

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический  
университет имени К. И. Сатпаева

Институт информационных и телекоммуникационных технологий

Кафедра «Программной инженерии»

Мұхит Тәңірберген Рызабекұлы

Разработка системы обработки и сжатия аэрокосмических изображений

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

Специальность 7M06101 – Software Engineering

Алматы 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический  
университет имени К. И. Сатпаева

Институт информационных и телекоммуникационных технологий

Кафедра «Программной инженерии»

УДК 004.415

На правах рукописи

Мұхит Тәңірберген Рызабекұлы

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

На соискание академической степени магистра

Название диссертации Разработка системы обработки и сжатия  
аэрокосмических изображений

Направление подготовки 7M06101 – Software Engineering

Научный руководитель,  
ассистент профессор PhD  
Ж. М. Алибиева  
«  »    2022г.

Рецензент  
Доктор PhD, профессор  
А. Ж. Аккалова  
(ученая степень, звание)  
«  »    2022 г.



**ДОПУЩЕН К ЗАЩИТЕ**  
Заведующий кафедрой ПИ  
канд. физ.-мат. наук,  
ассоциированный профессор  
А. Н. Молдагулова  
«01» 06 2022 г.

Алматы 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический  
университет имени К. И. Сатпаева

Институт информационных и телекоммуникационных технологий

Кафедра Информационные технологии

Специальность: 7M06101 – Software engineering



УТВЕРЖДАЮ

Заведующий кафедрой ПИ

канд. физ.-мат. наук, ассоциированный  
профессор

А. Н. Молдагулова

«01» 06 2022г.

**ЗАДАНИЕ**

**на выполнение магистерской диссертации**

магистранту Мұхит Тәңірберген Рызабекұлы \_\_\_\_\_

Тема: «Разработка системы обработки и сжатия аэрокосмических изображений»

Утверждена приказом Ректора Университета №330– М от "11" ноября 2019 г.

Срок сдачи законченной диссертации

"06" июня 2022г.

Исходные данные к магистерской диссертации. Дан анализ существующих алгоритмов и методов сжатия гиперспектральных аэрокосмических изображений. Поставленные цели и задачи диссертационной работы обуславливают комплексность методологии исследования, основанной на системном, процессно-ориентированном подходе к изучению различных алгоритмов обработки, сжатия и преобразования гиперспектральных аэрокосмических изображений.

Перечень подлежащих разработке в магистерской диссертации вопросов:

а) обзор существующих алгоритмов и методов сжатия гиперспектральных аэрокосмических изображений

б) методы и алгоритмы обработки и сжатия гиперспектральных изображений

в) результаты исследования системы обработки и алгоритмов сжатия гиперспектральных изображений

г) разработка системы обработки и сжатия преобразования Уолша-Адамара гиперспектральных изображений

Рекомендуемая основная литература:

1. Emmanuel Christophe. Hyperspectral Data Compression Tradeoff. Augmented Vision and Reality, 3, DOI: 10.1007/978-3-642-14212-3\_2, Springer-Verlag Berlin Heidelberg 2011. Lossless multispectral & hyperspectral image compression. CCSDS Recommended standard for lossless multispectral & hyperspectral image compression. Recommended Standard, Issue 1. CCSDS Secretariat. Space Communications and Navigation Office, 7L70. NASA Headquarters. Washington, DC 20546-0001, USA. May 2012. 52 p. 3. Anonymous *Image Data Compression*. Washington D.C.: Recommendation for Space Data Systems Standards. CCSDS 120.1-G-1 Green Book, June 2007. 4. D. Salomon, *Data Compression: The Complete Reference*. London: Springer, 2007.




## ГРАФИК

подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Реализация методов на языке программирования C#	30.10.2021 г.	Выполнено
Тестовое сжатие аэрокосмических изображений	13.11.2021 г.	Выполнено
Сравнение сжатого изображения с оригиналом в программе IDRISI Selva	20.11.2021 г.	Выполнено

### Подписи

консультантов и нормоконтролера на законченную магистерскую диссертацию с указанием относящихся к ним разделов диссертации

Наименования разделов	Консультанты Ф.И.О. (уч. степень, звание)	Дата подписания	Подпись
Стандартизация/нормоконтроль	Ахмедиярова Айнур Танатаровна – Ассистент проф.	26.04.2022 - 07.05.2022	
Программное обеспечение/нормоконтроль	Мукажанов Нуржан Какенович – Ассоц. профессор	26.04.2022 - 07.05.2022	
Антиплагиат	Әубәкіров Б. С. – Ассистент	09.05.2022 - 14.05.2022	

Дата выдачи задания "10" 01 2022г.

Заведующий кафедрой Молдагулов А. Н. Молдагулов

Научный руководитель Алибиева Ж. М. Алибиева

Задание принял к исполнению магистрант Мухит Т. Р. Мухит

Дата "01" 06 2022 г.





## СОДЕРЖАНИЕ

Введение	5
1 Обзор существующих алгоритмов и методов сжатия гиперспектральных аэрокосмических изображений	7
1.1 Характеристика бортовых и космических систем для обработки гиперспектральных изображений	7
1.2 Особенности гиперспектральных аэрокосмических изображений	10
1.3 Подходы к обработке и сжатия гиперспектральных изображений	11
Выводы по первому разделу	
2 Методы и алгоритмы обработки и сжатия гиперспектральных изображений	18
2.1 Способы кодирования гиперспектральных изображений на основе преобразования	18
2.2 Стандарт сжатия изображений JPEG	22
2.3 Преобразование вейвлет Хаара в обработке гиперспектральных изображений	23
2.4 Постановка задачи обработки и сжатия гиперспектральных изображений	27
Выводы по второму разделу	
3 Результаты исследования системы обработки и алгоритмов сжатия гиперспектральных изображений	31
3.1 Оценка эффективности предлагаемого алгоритма и преобразования Уолша-Адамара	31
3.2 Обработка гиперспектральных изображений и преобразование Уолша Адамара	32
3.3 Разработка программы и практическое применение в обработке гиперспектральных данных	35
3.4 Разработка системы обработки и сжатия преобразования Уолша-Адамара гиперспектральных изображений	42
3.5 Описание разработанных классов для обработки и сжатия гиперспектральных изображений	52
Выводы по третьему разделу	
Список использованных источников	62
Приложение А Листинг программы обработки и сжатия гиперспектральных изображений на основе преобразования Уолша-Адамара	67

## ВВЕДЕНИЕ

**Актуальность исследования.** Аэрокосмические данные используются для различных отраслей науки и хозяйства: прогнозирование погодных условий, определение границ климатических событий, в целях военной разведки и множестве других. Уменьшение объема передаваемых данных позволит более оптимально управлять дисковым пространством для их хранения и увеличить скорость обмена информацией. Собственные решения укрепляют позиции государства в сфере разработок программного обеспечения.

**Цель исследования:** Разработка программного обеспечения, позволяющего обрабатывать и сжимать данные дистанционного зондирования Земли, с целью уменьшения занимаемого ими объёма, используя математические методы. Функцией разрабатываемого программного комплекса является обработка аэрокосмических изображений с целью рационального использования устройств хранения и передачи данных, с возможностью преобразования данных и восстановления их исходного состояния. В разработке будут использоваться эффективные математические методы и новые подходы к обработке и сжатию данных.

### **Задачи исследования:**

- проведение анализа существующих систем работы с аэрокосмическими изображениями;
- разработка высокопроизводительного метода и алгоритма сжатия аэрокосмических данных на основе преобразования Уолша-Адамара, позволяющих оптимизировать их передачу и хранение;
- программная реализация высокопроизводительных алгоритмов сжатия в программной среде Visual Studio 2019 с помощью языка C#.
- проведение экспериментальных исследований по показателям разработанного метода и алгоритма на основе преобразования Уолша-Адамара;
- подготовка и оформление магистерской диссертации.

### **Научная новизна исследования** заключается в следующем:

- исследованы имеющиеся программные системы, методы, алгоритмы и способы обработки и сжатия аэрокосмических изображений, и на основе выявления их преимуществ и недостатков выбрано наиболее актуальное направление;

- проведено многоэтапное проектирование автоматизированной системы для сжатия аэрокосмических изображений с использованием преобразования Уолша-Адамара матриц  $8 \times 8$ ,  $4 \times 4$ ,  $2 \times 2$ , с выбором одного или нескольких каналов гиперспектрального изображения;

- разработана автоматизированная система для сжатия аэрокосмических изображений с использованием преобразования Уолша-Адамара матриц  $8 \times 8$ ,  $4 \times 4$ ,  $2 \times 2$ , отличающаяся эффективным функционалом, доступностью и простотой в использовании посредством программных средств бесплатного распространения Microsoft Visual Studio (язык программирования C#);

**Практическая значимость** диссертационной работы состоит в том, что разработана многофункциональная автоматизированная система, позволяющая



обрабатывать, хранить и выдавать необходимую информацию, повышая эффективность сжатия аэрокосмических изображений с использованием преобразования Уолша-Адамара матриц  $8*8$ ,  $4*4$ ,  $2*2$ .

**Теоретической значимостью** исследования явились научные работы, раскрывающие принципы разработки алгоритмов и методов для сжатия аэрокосмических изображений с использованием преобразования Уолша-Адамара матриц  $8*8$ ,  $4*4$ ,  $2*2$  автоматизированных систем (Р. Н. Ахметов, Н. Р. Стратилатов, А. А. Юдаков, Козинков, И. А., Саринова, А. Ж., С. Л. Погорельский, Е. А. Макарецкий, А. В. Овчинников, Н. Ying and L. Guizhong, Kai-Jen Cheng, Jeffrey C. Dill, Diego Valsesia, Enrico Magli D.S.Sujithra, T.Manickam, D.S.Sudheer, E. Christophe, С. М. Борзов, П. В. Мельников, И. А. Пестунов и многие др.). Для решения поставленных задач были использованы методы исследования: изучение источников, теоретический анализ и обобщение литературных данных с выявлением особенностей обработки и сжатия аэрокосмических изображений;

**Апробация работы.** Диссертационные исследования были опубликованы и апробированы на международной научно-практической конференции «XXI Сатпаевские чтения», тема статьи «Современные подходы к алгоритмам сжатия гиперспектральных аэрокосмических изображений» в 2022 году.

**Структура диссертации.** Диссертационная работа состоит из трех глав, введения, заключения, списка использованной литературы и приложения (листинг программы). Материал изложен на 82 страницах, включает 11 таблиц, 26 рисунков, а также одно приложение. Список использованной литературы содержит 56 наименований.

# **1 Обзор существующих алгоритмов и методов сжатия гиперспектральных изображений**

## **1.1 Характеристика бортовых и космических систем для обработки гиперспектральных изображений**

Дистанционное зондирование — это наука о получении информации о земной поверхности с большого расстояния с помощью оптических датчиков без физического контакта с объектом. То эволюция науки тесно связана с практикой фотографии [1]. В 1858 г. Первые аэрофотоснимки были сделаны Гаспаром-Феликсом Турнашоном с привязанного воздушного шара над Париж, Франция. Другие фотографы-первопроходцы также использовали ракеты, воздушных змеев и голубей для переноски их камеры. На начальном этапе аэрофотосъемки управление камерами на этих перевозчики были очень сложными и ненадежными. В начале 1900-х годов был изобретен самолет. В 1909 году Уилбур Райт сделал первые аэрофотоснимки с самолета на контролируемой скорости высота и направление. Во время Первой мировой войны аэрофотосъемки возросла важная роль. Многие приложения аэрофотосъемки использовались в военной разведке и наблюдении. Специально разработанные камеры и соответствующие воздушные приборы для самолетов были разработаны и разработаны правительством США. После окончания войны некоторые новаторские гражданские компании преобразовали эти военные приложения аэрофотосъемки в невоенные приложения, такие как геологическое картирование или серия почв, лесов и сельскохозяйственные обследования.

Во второй мировой войне значение воздушной разведки значительно возросло, потому что расширение воздушной разведки проникло вглубь вражеской территории собирать ценную информацию, такую как военное развертывание, а также промышленные и транспортные инфраструктуры. После войны многие опытные пилоты, операторы, и фотопереводчики, обученные правительством или военными, применяли свои ценные навыки и опыт для гражданских компаний, и где они процветали в индустрия дистанционного зондирования.

В 1960-е годы произошли значительные изменения и улучшения в истории дистанционное зондирование. Первый успешный и специализированный низкоорбитальный метеорологический спутник. (телевизионный инфракрасный спутник наблюдения, или TIROS-1) был запущен НАСА в апреле 1960 [2]. Впервые он был разработан для климатологических и метеорологических наблюдений за Земли и послужили основой для более позднего развития спутников наблюдения за землей. На TIROS-1 были установлены инфракрасные радиометры для наблюдения за распределением тепла на Земле. Аэрофотосъемка в инфракрасном и микроволновом диапазонах. Поскольку новые формы изображений были собраны с использованием излучения вне видимой области спектра, термин «воздушная фотография», больше не подходило для описания новой формы

изображения. Новый термин «дистанционное зондирование» возник и начал использоваться в 1960-х годах. В то время многие секретные методы дистанционного зондирования и инструменты, выпущенные военными ускоренными темпами рост применения дистанционного зондирования в научных и гражданских целях.

Находящийся на околоземной орбите спутник Landsat 1, первоначально называвшийся "Earth Resources Технологический спутник 1", был запущен 23 июля 1972 г. [3]. Программа Landsat была инициирована на изучение земной поверхности и ресурсов. Его космический датчик, называемый мультиспектральный сканер (MSS) позволил ему получать четырехдиапазонные мультиспектральные изображения на сельскохозяйственные и лесные ресурсы, геология и полезные ископаемые, гидрология и вода ресурсы, география и морские ресурсы, а также метеорологические явления. Помимо своего датчика космического базирования, самым важным вкладом Landsat 1 было обеспечение стандартный цифровой формат мультиспектральных изображений. Этот цифровой формат способствовал точному обработка, получение, воспроизведение и распространение изображений дистанционного зондирования; более того, этот стандартный цифровой формат способствовал росту цифрового изображения. поле обработки. Landsat 1 проложил путь другим спутникам программы Landsat, которая является самой продолжительной космической программой до настоящего времени. Две новые системы MSS, позднее были установлены Thematic Mapper (TM) и Enhanced Thematic Plus (ETM+).

Коммерческий спутник наблюдения Земли IKONOS впервые был запущен в эксплуатацию гражданских применений и имели высокое пространственное разрешение многоспектральных и панхроматические изображения с разрешением 1 м и 4 м [1]. Изображения IKONOS подходят для приложений, требующих высокого уровня детализации и точности, таких как картография и городская планирование.

В 1980-х годах Лаборатория реактивного движения (JPL) разработала гиперспектральный спектрометр, началась эра гиперспектрального дистанционного зондирования [4]. Передовой воздушно-десантный Спектрометр видимого/инфракрасного изображения (AVIRIS) был установлен на четырех самолетах платформы: реактивный самолет NASA ER-2, турбовинтовой Twin Otter International, Scaled Composites Proteus и НАСА WB-57. Спектрометр производит до 224 спектральных полос о шириной 10 нм, в диапазоне от 400 до 2500 нанометров (нм). С таким высоким спектральное разрешение, многие тонкие объекты и материалы могут быть идентифицированы. Он был использован в классификации местности, экологическом мониторинге, сельскохозяйственном мониторинге, геологическом разведка и наблюдение.

Эксперимент по сбору гиперспектральных цифровых изображений (HYDICE) был одним из бортовых гиперспектральных приборов, используемых на относительно малых высотах. Он эксплуатировался в 1994 г. и производит 210 спектральных диапазонов между 400 и 2500 нм [1].

Гиперион был на спутнике NASA Earth-Observing-1 (EO-1) в качестве первого гражданского гиперспектральной спутниковой системы, действовавшая в 2000 году. Она предоставила изображения в 220 спектральных полос в диапазоне от 400 до 2500 нм [2]. Глобальное дистанционное зондирование собирает и отслеживает широкомасштабные наблюдения за поверхность земли и окружающие прибрежные районы для исследования глобальных изменений, региональные исследования изменения окружающей среды и другие гражданские и коммерческие цели. В декабре в 1999 году НАСА запустило Terra-1, первый спутник системы, специально предназначенной для получения глобального охвата, чтобы обеспечить основу для документирования экологических изменений за последние десятилетия [5]. Спутник Terra-1 несет пять современных приборов, которые проводят совпадающие измерения земной системы: Advanced Spaceborne Thermal эмиссионно-отражательный радиометр (ASTER), Облака и лучистая энергия Земли системы (CERES), измерения загрязнения тропосферы (MOPITT), спектрометри с визуализацией среднего разрешения (MODIS) и многоугольная визуализация спектрометра (MISR).

В современных дистанционных датчиках усовершенствования сосредоточены на размере датчика, ширина полосы обзора, отношение сигнал/шум, точность радиометрической и спектральной калибровки и спектральные каналы. Показаны некоторые примеры бортовых и космических датчиков [6], ниже в таблице 1.

Table 1 Research and Commercial imaging spectrometers [6]

<b>AIRBORNE SENSOR</b>	<b>Explanation of Acronym</b>	<b>NUMBER OF BANDS</b>	<b>WAVELENGTH RANGE (<math>\mu\text{m}</math>)</b>
<b>AVIRIS</b>	Airborne Visible/InfraRed Imaging Spectrometer	224	0.4-2.5
<b>AHI</b>	Airborne Hyperspectral Imager	210	7.9-11.5
<b>ARCHER</b>	Airborne Real-time Cueing Hyperspectral Enhanced Reconnaissance	512	0.5-1.1
<b>AISA</b>	Airborne Imaging Spectrometer for Application	286	0.45-0.9
<b>CASI</b>	Compact Airborne Spectrographic Imager	288	0.43-1.1
<b>COMPASS</b>	COMPact Airborne Spectral Sensor	256	0.4-2.5
<b>HYDICE</b>	HYperspectral Digital Imagery Collection Experiment	210	0.4-2.5
<b>HyMap</b>	-	126	0.45-2.5
<b>SPACEBORNE SENSOR</b>	<b>STAND FOR</b>	<b>NUMBER OF BANDS</b>	<b>WAVELENGTH RANGE (<math>\mu\text{m}</math>)</b>
<b>Hyperion</b>	-	220	0.4-2.5
<b>FTHSI</b>	Fourier Transform <i>Hyperspectral</i> Imager	256	0.35-1.05

## 1.2 Особенности гиперспектральных аэрокосмических изображений

Как правило, гиперспектральный спектрометр анализирует и интерпретирует измерения электромагнитных излучений, которые отражаются или излучаются целями на Земле, океана или пустыни для создания гиперспектральных изображений. По сути, спектрометр, установленный на самолет или спутник рассеивают свет на сотни узких и смежных спектров.

Между тем, сотни датчиков внутри детектора спектрометра по отдельности соответствующий спектр, и они строят куб цифрового гиперспектрального изображения с сотнями смежных спектральных полос, как показано на рисунке 1. Каждый куб изображения состоит из трехмерного массива пикселей. Пиксель описывается значением интенсивности и местоположением в трехмерном изображении. Значение интенсивности пикселя представляет собой среднее значение измеренного физического сигнала, такого как солнечное излучение, испускаемое инфракрасное излучение, или интенсивность обратного радиолокационного рассеяния, отраженного от всей поверхности земли. Интенсивность пиксель оцифровывается и записывается как цифровое число.

AVIRIS, CCSDS и Hyperion. Бортовой спектрометр видимого/инфракрасного изображения (AVIRIS) был приобретен лабораторией реактивного движения НАСА (JPL) в 1987 году. Изображения AVIRIS обеспечивают более точную и подробную информацию, чем другие типы спектральных данных, из-за обильного спектральная информация. AVIRIS был доставлен на самолете НАСА ER-2 на высоту около 20 км при скорости около 730 км/ч. AVIRIS был первым спектрометром изображений, анализировать отраженную солнечную радиацию в 224 смежных спектральных диапазонах с длинами волн от 400 до 2500 нм.

В изображении AVIRIS разрешение каждого пикселя называется сценой, соответствующей площади примерно 11 км 10 км на земле. Встроенное хранилище сохраняет все собранные сцены вместе с навигационными и инженерными данными и показания бортового калибратора AVIRIS. Когда все данные обработаны и хранится на земле, каждая сцена занимает около 140 мегабайт (МБ).

Гиперспектральное изображение можно рассматривать как трехмерное (3D) кубическое изображение с пространственным измерения, образованные осями  $x$  и  $y$ , и спектральное измерение ( $z$ ). На рис. 1 показан Куб изображения AVIRIS, покрывающий территорию над Моффетт Филд, Калифорния, в южной части залива Сан-Франциско. Куб отображает стопку из 224 смежных спектральных полос от верхней части видимой области (400 нм) до нижней части инфракрасной (2500 нм). То стороны имеют псевдоцвет, варьирующийся от черного и синего (низкая частота) до красного (высокая частота). частота). Непрерывные спектральные трассы также показаны на рисунке 1 и образованы наблюдая столбец пикселей вдоль направления  $z$ . Три уникальные спектральные трассы представляют город, вода и земля, и можно

идентифицировать или различать различные материалы по сравнению спектральной трассы с заданной спектральной базой данных.

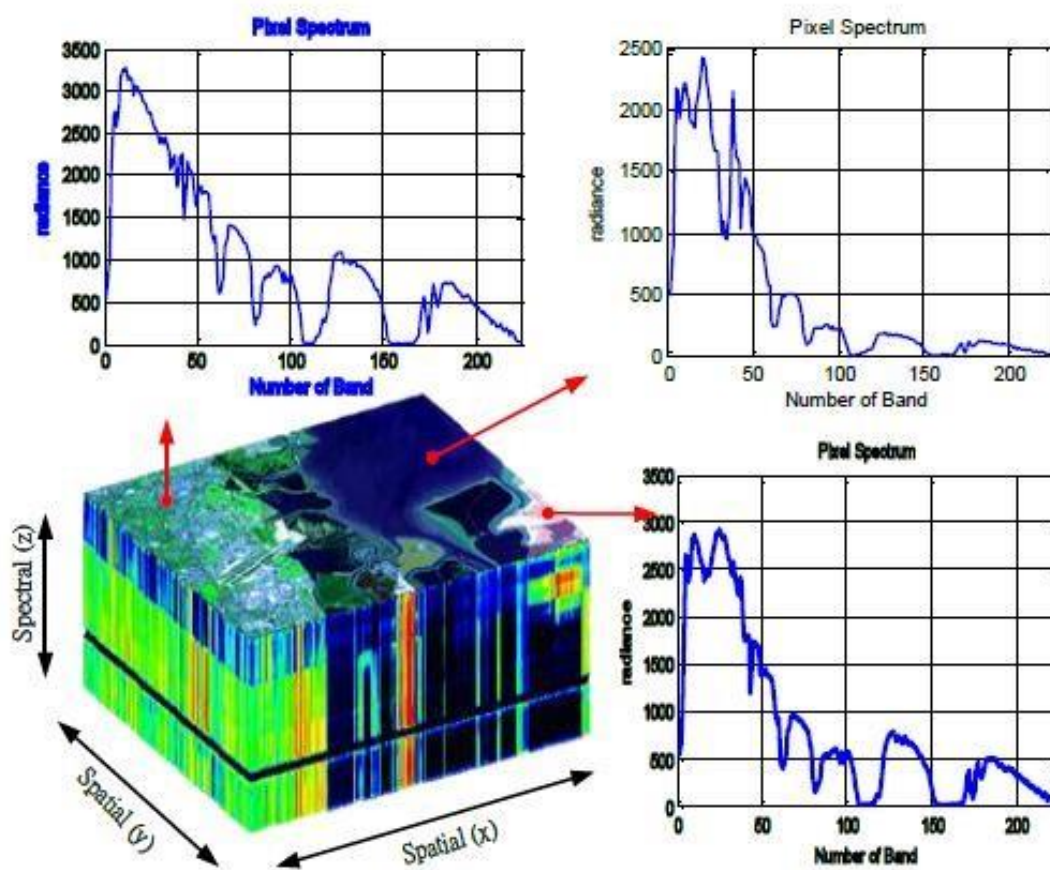


Рисунок 1. Перспектива куба гиперспектральных изображений AVIRIS пространственных данных (x и y) и спектральные данные (z).

Каждая непрерывная спектральная трасса для каждого пикселя изображения уникальна.



Рисунок 2 – Оригинальное изображение

### 1.3 Подходы к обработке и сжатию гиперспектральных изображений

К преимуществам сжатия гиперспектральных изображений [8] относятся: а) уменьшение пропускная способность канала передачи; б) сокращение буферизации и хранения требование; в) сокращение времени передачи данных на заданной скорости. Из-за ограниченных доступных ресурсов и возможности обработки в космическом и воздушном платформах, сложность встроенного кодирования является важной проблемой в гиперспектральных сжатия. Сжатие учитывает компромисс между этими факторами: обработка возможности, приложения для конечных пользователей и качество данных (без потерь или с потерями), а также ограничения (памяти и аппаратных средств) космических и бортовых приборов.

Самые большие проблемы адресованы встроенному сжатию изображений. Во-первых, из-за большого объема данных. собираются и ограниченная пропускная способность, нет сомнения, что данные должны быть хранится на спутнике или самолете. Однако это встроенное хранилище ограничено. Данные имеют обрабатываться «на лету» по мере их поступления. Алгоритм сжатия изображения должен выполняться на лету, то есть они должны начать сжиматься по мере того, как изображения приобретенный. Кроме того, сжатие должно быть алгоритмом низкой сложности и иметь постоянная пропускная способность из-за ограниченных возможностей обработки.

Сжатие изображений — это метод устранения избыточности. Изображение по своей сути методы сжатия делятся на две группы: сжатие без потерь и сжатие с потерями. Сжатие без потерь не влияет на исходное изображение. Самые свежие публикации показывают, что коэффициент сжатия без потерь может достигать 4:1. Однако может быть и недостаточно, чтобы соответствовать ограничениям для бортовых приложений. Наоборот, с потерями сжатие вносит небольшие искажения в исходное изображение и может достигать гораздо большей степени сжатия при отбрасывании большего количества информации. Компромисс с потерями сжатие — это возможность собрать больше изображений в ограниченном встроенном хранилище. То эффекты искажения собранных изображений различаются от одного случая к другому. Анализ и на понимание искажения может указывать либо субъективное, либо объективное изображение оценка качества. Субъективная оценка использует людей для оценки, сравнения или оценить качество изображений. Объективная оценка оценивает качества, принимая статистические измерения на тестовых изображениях.

Другими важными свойствами сжатия гиперспектральных изображений являются случайные доступ, прогрессивное декодирование, масштабируемость разрешения, стандартный формат и гибкость. То свойство произвольного доступа состоит в том, чтобы кодировать и декодировать выбранные части интереса в гиперспектральное изображение. Поскольку вместо всего изображения кодируется небольшая его часть, достигает высокой степени сжатия, не жертвуя какой-либо информацией в изображении и можно

реконструировать с высоким качеством. Особенность прогрессивного декодирования заключается в том, что точность реконструкции улучшается по мере восстановления большего количества информации. Разрешение масштабируемости позволяет генерировать изображение с низким разрешением или грубое изображение для быстрого предварительного просмотра всего изображения. При необходимости изображение низкого разрешения можно улучшить (меньше искажение), когда восстанавливается больше данных.

За последние несколько лет было предложено большое разнообразие методов сжатия, в то время как исследователи рассматривают возможные отношения между большим объемом пространственной и спектральной информации. В целом, эти методы сжатия изображений классифицируются на три группы: прогнозирующее кодирование, векторное квантование и кодирование на основе преобразования.

В следующих разделах представлена литература, относящаяся к компрессии гиперспектральных изображений.

Предиктивное кодирование. Предпочтительным методом сжатия гиперспектральных изображений без потерь является предиктивное кодирование. На первом этапе вычисляются один или несколько предикторов и используются для создания остаточных ошибок и удалить избыточность изображения. Эти ошибки затем кодируются энтропийным кодированием (кодированием Хаффмана, кодированием Райса или арифметическим кодированием).

Предыдущие работы показали [1-31], что алгоритм, исследующий спектральную корреляцию в гиперспектральных изображениях, могут вычислять оптимальные предикторы, чтобы уменьшить избыточность в изображениях и приводит к высоким коэффициентам сжатия. Обратите внимание, что все коэффициенты сжатия в основном зависят от данных и будут несколько отличаться от изображения к изображению для одного и того же алгоритма сжатия.

Роджер и Кавенор [14] впервые изучили адаптивный дифференциальный импульсный код. Модуляция (ADPCM) для сжатия без потерь гиперспектральных изображений AVIRIS в 1996. Они экспериментировали с гиперспектральными изображениями с пятью наборами линейных пространственных, спектральных и пространственно-спектральные предикторы. Остаточные значения кодируются кодированием переменной длины (VLC) алгоритм. Коэффициент сжатия указывается в битах на пиксель на полосу ( $b_{pprb}$ ), то есть количество выходного битового потока усредняется по всему изображению объема. Их коэффициенты сжатия без потерь для изображений AVIRIS находятся в диапазоне 1,6-2,0:1 Риццо и др. [15] предложили сжатие без потерь гиперспектрального изображения с помощью Linear Предсказание (LP). Простой спектральный LP предсказывает текущий пиксель по его причинно-следственному соседу набора данных в текущем диапазоне и предыдущем диапазоне. Второй метод Риццо и др. был назван предиктором спектрального метода наименьших квадратов (SLSQ), оптимальные коэффициенты которого равны определяется алгоритмом



наименьших квадратов (LS). Алгоритм исследования спектральных корреляций гиперспектрального изображения обеспечивает лучшую производительность. Каждый пиксель в каждой полосе был предсказан его собственными оптимальными предикторами SLSQ, так что степень сжатия была улучшилась в более высоком диапазоне 2–3: 1, но требования к памяти были больше.

Несколько исследователей [32-35] пытались найти оптимизированные коэффициенты для предикторов на основе группы пикселей с похожими характеристики, которые внесли вклад в исследование, посвященное оптимизированному коэффициенту предикторы методами классификации [16-18]. В [16] пространственно-спектральный нечеткий Введен ДИКМ. Трехмерная окрестность (совместно пространственная и спектральная) для каждого пикселя определены и сгруппированы в М кластеров с помощью Fuzzy C Means (FCM) [19]. Коэффициенты предикторов для каждого кластера вычисляются алгоритмом LS. Окончательная оценка вычисляется как взвешенная сумма всех результатов предикторов, где весами являются степени сходства. Остатки предсказания затем кодируются контекстно-ориентированным арифметическое кодирование (CBAC). Айацци и др. заявил, что сжатие данных AVIRIS примерно на 20% лучше, чем у Объединенной группы экспертов по фотографии (JPEG) без потерь [20].

После дальнейшей разработки Aiazzi et al. в [17, 18] введены без потерь и почти классифицированные предсказания без потерь для оптических данных и обсуждение их искажения и измерения.

В работе [21] представили Cluster DPCM (C-DPCM). То есть спектральные данные сгруппированы в 16 групп с помощью системы Linde-Vuzo-Gray (LBG) метод. Коэффициенты линейных предикторов 20-го порядка внутри каждого кластера оптимизируются с помощью минимизация среднеквадратичной ошибки (MSE) и ошибки прогнозирования кодируются диапазоном кодер. Коэффициент сжатия улучшен на 42% по сравнению с JPEG-LS. Миликайнен [22] также представил C-DPCM вместе с Adaptive Prediction Length (C-DPCM). На стадии кластеризации спектральные полосы группируются в 16 групп. Далее, линейные предикторы для каждой полосы находятся в смысле минимизации среднеквадратичной ошибки (MSE) внутри каждого кластера. Длина предиктора выбирается из диапазона от 10 до 200, и выбирается тот, который приводит к минимальной остаточной стоимости. Для Aviris гиперспектральных изображений имеется 224 предиктора разной длины, и результаты показывают, что C-DPCM-APL имеет среднее улучшение на 3% по сравнению с C-DPCM [24] упорядочили свои полосы в виде древовидных структур; это полоса связана с n каналами, которые определяются на основе силы спектральная корреляция между полосами. Таким же образом каждый подчиненная полоса также может быть распространяется на собственную полосу сыновей и ветви древовидной структуры. Из-за сильной корреляции оптимальный линейный предиктор для отцовского диапазона вычисляется из его подчиненных диапазонов и прогностическая производительность улучшается, когда размер диапазона вырос. В эксперименте Хуо и др. были

выбраны 4 подчиненные полосы для компромисса между производительность прогнозирования и сложность вычислений.

Однако перегруппировка группы никогда не уделялось большого внимания встроенному сжатию, потому что недостаточно памяти для хранения оптимального порядка. В работе [25] предложили блочный межполосный компрессор под названием ВН. Каждая полоса гиперспектрального изображения разбиты на квадратные блоки. Затем блоки прогнозируются на основе соответствующих блок в предыдущей полосе.

Ученые в [26] предложили 2D-контекстное адаптивное изображение без потерь. Кодирование (2D-CALIC), которое исследует контекст пространственного соседства в изображении и метод причинный. Другими словами, данный пиксель обычно имеет значение, близкое к одному из его соседей (горизонтальные или вертикальные ребра). Точно так же окрестности расширены до 3D образов и предикторов каузальны; то есть эталонные пиксели перекодируются перед текущий пиксель в том же канале или в предыдущих каналах. Следовательно, 3D-CALIC [27] расширенный 2D-CALIC для гиперспектральных изображений. Четыре важных компонента в этом 3DCALIC являются предсказание с поправкой на градиент (GAP), контекстное моделирование и квантование, смещение отмены и энтропийное кодирование. GAP определяет внутрдиапазонные или междиапазонные предикторы по зависимости в причинной окрестности. Прогнозируемое значение далее уточняется с помощью процедуры устранения смещения, которая включает среднее значение ошибки выборки в пределах контекстное моделирование. Поскольку классификация и переупорядочивание полос не задействованы, нет и дополнительных запросов на память, и он подходит для бортовой реализации космического корабля. Тем не менее предикторные коэффициенты и пороги, приведенные в [26] [27], были выбраны эмпирически.

Многоканальный-CALIC (M-CALIC) [28] использует только межполосный (спектральный) предиктор, но коэффициенты и пороги оптимизированы для гиперспектральных изображений, так что M-CALIC превосходит 3D-CALIC.

Артефакты, вызванные калибровкой, появляются во время радиометрической калибровки, который умножает значения цифрового числа (DN) на коэффициент, зависящий от полосы и изображения, который больше одного. Это приводит к тому, что некоторые пиксели появляются с высокой частотой в каждой полосе изображения.

Гиперспектральные изображения AVIRIS. Некоторые подходы к сжатию изображений обеспечивают значительное улучшение путем изучения артефактов, вызванных калибровкой [29-33]. Таблица поиска (LUT) сжатие гиперспектральных изображений предложено в [29-31], а также причинный метод. Таблица поиска ближайших соседей (LUT-NN) предложена Миликайненом. [29]. LUT-NN предсказывает текущий пиксель, ища ближайший соседний пиксель в предыдущая полоса, имеющая такое же значение пикселя, расположенного в том же пространственном положении текущий пиксель в предыдущих полосах. Предполагаемый пиксель находится в

соответствующем месте пиксель NN в текущей полосе. Все результаты поиска перекодируются в таблице поиска для заменить трудоемкую процедуру поиска. Чтобы повысить производительность LUT, двойной LUT перекодируют два ближайших соседних пикселя, и руководство по выбору основано на измерение локально усредненного межполосного масштабирования (LAIS) [30]. С учетом LUT и метод LAIS-LUT [31] разрабатывает несколько LUT для многоканальных изображений. Следовательно, в этом расширенном методе N предыдущих полос используются для генерации M количества LUT; следовательно, есть NM различных предикторов на выбор. Миликайнен также показал, что метод на основе LUT сочетается со спектральным RLP (S-RLP) и спектральным FMP (S-FMP) может превосходить LAIS-LUT до 20%.

В [32] подробно описана процедура калибровки и явление. Артефакты, вызванные калибровкой, обсуждались далее. Также предложено двухступенчатое прогнозирование (TSP) с использованием компрессора Fast Lossless (FL) для начального прогнозирования. Финальный прогноз получается путем поиска пикселей в текущей полосе, которые максимизируют произведение весовой функции и функции, перекодирующей количество остатков предсказания с первого этапа. Два типа весовой функции определяются с использованием структуры данных, вызванной калибровкой. Междиановый поиск третьего порядка (IP3) с обратным пиксельным поиском (BPS) [33] — аналогичный метод, направленный на использование данных, вызванных калибровкой.

Первый этап использует линейный междиановый предиктор IP3, коэффициенты которого решаются методом Винера. Уравнение Хопфа для получения начального прогнозируемого значения. Метод BPS включает поисковую модель алгоритма на основе LUT. В отличие от LUT, метод BPS выполняет поиск случайные пиксели текущего пикселя в текущей полосе для поиска наилучшего соответствия для значения прогноза с первого этапа. Пороговые значения используются для выбора наилучшего соответствующего пикселя, а также облегчить усилия по поиску. В этом результате сжатия IP3-BPS, коэффициент сжатия для стандартного изображения AVIRIS достигает очень низкого значения, что составляет около 3,76 бит на пиксель по сравнению с несжатыми 16 битами.

Однако AVIRIS изображения (с 2016 г.) не имеют заметных калибровочных артефактов, придающих особую разработанное сжатие не имеет преимуществ, и эти новые изображения AVIRIS (с 2016 г.) демонстрируют довольно низкую производительность.

Ван и др. представил метод сжатия без потерь, состоящий из условного среднего предсказания (CCAP) с последующим кодированием Голомба-Райса. Основная цель — разработать сжатие без потерь для приложений реального времени. Аналогично IP3-BPS имеет двухступенчатую компрессию. Первый этап выбирает между междиановым линейное предсказание, аналогичное IP3-BPS, и внутриволосное срединное предсказание, определенное в JPEG-LS [20], основанный на коэффициенте корреляции и генерирующий остаточное значение.

На втором этапе остаточная стоимость дополнительно уточняется с помощью ССАР. Оптимальная оценка остаточного значения, учитывая его соседние пиксели, является его условным ожидаемым значением, но оно упрощается с помощью метода контекстного сопоставления, основанного на корреляции между соседними группы.

### **Выводы по первому разделу**

Исходя из предыдущих проведенных исследований, можно сделать вывод, что наиболее эффективным предиктивное кодирование сжатия гиперспектральных изображений должно исследовать как внутриполосные и межполосное резервирование. В общем случае спектральная корреляция гиперспектральных изображений относительно сильнее, чем пространственная корреляция, так что эти алгоритмы, использующие чисто межполосную предикторы или адаптивное переключение между интер- и интра-предикторами приводит к состоянию высочайшее качество кодирования.

Ограничения прогностического кодирования заключаются в том, что они реализуются последовательно; то есть перед кодированием изображения необходимо выполнить предварительную обработку, предварительную классификацию.

Поэтому трудно распараллелить обработку, чтобы повысить скорость обработки кодирования. Во-вторых, они обладают высокой производительностью при использовании для сжатия с потерями. Это происходит потому, что они не реализуют масштабируемость скорости, восстанавливая частичные результаты. Если скорость реализована масштабируемость, что приведет к несоответствиям между предсказателями накодировщик и декодер и приводят к ухудшению качества реконструкции.

В следующей главе рассмотрим подробно существующие методы и алгоритмы в области сжатия гиперспектральных изображений для выявления сильных и слабых сторон характеристических особенностей в кодировании, основанном на преобразовании.

## **2 Методы и алгоритмы обработки и сжатия гиперспектральных изображений**

### **2.1 Способы кодирования гиперспектральных изображений на основе преобразования**

Сжатие сигнала на основе преобразования является вторым популярным методом сжатия гиперспектральных изображений благодаря его превосходной производительности при сжатии с потерями при низкой скорости передачи данных. Процесс кодирования в целом происходит на изображении, которое отображается из временной области в другое пространство, чтобы продемонстрировать статистические свойства в образцах, которые можно лучше понять, использовать и удалить. Вместо использования избыточности внутри- и междиапазонных каналов методы сжатия в большей степени полагаются на свойства преобразования, такие как анализ с несколькими разрешениями и компактность энергии.

Примеры преобразований включают Преобразование Кархунена-Лоева (KLT), Дискретное преобразование Фурье (DFT), Дискретное косинусное преобразование (DCT) и Дискретное вейвлет-преобразование (ДВТ). Коммерчески успешные промышленные стандарты сжатия изображений (стандарт JPEG) и сжатия видео (стандарт MPEG) [22] относятся к сжатию сигналов на основе преобразования

JPEG использует  $8 \times 8$  DCT, а более поздний JPEG2000 использует 2D DWT. В целом, эти методы сжатия на основе преобразования реализуются в три основных этапа: преобразование выборок сигнала, квантование коэффициентов преобразования и энтропийное кодирование квантованных коэффициентов. Привлекательными особенностями кодирования на основе преобразования являются обратимое сжатие, масштабируемость разрешения, прогрессивная передача, произвольный доступ к потоку битов и кодирование с учетом области интереса (ROI).

Шапиро [24] представил свое основанное на преобразовании сжатие сигнала, называемое встроенным кодированием изображений с использованием встроенных нулевых деревьев вейвлет-коэффициентов (EZW). Алгоритм EZW последовательно квантует и кодирует сигналы, включающие характеристики вейвлет-коэффициентов, в первую очередь компактное распределение и анализ с несколькими разрешениями.

В предыдущих экспериментах алгоритм EZW, кодирующий гиперспектральные изображения, изучались по разным вейвлет-фильтрам, разным уровням декомпозиции, разным структурам декомпозиции, различные конструкции древовидной структуры и модификации Алгоритм ЭЗВ. Эксперименты показали, что симметрия и линейная фаза биортогональные вейвлеты больше подходят для сжатия гиперспектральных изображений и высокий уровень разложения может иметь небольшое улучшение сжатия. Дизайн скважины древовидная структура должна включать изучение уплотнения энергии и самоподобия генерируются различными структурами

вейвлет-разложения. Асимметричное дерево лучше чем симметричное дерево, потому что асимметричное дерево длиннее в спектральном направлении и охватывает больше спектральной информации. Кроме того, можно изменить кодировку алгоритма, чтобы сделать его более эффективным. Идея добавления символов состоит в том, чтобы уменьшить ненужные символы и сделать выходную последовательность более компактной. Его результаты также доказывают что есть лучшие улучшения сжатия, чем другие методы. можно добавлять дополнительные символы в алгоритм кодирования, но при этом увеличивается и сложность кодирования.

Поскольку эти символы необходимо преобразовать в двоичную последовательность для арифметического кодирования, чем больше символов, тем длиннее кодовое слово. Нет никакого улучшения от определения слишком большого количества символов.

Эти свойства облегчают сжатие и приводят к повышению производительности и эффективности. Нулевое дерево относится к древовидной структуре незначимых коэффициентов. Сжатие может быть с потерями или без потерь в зависимости от приложения. Распространенный способ реализовать обратимое дискретное вейвлет-преобразование целого числа в целое число (DWT) можно по схеме подъема [25]. Преимущество схемы подъема заключается в том, что она позволяет полностью выполнять вычисления на месте для быстрой реализации DWT и экономит память; в то же время она реализована во временной области. Распространенной декомпозицией DWT является симметричная диадическая DWT. Для реализации 3D-DWT на гиперспектральных изображениях он выполняется тремя отдельными 1D-диадическими вейвлет-разложениями по каждому измерению.

Статистические свойства преобразованного изображения обычно считаются симметричными. Другие возможные декомпозиции включают равномерное вейвлет-разложение и адаптивные вейвлет-пакеты [19]. Аналогично диадическому DWT равномерное вейвлет-разложение выполняется таким образом, что каждое направление изображения равномерно разлагается на 1D-полный WT и предлагает больше поддиапазонов, чем симметричный DWT. Идея адаптивного вейвлет-пакета состоит в том, чтобы исключить поддиапазоны, создаваемые однородными вейвлетами, которые не вносят существенного вклада в уплотнение энергии. Основной проблемой адаптивной вейвлет-декомпозиции является нахождение эффективной функции затрат, которая будет определять, какой поддиапазон разбиения могут быть пропущены (также называемый алгоритмом выбора основы). Предложенный простой алгоритм использует вычисления энтропии [19].

Билгин и др. [20] сжатые гиперспектральные изображения с помощью симметричного целого числа Декомпозиция 3D-DWT и 3D-EZW, которые просто расширили 2D EZW Шапиро, определив 3D нулевые деревья. Поскольку предполагалась симметричная статистика, автор использовал аналогичное симметричное нулевое дерево, как определил Шапиро. Кристоф и др. [14] удалили модель поиска, которая определяет расположение значимых

пикселей в изображении, чтобы Алгоритм EZW был упрощен и не требовал дополнительного хранилища. Но результаты с потерями были примерно на 2 дБ ниже по сравнению с обычным алгоритмом EZW после преобразования изображения с использованием вейвлет-преобразования. Чжу и др. [15] применили DPCM к преобразованному изображению перед применением алгоритма EZW.

Целью DPCM является централизация энергии преобразованного изображения и экономия времени на поиск значимых пикселей. В целом производительность улучшается, но из-за алгоритма DPCM добавляются дополнительные сложности.

Алгоритм SPIHT, основанный на адаптивном вейвлет-пакете. Легко построить иерархическое нулевое дерево для симметричного WT из-за пирамидальной структуры. Выбор основы адаптивного вейвлет-пакета зависит от данных; поэтому нулевое дерево должно быть адаптировано для повышения производительности. Предложенная функция затрат на основе цепочки Маркова, которая оценивает квантованные коэффициенты, также предоставляет правила для генерации совместимых нулевых деревьев. Однако эти правила построения нулевых деревьев в адаптивном вейвлет-пакете слишком сложны в гиперспектральное изображение.

Ким и Перлман [47, 48] рассмотрели 3D-SPIHT для масштабируемого кодирования видео с низкой скоростью передачи данных. Благодаря отличной производительности 3D-SPIHT при применении к 2D-изображениям и видео, Сон и Ли [49] применили алгоритм 3D-SPIHT, основанный на симметричном 3DDWT, к гиперспектральным изображениям и показали, что 3D-SPIHT может быть успешно применен для сжатия гиперспектральных изображений. Танг и Перлман [50] представили предполагаемые результаты, согласно которым 3D-SPIHT дает 6,94 бита на пиксель при сжатии AVIRIS Moffett изображение. Два ранее обсуждавшихся метода основаны на структуре нулевого дерева, поэтому они называются кодированием на основе нулевого дерева.

Одним из потенциальных преимуществ использования симметричной древовидной структуры является то, что ее легче применять к различным измерениям. Но асимметричная древовидная структура лучше симметричной, потому что она длиннее и может охватывать больший набор коэффициентов по спектральному измерению. [10-13] изучали асимметричную декомпозицию 3D-DWT, которая вызывает асимметричную статистику преобразованного гиперспектрального изображения; таким образом, асимметричная древовидная структура более подходит для описания преобразованного гиперспектрального изображения.

Методы, основанные на преобразовании, наряду с асимметричными деревьями (AT), имеют AT47 3DSPIHT, AT-3DSPECK и AT-3DEZBC. Кристоф и др. [11] применили анизотропный 3D-DWT к алгоритмам 3D-SPIHT и 3D-EZW и пришли к выводу, что 3D асимметричная древовидная структура, использующая как спектральные, так и пространственные соотношения коэффициентов, более эффективна и использует меньше символов для

кодирования. Их результаты показывают, что 3D асимметричная древовидная структура позволила улучшить PSNR на 0,5-1,0 дБ по сравнению с 3D SPIHT (используя симметричное дерево). В [10] авторы реализовали три уровня пространственная декомпозиция и четыре уровня вдоль спектрального направления для создания асимметричной декомпозиции 3D-DWT и оценки производительности AT-3D SPIHT, 3D-SPIHT и 3D-SPECK. Результаты показывают, что AT-3D SPIHT достиг гораздо большего выигрыша в кодировании по сравнению с 3D-SPIHT и 3D-SPECK. Ву и др.[54] предложили алгоритм AT-3D SPECK, основанный на том же асимметричном 3D-DWT, что и [10], для сжатия гиперспектральных изображений с потерями и без потерь. При сравнении AT-3D SPECK с другими 3D кодерами, такие как AT-3D SPIHT и 3D-SPECK, AT-3D SPECK дали хорошие результаты с потерями сжатие при любой скорости передачи данных, особенно при более высоких скоростях передачи данных.

Хоу и Лю [53] представили AT-3DEZBC для сжатия гиперспектральных изображений и оценили производительность сжатия без потерь, используя следующие целочисленные вейвлет-преобразования, такие как S +P(B), (2+2,2), 5/3, и так далее. Их кодер превосходит другое государство-оф-искусство вейвлет алгоритмы (с 3D-спек, с 3D-SPIHT, по-3D SPIHT, и JPEG2000-MK) в основе целочисленных режим с потерями на 0.2 ~ 1.3 DB на средние и кодирование без потерь производительность 3D - EZBC составляет около 5 % ~ 7 % лучше, чем в 3D-спек, с 3D-SPIHT, и в-3D SPIHT.

В последнее время растет интерес к гиперспектральному сжатию изображений с использованием 3D-преобразования, включая спектральный KLT, поскольку KLT может обеспечить более высокое сжатие спектральной энергии.

Хао и Ши [41] представили целочисленный обратимый KLT, объединенный с частью 2 JPEG2000 для многокомпонентного сжатия изображений. В общем случае KLT используется в спектральном измерении, за которым следует 2D дискретное вейвлет-преобразование (2D-DWT) в пространственном измерении, и эти схемы могут значительно превосходить схемы, основанные на асимметричном 3D-DWT [56-58]. Пенна [42] предложил версию с низкой сложностью KLT для того, чтобы сократить вычисление ковариационной матрицы. Хао и Ши [43] представил обратимое целочисленное отображение KLT с помощью матричных факторизаций [44] и обсудил распределение энергии в спектральных и пространственных измерениях при использовании гибридных преобразований, KLT и 2D-DWT, а также асимметричную древовидную структуру, разработанную для гибридного преобразования, предложенного в [45]. Ченг и Дилл [63] представили трехмерный двоичный встроенный вейвлет с нулевым деревом без потерь (3D-BEZW), основанный на целочисленном преобразовании Кархунена-Лоева (IKLT) и целочисленном дискретном вейвлет-преобразовании (IDWT). Для эффективного кодирования гиперспектральных объемных изображений определен новый тип асимметричной древовидной структуры для адаптации оптимального гибридного преобразования. Его коэффициент сжатия без потерь



близок к методам прогностического кодирования, но вычислительная сложность алгоритма 3D-BEZW является умеренной. Таким образом, это новый и эффективный алгоритм сжатия гиперспектральных изображений без потерь.

## 2.2 Стандарт сжатия изображений JPEG

По мере увеличения размеров и возможностей цифровых датчиков изображения появляются новые способы эффективного хранения/передачи данных, которые они генерируют, должны быть проверены. JPEG2000 является последним стандарт сжатия изображений от Объединенной группы экспертов по фотографии, который улучшает по сравнению с более ранними стандартами в своей способности сжимать изображения при сохранении изображения качество.

Тем не менее, сжатие дает преимущество перед другими способами сжатия изображений. Стандарт требует дополнительных вычислительных затрат. Компрессор JPEG2000, значительно сложнее в вычислительном отношении, чем его предшественник, JPEG [12].

Существуют 2 основные процедуры для необратимого снижения скорости сжатия JPEG2000 изображения: квантование и оптимизация скорости искажения после сжатия (PCRD-Вариант).

Квантование – это метод уменьшения динамического диапазона преобразованного данные изображения перед кодированием. Квантование — это простой в вычислительном отношении метод данных, но не контролирует размер сжатого файла и является неоптимальным в с точки зрения качества изображения. PCRD-Opt, однако, дает пользователю точный контроль над размер выходного файла и обеспечивает сжатые изображения высочайшего качества для каждого вывода битрейт.

JPEG 2000 – это новейший стандарт сжатия изображений, разработанный Объединенной группой экспертов по фотографии (JPEG). Целью этого стандарта является улучшение качества изображения по сравнению с предыдущими стандартами (JPEG 6.2), а также расширение возможностей сжатого \_lstream [7]. Что касается отношения пикового сигнала к шуму (PSNR), JPEG2000 превосходит JPEG 6.2 как минимум на 2 дБ для всех степеней сжатия при использовании необратимых методов с потерями [10]. Кроме того, JPEG2000 добавляет возможности сжатому кодовому потоку, такие как прогрессивная передача по разрешению и кодированию интересующей области [3]. Благодаря расширенным возможностям и качеству алгоритмы, связанные со стандартом JPEG2000, также являются более сложными в вычислительном отношении по сравнению с предыдущими стандартами [9].

Стандарт JPEG определяет два различных варианта выполнения дополнительного цвета трансформация: обратимая и необратимая. Обратимое преобразование используется без потерь кодировщики, но ухудшает производительность сжатия по сравнению с необратимым преобразовать в сжатие с потерями. Хотя преобразование цвета не требуется, рекомендуется стандартом, так как это может улучшить производительность коэффициента

сжатия путем удаления потенциально избыточных данных из стандартного изображения RGB. Цвет преобразует имеющиеся изображения в цветовом пространстве RGB в YCrCb или яркостно-цветовое пространство цветности.

### **2.3 Преобразование вейвлет Хаара в обработке гиперспектральных изображений**

Гиперспектральные аэрокосмических снимки в настоящее время превратились в инструмент эффективного решения тематических задач в различных областях научной и хозяйственной деятельности. Развитие новых эффективных методов распознавания и классификации объектов земной поверхности, оценки их состояния с использованием гиперспектральных изображений дистанционного зондирования Земли (ДЗЗ) является актуальной задачей. Алгоритмы обработки гиперспектральных данных можно разделить на две категории: предварительной и тематической обработки.

Развитие новых эффективных методов распознавания и классификации объектов земной поверхности, оценки их состояния с использованием мульти- и гиперспектральных изображений ДЗЗ является актуальной задачей. Алгоритмы обработки мульти- и гиперспектральных данных можно разделить на две категории: предварительной и тематической обработки. В рамках НИР «Разработка методов и программного комплекса тематической обработки гиперспектральных данных, получаемых бортовым комплексом дистанционного зондирования земли для микроспутников» разработаны и используются программные средства для отображения ГС данных и результатов их обработки [1-4].

Основными видами предварительной обработки изображений являются: геометрическая коррекция спутниковых изображений, радиометрическая коррекция, атмосферная коррекция, восстановление пропущенных пикселей, контрастирование, фильтрация.

Быстрый прогресс в развитии методов анализа гиперспектральных изображений привел к созданию специализированных гиперспектральных систем. Это приводит к массовой передаче гиперспектральных изображений от датчиков в центры анализа и, наконец, в центры обработки данных. Хранение этих изображений большого размера – критическая проблема, которую решают методы сжатия. В этой статье рассматриваются различные алгоритмы сжатия гиперспектральных изображений, которые были разделены на две широкие категории сжатия без потерь и с потерями.

Датчики гиперспектрального аэрокосмического изображения (ГАИ) собирают данные в смежных диапазонах длин волн от 400 до 2500 нм, за пределами видимого диапазона человеческого зрения. Каждая полоса имеет одинаковое количество пикселей и фиксированное спектральное разрешение, зависящее от возможностей датчиков. Каждый пиксель имеет некоторое пространственное разрешение, которое определяет область поверхности, покрытую пикселем. Он собирает значение коэффициента отражения области для разных длин волн в разных диапазонах, формируя куб данных, который

полезен во многих приложениях. Например, гиперспектральные изображения используются в военных операциях для поиска войск и отслеживания их продвижения. В сельскохозяйственном секторе гиперспектральных изображений используется для мониторинга качества, борьбы с болезнями, классификации сельскохозяйственных культур и улучшения производства. Обнаружение неисправностей и в космической промышленности ГАИ используется при движении небесных тел. В случае дистанционного зондирования гиперспектральных изображений применяется для исследования поверхности Земли, классификации полезных ископаемых, отслеживания и отслеживания стихийных бедствий в виде наводнений, засуха и др.

Гиперспектральные аэрокосмические изображения – это изображения, полученные с космических аппаратов дистанционного зондирования Земли, предназначенные для решения задач в области прикладных исследований. Исследования для сжатия гиперспектральных АИ являются наиболее интересными, это подтверждается большим количеством публикаций [1-5]. В гиперспектральных АИ для каждого пикселя гиперспектральная камера принимает интенсивность света для большого числа смежных спектральных диапазонов, достигающих несколько сотен.

Гиперспектральная визуализация (ГВ) является важной концепцией дистанционного зондирования из-за ее способности хранить информацию в деталях. В последние годы эта тема вызвала большой интерес у исследователей, поскольку она находит свое применение в обнаружении целей, классификации, обнаружении аномалий и спектральном разделении.<sup>1</sup>

Гиперспектральные АИ в связи с богатым содержанием информации эффективно используются в задачах автоматизированной обработки изображений ДЗЗ. Этапами обработки гиперспектрального (АИ) могут являться: внутреннее представление изображений, преобразование снимков, геометрическая коррекция сцен и предварительная обработка изображения.

Сжатие с потерями. Сжатие с потерями можно определить как процесс сжатия, при котором исходное изображение не может быть восстановлено в реконструированном изображении. Некоторая информация отбрасывается при сжатии, которая не может быть восстановлена во время декомпрессии. В основном используется, когда приложение устойчиво к ошибкам, то есть конкретная потеря данных не влияет на вывод. Это результаты с высокой производительностью сжатия за счет уменьшения размера сжатого изображения и, таким образом, сохраняет компромисс между пространством и точностью. В сжатии с потерями преобладает аппаратное обеспечение и реализация [6-10].

Сжатие без потерь. Такое вид сжатия состоит из методов, которые могут точно восстановить исходное изображение без любая потеря информации. Используется в приложениях, где даже небольшая потеря ошибки недопустима. такие как военные операции, слежение за глобальной системой позиционирования и идентификация целей. Сжатие без потерь приводит к снижению производительности сжатия, то есть увеличение степени сжатия. Сжатие без потерь предпочтительнее, так как эти изображения хранят важную

информацию, которая используется при анализе, классификации, идентификации целей и т. д.[11-15].

Эффективность сжатия может состоять в совокупности этапов – это изменение внутреннего представления изображений, преобразование его к компактному виду и использование предварительной обработки. Сжатие представляет собой предобработку и удаление избыточности в изображениях. Существует два типа избыточности в гиперспектральных АИ, пространственные и спектральные. Эти избыточности позволяют разработку эффективных алгоритмов сжатия.

Один из способов получения двумерного вейвлет-преобразования изображения размером  $2^m \times 2^n$ , заключается в том, чтобы сначала применить одномерное вейвлет-преобразование к каждой из  $2^m$  строкканалов гиперспектральных АИ, а затем применить одномерное вейвлет-преобразование к каждому из  $2^n$  столбцов.

Первый способ заключается в том, чтобы сначала преобразовывались строки изображения, а затем преобразовывать столбцы изображения с преобразованными строками.

Рассмотрим детализацию этапов преобразования значений канала гиперспектрального АИ  $\mathbf{I}[m, n, k]$  на основе двумерного вейвлета Хаара.

Шаг 1. Сформируем последовательность на основе исходного изображения со следующей функцией:  $f(\mathbf{I}[m, n, k], \mathbf{I}'[m, n, k]) = \sum_{i=1}^{\lfloor \frac{i-1}{m}, \frac{i}{n} \rfloor} \sum_{j=1}^{\lfloor \frac{j-1}{m}, \frac{j}{n} \rfloor} \mathbf{I}[m, n, k]_{i,j} \mathbf{I}'[m, n, k]_{i,j}$  где  $I_i \times I_j \equiv \lfloor \frac{i-1}{m}, \frac{i}{n} \rfloor \times \lfloor \frac{j-1}{m}, \frac{j}{n} \rfloor = \{(\mathbf{I}[m, n, k], \mathbf{I}'[m, n, k]): \mathbf{I}[m, n, k] \in \lfloor \frac{i-1}{m}, \frac{i}{n} \rfloor \text{ и } \mathbf{I}'[m, n, k] \in \lfloor \frac{j-1}{m}, \frac{j}{n} \rfloor\}$ ,  
 $X_{I_i \times I_j}(\mathbf{I}[m, n, k], \mathbf{I}'[m, n, k]) = \begin{cases} 1 & \text{если } (\mathbf{I}[m, n, k], \mathbf{I}'[m, n, k]) \in I_i \times I_j \\ 0 & \text{в противном случае} \end{cases}$   
 $X_{I_i}(\mathbf{I}[m, n, k])X_{I_j}(\mathbf{I}'[m, n, k]) = \phi_{2,i-1}(\mathbf{I}[m, n, k])\phi_{2,j-1}(\mathbf{I}'[m, n, k])$

Шаг 2. На основе сформированной функции преобразуем строки изображения для получения вейвлет-коэффициентов:  $f(\mathbf{I}[m, n, k], \mathbf{I}'[m, n, k]) = \sum_{i=1}^4 \sum_{j=1}^4 \mathbf{I}[m, n, k]_{i,j-1} \phi_{2,j-1}(\mathbf{I}[m, n, k]) \times \phi_{2,i-1}(\mathbf{I}'[m, n, k]) = \sum_{i=1}^4 \{ \sum_{j=1}^4 x_{i,j} \phi_{2,i-1}(\mathbf{I}'[m, n, k]) \} \phi_{2,i-1}(\mathbf{I}[m, n, k]) = \sum_{i=1}^4 \mathbf{I}[\tilde{m}, n, k]_i(\mathbf{I}'[m, n, k]) \phi_{2,i-1}(\mathbf{I}[m, n, k])$ , где  $\mathbf{I}[\tilde{m}, n, k]_i(y) = \sum_{j=i}^4 \mathbf{I}[m, n, k]_{i,j} \phi_{2,j-1}(\mathbf{I}'[m, n, k])$ . В результате получили новый набор с коэффициентами, являющимися результатом вейвлет-преобразования  $\mathbf{I}[m, n, k] \cdot \tilde{\chi}(y) = a_{0,0}^i \phi_{0,0}(y) + d_{0,0}^i \psi_{0,0}(y) + d_{1,0}^i \psi_{1,0}(y) + d_{1,1}^i \psi_{1,1}(y)$

Шаг 3. На основе сформированных строк с коэффициентами преобразуем столбцы изображения:  $f(\mathbf{I}[m, n, k], \mathbf{I}'[m, n, k]) = \{ \sum_{i=1}^m a_{0,0}^i \phi_{2,i-1}(x) \} \phi_{0,0}(y) + \{ \sum_{i=1}^n d_{0,0}^i \phi_{2,i=j}(x) \} \psi_{0,0}(y) + \{ \sum_{i=1}^n d_{1,0}^i \phi_{1,0}(x) \} \psi_{1,0}(y) + \{ \sum_{i=1}^n d_{1,1}^i \phi_{2,i-1}(x) \} \psi_{1,1}(y)$ . Результат поместим в матрицы  $I_{Haar II}^\Psi [m, n, k]$

Итак, один из способов получения двумерного вейвлет-преобразования изображения размером  $m \times n$ , заключается в том, чтобы сначала применить

одномерное вейвлет-преобразование к каждой из  $m$  строк, а затем применить одномерное вейвлет-преобразование к каждому из  $n$  столбцов .

Рассмотрим на примере фрагмент гипеспектрального АИ:

$$I[m, n, k] = \begin{pmatrix} 128 & 123 & 0 & 105 \\ 0 & 105 & 0 & 105 \\ 128 & 121 & 128 & 121 \\ 0 & 103 & 128 & 121 \end{pmatrix}$$

При применении одномерного вейвлет-преобразования Хаара к первой строке, значения коэффициентов аппроксимации и различия вычисляются :

$$a_1 = \frac{128+0}{\sqrt{2}}, \quad a_1 = \frac{123+105}{\sqrt{2}}, \quad d_1 = \frac{128-0}{\sqrt{2}}, \quad d_1 = \frac{123-105}{\sqrt{2}}$$

Аналогично к остальным строкам матрицы  $I[m, n, k]$  применяется одномерное вейвлет-преобразование Хаара. В результате получается новая матрица, первые два столбца которой содержат значения коэффициентов аппроксимации каждой строки, а последующие два столбца – значения коэффициентов различия.

В результате получается преобразованная матрица первого уровня.

$$I[m, n, k] = \begin{pmatrix} 64 & 114 & 0 & 0 \\ 64 & 112 & 0 & 0 \\ -64 & -9 & 0 & 0 \\ -64 & -9 & 0 & 0 \end{pmatrix}$$

Преобразованная матрица делится на четыре фильтра, рис. 3

$LL = \begin{pmatrix} 64 & 114 \\ 64 & 112 \end{pmatrix}$	$LH = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
$HL = \begin{pmatrix} -64 & -9 \\ -64 & -9 \end{pmatrix}$	$HH = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

Рисунок 3. Матрица фильтров LL, HL, LH, HH

Из рис.1 видно что, целая четверть коэффициентов двумерного преобразования является результатом действия высокочастотного фильтра (H) на строки, а затем его действия на результирующие столбцы информации. Это блок коэффициентов в правом нижнем углу схемы, обозначенный HH Другая

четверть коэффициентов двумерного преобразования является результатом действия низкочастотного фильтра (L) на столбцы информации, которые уже прошли через высокочастотный фильтр. Этот блок, обозначенный LN находится в правом верхнем углу рис.1. Аналогично, блок HL расположенный в левом нижнем углу, является результатом низкочастотной обработки строк, с последующей высокочастотной обработкой столбцов.

## **2.4 Постановка задачи обработки и сжатия гиперспектральных изображений**

Основываясь на исследованиях гиперспектральных изображений в области сжатия [1-5], представленных в работах зарубежных и отечественных учёных, позволяют предположить, что разработанные методы и алгоритмы сжатия без потерь для гиперспектральных АИ можно улучшить путём уменьшения их вычислительной эффективности и повышения степени сжатия за счет существенных этапов предобработки.

Наряду с преимуществами у гиперспектральных изображений есть некоторые ограничения, которые приводят к концепции сжатия. Необходимость сжатия гиперспектральных изображений при дистанционном зондировании можно сформулировать следующим образом:

Размер гиперспектральных изображений, измеряемый датчиками, составляет сотни и тысячи мегабайт, гигабайт и т.д. Например, датчик бортового видимого / инфракрасного спектрометра (AVIRIS) захватывает 224 спектральных диапазона с  $614 \times 500$  пиксели. В каждой полосе (канале изображения), где каждый пиксель занимает 16 бит. Размер изображения с такого сенсора составляет  $224 \times 614 \times 500 \times 16 = 131,17$  МБ. Следовательно, хранение данных большого размера является актуальной проблемой.

Ограниченная полоса пропускания канала передачи: гиперспектральных изображений должны передаваться из одного места в другое, а большой размер данных требует высокой пропускной способности, что является дорогостоящим ресурсом в приложениях дистанционного зондирования.

Ограниченное время передачи данных: датчики очень часто охватывают гиперспектральных изображений, что требует высокой скорости обработки, что очень сложно для устройства захвата. Эти изображения, если они не передаются в течение ограниченного времени, могут привести к потере информации и калибровке в центрах обработки данных.

Необходимо разработка методов и алгоритмов сжатия, с помощью которых можно уменьшить размер гиперспектральных изображений без потери качества изображения сверх желаемого уровня. Это один из важнейших этапов технологии обработки данных гиперспектральных изображений, который используется в каждой космической системе, поскольку это снижает стоимость полосы пропускания и оборудования для хранения данных.

В режиме без потерь сжатие уменьшает размер, сохраняя одну и ту же информацию с небольшим количеством бит, двумя методами с использованием разных представлений и удаляя существующую избыточность. Высокая

избыточность помогает алгоритмам сжатия достичь высокой степени сжатия. По сути, статистическая избыточность и визуальная избыточность – это две широкие категории избыточности аэрокосмических изображений. В то время как первый играет важную роль в гиперспектральных изображениях, последний не имеет первостепенной важности из-за его ограничения воздействия только в видимом диапазоне. Статистическая избыточность возникает из-за почти одинаковой интенсивности пикселей в окрестности, за исключением мест, где изменяется освещение. Его можно разделить на межпиксельную избыточность и избыточность кодирования.

В гиперспектральных изображениях существует три типа межпиксельной избыточности: (1) пространственная избыточность: она возникает из-за внутриволновой зависимости, которая существует в пространственной области, (2) спектральная избыточность: возникает из-за зависимости между пикселями разных полос в одном и том же пространственном положении. И (3) временная избыточность: она возникает, когда гиперспектральных изображений одного и того же местоположения берется в разное время, зависимость во временной области (для соответствующих спектральных и пространственных пикселей) приводит к временной избыточности. Эти избыточности декоррелированы в алгоритмах сжатия, и, таким образом, размер данных уменьшается. Исходные данные можно восстановить с помощью декомпрессии, которая обычно является обратным процессом сжатия.

Категоризация и классификация алгоритмов сжатия гиперспектральных изображений.

Сжатие гиперспектральных изображений – это обширная область, которую можно разделить на различные категории. Мы классифицировали алгоритмы на три различных способа: классификация на основе различных параметров, набор показателей, используемых для оценки конкретного алгоритма. Методы сжатия HSI классифицируются по применяемой методологии. Существуют различные методы сжатия изображения, каждый из которых имеет как свои преимущества, так и ограничения. Мы разделили алгоритмы на семь широких категорий, а именно алгоритмы на основе преобразования, на основе прогнозирования, на основе векторного квантования (VQ), на основе сжатия данных, на основе тензорной декомпозиции, на основе разреженного представления, на основе нескольких элементов и на основе обучения. На рисунке 4 показаны различные методы сжатия, классифицированные на основе методологии и различных алгоритмов в каждой категории.

Метод на основе преобразования – самый популярный метод сжатия двумерных (2-D) изображений, который был расширен до трехмерного (3-D) сжатия или сжатия гиперспектральных изображений. Он известен как метод на основе преобразования, поскольку он преобразует значения пикселей в частотную область, применяя функцию преобразования ко всем трем измерениям изображения.

Есть несколько отличных методов преобразования, таких как дискретное косинусное преобразование (DCT), дискретное преобразование Фурье,

дискретное вейвлет-преобразование (DWT), преобразование Карунена – Лоэва (KLT), и Уолш-Адамара преобразование, которые используются при сжатии изображений. Некоторые современные алгоритмы в этой области включают 3D-DWT, 2D-KLT, 9 встроенных блоков с разбиением на разделы с 3D-набором (SPECK), 2-4-8-16 UAT.

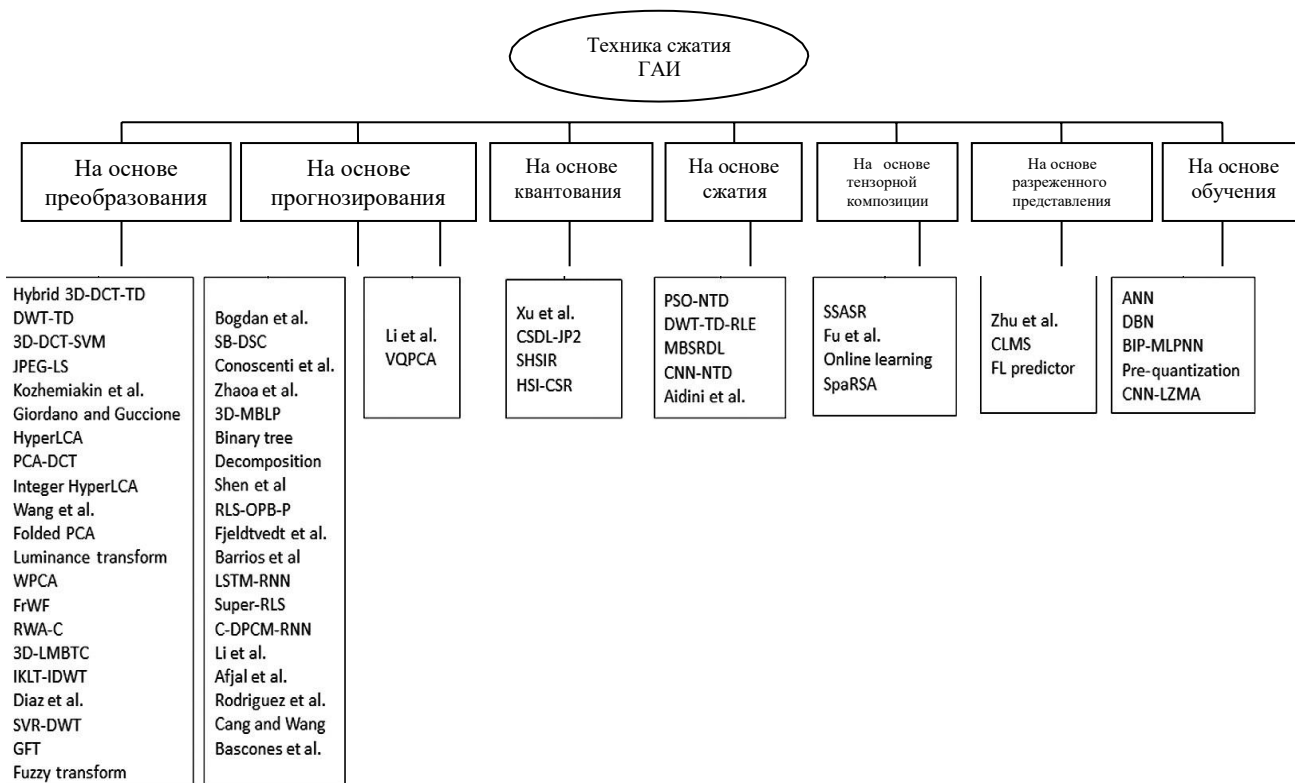


Рисунок 4. Классификация методов и алгоритмов ГАИ

Сжатие с использованием метода на основе преобразования включает несколько шагов, которые могут различаться для разных алгоритмов, но могут быть обобщены, как показано на рис. 5.

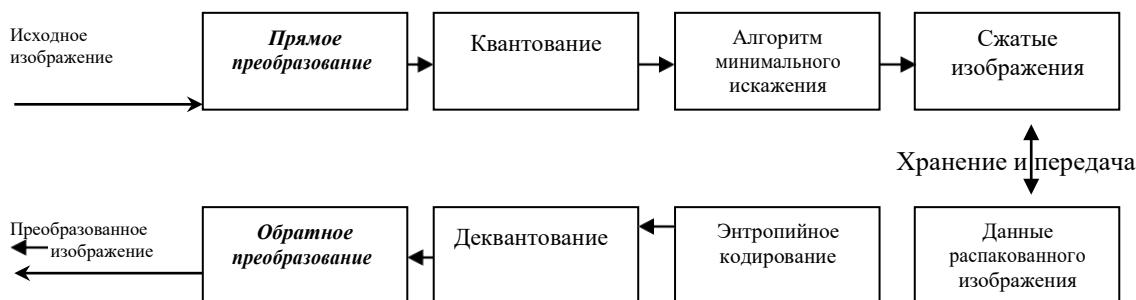


Рисунок 5. Этапы алгоритма сжатия на основе преобразования.

### Выводы к главе

На данный момент алгоритмы и методы сжатия определенно имеют преимущества и недостатки. Характеристика представленных методов,



алгоритмов и способов позволяют сформулировать требования к разработке и обработке гиперспектральных изображений с помощью преобразования Уолша-Адамара, которому посвящено незначительное количество исследования. В рамках проведенного исследования предложим методологию дальнейшего исследования:

- вейвлет-преобразования и/ или Уолша-Адамара преобразование;
- оценки критериев качества восстановленных изображений PSNR, MSE, PMSE и др.;
- адаптивное арифметическое кодирование, введение которого позволит улучшить результаты сжатия, т.к. он является одним из лучших среди статистических методов.

Предложенная методология может применяться в сочетании почти со всеми другими методами, такими как на основе прогнозирования, квантования, сжатия и в алгоритмах, основанных на обучении. Прямое преобразование применяет функцию преобразования (косинус, вейвлет, Фурье или Уолша-Адамара) к одной из пространственных или спектральных областей, затем выполняет декорреляцию и генерирует коэффициенты. После чего квантование завершается; это удаляет пиксели, близкие к нулю. На последнем этапе методы кодирования применяются к квантованным коэффициентам для генерации битовых потоков. Которые передаются или хранятся с уменьшенным числом бит на пиксель для экономии места (в хранилище) и полосы пропускания (при передаче) гиперспектральных изображений со спутниковых бортовых систем на наземные станции.

### 3 Результаты исследования системы обработки и алгоритмов сжатия гиперспектральных изображений

#### 3.1 Оценка эффективности предлагаемого алгоритма и преобразования Уолша-Адамара

Для определения эффективности предлагаемого алгоритма с точки зрения степени сжатия, а также пределов его применимости проведен ряд экспериментов на гиперспектральных АИ (Avisis), в табл. 1.

Таблица 1 - Характеристики тестовых гиперспектральных аэрокосмических изображений

Система ДЗЗ	Количество каналов	Размер изображения	Байт
AVIRIS	50	50*50	540800
AVIRIS	50	100*100	2040200
AVIRIS	50	200*200	8080200
AVIRIS	50	300*300	18120200
AVIRIS	50	400*400	32160200
AVIRIS	50	614*512	62873600
AVIRIS	100	50*50	1081600
AVIRIS	100	100*100	4080400
AVIRIS	100	200*200	16160400
AVIRIS	100	300*300	36240400
AVIRIS	100	400*400	64320400
AVIRIS	100	614*512	125747200
AVIRIS	150	50*50	1622400
AVIRIS	150	100*100	6120600
AVIRIS	150	200*200	24240600
AVIRIS	150	300*300	54360600
AVIRIS	150	400*400	96480600
AVIRIS	150	614*512	188620800
	200	50*50	2163200
AVIRIS	200	100*100	8160800
AVIRIS	200	200*200	32320800
AVIRIS	200	300*300	72480800
AVIRIS	200	400*400	128640800
AVIRIS	200	614*512	251494400
	224	50*50	2421632
AVIRIS	224	100*100	9140096
AVIRIS	224	200*200	36199296
AVIRIS	224	300*300	81178496
AVIRIS	224	400*400	144077696
AVIRIS	224	614*512	281673728

### 3.2 Обработка гиперспектральных изображений и преобразование Уолша Адамара

Преобразование быстрое, так как его можно вычислять, применяя только сложение, вычитание и, иногда, сдвиг вправо .

Для заданного блока  $N \times N$  пикселей  $P_{xy}$ , его двумерное прямое и обратное преобразование Уолша-Адамара определяются с помощью следующих уравнений:

$$H(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P_{xy} g(x, y, u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P_{xy} g(-1)^{i_0} \dots i_i \dots , \quad (1)$$

$$P_{xy} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} H(u,v) h(x, y, u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} H(u,v) (-1)^{i_0} \dots i_i \dots , \quad (2)$$

Где  $H(u, v)$  - это результат преобразования (т.е. коэффициенты WHT), величина  $b_i(u)$  равна биту  $i$  в двоичном представлении целого числа  $u$ , а  $p_i(u)$  определяется с помощью  $b_j(u)$  из следующих рекуррентных соотношений:

$$\begin{aligned} p_0(u) &= b_{n-1}(u), \\ p_1(u) &= b_{n-1}(u) + b_{n-2}(u), \\ p_2(u) &= b_{n-2}(u) + b_{n-3}(u), \\ &\dots \\ p_{n-1}(u) &= b_1(u) + b_0(u). \end{aligned} \quad (3)$$

Рассмотрим, например,  $u = 6 = 110_2$ . Нулевой, первый и второй биты равны соответственно, 0, 1, 1, поэтому  $b_0(6) = 0, b_1(6) = 1$  и  $b_2(6) = 1$ . Величины  $g(x, y, u, v)$  и  $h(x, y, u, v)$  называются ядрами (или базисными изображениями) WHT. Их матрицы совпадают. Элементами матриц служат числа +1 и -1, которые умножаются на  $1/N$ . в результате преобразование WHT состоит из умножения пикселей на +1 или -1, сложения и деления суммы на  $N$ . Но поскольку  $N = 2^n$ , деление можно делать, сдвигая разряды чисел вправо на  $n$  позиций.

Строки и столбцы в блоках занумерованы значениями  $u$  и  $v$  от 0 до  $N-1$ , соответственно. Число смен знаков в строке называется частотностью строки или столбца. Строки и столбцы упорядочены по возрастанию частотности.

Сжатие изображений с помощью WHT делается так же, как и для DCT с заменой уравнений на (1) и (2)

В настоящее время в области обработки гиперспектральных аэрокосмических изображений (АИ) наиболее актуальным является

применение преобразований Уолша при анализе эффективности его использования для решения задачи к подготовке сжатия.

В основе функций Уолша - Адамара лежат ортогональные бинарные матрицы Адамара  $H_N$ , которые определяются по простому правилу[1-2]:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix} \text{ и т.д.}$$

Матрицы Адамара можно получить другим путем, используя для этого операцию кронекеровского произведения матриц:

$$H_4 = H_2 \otimes H_2; H_8 = H_4 \otimes H_2 = H_2 \otimes H_2 \otimes H_2,$$

где  $\otimes$  - оператор кронекеровского произведения матриц.

Рассматривая элементы матриц Адамара как отсчеты непрерывных меандровых сигналов, можно получить в функции Уолша - Адамара  $\{had(k,t)\}$ .

Матрица дискретных функций Уолша - Адамара  $\{had(k,n)\}, t= n\Delta t$  примет вид

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} had(0,n) \\ had(1,n) \\ had(2,n) \\ \vdots \\ had(N-1,n) \end{bmatrix}$$

Введем двоичное представление номера функции

$$k = \sum_{j=0}^{r-1} k_j \cdot 2^j = (k_0, k_1, \dots, k_{r-1}) \stackrel{\Delta}{=} k$$

и двоичное представление номера отсчета

$$n = \sum_{j=0}^{r-1} n_j \cdot 2^j = (n_0, n_1, \dots, n_{r-1})$$

Тогда функции Уолша - Адамара можно определить как

$$had(k,n) = (-1)^{\sum_{j=0}^{r-1} n_j k_j}$$

где  $\binom{\vec{n}}{\vec{k}} = \sum_{j=0}^{n-1} n_j \cdot k_j$  - скалярное произведение векторов кодов номеров функции и отсчета, соответственно.

Например, пусть  $n = 3 \rightarrow 0 \ 1 \ 1, k = 2 \rightarrow 0 \ 1 \ 0$ , тогда  $\binom{\vec{n}}{\vec{k}} = 0 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 = 1$  и элемент матрица с координатами (3,2) равен  $had(2,3) = (-1)^1 = -1$ .

В зависимости от упорядочивания номера функций различают следующие системы ортогональных функций: система Адамара ( $\{had(k, n)\}$ ), система Пэли  $\{pal(k, n)\}$ , система Уолша  $\{wal(k, n)\}$ . Система функций Пэли имеет двоичную инверсию кода номера функции k. Номера функций Уолша изменяются по закону двоичной инверсии кода Грея. Например, для  $N = 8$  имеем

$had(k, n)$	$pal(k, n)$	$wal(k, n)$
000	000	000
001	100	100
010	010	110
011	110	010
100	001	011
101	101	111
110	011	101
111	111	001

Для вектора отсчетов  $\vec{x} = [x[0], \dots, x[N-1]]$ ,  $N = 2^m$  можно определить преобразование Уолша-Адамара следующим образом:

$$X_H(k) = \sum_{n=0}^{N-1} x[n] \cdot (-1)^{\binom{\vec{n}}{\vec{k}}},$$

$\vec{X}_H = H_N \cdot \vec{x}$  - прямое преобразование Уолша-Адамара;

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_H(k) \cdot (-1)^{\binom{\vec{n}}{\vec{k}}}, \quad \vec{x} = \frac{1}{N} H_N^{-1} \cdot \vec{X}_H = \frac{1}{N} H_N \cdot \vec{X}_H$$

обратное преобразование Уолша-Адамара.

### 3.3 Разработка программы и практическое применение в обработке гиперспектральных данных

Разработаем модуль прямого преобразования Уолша-Адамара. Исходная матрица:

0	128	155	68	0	128	155	68	0	128	155	68	0	96	166	68
0	128	155	68	0	96	166	68	0	128	155	68	0	128	155	68
0	160	165	68	0	192	154	68	0	160	165	68	0	0	144	68
0	160	165	68	0	192	154	68	0	192	154	68	0	192	154	68
0	64	153	68	0	64	153	68	0	64	153	68	0	96	142	68
0	96	142	68	0	64	153	68	0	32	164	68	0	64	153	68
0	160	165	68	0	192	154	68	0	192	154	68	0	0	144	68
0	160	165	68	0	192	154	68	0	192	154	68	0	160	165	68
0	0	156	68	0	224	166	68	0	32	145	68	0	0	156	68
0	224	166	68	0	32	145	68	0	0	156	68	0	0	156	68
0	64	152	68	0	64	152	68	0	64	152	68	0	64	152	68
0	64	152	68	0	0	174	68	0	96	141	68	0	32	163	68
0	160	155	68	0	160	155	68	0	192	144	68	0	128	166	68
0	192	144	68	0	160	155	68	0	160	155	68	0	128	166	68
0	160	144	68	0	128	155	68	0	96	166	68	0	160	144	68
0	160	144	68	0	128	155	68	0	128	155	68	0	128	155	68

Шаг 1. Для удобства преобразования формируем измененную матрицу, считывая из исходной по 4 столбца, попутно удаляя 1 и 4 столбцы:

128	155	128	155	128	155	96	166
128	155	96	166	128	155	128	155
160	165	192	154	160	165	0	144
160	165	192	154	192	154	192	154
64	153	64	153	64	153	96	142
96	142	64	153	32	164	64	153
160	165	192	154	192	154	0	144
160	165	192	154	192	154	160	165
0	156	224	166	32	145	0	156
224	166	32	145	0	156	0	156
64	152	64	152	64	152	64	152
64	152	0	174	96	141	32	163
160	155	160	155	192	144	128	166
192	144	160	155	160	155	128	166
160	144	128	155	96	166	160	144

$$\left| \begin{array}{cccccccc} 160 & 144 & 128 & 155 & 128 & 155 & 128 & 155 \end{array} \right|$$

Шаг 2. Делим полученную матрицу на подматрицы, размеры которых совпадают с размерами матрицы Уолша-Адамара. В примере это - 4x4:

128	155	128	155	128	155	96	166
128	155	96	166	128	155	128	155
160	165	192	154	160	165	0	144
160	165	192	154	192	154	192	154
64	153	64	153	64	153	96	142
96	142	64	153	32	164	64	153
160	165	192	154	192	154	0	144
160	165	192	154	192	154	160	165
0	156	224	166	32	145	0	156
224	166	32	145	0	156	0	156
64	152	64	152	64	152	64	152
64	152	0	174	96	141	32	163
160	155	160	155	192	144	128	166
192	144	160	155	160	155	128	166
160	144	128	155	96	166	160	144
160	144	128	155	128	155	128	155

Шаг 3. С каждой подматрицей производим вычисления, согласно которым каждый элемент преобразованной подматрицы вычисляется по формуле:

$$\frac{\sum_{i=1}^n Matrix[Row, i] * Hadamar[i, Column]}{2^x},$$

где  $n$  – размерность матрицы Адамара,  $Matrix$ –подматрица исходной матрицы,  $Hadamar$ –матрица Адамара,  $Row$ –строка текущего значения в подматрице,  $Column$  - столбец текущего значения в подматрице,  $x$ –базис матрицы.

128	155	128	155	36	40	38	39,3125
128	155	96	166	-4	-1,25	-10	0,8125
160	165	192	154	0	0	2	-0,6875
160	165	192	154	0	0	2	-0,6875

Шаг 4. Из полученных значений подматриц создаем конечную матрицу, вставляя значения в соответствующие места, относительно исходной матрицы:

36	40	38	39,31	38	39,31	26	38,69
-4	-1,25	-10	0,813	-6	-0,563	2	1,438
0	0	2	-0,688	-2	0,688	-14	0,063
0	0	2	-0,688	2	-0,688	10	1,313
30	39,06	32	38,38	30	39,06	20	37,75
-10	-2,188	-16	-0,125	-18	0,563	0	-0,875
-2	0,688	0	0	2	-0,688	-8	-2
-2	0,688	0	0	2	-0,688	12	0,625
22	39,13	20	39,81	12	37,13	6	39,19
6	1,125	12	-0,938	-8	0,5	-6	-0,188
-14	-0,625	16	-0,063	0	0	2	-0,688
-14	-0,625	8	2,688	4	-1,375	-2	0,688
42	36,69	36	38,75	36	38,75	34	39,44
2	0,688	4	0	8	-1,375	-2	2,063
-2	0,688	0	0	0	0	2	-0,688
-2	0,688	0	0	4	-1,375	-2	0,688

Шаг 5. Квантуем значения преобразованной матрицы: (до 1)

36	40	38	39	38	39	26	39
-4	-1	-10	1	-6	-1	2	1
0	0	2	-1	-2	1	-14	0
0	0	2	-1	2	-1	10	1
30	39	32	38	30	39	20	38
-10	-2	-16	0	-18	1	0	-1
-2	1	0	0	2	-1	-8	-2
-2	1	0	0	2	-1	12	1
22	39	20	39	12	37	6	39
6	1	12	-1	-8	1	-6	0
-14	-1	16	0	0	0	2	-1
-14	-1	8	3	4	-1	-2	1
42	36	36	39	36	39	34	39
2	1	4	0	8	-1	-2	2
-2	1	0	0	0	0	2	-1
-2	1	0	0	4	-1	-2	1



Шаг 6. Добавляем 1 и 4 столбцы из исходной матрицы в преобразованную через каждые 2 столбца:

0	36	40	68	0	38	39	68	0	38	39	68	0	26	39	68
0	-4	-1	68	0	-10	1	68	0	-6	-1	68	0	2	1	68
0	0	0	68	0	2	-1	68	0	-2	1	68	0	-14	0	68
0	0	0	68	0	2	-1	68	0	2	-1	68	0	10	1	68
0	30	39	68	0	32	38	68	0	30	39	68	0	20	38	68
0	-10	-2	68	0	-16	0	68	0	-18	1	68	0	0	-1	68
0	-2	1	68	0	0	0	68	0	2	-1	68	0	-8	-2	68
0	-2	1	68	0	0	0	68	0	2	-1	68	0	12	1	68
0	22	39	68	0	20	39	68	0	12	37	68	0	6	39	68
0	6	1	68	0	12	-1	68	0	-8	1	68	0	-6	0	68
0	-14	-1	68	0	16	0	68	0	0	0	68	0	2	-1	68
0	-14	-1	68	0	8	3	68	0	4	-1	68	0	-2	1	68
0	42	36	68	0	36	39	68	0	36	39	68	0	34	39	68
0	2	1	68	0	4	0	68	0	8	-1	68	0	-2	2	68
0	-2	1	68	0	0	0	68	0	0	0	68	0	2	-1	68
0	-2	1	68	0	0	0	68	0	4	-1	68	0	-2	1	68

Обратное преобразование Уолша-Адамара. Исходная матрица:

0	36	40	68	0	38	39	68	0	38	39	68	0	26	39	68
0	-4	-1	68	0	-10	1	68	0	-6	-1	68	0	2	1	68
0	0	0	68	0	2	-1	68	0	-2	1	68	0	-14	0	68
0	0	0	68	0	2	-1	68	0	2	-1	68	0	10	1	68
0	30	39	68	0	32	38	68	0	30	39	68	0	20	38	68
0	-10	-2	68	0	-16	0	68	0	-18	1	68	0	0	-1	68
0	-2	1	68	0	0	0	68	0	2	-1	68	0	-8	-2	68
0	-2	1	68	0	0	0	68	0	2	-1	68	0	12	1	68
0	22	39	68	0	20	39	68	0	12	37	68	0	6	39	68
0	6	1	68	0	12	-1	68	0	-8	1	68	0	-6	0	68
0	-14	-1	68	0	16	0	68	0	0	0	68	0	2	-1	68
0	-14	-1	68	0	8	3	68	0	4	-1	68	0	-2	1	68
0	42	36	68	0	36	39	68	0	36	39	68	0	34	39	68
0	2	1	68	0	4	0	68	0	8	-1	68	0	-2	2	68
0	-2	1	68	0	0	0	68	0	0	0	68	0	2	-1	68
0	-2	1	68	0	0	0	68	0	4	-1	68	0	-2	1	68

Шаг 1. Для удобства преобразования формируем измененную матрицу, считывая из исходной по 4 столбца и убирая 1 и 4 столбцы:

36	40	38	39	38	39	26	39
-4	-1	-10	1	-6	-1	2	1
0	0	2	-1	-2	1	-14	0
0	0	2	-1	2	-1	10	1
30	39	32	38	30	39	20	38
-10	-2	-16	0	-18	1	0	-1
-2	1	0	0	2	-1	-8	-2
-2	1	0	0	2	-1	12	1
22	39	20	39	12	37	6	39
6	1	12	-1	-8	1	-6	0
-14	-1	16	0	0	0	2	-1
-14	-1	8	3	4	-1	-2	1
42	36	36	39	36	39	34	39
2	1	4	0	8	-1	-2	2
-2	1	0	0	0	0	2	-1
-2	1	0	0	4	-1	-2	1

Шаг 2. Делим полученную матрицу на подматрицы, размеры которых совпадают с размерами матрицы Уолша-Адамара. В нашем примере это - 4x4:

36	40	38	39	38	39	26	39
-4	-1	-10	1	-6	-1	2	1
0	0	2	-1	-2	1	-14	0
0	0	2	-1	2	-1	10	1
30	39	32	38	30	39	20	38
-10	-2	-16	0	-18	1	0	-1
-2	1	0	0	2	-1	-8	-2
-2	1	0	0	2	-1	12	1
22	39	20	39	12	37	6	39
6	1	12	-1	-8	1	-6	0
-14	-1	16	0	0	0	2	-1
-14	-1	8	3	4	-1	-2	1
42	36	36	39	36	39	34	39
2	1	4	0	8	-1	-2	2
-2	1	0	0	0	0	2	-1
-2	1	0	0	4	-1	-2	1

Шаг 3. С каждой подматрицей производим вычисления, согласно которым каждый элемент преобразованной подматрицы вычисляется по формуле:

$$\sum_{i=1}^n Matrix[Row, i] * Hadamar[i, Column] * x,$$

где  $n$  – размерность матрицы Адамара,  $Martix$ –подматрица исходной матрицы,  $Hadamar$ –матрица Адамара,  $Row$ –строка текущего значения в подматрице,  $Column$  - столбец текущего значения в подматрице,  $x$ –базис матрицы.

$$\begin{vmatrix} 36 & 40 & 38 & 39 \\ -4 & -1 & -10 & 1 \\ 0 & 0 & 2 & -1 \\ 0 & 0 & 2 & -1 \end{vmatrix} \Rightarrow \begin{vmatrix} 128 & 156 & 128 & 152 \\ 128 & 156 & 96 & 168 \\ 160 & 164 & 192 & 152 \\ 160 & 164 & 192 & 152 \end{vmatrix}$$

Шаг 4. Из полученных значений подматриц создаем конечную матрицу, вставляя значения в соответствующие места, относительно исходной матрицы:

$$\begin{vmatrix} 128 & 156 & 128 & 156 & 128 & 156 & 96 & 160 \\ 128 & 156 & 96 & 156 & 128 & 156 & 128 & 152 \\ 160 & 164 & 192 & 156 & 160 & 156 & 0 & 144 \\ 160 & 164 & 192 & 156 & 192 & 156 & 192 & 152 \\ 64 & 148 & 64 & 152 & 64 & 156 & 96 & 140 \\ 96 & 148 & 64 & 152 & 32 & 156 & 64 & 156 \\ 160 & 164 & 192 & 152 & 192 & 156 & 0 & 140 \\ 160 & 164 & 192 & 152 & 192 & 156 & 160 & 156 \\ 0 & 160 & 224 & 164 & 32 & 144 & 0 & 156 \\ 224 & 160 & 32 & 148 & 0 & 152 & 0 & 156 \\ 64 & 152 & 64 & 148 & 64 & 152 & 64 & 156 \\ 64 & 152 & 0 & 164 & 96 & 144 & 32 & 156 \\ 160 & 144 & 160 & 152 & 192 & 144 & 128 & 164 \\ 192 & 144 & 160 & 152 & 160 & 152 & 128 & 164 \\ 160 & 144 & 128 & 152 & 96 & 160 & 160 & 148 \\ 160 & 144 & 128 & 152 & 128 & 152 & 128 & 148 \end{vmatrix}$$

Шаг 5. Добавляем 1 и 4 столбцы из исходной матрицы в преобразованную через каждые 2 столбца:

$$\begin{vmatrix} 0 & 128 & 156 & 68 & 0 & 128 & 156 & 68 & 0 & 128 & 156 & 68 & 0 & 96 & 160 & 68 \\ 0 & 128 & 156 & 68 & 0 & 96 & 156 & 68 & 0 & 128 & 156 & 68 & 0 & 128 & 152 & 68 \\ 0 & 160 & 164 & 68 & 0 & 192 & 156 & 68 & 0 & 160 & 156 & 68 & 0 & 0 & 144 & 68 \end{vmatrix}$$

0	160	164	68	0	192	156	68	0	192	156	68	0	192	152	68
0	64	148	68	0	64	152	68	0	64	156	68	0	96	140	68
0	96	148	68	0	64	152	68	0	32	156	68	0	64	156	68
0	160	164	68	0	192	152	68	0	192	156	68	0	0	140	68
0	160	164	68	0	192	152	68	0	192	156	68	0	160	156	68
0	0	160	68	0	224	164	68	0	32	144	68	0	0	156	68
0	224	160	68	0	32	148	68	0	0	152	68	0	0	156	68
0	64	152	68	0	64	148	68	0	64	152	68	0	64	156	68
0	64	152	68	0	0	164	68	0	96	144	68	0	32	156	68
0	160	144	68	0	160	152	68	0	192	144	68	0	128	164	68
0	192	144	68	0	160	152	68	0	160	152	68	0	128	164	68
0	160	144	68	0	128	152	68	0	96	160	68	0	160	148	68
0	160	144	68	0	128	152	68	0	128	152	68	0	128	148	68

Показатели метрик качества восстановленных изображений были определены с помощью *PSNR* и *SKO*. Степень искажения сравнивает соотношение между сжатием и искажением в алгоритмах с потерями. Оценка определяется как среднее значение числа бит, необходимого для представления каждого пиксела. Измеряется в битах на пиксел (*bpp* – bits per pixel). Искажение обычно измеряется с помощью *PSNR*. На рис.1 приведена зависимость *PSNR* от степени сжатия *D* для сравнения дискретно-косинусного преобразования (ДКП) и преобразования Уолша-Адамара. Данные показывают что, при Уолша – Адамара преобразовании сохраняется высокое качество при степени сжатия 8.

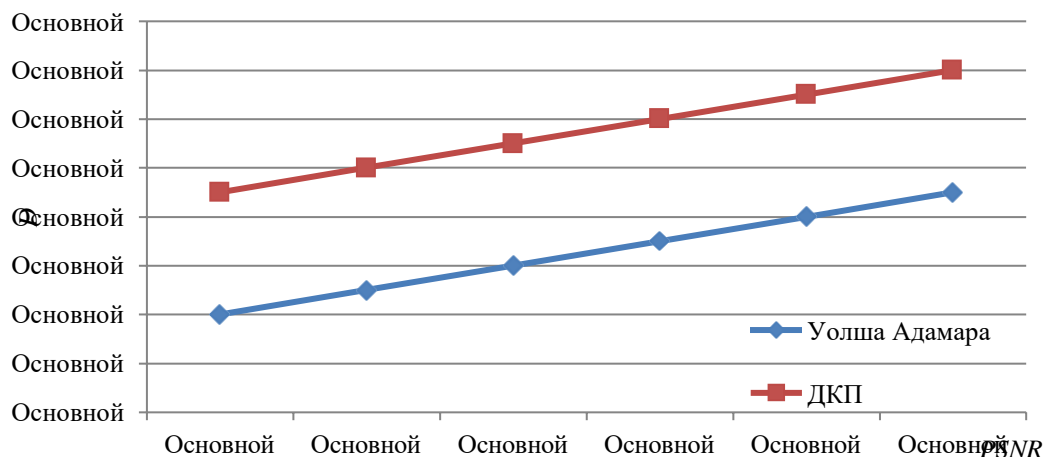


Рисунок 6. Зависимость *PSNR* от *D* для Уолша-Адамара и ДКП

Представлена обработка и методика разработки алгоритмов сжатия с потерями и высоким качеством при восстановлении преобразованием Уолша-Адамара. Проведенные исследования показали, что предложенные алгоритмы с потерями имеют эффективность, достаточную для использования и могут быть применены при передачи гиперспектральных данных ДЗЗ в условиях

ограниченной емкости буферной памяти и пропускной способности канала связи.

### 3.4 Разработка системы обработки и сжатия преобразования Уолша-Адамара гиперспектральных изображений

Проведенные эксперименты разработанной программы, рис.7- 8.

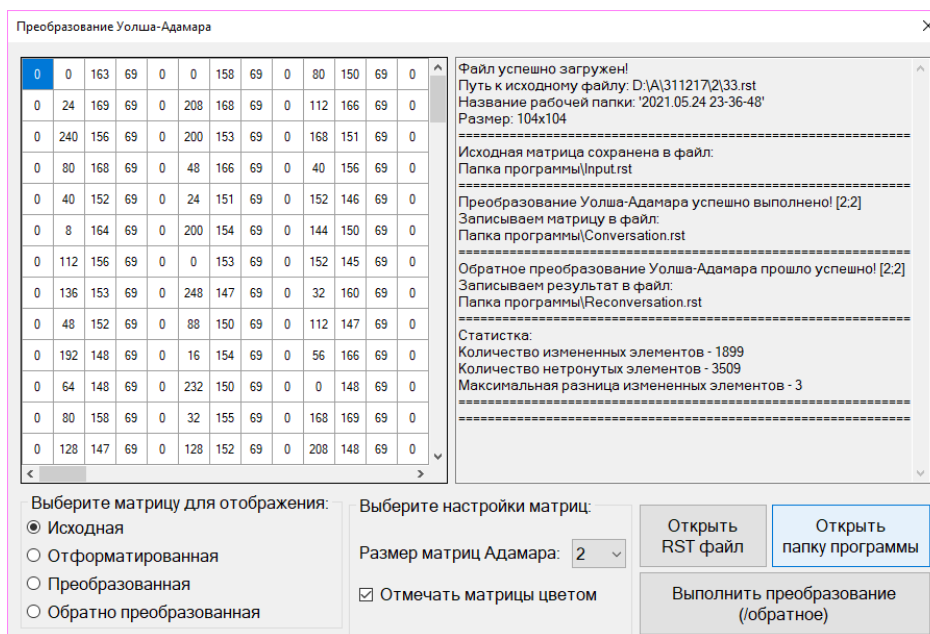


Рисунок 7. Преобразование матрицы гиперспектральных изображений  
Обратное преобразование Уолша-Адамара гиперспектральных изображений.

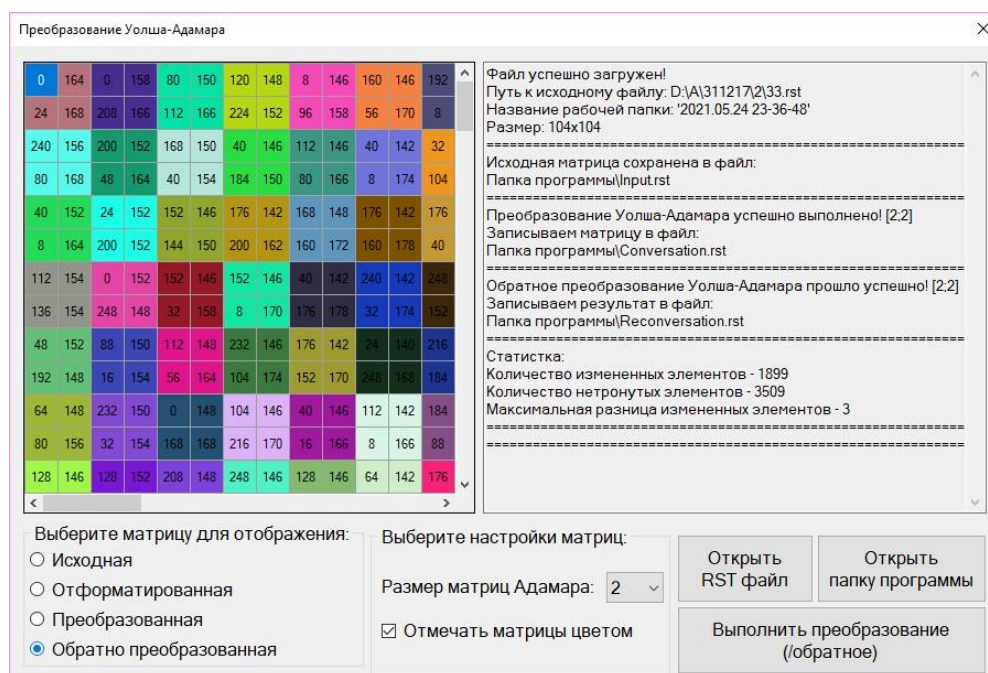


Рисунок 8. Обратное преобразование матрицы гиперспектральных изображений

На рис. 9 представлены исходные и преобразованные изображения после преобразования Уолша-адамара и восстановленных файлов.

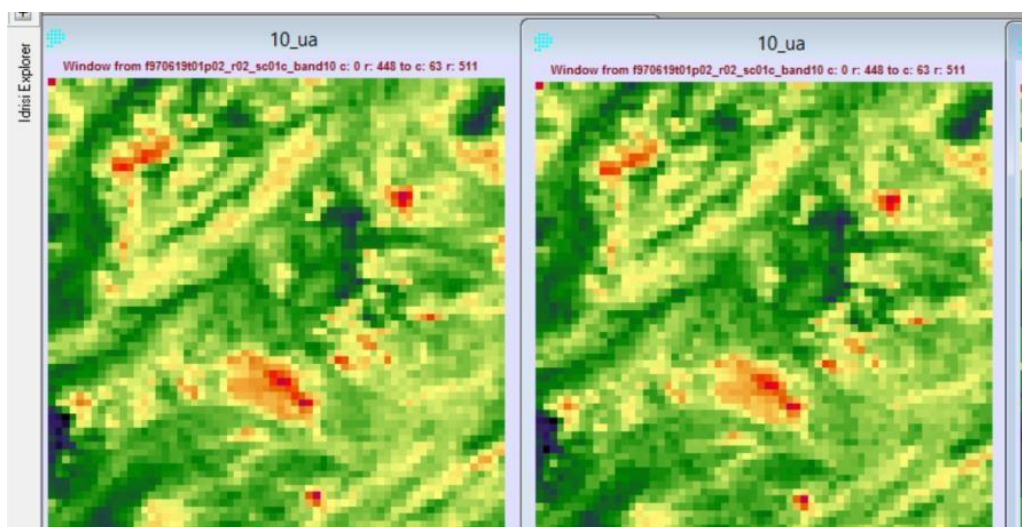


Рисунок 9. Исходные и преобразованные изображения

При открытии программы обработки и сжатия появляется окно с выбором одного канала или нескольких файлов, рис. 10.

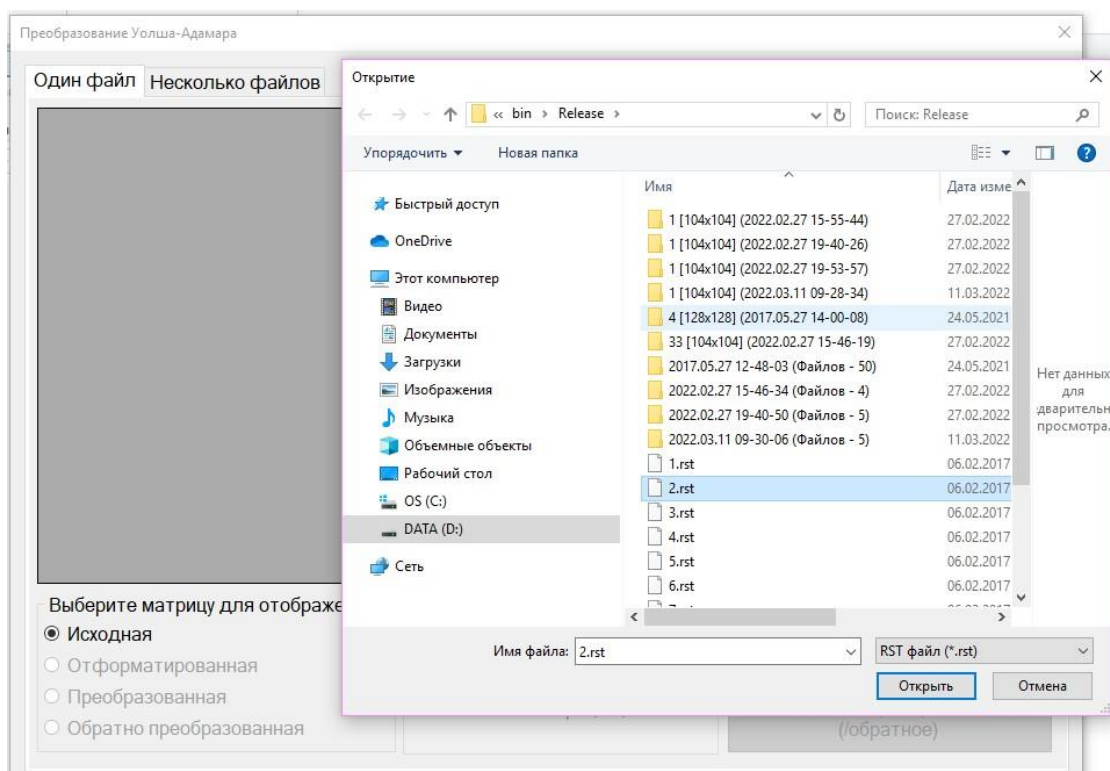


Рисунок 10. Выбор одного файла (канала) гипеспектральных изображений

На рис. 11-14 представлены матрицы, на которых отображаются исходные, отформатированные и преобразованные изображения после преобразования Уолша-адамара с матрицей 2\*2.

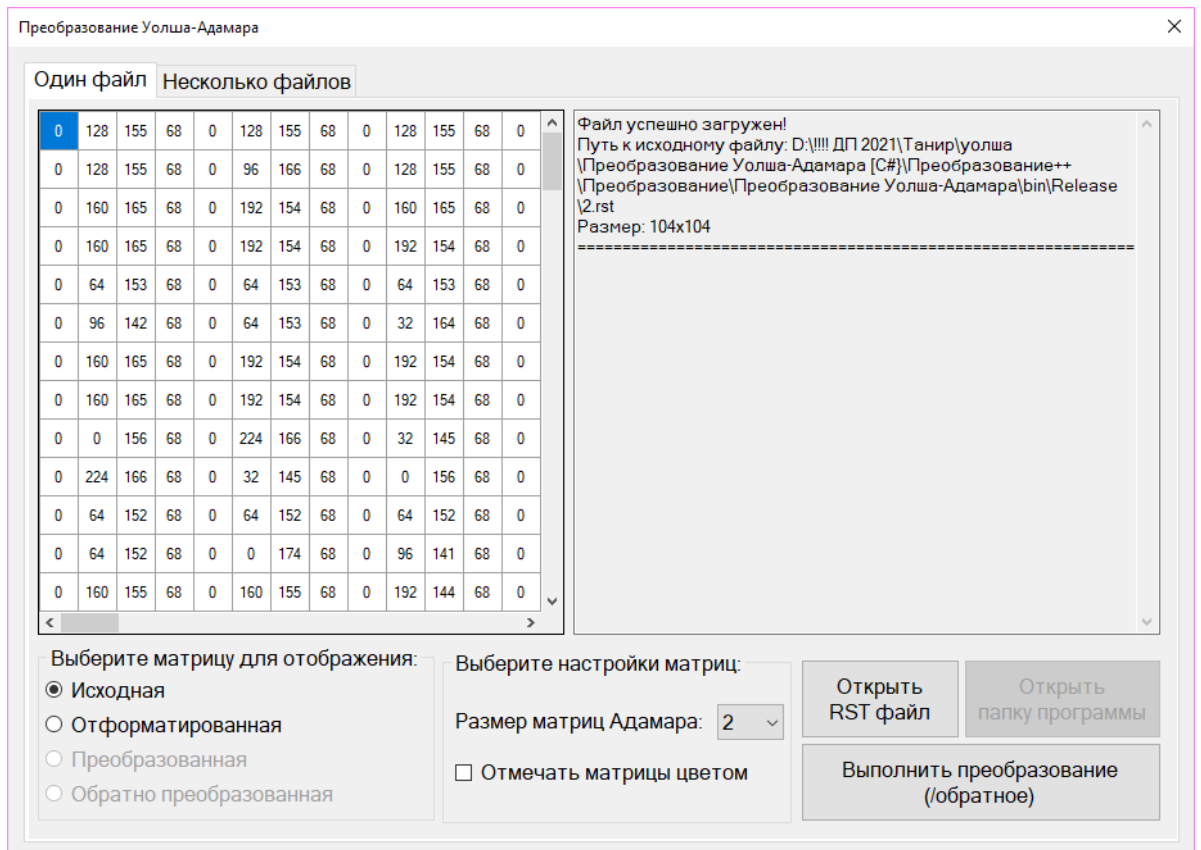


Рисунок 11. Преобразование матрицы 2\*2 гиперспектральных изображений

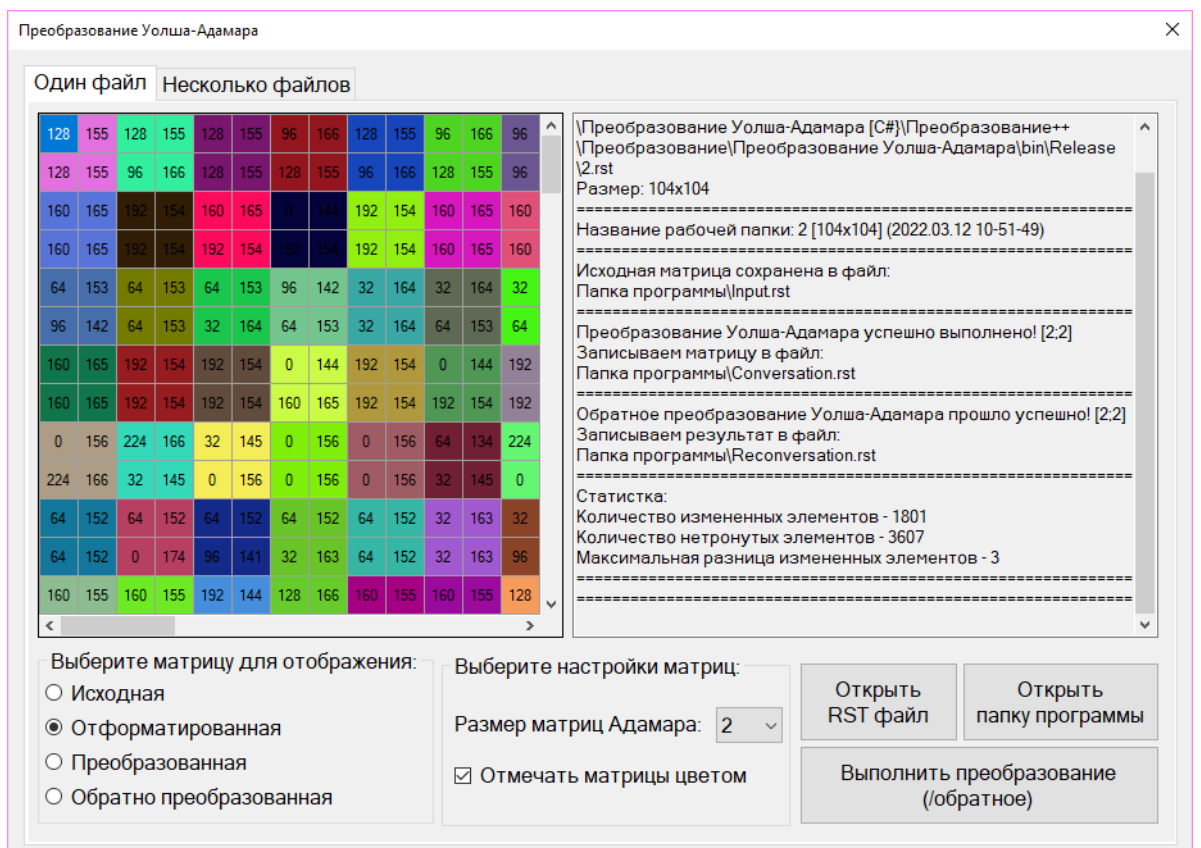


Рисунок 12. Отформатированные матрицы 2\*2

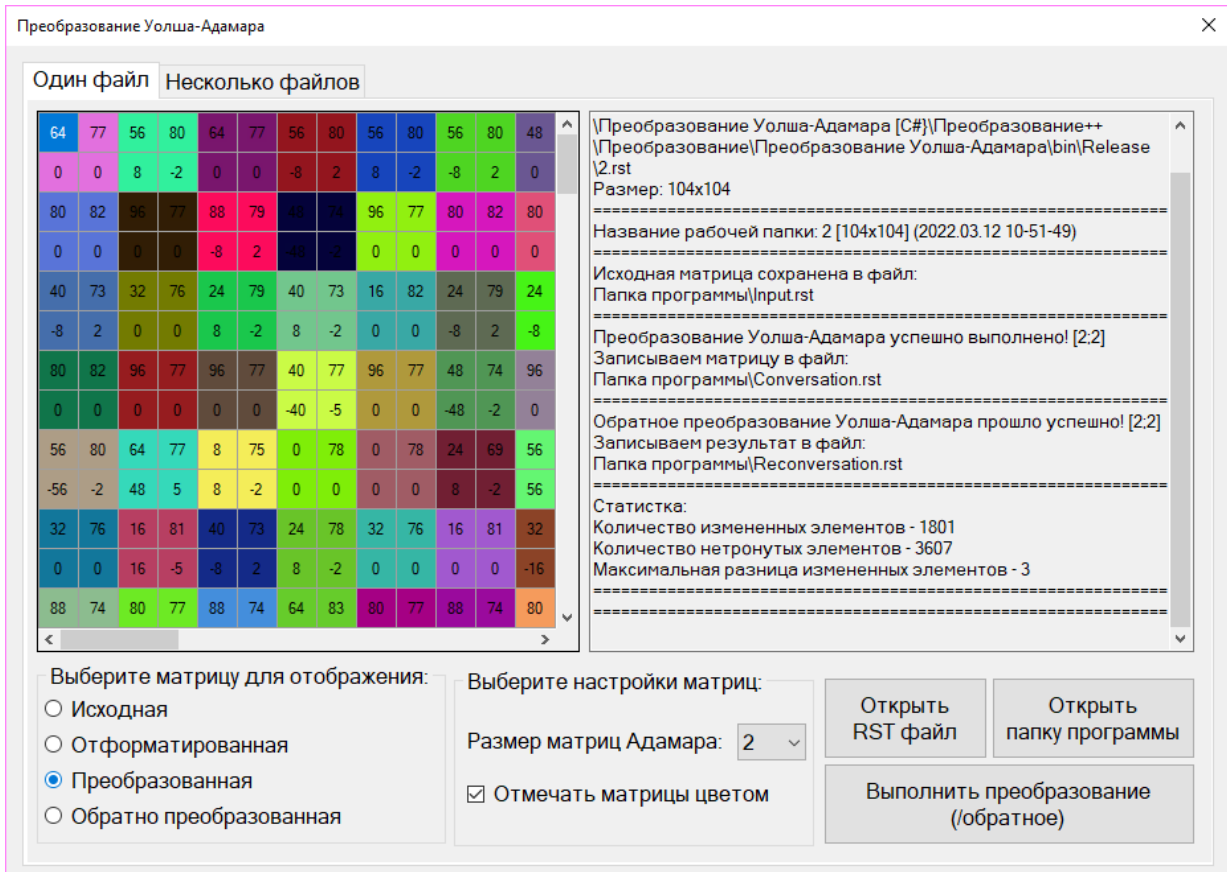


Рисунок 13. Преобразование квантования матрицы 2\*2

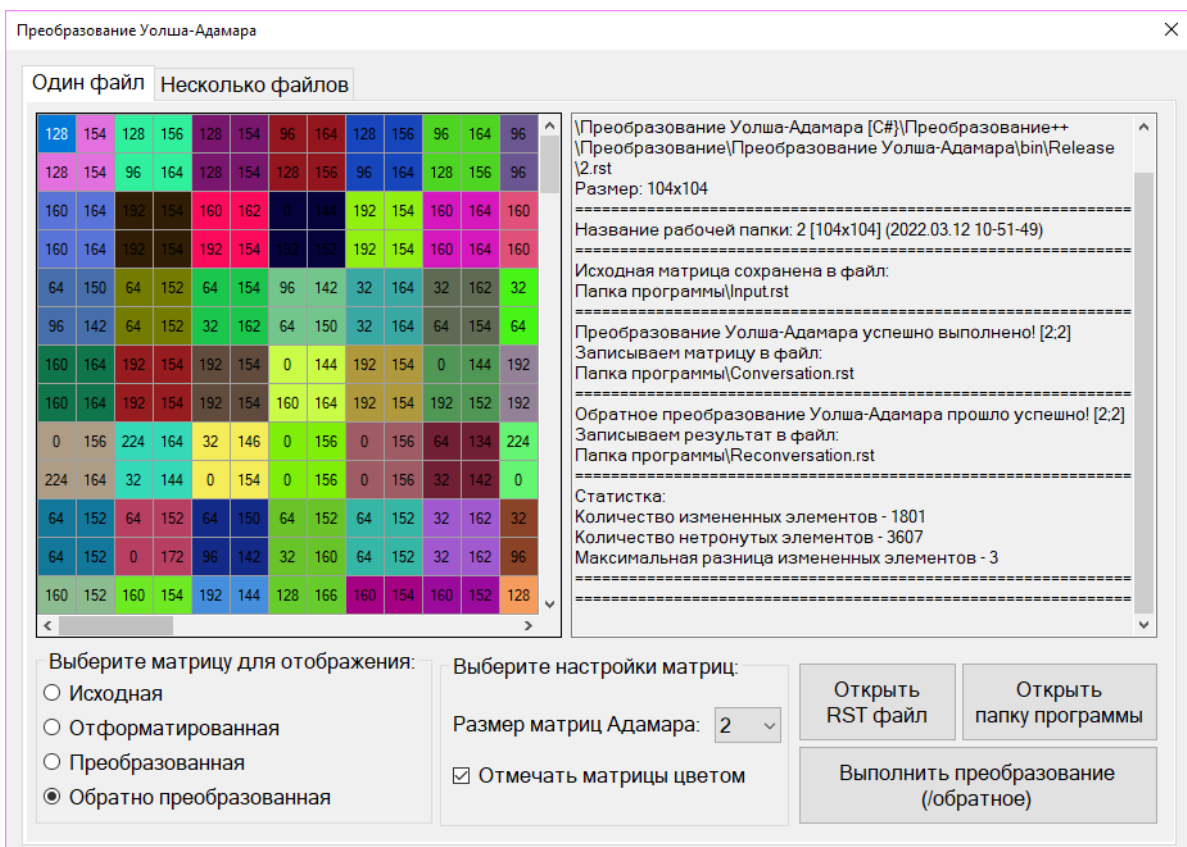


Рисунок 14. Обратное преобразование матрицы 2\*2



На рис. 15-18 представлены матрицы, на которых отображаются исходные, отформатированные и преобразованные изображения после преобразования Уолша-адамара с матрицей 4\*4.

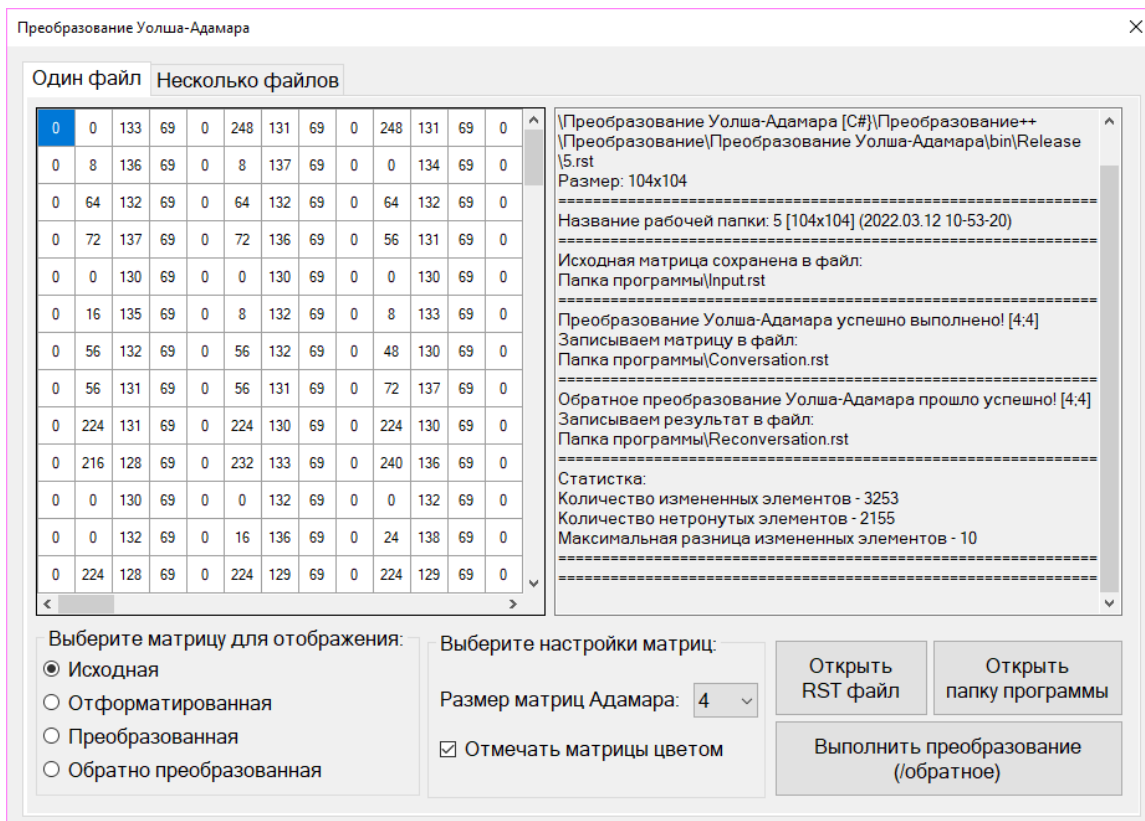


Рисунок 15. Преобразование матрицы 4\*4 гиперспектральных изображений

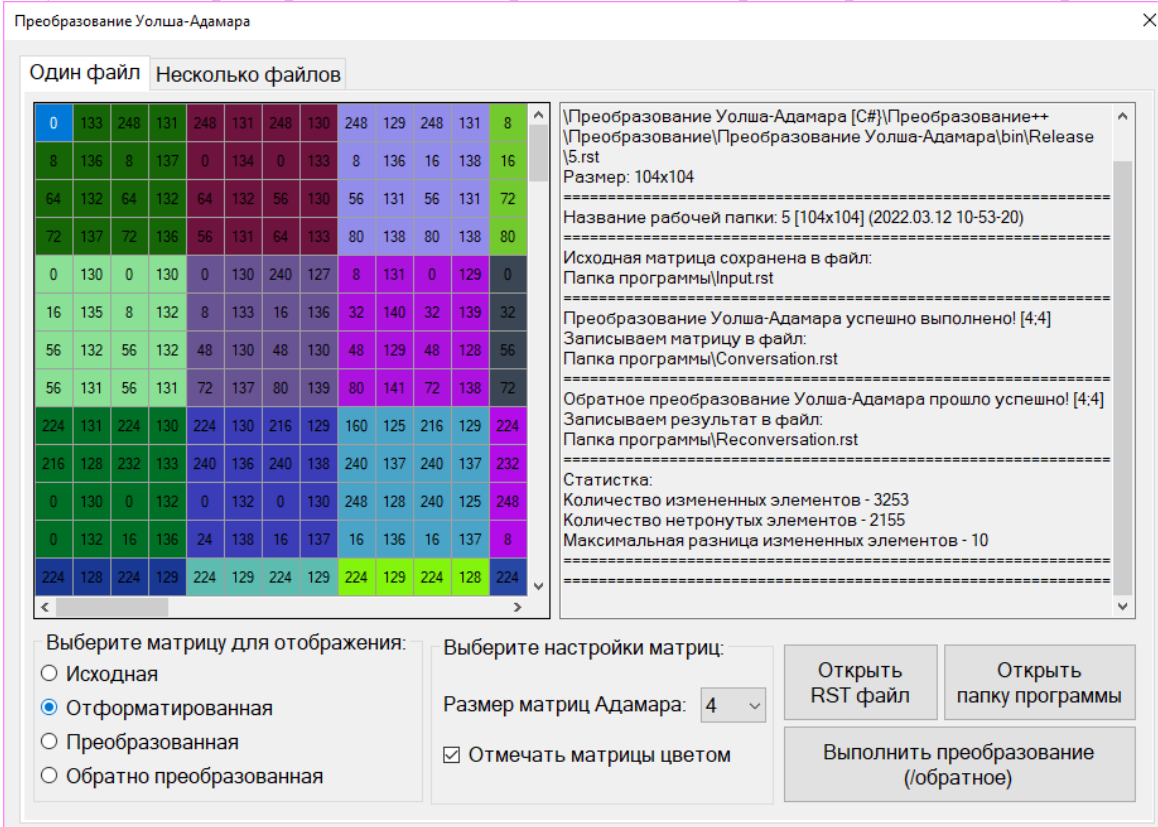


Рисунок 16. Отформатированные матрицы 4\*4

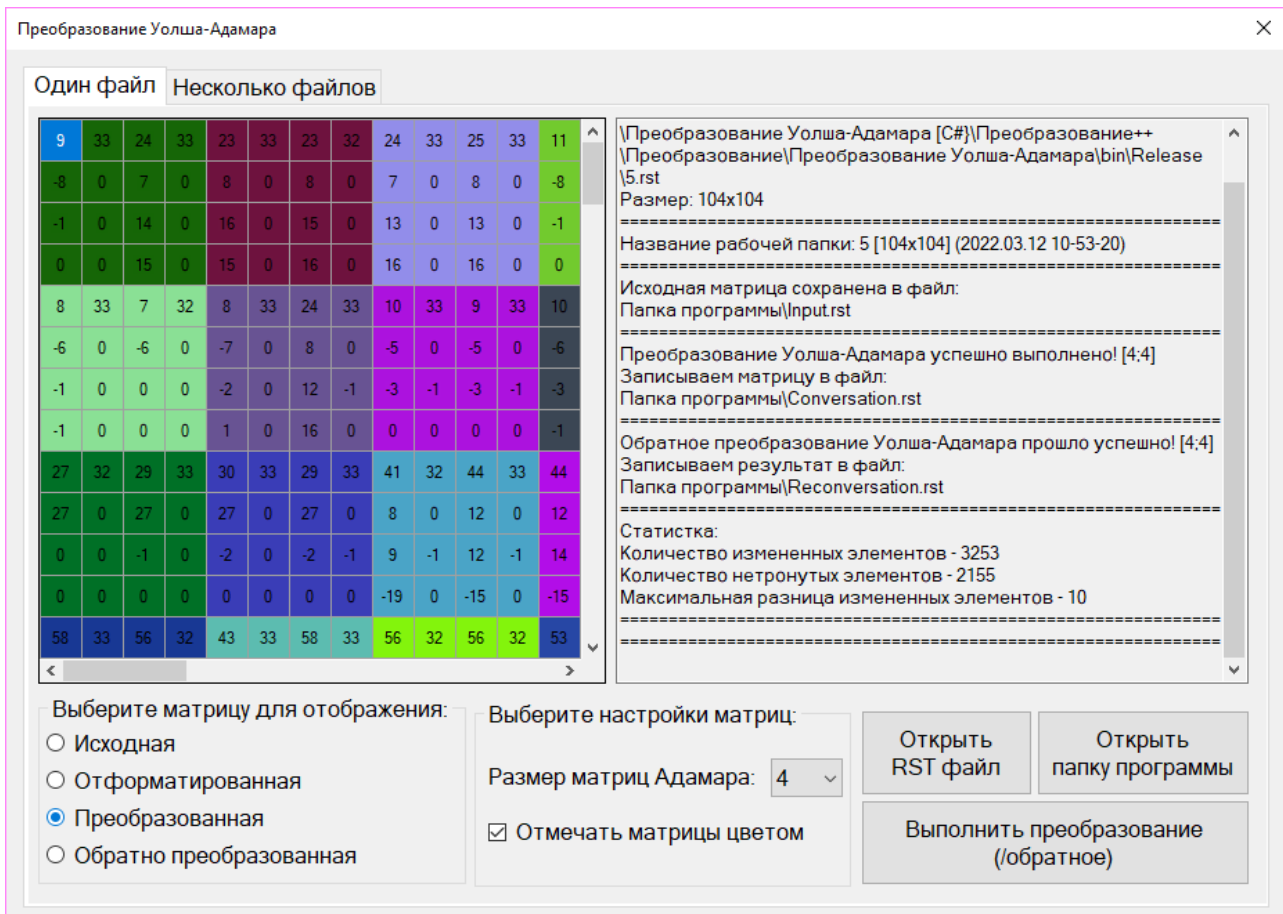


Рисунок 17. Преобразование квантования матрицы 4\*4

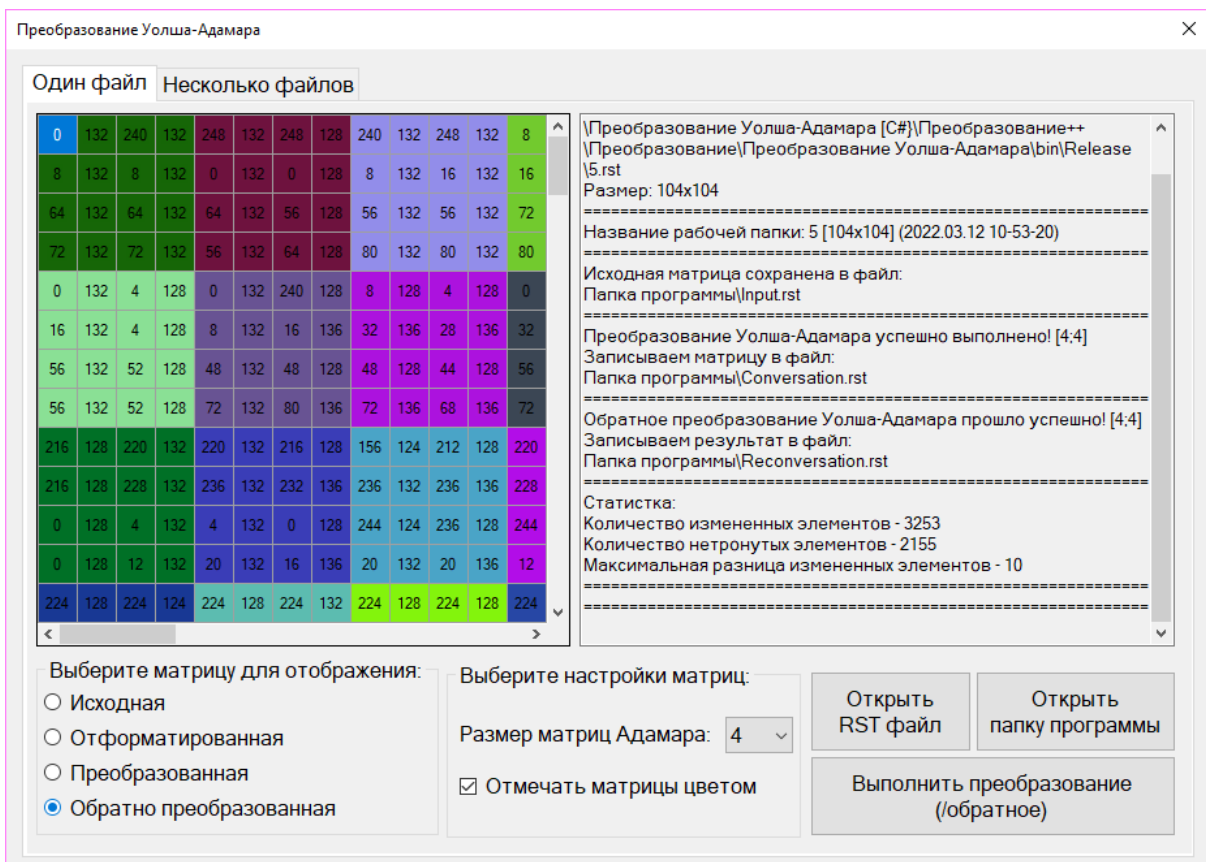


Рисунок 18. Обратное преобразование квантования матрицы 4\*4

На рис. 19-22 представлены матрицы, на которых отображаются исходные, отформатированные и преобразованные изображения после преобразования Уолша-адамара с матрицей  $8 \times 8$ .

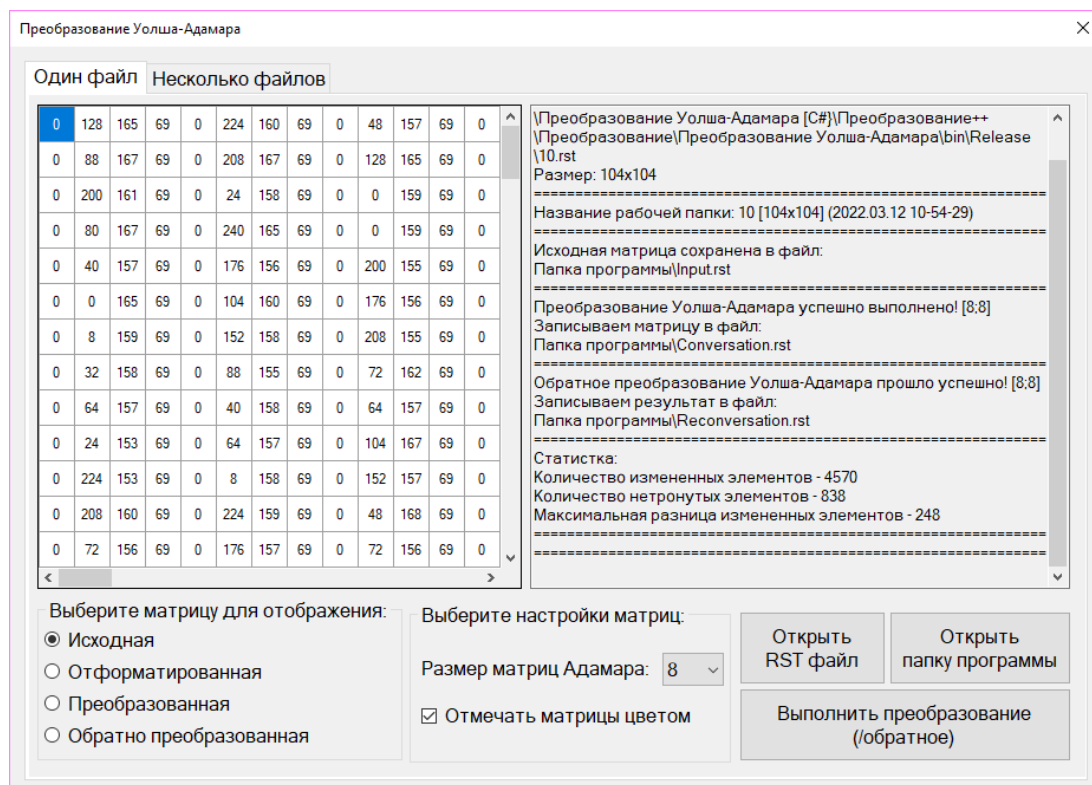


Рисунок 19. Преобразование матрицы  $8 \times 8$  гиперспектральных изображений

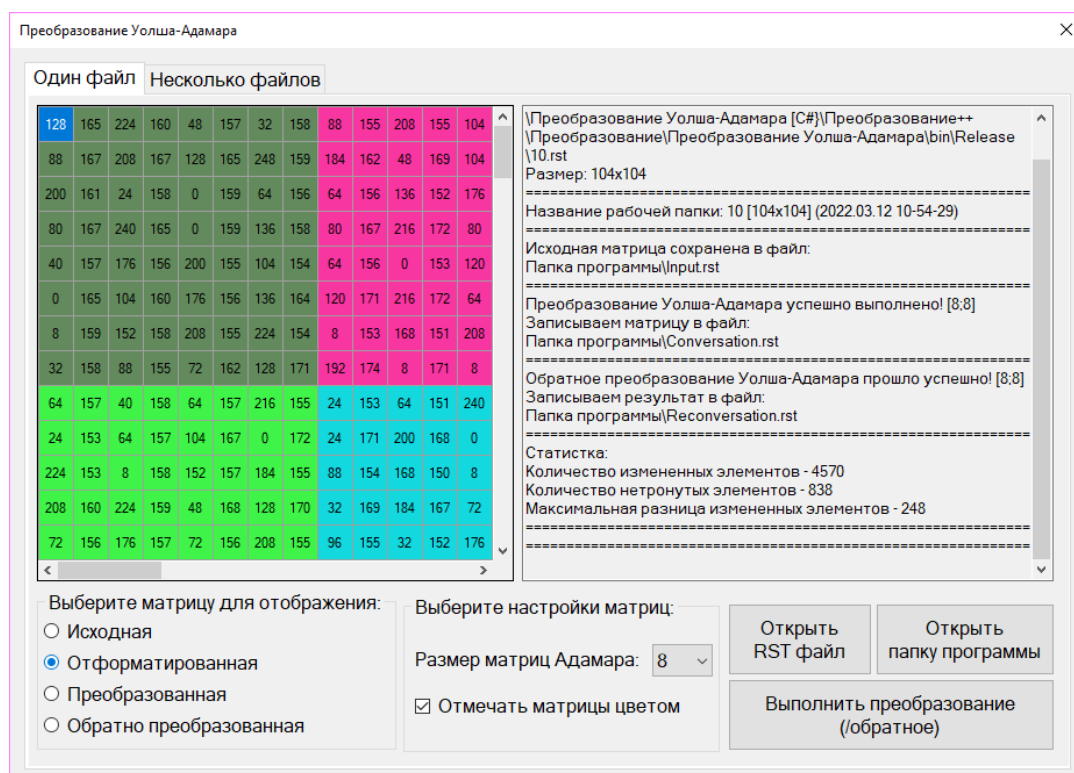


Рисунок 20. Отформатированные матрицы  $8 \times 8$

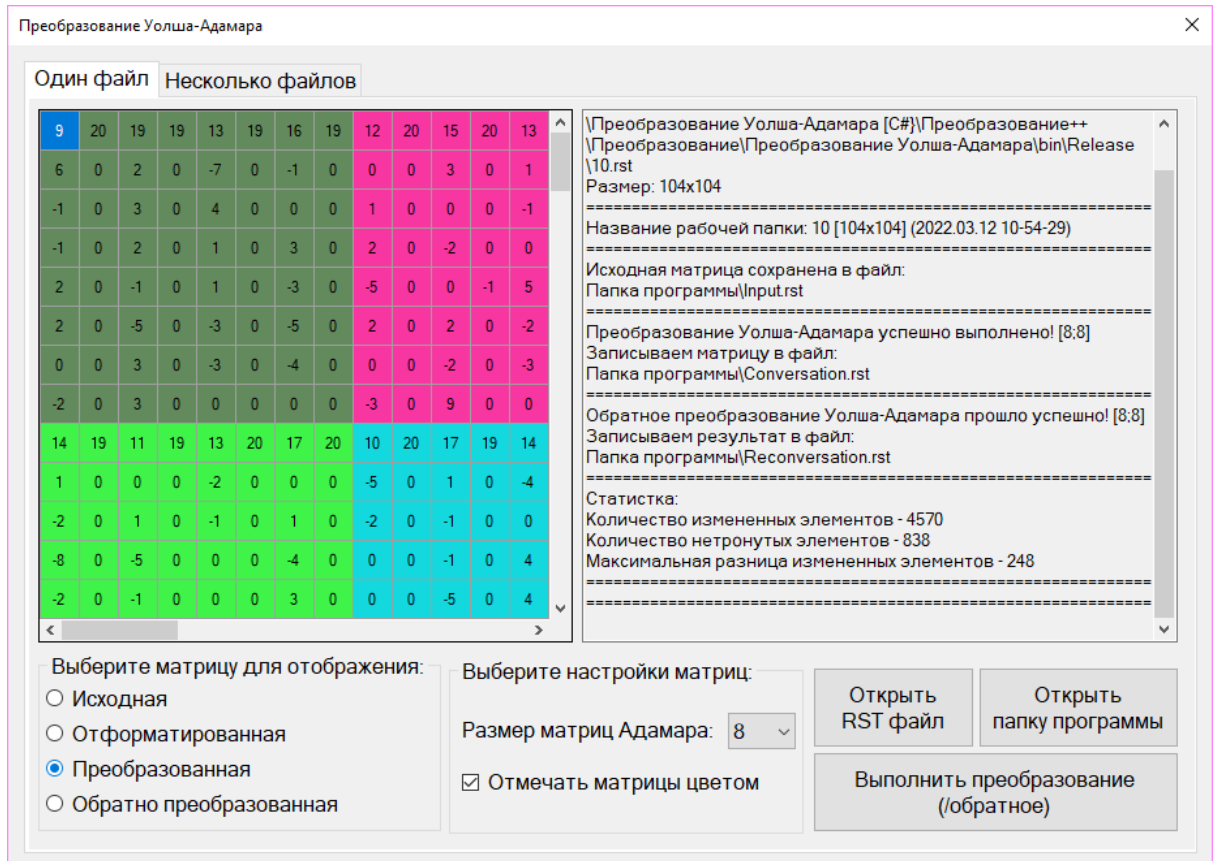


Рисунок 21. Преобразование квантования матрицы 8\*8

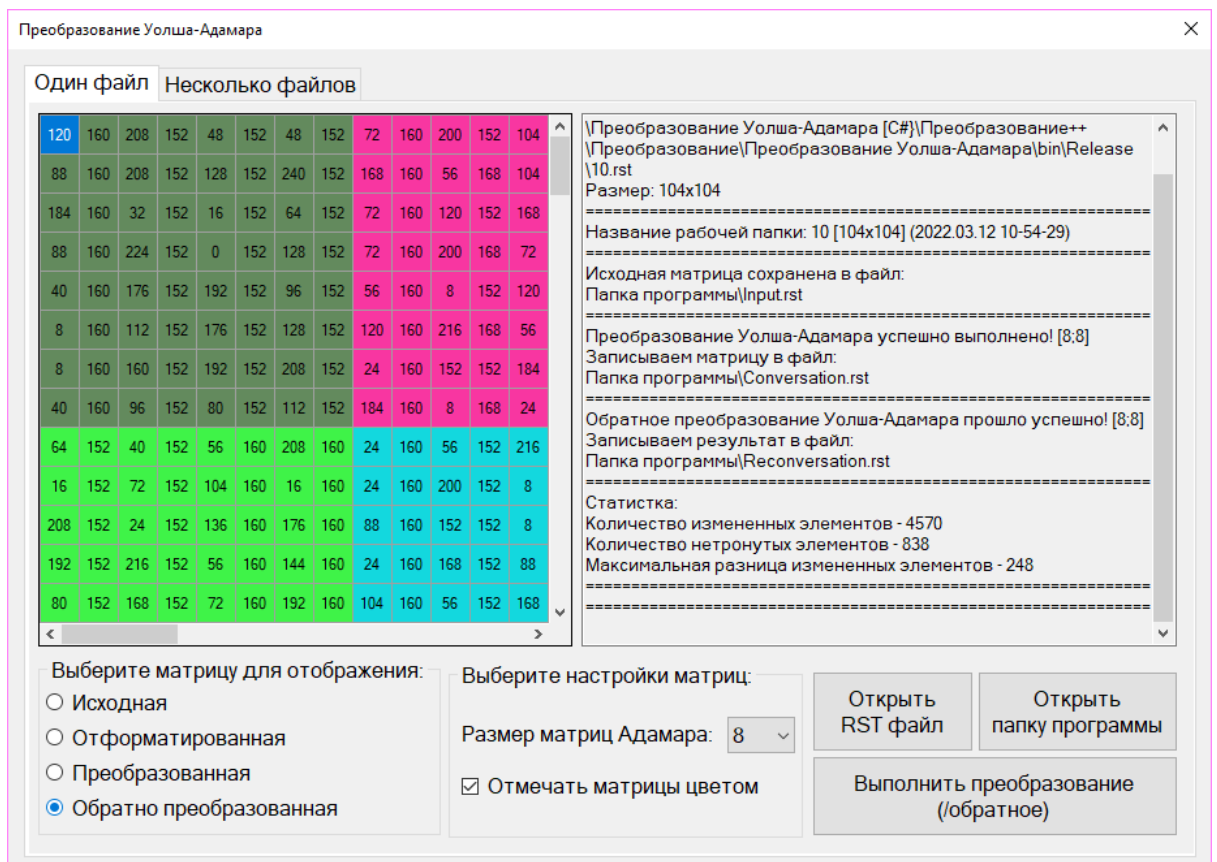


Рисунок 22. Обратное преобразование квантования матрицы 4\*4

Также визуально представленный интерфейс позволяет отображать и открыть папку с преобразованными и обратно преобразованными каналами гиперспектральных изображений, в соответствии с рисунком 23.

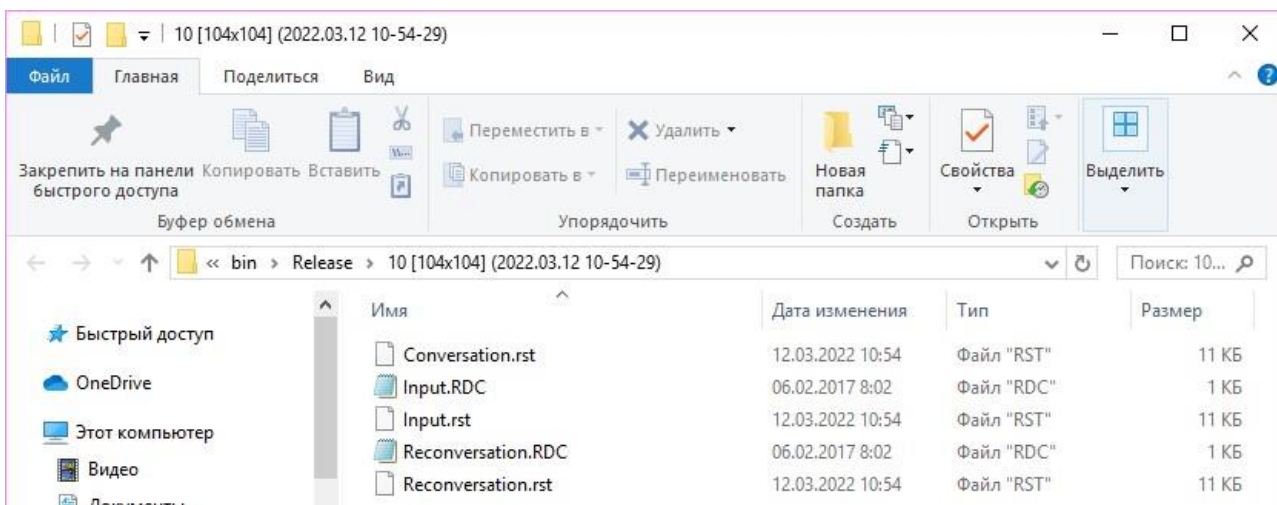


Рисунок 23. Отображаемая папка с преобразованным каналом гиперспектрального изображения

Ввиду того что, гиперспектральные изображения могут состоять из сотен каналов, для этого был создан модуль с выбором нескольких файлов (каналов), рис. 24.

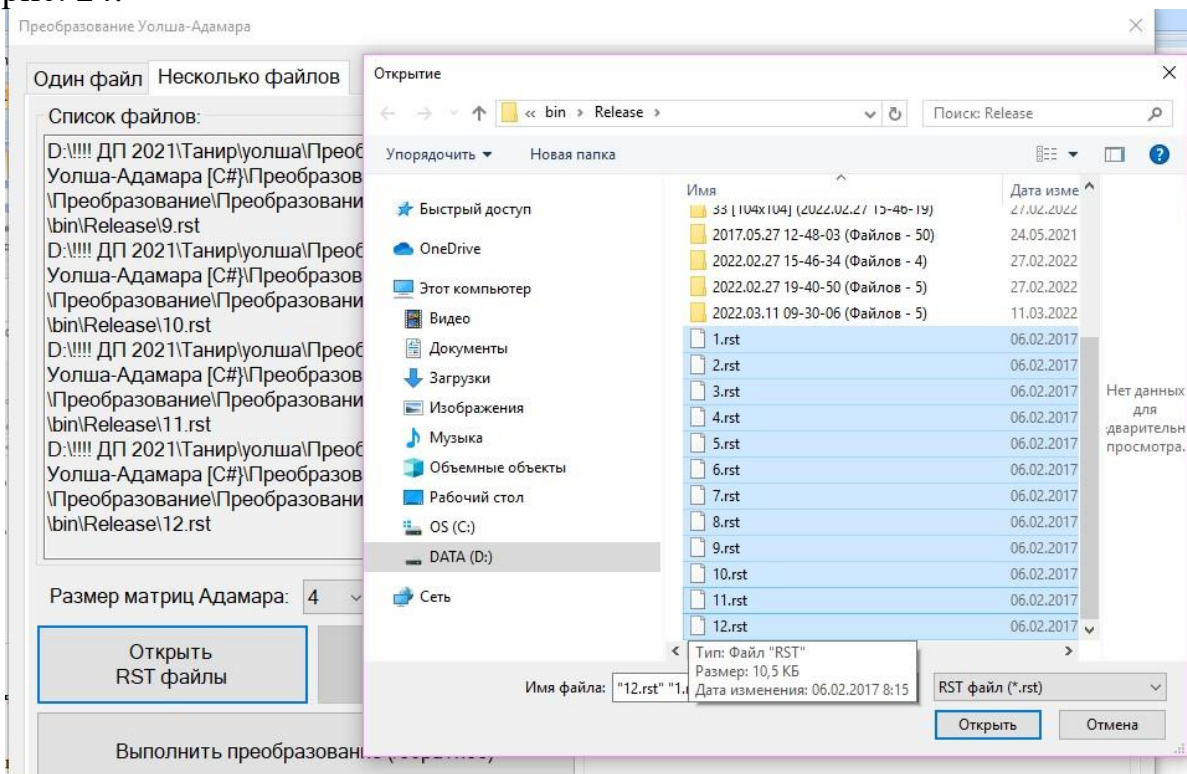


Рисунок 24. Выбор пути с выбором нескольких файлов (каналов), гиперспектрального изображения

На рис. 25 представлен интерфейс отображающий лог программы в котором имеется информация обо всех преобразованных каналах гиперспектральных изображений

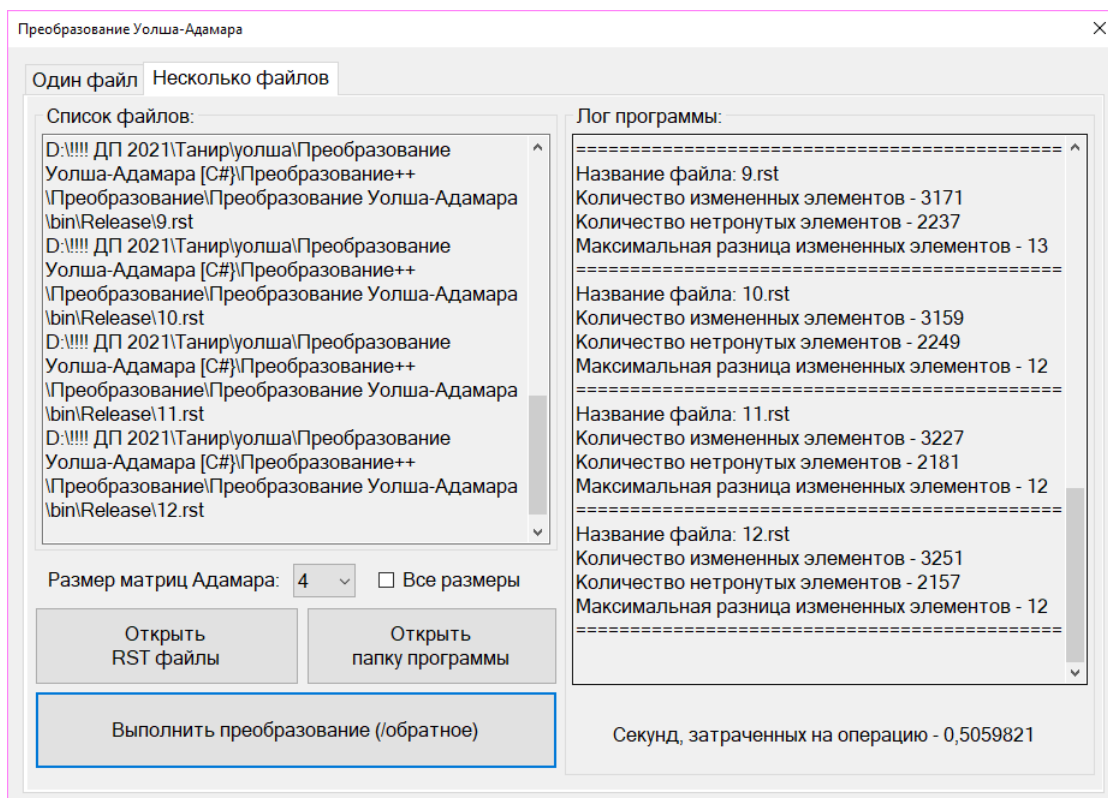


Рисунок 25. Отображаемая папка с преобразованными каналами гиперспектрального изображения

Далее на рисунке 26 можно увидеть преобразованные папки с каналами в отдельности для каждого гиперспектральных изображений.

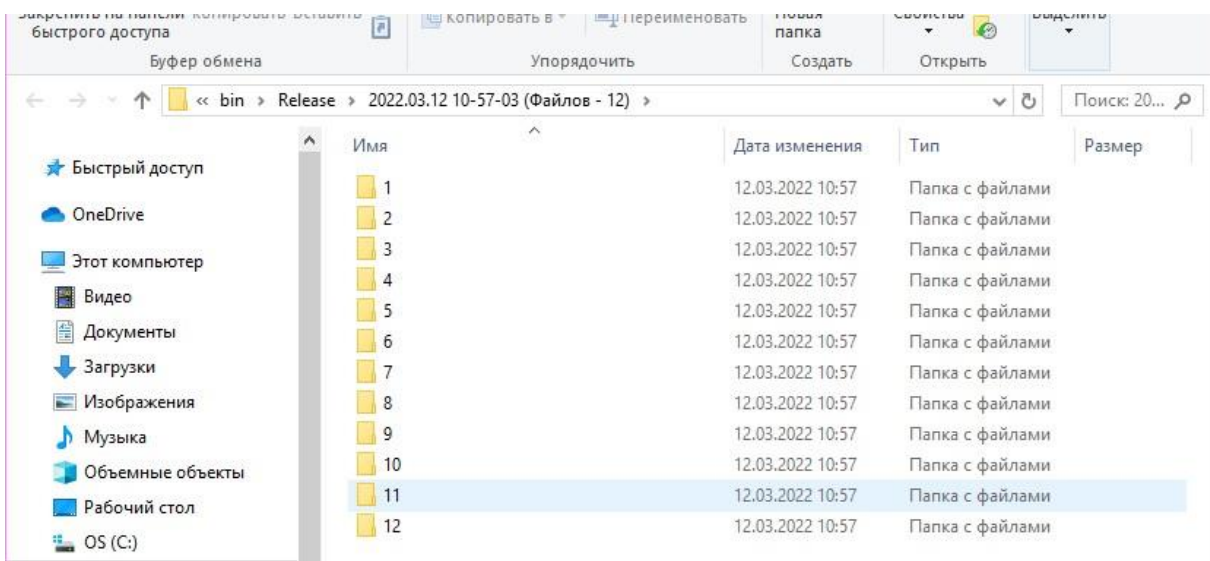


Рисунок 26. Папка с закодированными несколькими файлами (каналами)

### 3.5 Описание разработанных классов для обработки и сжатия гипеспектральных изображений

Класс `WHConversation` содержит основные методы для прямого преобразования матриц гипеспектральных АИ, согласно размерам матрицы Уолша-Адамара.

Класс <i>WHConversation</i>	
<i>ConversationFromFile2</i>	Метод реализует преобразование матрицы, которая содержится в файле. Прямое преобразование производится с помощью матрицы Уолша-Адамара 2x2. В параметрах функции принимается полный путь к файлу матрицы гипеспектрального АИ.
<i>ConversationFromMatrix2</i>	Метод реализует преобразование матрицы, которая содержится матрице. Прямое преобразование производится с помощью матрицы Уолша-Адамара 2x2. В качестве аргументов передается полный путь к файлу матрицы гипеспектрального АИ и ее размер.
<i>ConversationSubMatrix2</i>	Метод производит вычисления с подматрицей основной матрицы. В данной функции данные действия производятся с помощью матрицы Уолша-Адамара 2x2. Метод принимает в качестве аргументов подматрицу размером 2x2, ее размер и положение в исходной матрице.
<i>ConversationFromFile4</i>	Метод реализует преобразование матрицы, которая содержится в файле. Прямое преобразование производится с помощью матрицы Уолша-Адамара 4x4. В параметрах функции принимается полный путь к файлу матрицы гипеспектрального АИ.
<i>ConversationFromMatrix4</i>	Метод реализует преобразование матрицы, которая содержится матрице. Прямое преобразование производится с помощью матрицы Уолша-Адамара 4x4. В параметрах функции принимается полный матрица гипеспектрального АИ и ее размер.
<i>ConversationSubMatrix4</i>	Метод производит вычисления с подматрицей основной матрицы. В данной функции данные действия производятся с помощью матрицы Уолша-Адамара 4x4. Метод принимает в качестве аргументов подматрицу размером 4x4, ее размер и положение в исходной матрице.
<i>ConversationFromFile8</i>	Метод реализует преобразование матрицы, которая содержится в файле. Прямое преобразование производится с помощью матрицы Уолша-Адамара

	8x8. В параметрах функции принимается полный путь к файлу матрицы гиперспектрального АИ.
<i>ConversationFromMatrix8</i>	Метод реализует преобразование матрицы, которая содержится матрице. Прямое преобразование производится с помощью матрицы Уолша-Адамара 8x8. В параметрах функции принимается полный путь к файлу матрицы гиперспектрального АИ и ее размер.
<i>ConversationSubMatrix8</i>	Метод производит вычисления с подматрицей основной матрицы. В данной функции данные действия производятся с помощью матрицы Уолша-Адамара 8x8. Метод принимает в качестве аргументов подматрицу размером 8x8, ее размер и положение в исходной матрице.



Класс **WHReconversation** содержит основные методы для обратного преобразования гиперспектральных АИ, согласно размерам матрицы Уолша-Адамара.

Класс <i>WHReconversation</i>	
<i>ReconversationFromFile2</i>	Метод реализует преобразование матрицы, которая содержится в файле. Обратное преобразование производится с помощью матрицы Уолша-Адамара 2x2. В параметрах функции принимается полный путь к файлу матрицы гиперспектрального АИ.
<i>ReconversationFromMatrix2</i>	Метод реализует преобразование матрицы, которая содержится матрице. Обратное преобразование производится с помощью матрицы Уолша-Адамара 2x2. В качестве аргументов передается полный путь к файлу

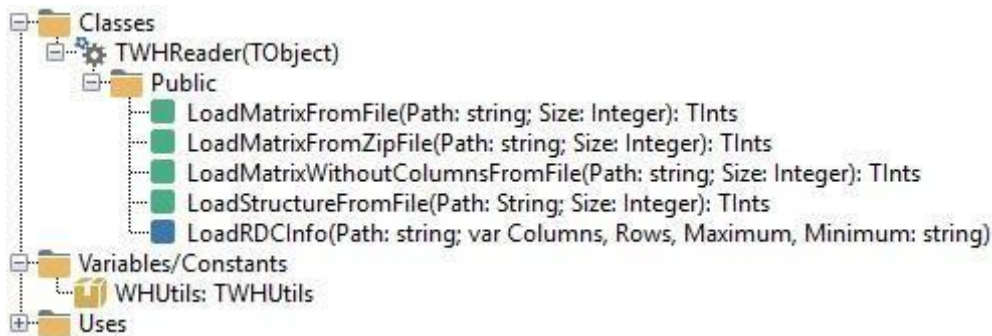


	матрицы гиперспектрального АИ и ее размер.
<i>ReconversationSubMatrix2</i>	Метод производит вычисления с подматрицей основной матрицы. В данной функции данные действия производятся с помощью матрицы Уолша-Адамара 2x2. Метод принимает в качестве аргументов подматрицу размером 2x2, ее размер и положение в исходной матрице.
<i>ReconversationFromFile4</i>	Метод реализует преобразование матрицы, которая содержится в файле. Обратное преобразование производится с помощью матрицы Уолша-Адамара 4x4. В параметрах функции принимается полный путь к файлу матрицы гиперспектрального АИ.
<i>ReconversationFromMatrix4</i>	Метод реализует преобразование матрицы, которая содержится матрице. Обратное преобразование производится с помощью матрицы Уолша-Адамара 4x4. В параметрах функции принимается полный матрица гиперспектрального АИ и ее размер.
<i>ReconversationSubMatrix4</i>	Метод производит вычисления с подматрицей основной матрицы. В данной функции данные действия производятся с помощью матрицы Уолша-Адамара 4x4. Метод принимает в качестве аргументов подматрицу размером 4x4, ее размер и положение в исходной матрице.
<i>ReconversationFromFile8</i>	Метод реализует преобразование матрицы, которая содержится в файле. Обратное преобразование производится с помощью матрицы Уолша-Адамара 8x8. В параметрах функции принимается полный путь к файлу матрицы гиперспектрального АИ.
<i>ReconversationFromMatrix8</i>	Метод реализует преобразование матрицы, которая содержится матрице. Обратное преобразование производится с помощью матрицы Уолша-Адамара 8x8. В параметрах функции принимается полный путь к файлу матрицы гиперспектрального АИ и ее размер.
<i>ReconversationSubMatrix8</i>	Метод производит вычисления с подматрицей основной матрицы. В данной функции данные действия производятся с помощью матрицы Уолша-Адамара 8x8. Метод принимает в качестве аргументов подматрицу размером 8x8, ее размер и положение в исходной матрице.



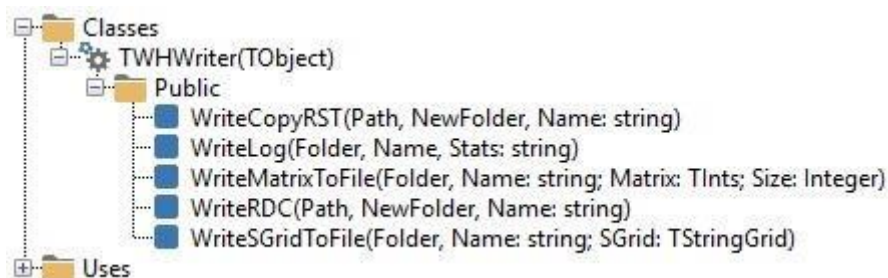
Класс **WHReader** реализует методы для чтения матриц гиперспектральных АИ и файла конфигурации из всех доступных источников.

Класс <i>WHReader</i>	
<i>LoadMatrixFromFile</i>	Метод необходим для загрузки гиперспектрального АИ из бинарного файла. В качестве аргументов передается полный путь до файла и размеры матрицы.
<i>LoadMatrixFromZipFile</i>	Метод необходим для загрузки гиперспектрального АИ из архива. В качестве параметров принимается полный путь до заархивированного файла и размеры матрицы.
<i>LoadMatrixWithout ColumnsFromFile</i>	Метод необходим для загрузки гиперспектрального АИ из бинарного файла. В отличие от других методов, данный – возвращает матрицу без лишних столбцов. В качестве параметров принимается полный путь до файла и размеры матрицы.
<i>LoadStructureFromFile</i>	Метод необходим для загрузки структуры гиперспектрального АИ из бинарного файла. В качестве аргументов передается полный путь до файла и размеры структуры.
<i>LoadRDCInfo</i>	Метод считывает и возвращает основную информацию из файла конфигурации: количество строк и столбцов, минимальное и максимальное значения структуры.



Класс WHWriter реализует методы для записи матриц гиперспектральных АИ и файла конфигурации в конечные файлы, а также перезаписи файла конфигурации.

Класс <i>WHWriter</i>	
WriteCopyRST	Метод копирует исходный файл матрицы из директории в папку текущей сессии.
WriteLog	Метод записывает лог сессии в соответствующий файл.
WriteMatrixToFile	Метод производит запись матрицы в файл. В качестве аргументов процедуре передается папка, имя файла, матрица и ее размер.
WriterRDC	Метод перезаписывает файл конфигурации из исходной директории в директорию с преобразованным файлом.
WriteSGridToFile	Метод перезаписывает файл конфигурации из исходной директории в директорию с преобразованным файлом. В качестве параметров принимает компонент типа TStringGrid.



Класс WHFill необходим для визуального представления матриц и структур в программе с помощью таблиц.

Класс <i>WHFill</i>	
<i>Structure</i>	Метод производит загрузку структуры в таблицу, которая расположена на главной форме в соответствующей вкладке.
<i>Matrix</i>	Метод производит загрузку исходной матрицы в таблицу, которая расположена на главной форме в

	соответствующей вкладке.
<i>MatrixWithoutColumns</i>	Метод производит загрузку матрицы без лишних столбцов в таблицу, которая расположена на главной форме в соответствующей вкладке.
<i>RDCInfo</i>	Метод с помощью визуальных компонентов показывает основную информацию о текущей матрице гиперспектрального АИ: количество строк и столбцов, максимальный и минимальный элементы структуры.



Класс *WHHadamarMatrixes* содержит вспомогательные для преобразования матрицы Уолша-Адамара всех необходимых размеров.

Класс <i>WHHadamarMatrixes</i>	
<i>Hadamar2</i>	Поле класса содержит матрицу Уолша-Адамара размером 2x2
<i>Hadamar4</i>	Поле класса матрицу Уолша-Адамара размером 4x4
<i>Hadamar8</i>	Поле класса матрицу Уолша-Адамара размером 8x8



Класс *WHMatrixUtils* предоставляет методы для работы с загруженными матрицами.

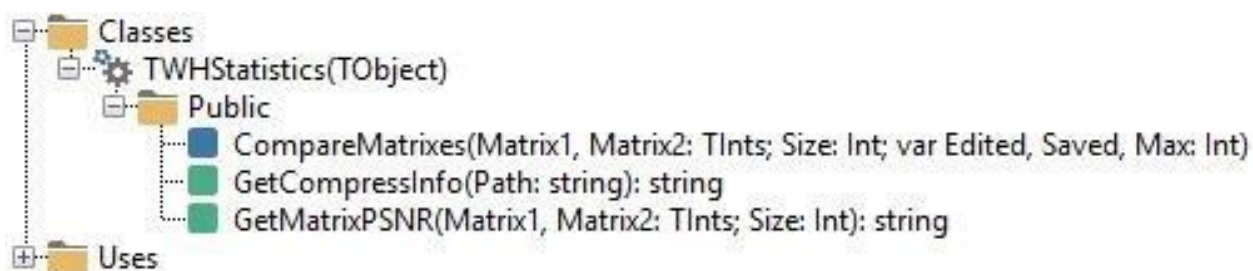
Класс <i>WHMatrixUtils</i>	
<i>GetMatrixSize</i>	Метод возвращает размеры матрицы, которую ему передают в параметрах.
<i>ConvertMatrixToStructure</i>	Метод преобразует исходную матрицу в структуру. В качестве аргументов получает саму матрицу и

	ее размер.
<i>AddColumnsToMatrix</i>	Метод удаляет лишние столбцы из матрицы.
<i>RemoveColumnsFromMatrix</i>	Метод добавляет столбцы из одной матрицы в другую.
<i>FreeMatrix</i>	Метод освобождает оперативную память, занимаемую матрицей.



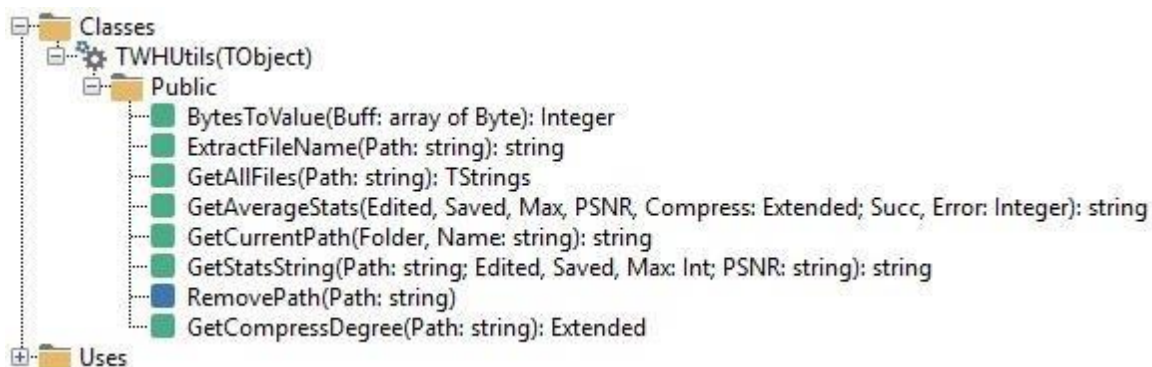
Класс WHStatistics реализует методы, которые производят вычисления, необходимые для сравнения исходных и конечных файлов.

Класс <i>WHStatistics</i>	
<i>CompareMatrixes</i>	Метод сравнивает две матрицы между собой и возвращает количество измененных, сохраненных элементов, а так же их максимальную разницу.
<i>GetCompressInfo</i>	Метод вычисляет степень сжатия исходного файла и заархивированного преобразованного.
<i>GetMatrixPSNR</i>	Метод вычисляет СКО и PSNR двух матриц.



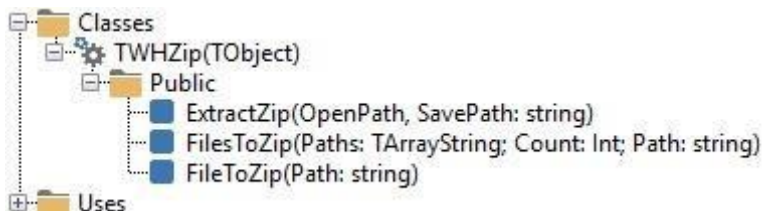
Класс WHUtils содержит в себе методы, которые производят малые вычисления для успешного преобразования.

Класс <i>WHUtils</i>	
<i>BytesToValue</i>	Метод вычисляет значение структуры матрицы. Принимает в качестве аргумента байтовый массив.
<i>ExtractFileName</i>	Метод возвращает название файла из заданной директории.
<i>GetAllFiles</i>	Метод позволяет получить список всех файлов в конкретной папке, которая является параметром функции.
<i>GetAverageStats</i>	Метод форматирует данные, полученные из параметров функции и возвращает готовую строку для записи в лог программы. Необходима для конечного итога сессии.
<i>GetCurrentPath</i>	Метод формирует название папки для записи данных.
<i>GetStatsString</i>	Метод форматирует данные, полученные из параметров функции и возвращает готовую строку для записи в лог программы. Необходима для каждого файла отдельно.
<i>RemovePath</i>	Метод удаляет временные папки, которые необходимы были в ходе преобразования.
<i>GetCompressDegree</i>	Метод позволяет узнать степень сжатия исходного и преобразованного файлов.



Класс WHZip служит для архивации файлов, а также для их извлечения.

Класс WHZip	
ExtractZip	Метод извлекает все файлы из архива для обратного преобразования.
FilesToZip	Метод архивирует все преобразованные файлы в конце сессии в единый архив.
FileToZip	Метод архивирует один файл, указанный в параметрах.



### Выводы по главе

Разработаны соответствующие классы, функции которых представлены в п.п.3.4.

Разработана программа преобразования Уолша-Адамара для кодирования аэрокосмических гипеспектральных изображений включают в себя 2 на 2, 4 на 4, 8 на 8 преобразованных пикселей матриц, в проанализированном обзоре имеющихся научных разработок в таком преобразовании не существует попыток 8 на 8, данный факт является новизной в проведенном исследовании.

## ЗАКЛЮЧЕНИЕ

Основная цель этой диссертации – изучить преобразованное кодирование, которое сжимает или уменьшает объем гиперспектральных данных для достижения таких преимуществ, как уменьшение пропускной способности канала передачи, уменьшение буферизации и хранения требования и сокращение задержки передачи данных на фиксированной скорости. В этой диссертации три набора гиперспектральных изображений спутниковых систем AVIRIS, CCSDS и Hyperion, сжаты алгоритмом на основе преобразования Уолша-Адамара.

Модифицированный алгоритм преобразования Уолша-Адамара обеспечивает лучшую производительность для сжатия гиперспектральных изображений. К преимуществам относятся: оптимизация сжатия энергии, полученного в результате гибридного преобразования, эффективность значительного поиска на основе хорошо спроектированных древовидных структур и упрощения алгоритм кодирования. Производительность сжатия с потерями конкурентоспособна с лучшими алгоритмами предиктивного кодирования со значительно меньшими вычислительными затратами, и превосходит другие алгоритмы на основе преобразования Уолша-Адамара. Производительность скорости искажения с потерями преобразования Уолша-Адамара также постоянно превосходит другие алгоритмы на основе преобразования дискретно-косинусного преобразования. Особенностью гиперспектрального изображения является наличие сотен спектральных каналов, таких, что производительность сжатия с потерями оценивается по некоторым спектральным критериям. Однако, более значимые критерии смогут определить разницу в зависимости от конечного пользователя приложения. Использование методов классификации более полезно на практике.

В данной диссертации был реализован предложенный алгоритм преобразования Уолша-Адамара с матрицей  $8 \times 8$ . Предлагаемый алгоритм способен обеспечить очень высокое сжатие за счет использования данных с потерями. Компрессия и искажение оцениваются по некоторым критериям спектрального искажения. На будущее работы, было бы полезно исследовать, как сжатие изображений с потерями повлияет на последующие приложения, такие как классификация изображений, обнаружение целей и сегментация гиперспектральных изображений. В будущем перспективно проанализировать и разобраться с тем, на каком уровне искажения является допустимым, прежде чем последующая обработка даст ложные результаты. Кроме того, это важно разработать подходящие методы спектрального анализа для обеспечения подробного и широкого анализ искаженных гиперспектральных изображений, который может учитывать другие факторы, такие как условия датчика и атмосферные условия.

Наконец, предлагаемый алгоритм преобразования Уолша-Адамара использует свойства спектральных данных. Также возможно сжатие других объемных изображений ДЗЗ и спектральных медицинских изображения, такие как магнитно-резонансная томография (МРТ), спектральная КТ и рентген.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Модели формирования и некоторые алгоритмы обработки гиперспектральных изображений / Р. Н. Ахметов, Н. Р. Стратилатов, А. А. Юдаков [и др.] // Исследование Земли из космоса. – 2014. – № 1. – С. 17.
2. Козинов, И. А. Формирование и обработка гиперспектральных изображений в оптико-электронных системах дистанционного зондирования земли / И. А. Козинов, Г. Н. Мальцев // Оптика и спектроскопия. – 2016. – Т. 121. – № 6. – С. 1005-1019.
3. Комплексная обработка гиперспектральных изображений на основе спектральной и пространственной информации / С. М. Борзов, П. В. Мельников, И. А. Пестунов [и др.] // Вычислительные технологии. – 2016. – Т. 21. – № 1. – С. 25-39.
4. Саринаова, А. Ж. Подготовительная обработка алгоритмов сжатия гиперспектральных аэрокосмических изображений с учетом междиапазонной корреляции с потерями и без потерь / А. Ж. Саринаова // Информационные технологии и математическое моделирование (ИТММ-2016) : Материалы XV Международной конференции имени А.Ф. Терпугова, – Алтайский край, Алтайский р-он, пос. Катунь: Национальный исследовательский Томский государственный университет, 2016. – С. 123-128. – DOI 10.17223/9785751124335/24.
5. Программный комплекс для формирования и обработки гиперспектральных изображений / С. Л. Погорельский, Е. А. Макарецкий, А. В. Овчинников [и др.] // Известия Тульского государственного университета. Технические науки. – 2020. – № 11. – С. 79-84.
6. Ouahioune mohand, Akroure leila, Lahdir mourad, Ameer soltane. Aviris Hyperspectral Images Compression Using 3d Spiht Algorithm. IOSR Journal of Engineering (IOSRJEN) e-ISSN: 2250-3021, p-ISSN: 2278-8719, www.iosrjen.org Volume 2, Issue 10 (October 2012), PP 31-36.
7. Lossless Compression of Hyperspectral Images Using Adaptive Prediction and Backward Search Schemes. JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 419-435 (2011).
8. Jordi Muñoz-Marí, Maciel Zortea, Ian Blanes, Vicente González-Ruiz, Gustavo Camps-Valls, Antonio Plaza. On the Impact of Lossy Compression on Hyperspectral Image Classification and Unmixing. IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, VOL. 8, NO. 2, MARCH 2011. Fernando García-Vilchez.
9. Emmanuel Christophe. Hyperspectral Data Compression Tradeoff. Augmented Vision and Reality, 3, DOI: 10.1007/978-3-642-14212-3\_2, Springer-Verlag Berlin Heidelberg 2011.
10. Lossless multispectral & hyperspectral image compression. CCSDS Recommended standard for lossless multispectral & hyperspectral image compression. Recommended Standard, Issue 1. CCSDS Secretariat. Space Communications and Navigation Office, 7L70. NASA Headquarters. Washington, DC 20546-0001, USA. May 2012. 52 p.

11. N.M.Mary Sindhuja, A.S.Arumugam. SPIHT BASED COMPRESSION OF HYPER SPECTRAL IMAGES. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. Vol. 2, Issue 10, October 2013.
12. D.S.Sujithra, T.Manickam, D.S.Sudheer. Compression Of Hyperspectral Image Using Discrete Wavelet Transform And Walsh Hadamard Transform. International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 2, Issue 3, March 2013. ISSN:2278 – 909X
13. SANJITH S, GANESAN R. A Review on Hyperspectral Image Compression. 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT).
14. Diego Valsesia, Enrico Magli . A Novel Rate Control Algorithm for Onboard Predictive Coding of Multispectral and Hyperspectral Images. 2014.
15. Sarinova A. Zh. The Compression Algorithm of Hyperspectral Aerospace Images with Use of Mathematical Processing and Intrabands Correlation. Journal of Siberian Federal University. Engineering & Technologies, 2018, 11(8). pp.882-891. ВАК РФ.
16. Саринова А. Ж. Подготовительная обработка для сжатия гиперспектральных изображений с потерями // Региональные проблемы дистанционного зондирования Земли : материалы VI Междунар. науч. конф., Красноярск, 10–13 сентября 2019 г. – Красноярск : Сиб. федер. ун-т, 2019. с.149-152.
17. The use of correlation analysis in the algorithm of dynamic gestures recognition in video sequence. The International Conference on Engineering & MIS 2019. L.N.Gumilyov Eurasian National University, 06-08 june. Scopus.
18. JMIT, Radaur, INDIA. JMIT, Radaur, INDIA. Compression and Classification of Hyperspectral Images using an Algorithm based on DWT and NTD. Advance in Electronic and Electric Engineering. ISSN 2231-1297, Volume 3, Number 4 (2013), pp. 447-456.
19. Lucana Santos, Enrico Magli, Raffaele Vitulli, Antonio Núñez, José F. López, Roberto Sarmiento. Lossy hyperspectral image compression on a graphics processing unit: parallelization strategy and performance evaluation. Journal of Applied Remote Sensing 074599-1 Vol. 7, 2013.
20. Kai-Jen Cheng\*, Jeffrey C. Dill\*. An Improved EZW Hyperspectral Image Compression. School of Electrical Engineering and Computer Science, Ohio University, Athens, USA. Journal of Computer and Communications, 2014, 2, 31-36. Published Online January 2014 (<http://www.scirp.org/journal/jcc>) <http://dx.doi.org/10.4236/jcc.2014.22006>
21. Changguo Li<sup>1,2,\*</sup> and Ke Guo<sup>1</sup>. Lossless Compression of Hyperspectral Images Using Three-Stage Prediction with Adaptive Search Threshold. International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.7, No.3 (2014), pp.305-316 <http://dx.doi.org/10.14257/ijcip.2014.7.3.25>.
22. Dr. S.M.Ramesh (Assistant Professor), P.Bharat (P.G. Scholar), J.Anand (P.G.Scholar), J.AnbuSelvan (P.G.Scholar) . Analysis of Lossy Hyperspectral Image Compression Techniques. P.Bharat et al, International Journal of Computer

Science and Mobile Computing, Vol.3 Issue.2, February- 2014, pg. 302-307. India. ISSN 2320–088X.

23. SANJITH S, GANESAN R. A Review on Hyperspectral Image Compression. 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT).

24. Diego Valsesia, Enrico Magli . A Novel Rate Control Algorithm for Onboard Predictive Coding of Multispectral and Hyperspectral Images. 2014.

25. NutanC.Malekar, Dr.R.R.Sedamkar. Novel K-Means Clustering Approach for Compressing Hyperspectral Image. International Journal of Advancements in Research & Technology, Volume 3, Issue 1, January-2014. ISSN 2278-7763. Pp 140-145.

26. Dr. S.M.Ramesh, P.Bharat, J.Anand, J.AnbuSelvan. Analysis of Lossy Hyperspectral Image Compression Techniques. International Journal of Computer Science and Mobile Computing, Vol.3 Issue.2, February- 2014, pg. 302-307. ISSN 2320–088X

27. Raffaele Pizzolante and Bruno Carpentieri. Band Clustering for the Lossless Compression of AVIRIS Hyperspectral Images. ACEEE Int. J. on Signal and Image Processing , Vol. 5, No. 1, January 2014. DOI: 01.IJSIP.5.1.1532.

28. ShipingZhu, DongyuZhao, and Fengchao Wang. Hybrid Prediction and Fractal Hyperspectral Image Compression. Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2015, Article ID 950357, 10 pages <http://dx.doi.org/10.1155/2015/950357>.

29. LefeiZhanga, LiangpeiZhangb, DachengTaoc, Xin Huangb, Bo Dua,n. Compression of hyperspectral remote sensing images by tensor approach. Neurocomputing. 2015. <http://dx.doi.org/10.1016/j.neucom.2014.06.052>

30. Dharam Shah, KuhelikaBera, Sanjay Joshi. Software Implementation of CCSDS Recommended Hyperspectral Lossless Image Compression. I.J. Image, Graphics and Signal Processing, 2015, 4, 35-41 Published Online March 2015 in MECS (<http://www.mecs-press.org/>) DOI: 10.5815/ijigsp.2015.04.04.

31. Haar Wavelets. Richard W. Hamming. Lloyd N. Trefethen1. 1999 CRC Press LLC

32. Daubechies, I. 1992. Ten Lectures on Wavelets, SIAM.

33. B. Campbell and R. H. Wynne, *Introduction to Remote Sensing*. New York:Guilford Press, 2011 (2013, April 30).

34. *NASA Science Mission* [Online]. Available: <http://science1.nasa.gov/>

35. *Landsat Missions* [Online]. Available: <http://landsat.usgs.gov/>

36. *AVIRIS - Airborne Visible / Infrared Imaging Spectrometer* [Online]. Available: <http://aviris.jpl.nasa.gov/>

37. *Terra-The EOS Flagship* [Online]. Available: <http://terra.nasa.gov>

38. C. Chang, *Hyperspectral Data Exploitation: Theory and Applications*. Hoboken, N.J.: Wiley-Interscience, 2007.

39. *Jet Propulsion Laboratory* [Online]. Available: <http://compression.jpl.nasa.gov/hyperspectral/>

40. Anonymous *Image Data Compression*. Washington D.C.: Recommendation for Space Data Systems Standards. CCSDS 120.1-G-1 Green Book, June 2007.
41. E. Christophe and W. A. Pearlman, "Three-dimensional SPIHT coding of hyperspectral images with random access and resolution scalability," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, 2006, pp. 1897-1901.
42. X. Tang, C. Sungdae and W. A. Pearlman, "3D set partitioning coding methods in hyperspectral image compression," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, 2003, pp. II-239-42 vol.3.
43. E. Christophe, C. Mailhes and P. Duhamel, "Hyperspectral Image Compression: Adapting SPIHT and EZW to Anisotropic 3-D Wavelet Coding," *Image Processing, IEEE Transactions on*, vol. 17, pp. 2334-2346, 2008.
44. G. Liu and F. Zhao, "Efficient compression algorithm for hyperspectral images based on correlation coefficients adaptive 3D zerotree coding," *Image Processing, IET*, vol. 2, pp. 72-82, 2008.
45. H. Ying and L. Guizhong, "Lossy-to-lossless compression of hyperspectral image using the improved AT-3D SPIHT algorithm," in *Computer Science and Software Engineering, 2008 International Conference on*, 2008, pp. 963-966.
46. R. E. Roger and M. Cavenor, "Lossless compression of AVIRIS images," *Image Processing, IEEE Transactions on*, vol. 5, pp. 713-719, 1996.
47. W. Peizhuang, "Pattern Recognition with Fuzzy Objective Function Algorithms (James C. Bezdek)," *SIAM Rev. SIAM Review*, vol. 25, pp. 442, 1983.
48. M. J. Weinberger, G. Seroussi and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *Image Processing, IEEE Transactions on*, vol. 9, pp. 1309-1324, 2000.
49. J. Mielikainen and P. Toivanen, "Clustered DPCM for the lossless compression of hyperspectral images," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 41, pp. 2943-2946, 2003.
50. J. Mielikainen, "Clustered linear prediction for lossless compression of hyperspectral images using adaptive prediction length," pp. 78100M-78100M, August 19, 2010.
51. Z. Jing and L. Guizhong, "An Efficient Reordering Prediction-Based Lossless Compression Algorithm for Hyperspectral Images," *Geoscience and Remote Sensing Letters, IEEE*, vol. 4, pp. 283-287, 2007.
52. H. Chengfu, Z. Rong and P. Tianxiang, "Lossless Compression of Hyperspectral Images Based on Searching Optimal Multibands for Prediction," *Geoscience and Remote Sensing Letters, IEEE*, vol. 6, pp. 339-343, 2009.
53. M. Slyz and D. Zhang, "A block-based inter-band lossless hyperspectral image compressor," in *Data Compression Conference, 2005. Proceedings. DCC 2005*, 2005, pp. 427-436.
54. W. Xiaolin and N. Memon, "Context-based, adaptive, lossless image coding," *Communications, IEEE Transactions on*, vol. 45, pp. 437-444, 1997.

55. W. Xiaolin and N. Memon, "Context-based lossless interband compression extending CALIC," *Image Processing, IEEE Transactions on*, vol. 9, pp. 994-1001, 2000.
56. D. Salomon, *Data Compression: The Complete Reference*. London: Springer, 2007.
57. Модели формирования и некоторые алгоритмы обработки гиперспектральных изображений / Р. Н. Ахметов, Н. Р. Стратилатов, А. А. Юдаков [и др.] // *Исследование Земли из космоса*. – 2014. – № 1. – С. 17.
58. Козинов, И. А. Формирование и обработка гиперспектральных изображений в оптико-электронных системах дистанционного зондирования земли / И. А. Козинов, Г. Н. Мальцев // *Оптика и спектроскопия*. – 2016. – Т. 121. – № 6. – С. 1005-1019.
59. Комплексная обработка гиперспектральных изображений на основе спектральной и пространственной информации / С. М. Борзов, П. В. Мельников, И. А. Пестунов [и др.] // *Вычислительные технологии*. – 2016. – Т. 21. – № 1. – С. 25-39.
60. Саринава, А. Ж. Подготовительная обработка алгоритмов сжатия гиперспектральных аэрокосмических изображений с учетом междиапазонной корреляции с потерями и без потерь / А. Ж. Саринава // *Информационные технологии и математическое моделирование (ИТММ-2016) : Материалы XV Международной конференции имени А.Ф. Терпугова*, – Алтайский край, Алтайский р-он, пос. Катунь: Национальный исследовательский Томский государственный университет, 2016. – С. 123-128. – DOI 10.17223/9785751124335/24.
61. Программный комплекс для формирования и обработки гиперспектральных изображений / С. Л. Погорельский, Е. А. Макарецкий, А. В. Овчинников [и др.] // *Известия Тульского государственного университета. Технические науки*. – 2020. – № 11. – С. 79-84.

## ПРИЛОЖЕНИЕ А

### Листинг программы обработки и сжатия гиперспектральных изображений на основе преобразования Уолша-Адамара

```
using System;
using System.IO;
using System.Windows.Forms;

namespace Преобразование_Уолша_Адамара.Classes
{
    public class WalshHadamarSeveral
    {
        #region Переменные
        int HadamarSize;
        string[] PathsToMatrix;
        string SaveFolder;
        bool SeveralMatrix;
        TextBox TBoxResult;
        #endregion

        public WalshHadamarSeveral(int HadamarSize, string[] PathsToMatrix, TextBox
TBoxResult, string FolderName) // Функция для одного размера
        {
            this.HadamarSize = HadamarSize;
            this.PathsToMatrix = PathsToMatrix;
            this.TBoxResult = TBoxResult;
            SaveFolder = FolderName;
            SeveralMatrix = false;
        }

        public WalshHadamarSeveral(string[] PathsToMatrix, TextBox TBoxResult, string
FolderName, bool SeveralMatrix) // Функция для всех размеров
        {
            this.PathsToMatrix = PathsToMatrix;
            this.TBoxResult = TBoxResult;
            SaveFolder = FolderName;
            this.SeveralMatrix = SeveralMatrix;
        }

        public void MainFunctionSingleHadamar() // Функция для одного размера
        {
            int[,] MatrixWithoutColumns;
            int[,] Matrix;
            int MatrixSize = 0;
            string TempFolder;

            CreateDirectory();

            TempFolder = SaveFolder;
            for (int i = 0; i < PathsToMatrix.Length; i++)
            {
```

```

MatrixSize = GetMatrixSize(PathsToMatrix[i]) * 2;
Matrix = new int[MatrixSize, MatrixSize];
MatrixWithoutColumns = new int[MatrixSize / 2, MatrixSize];
TBoxResult.AppendText("Название файла: " +
Path.GetFileName(PathsToMatrix[i]) + "\n");
if ((MatrixSize / 2) % HadamarSize != 0)
{
TBoxResult.AppendText(string.Format("Не удается выполнить
преобразование с матрицей Адамара [{0}x{0}]{1}{2}{1}",
Environment.NewLine, new string('=', 45))), HadamarSize,
continue;
}
SaveFolder = string.Format("{0}/{1}", TempFolder,
Path.GetFileNameWithoutExtension(PathsToMatrix[i]));
Directory.CreateDirectory(SaveFolder);

#region Функция обратного и прямого преобразования, сравнение
GetMatrixFromFile(Matrix, MatrixSize, PathsToMatrix[i]); // В Matrix -
получаем исходную матрицу

PrintMatrixToFile(Matrix, "Input.rst", MatrixSize);
CopyRDC(PathsToMatrix[i], "Input.RDC");

MakeMatrixWithoutColumns(Matrix, MatrixWithoutColumns, MatrixSize); // В
MatrixWithoutColumns - получаем исходную матрицу без лишних столбцов

int[,] ConversationMatrix = MakeConversation(Matrix, MatrixWithoutColumns,
MatrixSize, MatrixSize / 2, i); // Преобразованная матрица

int[,] ReconverSationMatrix = MakeReconverSation(Matrix, ConversationMatrix,
MatrixSize, MatrixSize / 2, i); // Обратно преобразованная матрица

int[] CompareResult = CompareMatrix(MatrixWithoutColumns,
ReconverSationMatrix, MatrixSize);

TBoxResult.AppendText(string.Format("Количество измененных элементов -
{1}{0}Количество нетронутых элементов - {2}{0}Максимальная разница измененных
элементов - {3}{0}{4}{0}", Environment.NewLine, CompareResult[1], CompareResult[0],
CompareResult[2], new string('=', 45)));
#endregion
}
}

public void MainFunctionSeveralHadamard() // Функция для всех размеров
{
int[,] MatrixWithoutColumns;
int[,] Matrix;
int MatrixSize = 0;
string TempFolder;

CreateDirectory();

```

```

TempFolder = SaveFolder;

for (int i = 0; i < PathsToMatrix.Length; i++)
{
    TBoxResult.AppendText("Название файла: " +
Path.GetFileName(PathsToMatrix[i]) + "\n" + new string('-', 81) + "\n");

    MatrixSize = GetMatrixSize(PathsToMatrix[i]) * 2;
    Matrix = new int[MatrixSize, MatrixSize];
    MatrixWithoutColumns = new int[MatrixSize / 2, MatrixSize];

    SaveFolder = string.Format("{0}/{1}", TempFolder,
Path.GetFileNameWithoutExtension(PathsToMatrix[i]));
    Directory.CreateDirectory(SaveFolder);

    GetMatrixFromFile(Matrix, MatrixSize, PathsToMatrix[i]); // В Matrix -
получаем исходную матрицу

    PrintMatrixToFile(Matrix, "Input.rst", MatrixSize);
    CopyRDC(PathsToMatrix[i], "Input.RDC");

    for (HadamardSize = 2; HadamardSize <= 8; HadamardSize *= 2)
    {
        if ((MatrixSize / 2) % HadamardSize != 0)
        {
            TBoxResult.AppendText(string.Format("Не удается выполнить
преобразование с матрицей Адамара [{0}x{0}]{1}", HadamardSize, Environment.NewLine));
            continue;
        }
        #region Функция обратного и прямого преобразования, сравнение

        MakeMatrixWithoutColumns(Matrix, MatrixWithoutColumns, MatrixSize); //
В MatrixWithoutColumns - получаем исходную матрицу без лишних столбцов

        int[,] ConversationMatrix = MakeConversation(Matrix,
MatrixWithoutColumns, MatrixSize, MatrixSize / 2, i); // Преобразованная матрица

        int[,] ReconversionMatrix = MakeReconversion(Matrix,
ConversationMatrix, MatrixSize, MatrixSize / 2, i); // Обратное преобразованная матрица

        int[] CompareResult = CompareMatrix(MatrixWithoutColumns,
ReconversionMatrix, MatrixSize);
        TBoxResult.AppendText(string.Format("Размерность матрицы Адамара:
{5}x{5}{0}Количество измененных элементов - {1}{0}Количество нетронутых элементов -
{2}{0}Максимальная разница измененных элементов - {3}{0}", Environment.NewLine,
CompareResult[1], CompareResult[0], CompareResult[2], new string('-', 81), HadamardSize));
        if (HadamardSize != 8)
            TBoxResult.AppendText(new string('-', 81) + "\n");
        #endregion
    }
    TBoxResult.AppendText(new string('-', 45) + "\n");
}

```



```

    }

    #region Доп. функции
    private void CreateDirectory()
    {
        TBoxResult.AppendText(string.Format("Название папки: {0} {2} {0} {1} {0}", Environment.NewLine, new string( '=', 45), SaveFolder));
        Directory.CreateDirectory(SaveFolder);
    }

    private int[] CompareMatrix(int[,] InputMatrix, int[,] ReconversionMatrix, int MatrixSize) // Функция для вывода статистики по окончанию работы
    {
        int[] Result = new int[3] { 0, 0, 0 }; //1 - Good | 2 - Bad | 3 - Diff
        int TempW = 0, TempR = 0;
        for (int i = 0; i < MatrixSize / 2; i++)
        {
            for (int j = 0; j < MatrixSize; j++)
            {
                TempW = Convert.ToInt16(InputMatrix[i, j]);
                TempR = Convert.ToInt16(ReconversionMatrix[i, j]);
                if (TempW == TempR)
                    Result[0]++;
                else
                {
                    Result[1]++;
                    if (Math.Abs(TempW - TempR) > Result[2])
                        Result[2] = Math.Abs(TempW - TempR);
                }
            }
        }
        return Result;
    }

    private void CopyRDC(string Path, string FileName) // Копируем файл RDC в папку
    с программой
    {
        try
        {
            string FilePath = System.IO.Path.GetFileNameWithoutExtension(Path);
            File.Copy(System.IO.Path.GetDirectoryName(Path) + @"\" + FilePath + ".RDC",
            SaveFolder + "/" + FileName);
        }
        catch { }
    }
    #endregion

    #region Преобразование
    int[,] MakeConversation(int[,] MainMatrix, int[,] Matrix, int MatrixRow, int
    MatrixColumn, int Index) // Главная матрица преобразования
    {

```

```

int[,] TempMatrix = new int[HadamarSize, HadamarSize]; // Дополнительная
матрица для вывода
int[,] TempMatrixHadamar = new int[HadamarSize, HadamarSize];
int[,] ConversationMatrix = new int[MatrixColumn, MatrixRow];

for (int i = 0; i < MatrixColumn - (HadamarSize - 1); i += HadamarSize)
{
    for (int j = 0; j < MatrixRow - (HadamarSize - 1); j += HadamarSize)
    {
        MakeHadamarMatrix(TempMatrix, Matrix, i, j);

        AdamanFunctionMain(TempMatrix, TempMatrixHadamar);

        SetValueToDgvOut(ConversationMatrix, TempMatrixHadamar, i, j);
    }
}
int[,] PrintMatrix = new int[MatrixRow, MatrixRow];
MakeMatrixForPrint(PrintMatrix, MainMatrix, ConversationMatrix, MatrixRow);
PrintMatrixToFile(PrintMatrix, SeveralMatrix ? "Conversation" + HadamarSize +
".rst" : "Conversation.rst", MatrixRow);
return ConversationMatrix;
}

private void MakeHadamarMatrix(int[,] MatrixOut, int[,] MatrixIn, int X, int Y) //
Получение временной матрицы Адамара
{
    for (int Row = 0; Row < HadamarSize; Row++)
    {
        for (int Column = 0; Column < HadamarSize; Column++)
        {
            MatrixOut[Row, Column] = MatrixIn[X + Row, Y + Column];
        }
    }
}

private void AdamanFunctionMain(int[,] Matrix, int[,] TempMatrix) // Главная
функция, в которой происходит преобразование матриц
{
    int[,] TempHadamarMatrix = new int[,] { };

    ChangeHadamarMatrix(ref TempHadamarMatrix);

    for (int i = 0; i < HadamarSize; i++)
    {
        for (int j = 0; j < HadamarSize; j++)
        {
            TempMatrix[i, j] = GetMatrixSum(HadamarSize, j, i, Matrix,
TempHadamarMatrix) / (HadamarSize * HadamarSize);
        }
    }
}

```

```

private int GetMatrixSum(int Size, int Row, int Column, int[,] MatrixMain, int[,]
HadamarMatrix) // Получение значения элемента после преобразования
{
    int Result = 0;
    for (int i = 0; i < Size; i++)
        Result += MatrixMain[Column, i] * HadamarMatrix[i, Row];
    return Result;
}

void GetMatrixFromFile(int[,] Matrix, int MatrixSize, string PathMatrix) // В матрице
Matrix - исходная матрица из файла
{
    using (FileStream fs = File.OpenRead(PathMatrix))
    {
        BinaryReader br = new BinaryReader(fs);
        for (int Row = 0; Row < MatrixSize; Row++)
        {
            for (int Column = 0; Column < MatrixSize; Column++)
            {
                Matrix[Column, Row] = br.ReadByte();
            }
        }
    }
}

void MakeMatrixWithoutColumns(int[,] InputMatrix, int[,] OutputMatrix, int
MatrixSize) // Убираем "лишние" столбцы
{
    for (int i = 0, OrigColumn = 0, TempColumn = 0; i < MatrixSize; i++)
    {
        if (TempColumn == 0)
            TempColumn++;
        else if (TempColumn == 3)
            TempColumn = 0;
        else
        {
            for (int j = 0; j < MatrixSize; j++)
            {
                OutputMatrix[OrigColumn, j] = InputMatrix[i, j];
            }
            TempColumn++;
            OrigColumn++;
        }
    }
}

int GetMatrixSize(string PathMatrix) // Получить размер матрицы
{
    using (FileStream fs = File.OpenRead(PathMatrix))
    {
        BinaryReader br = new BinaryReader(fs);
        return (int)Math.Sqrt(fs.Length / 4);
    }
}

```

```

    }
}
#endregion

#region Обратное преобразование
public int[,] MakeReconversation(int[,] MainMatrix, int[,] ConversMatrix, int
MatrixRow, int MatrixColumn, int Index) // Главная функция обратного преобразования
{
    int[,] TempMatrix = new int[HadamarSize, HadamarSize]; // Дополнительная
матрица для вывода
    int[,] TempMatrixHadamar = new int[HadamarSize, HadamarSize];
    int[,] ReconversationMatrix = new int[MatrixColumn, MatrixRow];

    for (int i = 0; i < MatrixColumn - (HadamarSize - 1); i += HadamarSize)
    {
        for (int j = 0; j < MatrixRow - (HadamarSize - 1); j += HadamarSize)
        {
            GetReconversationMatrix(TempMatrix, ConversMatrix, i, j);

            ReconvFunctionMain(TempMatrix, TempMatrixHadamar);

            SetValueToDgvOut(ReconversationMatrix, TempMatrixHadamar, i, j);
        }
    }

    string FileName = SeveralMatrix ? ("Reconversation" + HadamarSize + ".") :
("Reconversation.");
    int[,] PrintMatrix = new int[MatrixRow, MatrixRow];
    MakeMatrixForPrint(PrintMatrix, MainMatrix, ReconversationMatrix, MatrixRow);
    PrintMatrixToFile(PrintMatrix, FileName + "rst", MatrixRow);
    CopyRDC(PathsToMatrix[Index], FileName + "RDC");
    return ReconversationMatrix;
}

private void ReconvFunctionMain(int[,] Matrix, int[,] TempMatrix) // Главная
функция, в которой происходит обратное преобразование матриц
{
    int[,] TempHadamarMatrix = new int[,] { };

    ChangeHadamarMatrix(ref TempHadamarMatrix);

    for (int i = 0; i < HadamarSize; i++)
    {
        for (int j = 0; j < HadamarSize; j++)
        {
            TempMatrix[i, j] = GetReconvSum(HadamarSize, j, i, Matrix,
TempHadamarMatrix) * HadamarSize;
        }
    }
}
}

```

```

private int GetReconvSum(int Size, int Row, int Column, int[,] MatrixMain, int[,]
HadamarMatrix) // Получение значения элемента обратного преобразования
{
    int Result = 0;
    for (int i = 0; i < Size; i++)
        Result += MatrixMain[Column, i] * HadamarMatrix[i, Row];
    return Result;
}

void GetReconversionMatrix(int[,] MatrixOut, int[,] MatrixIn, int X, int Y) //
Получение временной матрицы, в зависимости от размера выбранной матрицы
{
    for (int Row = 0; Row < HadamarSize; Row++)
    {
        for (int Column = 0; Column < HadamarSize; Column++)
        {
            MatrixOut[Row, Column] = MatrixIn[X + Row, Y + Column]; // MatrixIn -
уже преобразованная матрица
        }
    }
}
#endregion

#region Печать в файл
private void PrintMatrixToFile(int[,] Matrix, string FileName, int MatrixSize) //
Печать матрицы в файл
{
    using (FileStream fw = File.OpenWrite(SaveFolder + @"\" + FileName))
    {
        BinaryWriter bw = new BinaryWriter(fw);
        {
            for (int i = 0; i < MatrixSize; i++)
                for (int j = 0; j < MatrixSize; j++)
                {
                    bw.Write((byte)Matrix[j, i]);
                }
        }
    }
}

private void MakeMatrixForPrint(int[,] Matrix, int[,] MainMatrix, int[,]
MatrixWithoutCol, int MatrixSize) // Создаем отформатированную матрицу, прибавляя лишние
столбцы
{
    for (int Column = 0, OrigSub = 0, TempColumn = 0; Column < MatrixSize;
Column++)
    {
        if (TempColumn == 0)
        {
            for (int Row = 0; Row < MatrixSize; Row++)
                Matrix[Column, Row] = MainMatrix[Column, Row];
            TempColumn++;
        }
    }
}

```

```

    }
    else if (TempColumn == 3)
    {
        for (int Row = 0; Row < MatrixSize; Row++)
            Matrix[Column, Row] = MainMatrix[Column, Row];
        TempColumn = 0;
    }
    else
    {
        for (int Row = 0; Row < MatrixSize; Row++)
            Matrix[Column, Row] = MatrixWithoutCol[OrigSub, Row];
        TempColumn++;
        OrigSub++;
    }
}

private void SetValueToDgvOut(int[,] MatrixConversation, int[,] Matrix, int Row, int
Column) // Присвоение массиву значения, указанного в параметрах, печать поблочно
{
    int TempColumn = Column;
    for (int i = 0; i < HadamarSize; i++, Row++)
    {
        for (int j = 0; j < HadamarSize; j++, Column++)
        {
            MatrixConversation[Row, Column] = Matrix[i, j];
        }
        Column = TempColumn;
    }
}
#endregion

#region Матрицы
private void ChangeHadamarMatrix(ref int[,] TempMatrix) // Выбор нужной
матрицы Адамара, в зависимости от выбранного числа элементов
{
    switch (HadamarSize)
    {
        case 2:
            TempMatrix = GetHadamar2();
            break;
        case 4:
            TempMatrix = GetHadamar4();
            break;
        case 8:
            TempMatrix = GetHadamar8();
            break;
        default:
            break;
    }
}
}

```

```

private int[,] GetHadamar2() // Получить матрицу Адамара 2x2
{
    return new int[,]
    {
        { 1, 1 },
        { 1, -1 }
    };
}

private int[,] GetHadamar4() // Получить матрицу Адамара 4x4
{
    return new int[,]
    {
        { 1, 1, 1, 1 },
        { 1, 1, -1, -1 },
        { 1, -1, 1, -1 },
        { 1, -1, -1, 1 }
    };
}

private int[,] GetHadamar8() // Получить матрицу Адамара 8x8
{
    return new int[,]
    {
        { 1, 1, 1, 1, 1, 1, 1, 1 },
        { 1, 1, 1, 1, -1, -1, -1, -1 },
        { 1, 1, -1, -1, 1, 1, -1, -1 },
        { 1, 1, -1, -1, -1, -1, 1, 1 },
        { 1, -1, 1, -1, 1, -1, 1, -1 },
        { 1, -1, 1, -1, -1, 1, -1, 1 },
        { 1, -1, -1, 1, 1, -1, -1, 1 },
        { 1, -1, -1, 1, -1, 1, 1, -1 }
    };
}
#endregion
}
}

```

```

using System;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Windows.Forms;
using Преобразование_Уолша_Адамара.Classes;

```

```

namespace Преобразование_Уолша_Адамара
{
    public partial class MainForm : Form
    {
        #region Переменные
        Random random = new Random(); // Рандом для выбора цвета
        int HadamarSize; // Кол-во столбцов/строк в матрице Адамара

```

```

int SizeMain; // Кол-во столбцов/строк в исходной матрице изображения
bool MessageColor; // Для показа сообщения о цвете
string FolderPath; // Рабочая папка для файлов
string SeveralFolder; // Рабочая папка для файлов (несколько файлов)
#endregion

public MainForm()
{
    InitializeComponent();

    CBoxMatrixSize.SelectedIndex = 0;
    CBoxHadamardSizeSeveral.SelectedIndex = 0;
    HadamardSize = Convert.ToInt16(CBoxMatrixSize.Text);
    MessageColor = false;
}

#region Single
#region Цвет таблиц
private void CheckForColorDgv(bool Check) // Выбор цветных/белых таблиц
{
    if (Check)
    {
        if (!MessageColor)
            MessageBox.Show("Цветом отмечаются все матрицы, кроме исходной");
        MessageColor = true;
        int HadamardSize = Convert.ToInt16(CBoxMatrixSize.Text);
        for (int Row = 0; Row < (SizeMain - (HadamardSize - 1)) / 2; Row += HadamardSize)
        {
            for (int Column = 0; Column < SizeMain - (HadamardSize - 1); Column +=
HadamardSize)
            {
                Color CellColor = Color.FromArgb(random.Next(255), random.Next(255),
random.Next(255));
                MakeDgvColored(DgvConversation, Row, Column, CellColor);
                MakeDgvColored(DgvWithoutColumns, Row, Column, CellColor);
                MakeDgvColored(DgvReconversation, Row, Column, CellColor);
            }
        }
    }
    else
    {
        MakeDgvWhite(DgvConversation);
        MakeDgvWhite(DgvReconversation);
        MakeDgvWhite(DgvWithoutColumns);
    }
}

private void MakeDgvColored(DataGridView Dgv, int Row, int Column, Color CellColor) //
Сделать таблицы цветными
{
    int TempColumn = Column;

```



```

for (int i = 0; i < HadamarSize; i++, Row++)
{
    for (int j = 0; j < HadamarSize; j++, Column++)
    {
        Dgv[Row, Column].Style.BackColor = CellColor;
    }
    Column = TempColumn;
}
}

private void MakeDgvWhite(DataGridView Dgv) // Сделать таблицы белыми
{
    for (int Row = 0; Row < DgvWithoutColumns.Rows.Count / 2; Row++)
    {
        for (int Column = 0; Column < DgvWithoutColumns.Columns.Count; Column++)
        {
            Dgv[Row, Column].Style.BackColor = Color.White;
        }
    }
}
#endregion

#region События формы
private void DgvVisibleControl(object sender, EventArgs e) // Контроль за видимостью
таблиц
{
    DgvInput.Visible = RButtonInput.Checked;
    DgvWithoutColumns.Visible = RButtonWithoutColumns.Checked;
    DgvConversation.Visible = RButtonConversation.Checked;
    DgvReconversation.Visible = RButtonReconversation.Checked;
}

private void CBoxDgvColorManager_CheckedChanged(object sender, EventArgs e) // Выбор
цветных/белых таблиц
{
    CheckForColorDgv(CBoxDgvColorManager.Checked);
}

private void CBoxMatrixSize_SelectedIndexChanged(object sender, EventArgs e) //
Изменение матрицы Адамара
{
    HadamarSize = Convert.ToInt16(CBoxMatrixSize.Text);
}

private void ButtonMakeTransformation_Click(object sender, EventArgs e) // Выполнить
преобразование/обратное преобразование
{
    int Size = Convert.ToInt16(CBoxMatrixSize.Text);
    FolderPath = GetPathNameNow();
}

```

```

        WalshHadamarSingle WH = new WalshHadamarSingle(SizeMain, SizeMain / 2, Size,
DgvInput, DgvWithoutColumns, DgvConversation, DgvReconversation, TBoxMain, FolderPath,
OpenFileSingle.FileName);
        WH.MainFunction();
        if (CBoxDgvColorManager.Checked)
            CheckForColorDgv(true);
        ButtonOpenCurrFolder.Enabled = true;
        RButtonConversation.Enabled = true;
        RButtonReconversation.Enabled = true;
    }

private void ButtonOpenRST_Click(object sender, EventArgs e) // Открыть RST
{
    TBoxMain.Clear();
    if (OpenFileSingle.ShowDialog() == DialogResult.OK)
    {
        SizeMain = MatrixSize(OpenFileSingle.FileName) * 2;
        AddRowsColumnsMain(DgvInput, SizeMain);
        AddRowsColumnsAnother(DgvWithoutColumns, SizeMain);
        AddRowsColumnsAnother(DgvConversation, SizeMain);
        AddRowsColumnsAnother(DgvReconversation, SizeMain);
        AddMatrixToDgv(OpenFileSingle.FileName);
        MakeMatrixWithoutColumns();
        CBoxDgvColorManager.Checked = false;
        RButtonWithoutColumns.Enabled = true;
        ButtonConversation.Enabled = true;
        CBoxDgvColorManager.Enabled = true;
        TBoxMain.AppendText(string.Format("Файл успешно загружен! {0} Путь к исходному
файлу: {1} {0} Размер: {2}x{2} {0} {3} {0}", Environment.NewLine, OpenFileSingle.FileName,
SizeMain, new string('=', 62)));
    }
    else
    {
        ButtonOpenCurrFolder.Enabled = false;
        ButtonConversation.Enabled = false;
        RButtonWithoutColumns.Enabled = false;
        CBoxDgvColorManager.Enabled = false;
    }
    RButtonConversation.Enabled = false;
    RButtonReconversation.Enabled = false;
    RButtonInput.Checked = true;
}

private void ButtonOpenCurrFolder_Click(object sender, EventArgs e) // Открыть папку
программы
{
    Process.Start("explorer", FolderPath);
}
#endregion

#region Форматирование таблиц
void MakeMatrixWithoutColumns() // Сделать вторую матрицу без лишних столбцов

```

```

{
    for (int i = 0, OrigColumn = 0, TempColumn = 0; i < SizeMain; i++)
    {
        if (TempColumn == 0)
            TempColumn++;
        else if (TempColumn == 3)
            TempColumn = 0;
        else
        {
            for (int j = 0; j < SizeMain; j++)
            {
                DgvWithoutColumns[OrigColumn, j].Value = DgvInput[i, j].Value;
            }
            TempColumn++;
            OrigColumn++;
        }
    }
}

int MatrixSize(string s) // Определить кол-во столбцов/строк матрицы
{
    using (FileStream fs = File.OpenRead(s))
    {
        BinaryReader br = new BinaryReader(fs);
        return (int)Math.Sqrt(fs.Length / 4);
    }
}

void AddMatrixToDgv(string s) // Добавить матрицу в главную таблицу
{
    using (FileStream fs = File.OpenRead(OpenFileSingle.FileName))
    {
        BinaryReader br = new BinaryReader(fs);
        for (int Row = 0; Row < SizeMain; Row++)
        {
            for (int Column = 0; Column < SizeMain; Column++)
            {
                DgvInput[Column, Row].Value = br.ReadByte();
            }
        }
    }
}

public void AddRowsColumnsMain(DataGridView dgv, int Size) // Добавление строк,
столбцов в главную таблицу
{
    dgv.Columns.Clear();
    dgv.Rows.Clear();

    for (int i = 0; i < Size; i++)
    {
        dgv.Columns.Add("", "");
    }
}

```

```

        dgv.Rows.Add();
        dgv.Columns[i].Width = 30;
    }
}

void AddRowsColumnsAnother(DataGridView dgv, int Size) // Добавление строк в другие
таблицы
{
    dgv.Columns.Clear();
    dgv.Rows.Clear();

    for (int i = 0; i < Size / 2; i++)
    {
        dgv.Columns.Add("", "");
        dgv.Columns[i].Width = 30;
    }

    for (int i = 0; i < Size; i++)
    {
        dgv.Rows.Add();
    }
}
#endregion

private string GetPathNameNow() // Возвращает имя рабочей папки для данного
преобразования
{
    return string.Format("{0} [{1}x{1}] ({2})",
    Path.GetFileNameWithoutExtension(OpenFileSingle.FileName), SizeMain,
    DateTime.Now.ToString("yyyy.MM.dd HH-mm-ss"));
}
#endregion

#region Several
private void ButtonOpenSeveralRST_Click(object sender, EventArgs e) // Открытие RST
файлов
{
    if (OpenFileSeveral.ShowDialog() == DialogResult.OK)
    {
        if (OpenFileSeveral.FileNames.Length == 1)
            MessageBox.Show("Вы выбрали всего 1 файл\nДля одного файла корректнее
использовать\nпреобразование на вкладке 'Один файл'");
        else
            MessageBox.Show("Выбрано файлов: " + OpenFileSeveral.FileNames.Length);
        TBoxFiles.Clear();
        ButtonSeveralConversation.Enabled = true;
        foreach (string FileName in OpenFileSeveral.FileNames)
            TBoxFiles.AppendText(FileName + "\n");
    }
    else
        ButtonSeveralConversation.Enabled = false;
}

```

```

    }

    private void ButtonSeveralConversation_Click(object sender, EventArgs e) // Произвести
преобразования
    {
        TBoxSeveralResult.Clear();
        HadamarSize = Convert.ToInt16(CBoxHadamarSizeSeveral.Text);
        SeveralFolder = DateTime.Now.ToString("yyyy.MM.dd HH-mm-ss") + " (Файлов - " +
OpenFileSeveral.FileNames.Length + ")";
        Stopwatch stopWatch = new Stopwatch();
        stopWatch.Start();
        if (CBoxHadamarAllSizes.Checked)
        {
            WalshHadamarSeveral WS = new WalshHadamarSeveral(OpenFileSeveral.FileNames,
TBoxSeveralResult, SeveralFolder, true);
            WS.MainFunctionSeveralHadamar();
        }
        else
        {
            WalshHadamarSeveral WS = new WalshHadamarSeveral(HadamarSize,
OpenFileSeveral.FileNames, TBoxSeveralResult, SeveralFolder);
            WS.MainFunctionSingleHadamar();
        }
        stopWatch.Stop();
        TimeSpan timeSpan = stopWatch.Elapsed;
        LabelResultTime.Text = "Секунд, затраченных на операцию - " +
timeSpan.TotalSeconds;
        ButtonOpenSeveralFolder.Enabled = true;
    }

    private void ButtonOpenSeveralFolder_Click(object sender, EventArgs e) // Открыть
текущую рабочую папку
    {
        Process.Start("explorer", SeveralFolder);
    }

    private void CBoxHadamarAllSizes_CheckedChanged(object sender, EventArgs e) //
Переключатель для всех размеров
    {
        LabelMatrixSizeSeveral.Enabled = !CBoxHadamarAllSizes.Checked;
        CBoxHadamarSizeSeveral.Enabled = !CBoxHadamarAllSizes.Checked;
    }
    #endregion
}
}

```