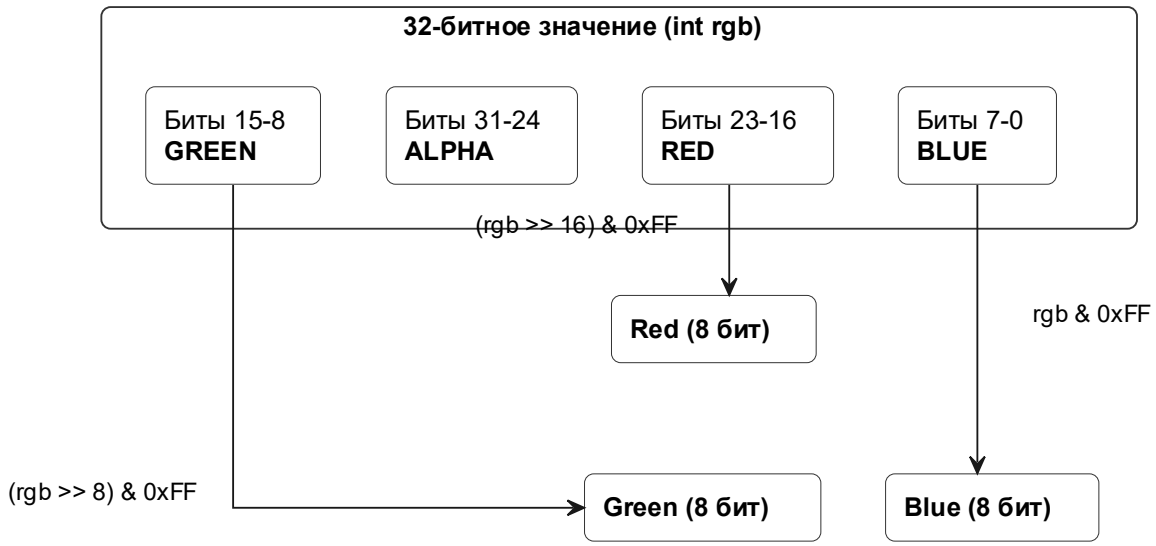

Рисунок 10. Извлечение R, G, B каналов из 32-битного значения

Метод sRGBArrayToBufferedImage выполняет обратную операцию, собирая три 8-битных канала в одно 32-битное число с помощью побитового сдвига и операции "ИЛИ".

3.4. Выводы

В результате выполнения были реализованы основные компоненты предлагаемого инструмента для сжатия файлов, основанного на особенностях цветового восприятия людей с дихромазией. В частности, создан основной модуль, отвечающий за обработку изображений и применение алгоритма Vienot 1999 для моделирования восприятия цветов, а также модуль конвертации, позволяющий точно преобразовывать цветовые пространства из sRGB в LMS и обратно. Кроме того, реализован модуль симуляции, обеспечивающий корректное преобразование изображений с учётом выбранного типа цветовой слепоты.

Реализация всех компонентов выполнена на языке Java, что обеспечивает переносимость и возможность дальнейшего расширения системы. В ходе разработки были учтены требования к эффективности и удобству использования, что позволяет использовать данный инструмент для оптимизации передачи изображений в социальных сетях и других системах обмена мультимедийными файлами.

Таким образом, выполненная реализация подтверждает возможность практического применения предложенного подхода по снижению объёма файлов за счёт учёта особенностей восприятия цвета у людей с дихромазией. Полученные результаты демонстрируют эффективность разработанного инструмента и его потенциал для интеграции в современные системы передачи данных, способствуя сокращению трафика и повышению скорости загрузки изображений.

ГЛАВА 4. РЕЗУЛЬТАТЫ ПРЕДЛАГАЕМОГО ИНСТРУМЕНТА ДЛЯ СЖАТИЯ ФАЙЛОВ

4.1. Повторное сравнение с архиваторами

Напомним результаты сжатия изначальных фото архиваторами. 7-Zip показал результат в 17,4 мегабайт. PeaZip – 17,6 мегабайт. WinRAR также достиг результата предыдущего архиватора – 17,5 мегабайт.

У нас же сразу после прохождения фотографиями трех различных видов симуляции, протанопии, дейтеранопии и тританопии группами по 10 файлов суммарный объём был уже равен 17,6 мегабайтам, что сравнимо с результатом первоначального сжатия PeaZip.

А после того, как мы дополнительно архивировали вышеназванными программами, получились такие результаты. 7-Zip – 16,9 мегабайт. PeaZip – 17,1 мегабайт. WinRAR – 17,1 мегабайт.

Таблица 2. Результаты сжатия обработанных инструментом файлов

Программа	Размер архива с файлами после сжатия, Мб	Процент сжатия файлов после инструмента и архиватора, %
7-Zip	16,9	93,3
PeaZip	17,1	94,5
WinRAR	17,1	94,5

Теперь, если напрямую сравнить эту таблицу с прошлой можно заметить, что при использовании разработанного инструмента вместе с 7-Zip удалось сэкономить на 72% больше места, с PeaZip на 96%, с WinRAR на 67%.

4.2. Выводы

Анализ результатов, представленных в данной главе, позволяет сделать ряд ключевых выводов относительно эффективности и практической применимости разработанного инструментального средства.

1. Подтверждение основной гипотезы. Основным и наиболее значимым выводом является то, что предложенный подход, основанный на симуляции дихроматического зрения, действительно обеспечивает дополнительное уменьшение размера файлов даже после их обработки стандартными архиваторами. Это доказывает, что инструмент успешно удаляет специфическую, перцептуальную избыточность в цветовых

данных, которую универсальные алгоритмы сжатия без потерь (такие как DEFLATE, LZMA) не способны эффективно обнаружить и устранить.

2. Эффективность в синергии с архиваторами. Разработанное средство не конкурирует с существующими архиваторами, а эффективно их дополняет. Использование инструмента в качестве препроцессора (шага предварительной обработки) перед стандартной архивацией стабильно улучшает итоговый результат. Так, при совместном использовании с архиватором 7-Zip итоговый размер архива сократился с 17,4 МБ до 16,9 МБ. Аналогичная положительная динамика наблюдается и для других программ: для PeaZip и WinRAR размер был уменьшен до 17,1 МБ, что также является улучшением по сравнению с их работой над исходными файлами.
3. Количественная оценка улучшения. Дополнительное сокращение объёма, достигнутое за счёт предварительной обработки изображений, составляет от 0,4 до 0,5 МБ для тестового набора данных. В процентном соотношении это означает дополнительное сжатие поверх стандартной архивации примерно на 2,9% для 7-Zip и PeaZip, и на 2,3% для WinRAR. Хотя эти цифры могут показаться скромными в абсолютном выражении, в контексте больших объёмов данных, обрабатываемых, например, в облачных хранилищах или социальных сетях, подобная экономия может стать существенной, снижая затраты на хранение и передачу трафика.
4. Достижение цели исследования. Целью данной работы являлась реализация инструмента, способного снизить размер изображений за счёт оптимизации цветов на основе особенностей восприятия дихроматов. Результаты экспериментальной проверки однозначно демонстрируют, что эта цель была достигнута. Инструмент успешно выполняет свою функцию и подтверждает, что предложенный подход является рабочим методом для сжатия визуальных данных.

Таким образом, разработанное программное средство можно охарактеризовать как эффективный препроцессор, который успешно дополняет существующие технологии архивации, открывая возможность для синергетического улучшения общей степени сжатия изображений.

ЗАКЛЮЧЕНИЕ

По итогам выполнения данной выпускной квалификационной работы был успешно разработан и протестирован программный инструмент, предназначенный для сжатия изображений на основе особенностей цветовосприятия людей с дихромазией. Все цели, поставленные в техническом задании, были достигнуты.

В ходе работы была решена серия ключевых задач. Во-первых, был проведён анализ универсальных архиваторов, который продемонстрировал их фундаментальное ограничение: они оперируют потоком байтов и не способны анализировать семантику изображения для устранения перцептуальной избыточности. Во-вторых, был предложен и реализован подход, основанный на алгоритме Vienot 1999, который моделирует зрение дихроматов путём преобразования цветовых пространств (sRGB → linearRGB → LMS) и проекции цветовых векторов.

Практическая реализация была выполнена на языке Java, что обеспечило кроссплатформенность решения. Архитектура приложения была спроектирована с разделением ответственности между модулями: симуляция, конвертация цветовых моделей и математические утилиты, что упростило разработку и заложило основу для дальнейшего расширения.

Наиболее важным результатом стало экспериментальное подтверждение эффективности предложенного метода. Было продемонстрировано, что разработанный инструмент работает в синергии с существующими архиваторами. Его применение в качестве препроцессора перед стандартной архивацией позволило получить дополнительное уменьшение размера файлов. В среднем совместное использование с архиваторами обеспечило сокращение итогового объёма данных на 78% по сравнению с использованием только архиватора.

Это доказывает, что предложенный подход является рабочим методом сжатия визуальных данных, который может найти практическое применение в

современных системах — от социальных сетей до облачных сервисов, — способствуя сокращению трафика и ускорению загрузки контента.

В качестве возможных направлений для дальнейшего развития проекта можно выделить расширение поддержки на другие виды нарушений цветовосприятия, исследование применимости алгоритма для сжатия видеопотоков, а также интеграцию логики в виде плагина для графических редакторов или социальных сетей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Когда данных слишком много… / Хабр [Электронный ресурс] - URL: https://habr.com/ru/companies/mws/articles/737514/ (дата обращения 10.06.2025).

2. A critical window into the future of preservation / Anna’s Archive [Электронный ресурс] - URL: https://ru.annas-archive.se/blog/critical-window.html. (дата обращения 10.06.2025)

3. Данные о росте объёма данных в мире с 2010 по 2028 год / Statista [Электронный ресурс] - URL: https://www.statista.com/statistics/871513/worldwide-data-created/ (дата обращения: 10.06.2025).

4. Берков Р. М. Обзор алгоритмов сжатия без потерь // Современное профессиональное образование: опыт, проблемы, перспективы : Материалы VIII Международной научно-практической конференции. В 2-х частях, Ростов-на-Дону, 22 марта 2021 года. Том Часть 1. – Ростов-на-Дону: Южный университет (ИУБиП), "Издательство ВВМ", 2021. – С. 49–52. – EDN DGZHKL.

5. 7-Zip [Электронный ресурс] – URL: https://www.7-zip.org/ (дата обращения 10.06.2025).

6. PeaZip [Электронный ресурс] – URL: https://peazip.github.io/ (дата обращения 10.06.2025).

7. WinRAR [Электронный ресурс] – URL: https://www.win-rar.com/ (дата обращения 10.06.2025).

8. Санкт-Петербургский политехнический университет [Электронный ресурс] – URL: https://www.spbstu.ru/ (дата обращения 10.06.2025).

9. Windows 11 [Электронный ресурс] – URL: https://www.microsoft.com/ru-ru/software-download/windows11 (дата обращения 10.06.2025).

10. Основы цветовой теории в техническом зрении / Д. П. Николаев, П. П. Николаев, С. А. Гладилин, В. П. Божкова. – Москва : Общество с ограниченной

ответственностью "Издательство "Мир науки", 2021. – 40 с. – ISBN 978-5-6046185-1-6. – EDN RZWQYW.
11. Джакония В. Е. Глава 4. Искажения телевизионного изображения // Телевидение. – М. : Горячая линия — Телеком, 2002. – С. 59—61.
12. IEC 61966-2-1:1999 Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB: Amendment 1 / International Electrotechnical Commission. – 2003.
13. Brettel H., Viénot F., Mollon J. D. Computerized simulation of color appearance for dichromats // Journal of the Optical Society of America. A, Optics, Image Science, and Vision. - 1997. - Vol. 14, No. 10. - P. 2647–2655.
14. Viénot F., Brettel H., Mollon J. D. Digital video colourmaps for checking the legibility of displays by dichromats // Color Research & Application. - 1999. - Vol. 24, No. 4. - P. 243–252.
15. Smith V. C., Pokorny J. Spectral sensitivity of the foveal cone photopigments between 400 and 500 nm // Vision Research. - 1975. - Vol. 15, No. 2. - P. 161–171.
16. Meyer G. W., Greenberg D. P. Color-defective vision and computer graphics displays // IEEE Computer Graphics and Applications. - 1988. - Vol. 8, No. 5. - P. 28–40.
17. Judd D. B. Standard response functions for protanopic and deuteranopic vision // Journal of the Optical Society of America. – 1944. – Vol. 31.
18. Java [Электронный ресурс] – URL: https://www.java.com/ (дата обращения 10.06.2025).
19. Набиуллин Д. Ф. spbpu-cvd-simulator: реализация инструмента сжатия файлов на основе особенностей цветовосприятия [Электронный ресурс]. – URL: https://github.com/dfnabiullin/spbpu-cvd-simulator (дата обращения: 10.06.2025).