МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.04.02 Прикладная математика и информатик
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Исследование алгоритмов обработки медицинских снимков на мобильных устройствах»

Студент Д.В. Шульженко _____
(И.О. Фамилия) (личная подпись)

Научный руководитель к.т.н., Т. Г. Султанов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время, для получения сохранённых на серверах снимков медицинских исследований необходимо использование стационарных рабочих ПК. Применение мобильного устройства в качестве переносной системы мобильного доступа к данным медицинских исследований имеет ряд ограничений, таких как: ограниченная производительность центрального процессора, ограниченный объем оперативной и постоянной памяти. Перечисленные ограничения создают проблему при использовании формата DICOM без применения дополнительных оптимизаций с учётом аппаратных особенностей мобильных устройств.

Актуальность темы исследования заключается в отсутствии готового решения для быстрого получения большого объёма медицинских исследований на мобильном устройстве.

Объектом исследования будет являться процесс получения и обработки снимков медицинских исследований.

Субъектом исследования будет являться реализация оптимизированного алгоритма получения и обработки снимков медицинских исследований.

Предметом исследования будет являться оптимизация процесса получения и обработки снимков медицинских исследований.

Методы исследования: теоретический анализ, синтез и обобщение полученного материала, общенаучные методы: анализ и синтез, индукция и дедукция.

Цель исследования: оптимизация алгоритма получения и обработки медицинских исследований DICOM для мобильных устройств.

Область исследования: алгоритмы обработки графических данных.

Гипотеза: повышение скорости получения и обработки снимков медицинских исследований повысит качество работы врачей, при выполнении следующих задач:

- проанализированы существующие алгоритмы обработки изображений формата DICOM;
- определены основные требования к искомой математической модели обработки исследований и, основанному на ней, техническому решению;
- оптимизирован процесс получения медицинских исследований;
- изучены авторитетные источники в исследуемой области;
- составлены необходимые математические модели обработки медицинских исследований;
- выбран оптимальный вариант решения проблемы, реализовано соответствующее программное обеспечение.

Задачи исследования:

1. провести анализ существующих подходов систем обработки информации в учреждениях здравоохранения;
2. провести анализ существующих программных решений для формализации требований к исследованию;
3. оптимизировать алгоритм получения медицинских исследований;
4. разработать математическую модель преобразования графической информации согласно параметрам центра и ширины окна денситометрических показателей;
5. реализовать разработанную математическую модель с целью анализа и тестирования в сравнении с существующими аналогами.

Публикации по теме исследования. Основные результаты теоретической части исследования изложены в статьях:

1. Шульженко Д.В. Сравнительное исследование эффективности циклического и рекурсивного алгоритмов : сб. науч. тр. / Прикладная

математика и информатика: современные исследования в области естественных и технических наук: материалы V Международной научно-практической конференции (школы-семинара) молодых ученых: 22-24 апреля 2019 г. – Тольятти: Издатель Качалин Александр Васильевич, 2019. 660 с.

2. Шульженко Д.В. Исследование алгоритмов мобильного доступа к сведениям по пациенту : сб. науч. тр. / Прикладная математика и информатика: современные исследования в области естественных и технических наук: материалы VI Международной научно-практической конференции (школы-семинара) молодых ученых: 23-25 апреля 2020 г. – Тольятти: Издатель Качалин Александр Васильевич, 2020.

Научная новизна исследования состоит в том, что в нём обоснована и реализована оптимизация алгоритма получения медицинских исследований, описана математическая модель обработки медицинских данных на мобильном устройстве, позволяющая правильно отображать окно плотности тканей.

Практическая значимость проявится:

- при исследовании целесообразности применения нового подхода в разрезе практической эффективности;
- при получении более высокой скорости получения снимков исследований;
- при повышении качества обслуживания пациентов медицинскими работниками, которые связанны с использованием цифрового медицинского оборудования.

На защиту выносятся:

1. Результаты исследования алгоритма получения медицинских исследований, с обоснованием выбор архитектуры «клиент-сервер», с

применением дополнительного сервера предобработки данных, для решения поставленной задачи.

2. Оптимизированный алгоритм получения медицинских исследований для мобильного устройства
3. Оптимизированный алгоритм обработки медицинских снимков.

Работа состоит из четырех глав. Первая глава является теоретической и описывает исследование и сравнительный анализ существующих мобильных решений для доступа к медицинским исследованиям.

Вторая глава является теоретической и описывает алгоритм мобильного доступа к медицинским исследованиям. Описана и обоснована оптимизированная архитектура с применением сервера предобработки данных. Описана математическая модель растрового изображения. Исследована возможность преобразования отображаемой плотности тканей. Исследована возможность распараллеливания алгоритма обработки плотностей снимков исследований.

Третья глава носит практический (технологический) характер. В этой главе описывается выбор средств реализации, описан процесс разработки архитектуры мобильного клиента. Описан процесс реализации математических моделей обработки снимков исследований.

Четвертая глава является заключительной и в ней представлено обоснование применения архитектуры с применением сервера предобработки данных, а также применения распараллеливания для обработки снимков исследований.

Работа изложена на 100 страницах и включает 49 рисунков, 27 таблиц, 35 источников, 1 приложение.

1 АНАЛИЗ СУЩЕСТВУЮЩЕЙ СИСТЕМЫ ПРИМЕНЕНИЯ МОБИЛЬНЫХ УСТРОЙСТВ В МЕДИЦИНЕ

1.1 Обзор статистики и исследований по вопросу заинтересованности в применении мобильных устройств в медицине

По статистике Росстата в России насчитывается 5.3 тыс. больничных организаций [16].

Для оценки заинтересованности в данном исследовании возьмём существующие исследования по вопросу использования мобильных устройств в медицинских целях.

По данным Manhattan Research 81% врачей в 2011 году использовали смартфоны/планшеты для доступа к медицинским данным. Причём в 2010 году данный показатель был на уровне 72%.

American Electronics Association было проведено исследование. Докторов и пациентов спрашивали насколько им нравится использование беспроводных устройств (смартфонов/планшетов). Большинство респондентов хотели бы использовать их для связи с доктором, фармацевтом, медсестрой и наоборот. Многие хотели бы иметь возможность хранить и получать медицинские записи, включая диагностические исследования с помощью мобильного устройства.

На рисунке 1.1.1 изображены результаты опроса по каждому из вопросов [35].

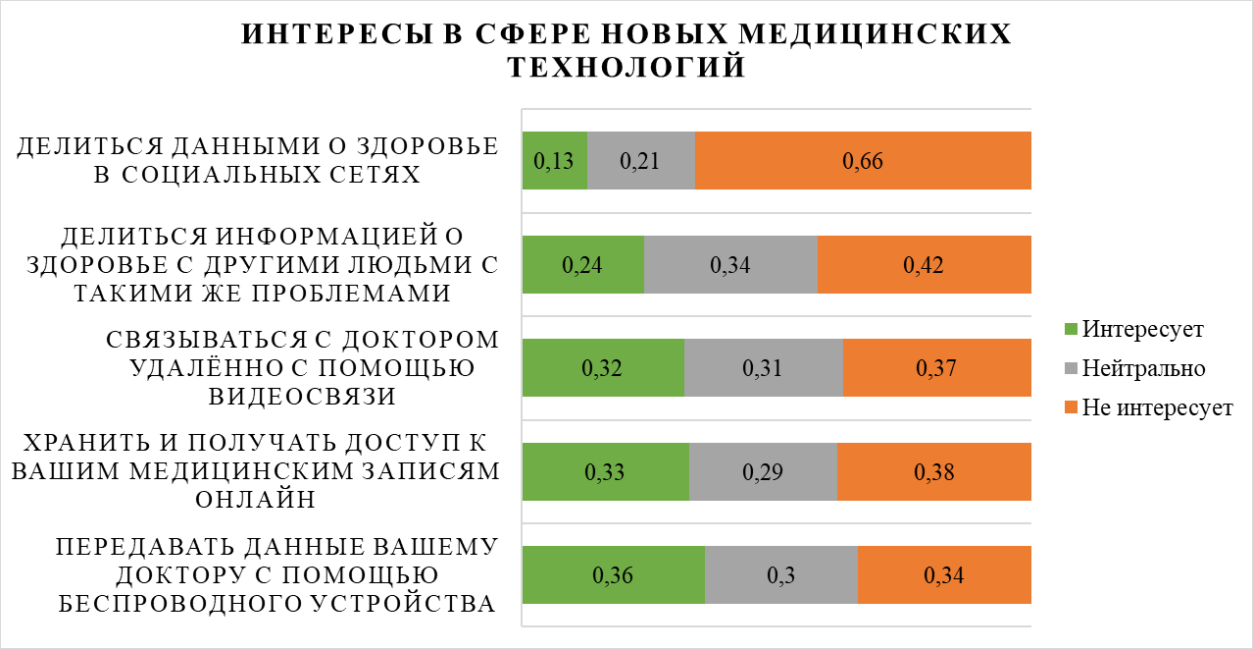

Рисунок 1.1.1 - Интересы в сфере новых медицинских технологий

Таким образом, больше трети респондентов заинтересованы в использовании мобильных устройств в медицине.

Ниже приведены основные преимущества использования смартфонов/планшетов:

- планшеты избавляют от необходимости записывать информацию на бумаге и вводить в системы (это экономит время, энергию, деньги и повышает эффективность);
- легкий доступ к информации: планшеты обеспечивают быстрый доступ к информации там, где это требуется медицинскому персоналу;
- минимизация бумажной работы: планшеты помогают минимизировать грязную бумажную работу и процесс ручного рабочего процесса - опять же, повышая эффективность;
- коммуникационные возможности: позволяет врачам, медсестрам и другому медицинскому персоналу общаться виртуально и более эффективно;

- конфиденциальность и безопасность данных: планшеты обеспечивают 128 и 256-битное шифрование данных при хранении и передаче (это минимизирует утечку данных и нарушения безопасности из-за ручной обработки незашифрованных бумажных форм, и других печатных документов).

Учитывая опыт медицинских учреждений в других странах и оценку заинтересованности в новых технологиях, в России потенциально заинтересованными могут быть:

$$5300 \times 0.8 \times 0.4 = 1696 \text{ учреждений.}$$

По данным Росстата количество частных клиник на 2016 год составляет 23173.

Потенциально заинтересованными могут быть:

$$23173 \times 0.8 \times 0.4 = 7415 \text{ учреждений.}$$

Таким образом, около 9111 учреждений могут быть потенциально заинтересованы в использовании результатов данного исследования.

1.2 Анализ существующих мобильных решений для работы с медицинскими исследованиями

Для определения оптимальных характеристик и возможностей разрабатываемого решения необходимо проанализировать существующие аналоги. В результате необходимо провести сравнительный анализ рассмотренных аналогов.

1.2.1 Анализ программного обеспечения mRay

mRay является CE-сертифицированным DICOM (Digital Imaging and Communications in Medicine) Viewer и клиентом PACS (Picture Archiving and Communication System), который включает в себя множество функций, таких как mRay-VEOcore, который поддерживает планирование терапии при остром инсульте, KI-Tool, который обеспечивает оптимизированный CT и

MRT перфузионный анализ. Не поддерживает открытие локальных файлов. Чтобы иметь возможность получать изображения в mRay с PACS сервера, необходимо программное обеспечение mRay Server, которое является платным. Окно просмотра исследований mRay представлено на рисунке 1.2.1.

Рисунок 1.2.1 – Окно просмотра исследований mRay

Особенности:

- CE-сертифицированный;
- просмотр магнитно-резонансной томографии (МРТ), флюороскопии (ХА), цифровой рентгенографии (КР), компьютерной томографии (КТ), ультразвука (США);
- полнофункциональный клиент, без просмотра удаленного рабочего стола;
- безопасное многоуровневое шифрование;
- система mRay - VEOcore - поддерживает планирование терапии острого инсульта;

- система KI-Tool - обеспечивает оптимизированный КТ и МРТ перфузионный анализ;
- сканер штрих-кода;
- автоматически создавать структурированные отчеты из заказов HL7;
- интегрированный Instant Messenger, который предлагает безопасную коммуникационную платформу для обмена изображениями DICOM, а также ключевыми изображениями и простыми текстовыми и аудио сообщениями;
- регулировка окна, измерения, толстые срезы и аннотации;
- поддержка Query/Retrieve.

В результате рассмотрено программное обеспечение mRay.

1.2.2 Анализ программного обеспечения Symmerty

Symmetry - это мощное приложение для визуализации, обработки, организации и взаимодействия с наборами научных / медицинских данных. Он основан на библиотеке Simulacrum, которая представляет собой пакет с открытым исходным кодом для взаимодействия тома и пространства изображения.

Окно просмотра исследований Symmetry DICOM Viewer представлено на рисунке 1.2.2.

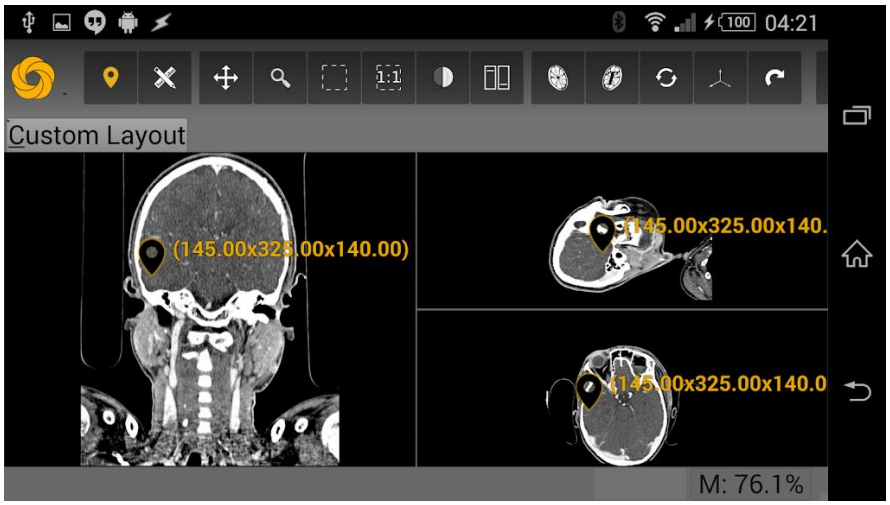

Рисунок 1.2.2 - Окно просмотра исследований Symmetry DICOM Viewer

Работа с удаленным хранилищем данных представлена на рисунке 1.2.3.

Рисунок 1.2.3 - Symmetry DICOM Viewer - Работа с удаленным хранилищем данных

Особенности:

- поддерживает открытие файлов из локального хранилища;
- кроссплатформенность: Windows, Linux и Android;
- простой просмотр на одной панели или расширенная конфигурация столов с несколькими панелями;
- стандартные инструменты для работы с изображениями (масштабирование, панорамирование, уровень окна, прокрутка томов);

- интеграция с DICOM / PACS (команды Query/Retrieve, WADO fulfilment, DICOM networking send/receive/echo);
- поддержка пользовательских расширений – плагины, скрипты.

В результате рассмотрено программное обеспечение Symmetry DICOM Viewer.

1.2.3 Анализ программного обеспечения iPaxera

iPaxera может добавлять исследования (изображения DICOM) с ПК, компакт-диска DICOM или флэш-памяти напрямую на устройство через локальную синхронизацию с помощью кабеля USB (не требуется подключение к Wi-Fi / 3G или сервер PACS). Окно просмотра исследований iPixera представлено на рисунке 1.2.4.

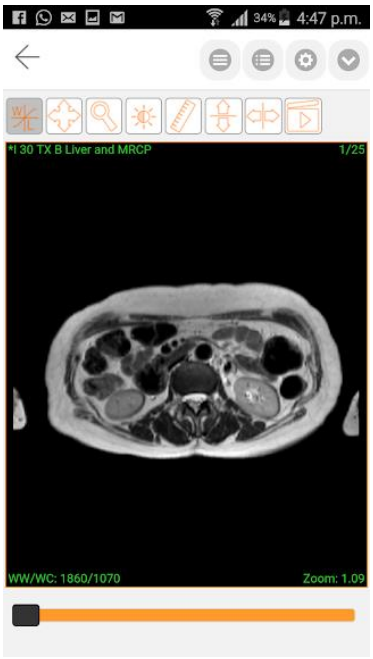

Рисунок 1.2.4 – Окно просмотра исследований iPixera

Особенности:

- создание отчетов с помощью встроенного инструмента отчетности и их отправка по электронной почте или сохранение с исследованиями;

- встроенный инструмент диктовки для записи голосовых файлов, позволяющий создавать отчеты на ходу;
- связь с любым PACS через DICOM Query/Retrieve, позволяя вам добавлять столько параметров узла DICOM, сколько необходимо
- поддерживает несколько модальностей, включая DICOM Cine (многокадровый) с интерактивными элементами управления воспроизведением;
- имеет простые в использовании инструменты обработки, такие как ширина окна и выравнивание с визуальными предустановками, измерение расстояния, панорамирование, масштабирование, свободное вращение, захват снимков и быстрая навигация для изображений/серий;
- поиск по типу модальности, дате (все, сегодня, вчера, 2 дня, 1 неделя, 2 недели, 1 месяц и 2 месяц).

В результате рассмотрено программное обеспечение iPixera.

1.2.4 Анализ программного обеспечения DroidRender

DroidRender - программа для просмотра 3D DICOM для Android, которая имеет 2D/3D мульти плоскость реконструкции и полнофункциональный движок 2D рендеринга. Окно просмотра исследований DroidRender на рисунке 1.2.5.

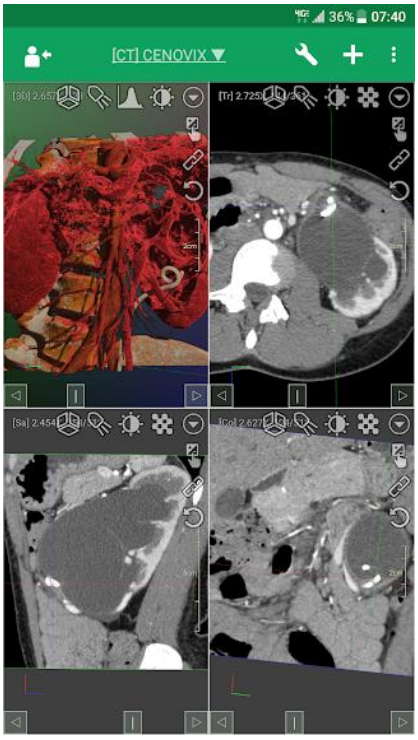

Рисунок 1.2.5 - Окно просмотра исследований DroidRender

Особенности:

- поддержка файлов DICOM, поддержка несжатого/сжатого формата DICOM;
- 2D / 3D мульти плоскостная реконструкция;
- полнофункциональный движок 2D рендеринга (шкала серого, таблица соответствия цветов, проекция максимальной интенсивности, проекция минимальной интенсивности, среднее значение);
- сегментация и отображение тканей.

В результате рассмотрено программное обеспечение DroidRender.

1.2.5 Анализ программного обеспечения simplyDICOM

simplyDICOM Viewer – программа для просмотра DICOM снимков с открытым исходным кодом и минимальным интерфейсом. Использует библиотеки OpenCV и dcm4che. Окно просмотра исследований simplyDICOM представлено на рисунке 1.2.6.

Рисунок 1.2.6 – Окно просмотра исследований simplyDICOM

Особенности:

- простой интерфейс;
- поддерживает открытие файлов из локального хранилища.

В результате рассмотрено программное обеспечение simplyDICOM.

1.3 Сравнительный анализ основных возможностей программного обеспечения для работы с медицинскими исследованиями

Обобщим представленные выше описания программ в единую таблицу 1.3.1.

Выделим основные задачи таких программ:

- обеспечивать работу с PACS сервером;
- позволять регулировать окно денситометрических показателей.

Также дополнительно выделим следующие параметры:

- локальный просмотр DICOM;
- поддержка 3D визуализации;
- поддержка измерения расстояний на снимке.

Таблица 1.3.1 – Сравнительный анализ характеристик существующих аналогов

	Работа с PACS сервером	**Регулировка окна денситометрических показателей**	**Локальный просмотр DICOM**	**Поддержка 3D визуализации**	**Поддержка измерения расстояний на снимке**	**Скорость доступа к удалённым данным**	**Возможность удалённого доступа к отдельному исследованию**
mRay	- (только через платный сервер mRay Server)	+	-	-	+	5	+
Symmetry	+/-	+	+	-	+	3	-
iPaxera	+ (требует регистрацию)	+	-	-	+	3	-
DroidRender	-	+	+	+	+	-	-
simplyDICOM	-	-	+	-	-	-	-

Из сравнения видно, что четыре из пяти, представленных для сравнения, программ позволяют регулировать окно плотностей тканей, две поддерживают работу с PACS сервером. Только две из представленных программ поддерживают и регулировку окна плотностей, и работу с PACS сервером, однако работа с PACS сервером поддерживается с ограничениями.

Только одна из представленных программ – mRay, позволяет организовать оперативный доступ к данным по пациенту через свой промежуточный сервер mRay Server (на платной основе). Также mRay позволяет получать удалённый доступ к конкретному исследованию, не выгружая лишних данных, что реализовано собственным сервером предобработки данных mRay Server.

1.4 Исследование текущей технологии доступа к исследованиям

Для определения цели модели, составим список вопросов, на которые она должна отвечать:

1. Кто проводит исследования?
2. Кто использует исследования?
3. Кто имеет доступ к исследованиям?
4. Кто проводит обслуживание системы?

Исходя из поставленных вопросов, следует выбрать точку зрения инженера. Это даст возможность полностью на них ответить.

Сформулируем цель моделирования: определение, каким образом происходит получение сведений о проведённых исследованиях.

Данная система будет рассматриваться с точки зрения вариантов использования. Диаграмма прецедентов даст возможность определить функциональную структуру системы, не углубляясь в детали её реализации [15].

Выделим актёров системы:

1. лечащий врач;
2. оператор оборудования (врач, проводящий исследования);
3. программный инженер (программист);
4. системный администратор оборудования (инженер).

Опишем методы использования автоматизированной системы:

Врач (лечащий) использует систему для проведения исследований пациентов, чтобы сформировать заключение и провести лечение.

Врач, проводящий исследование, проводит исследование пациентов с использованием медицинского оборудования, подключенного к информационной системе.

Инженер-программист и системный администратор поддерживают систему в рабочем состоянии, осуществляют ее обслуживание и настройку, а также следят за функционированием системы.

Таким образом, можно выделить следующие прецеденты, которые представлены в таблице 1.4.1.

Таблица 1.4.1 – Описание прецедентов

ID	Прецедент	Краткое описание
1	Проведение исследования	Обследование пациента с использованием медицинского оборудования
2	Просмотр сведений по пациенту	Анализ исследования с целью установления диагноза
3	Обслуживание системы	Проведение своевременного обслуживания системы

На рисунке 1.4.2 представлена диаграмма прецедентов.

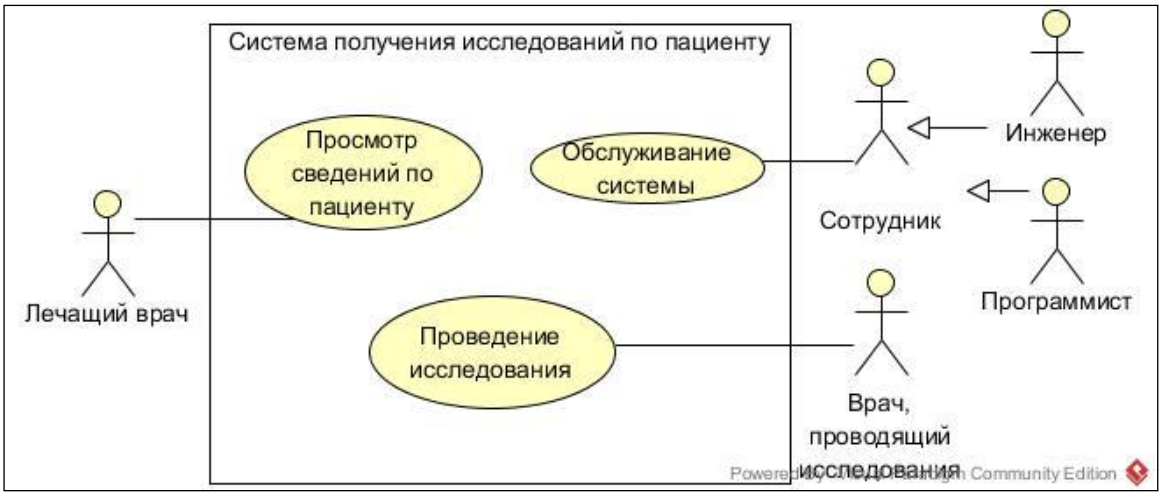

Рисунок 1.4.2 – Диаграмма прецедентов системы получения информации и исследований пациента

В таблицах 1.4.2-1.4.4 приведена спецификация прецедентов.

Таблица 1.4.2 – Прецедент: Проведение исследования

Прецедент: Проведение исследования
ID: 1
Краткое описание: Обследование пациента с использованием медицинского оборудования
Главные актеры: Оператор оборудования (врач)
Второстепенные актеры: Нет
Предусловия: Прецедент начинается по инициативе пользователя
Основной поток: 1. Пользователь проводит исследование пациента при помощи медицинского оборудования 2. Данные исследований передаются с устройства в базу данных информационной системы.
Постусловия: Исследование проведено
Альтернативные потоки: Нет

Таблица 1.4.3 – Прецедент: Просмотр сведений по пациенту

Прецедент: Просмотр сведений по пациенту
ID: 2
Краткое описание: Врачебный анализ исследования для установления диагноза
Главные актеры: Лечащий врач
Второстепенные актеры: Нет
Предусловия: Прецедент начинается по инициативе пользователя
Основной поток: 1. Поиск нужного исследования. 2. Исследование не найдено, выход из прецедента. 3. Исследование найдено: 1. Провести анализ исследования 2. Сделать заключение
Постусловия: Врач получил данные исследования
Альтернативные потоки: Нет

Таблица 1.4.4 – Прецедент: Обслуживание системы

Прецедент: Обслуживание системы
ID: 3
Краткое описание: Своевременное обслуживание системы
Главные актеры: Программный инженер (Программист), Системный администратор оборудования (Инженер)
Второстепенные актеры: Нет
Предусловия: Прецедент начинается по регламенту или по состоянию системы
Основной поток: 1. Поиск неисправности. 2. Неисправность найдена: 1. Проведение ремонта 3. Неисправности не найдены: 1. Выполняется выход из прецедента
Постусловия: Обслуживание системы проведено
Альтернативные потоки: Нет

Определены основные технологические этапы:

1. проведение исследования с помощью медицинского оборудования;
2. анализ врачом произведённых исследований с целью установления диагноза.

В результате были проанализированы основные технологические этапы, которые используются для получения исследований по пациентам.

1.5 Исследование алгоритма работы существующей системы доступа к сведениям по пациенту

Поведение системы – это «описание последовательности действий, без определения механизма их реализации». Одной из его частей является диаграмма последовательности.

Далее приведены диаграммы последовательностей для трёх основных прецедентов, выделенных в параграфе 1.2. Диаграммы основных прецедентов представлены на рисунке 1.5.1, 1.5.2, 1.5.3.

Рисунок 1.5.1 – Диаграмма по прецеденту – Проведение исследования

Рисунок 1.5.2 – Диаграмма по прецеденту – Просмотр сведений по пациенту

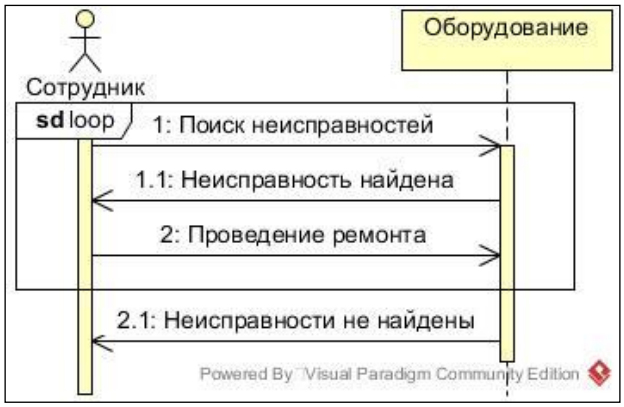

Рисунок 1.5.3 – Диаграмма по прецеденту – Обслуживание системы

В результате на данном этапе разработки были выделены основные этапы работы сотрудников по проведению и получению доступа к исследованиям с использованием медицинской автоматизированной информационной системы. Этапы рассматривались по принципу «чёрного ящика», это позволило определить последовательность работы сотрудников с системой. В этом случае можно выделить наиболее важные этапы – проведение исследования, просмотр информации о пациенте и обслуживание системы.

Таким образом, исходя из данных п. 1.3 и, в частности, таблицы 1.3.1, понятно, что оптимальным решением для реализации мобильного доступа к удаленным исследованиям по пациенту является создание программного комплекса, включающего в себя мобильное приложение – клиент и промежуточный сервер предобработки данных, позволяющий ускорить получение необходимых сведений.

2 РАЗРАБОТКА АЛГОРИТМА МОБИЛЬНОГО ДОСТУПА И МАТЕМАТИЧЕСКОЙ МОДЕЛИ РЕГУЛИРОВКИ ПЛОТНОСТЕЙ ТКАНЕЙ ПРИ ИСПОЛЬЗОВАНИИ РАСТРОВОГО ИЗОБРАЖЕНИЯ

2.1 Разработка архитектуры мобильного доступа к серверу хранения данных по пациентам PACS

2.1.1 Исследование архитектуры системы мобильного доступа к сведениям по пациенту

Разработка архитектуры является фундаментальным фактором при реализации проектов. Специфика разработки медицинского приложения определяет необходимость поддержки системой формата DICOM в связи с тем, что он является международным отраслевым стандартом для передачи, хранения, обработки и отображения медицинских данных [4], а также реализация обработки снимков исследований. Стандартные инструменты, включенные в SDK, не позволяют работать с этим форматом, но есть ряд сторонних библиотек. На рисунке 2.1.1 представлена диаграмма компонентов системы на основе архитектуры клиент-сервер.

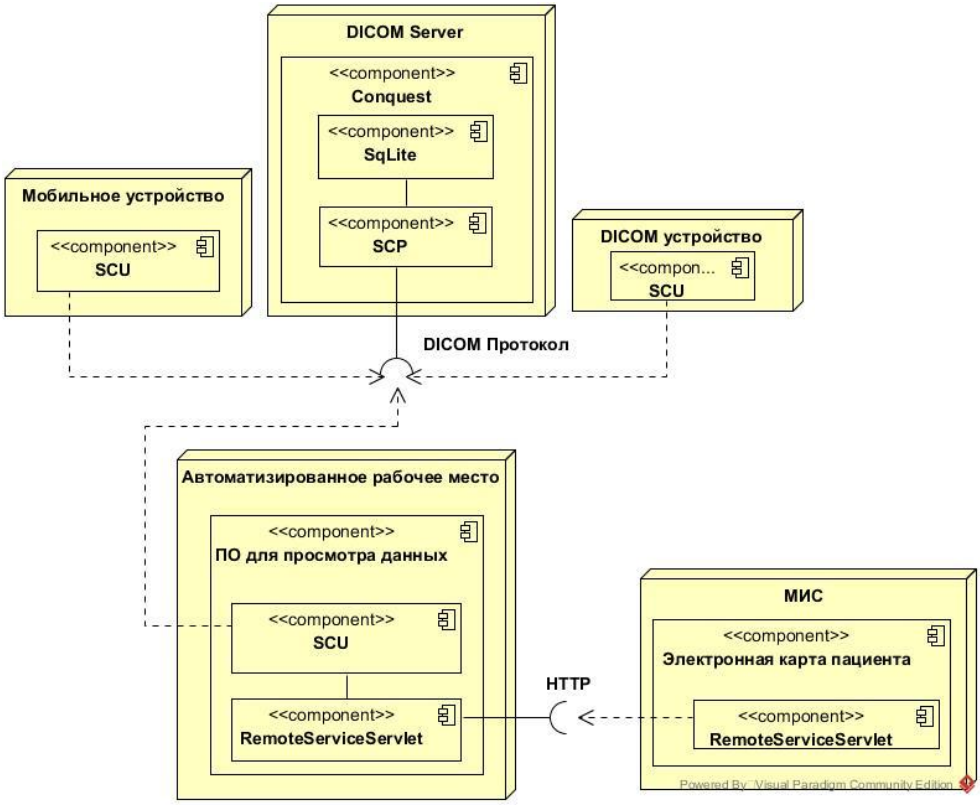

Рисунок 2.1.1 – Диаграмма компонентов системы на основе архитектуры клиент-сервер

Как видно из диаграммы на рисунке 2.1.1, мобильное устройство с установленным клиентом DICOM напрямую связывается с сервером DICOM по протоколу DICOM.

Чтобы оценить производительность системы, построенной на основе этой архитектуры, мы будем измерять время обработки изображения на мобильном устройстве, как критерий, наиболее требовательный к производительности системы. Мы также оценим время загрузки изображений.

Напишем приложение, которое будет использовать данные, хранящиеся в памяти устройства (время передачи исследований будет замеряться отдельно). Анализ будет проводиться для инструментов DicomDroid и Imebra Software Development Kit.

DicomDroid – библиотека, распространяющаяся под лицензией свободного программного обеспечения GNU GPL v3, написана на языке программирования Java. DicomDroid может быть включена в качестве внешнего инструментария в проекты Android. Позволяет производить экспорт изображений DICOM в виде растрового изображения (android.graphics.Bitmap). Также позволяет извлекать из DICOM параметры исследования и атрибутику [5].

В настоящее время поддерживаются только несжатые изображения. Также обработка изображения может быть достаточно медленной на относительно слабых устройствах (один снимок может обрабатываться до одной минуты).

Серьёзным недостатком библиотеки DicomDroid являются артефакты при отображении DICOM снимков. Артефакты изображены на рисунке 2.1.2. Также данная библиотека не поддерживает регулировку окна плотности тканей на снимке исследования.

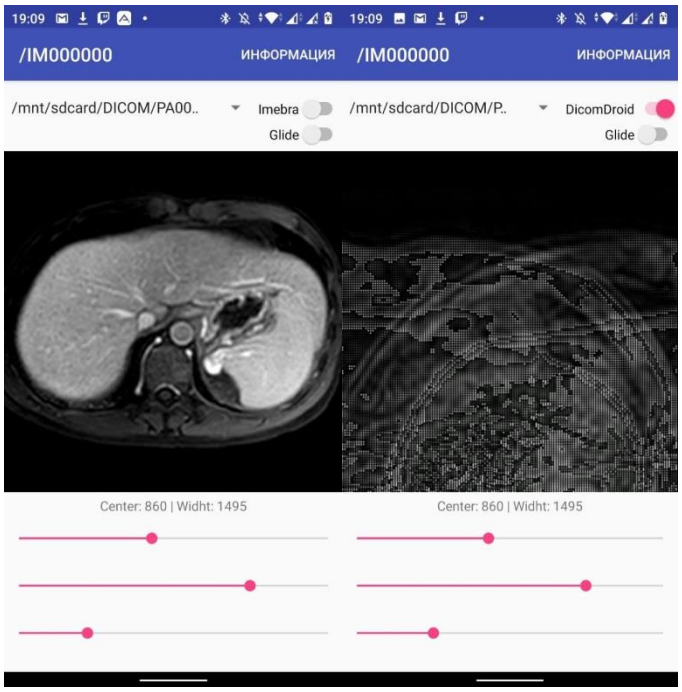

Рисунок 2.1.2 – Артефакты при работе библиотеки DicomDroid

(слева Imebra, справа - DicomDroid)

Поддержка проекта DicomDroid завершена 19 января 2011 года.

Imebra – библиотека, написанная на языке программирования C++. «Способна анализировать и создавать файлы DICOM» (включая файлы DICOMDIR – образ с массивом файлов DICOM). «Она также предоставляет необходимые кодеки для кодирования и декодирования изображений, встроенных в файлы DICOM» (поддерживает сжатые и несжатые изображения) [26].

Кроме того, инструменты «библиотеки позволяют приложению читать или записывать теги DICOM или сохранять новые изображения в наборе данных». Imebra предоставляется бесплатно в соответствии с GNU GPL и коммерческой лицензией. Стоимость коммерческой лицензии составляет: для пяти разработчиков (подписка на год) – 2499$, для неограниченного числа разработчиков (подписка на год) – 4999$.

Описание разрешений согласно типу исследования, представлено в таблице 2.1.1.

Таблица 2.1.1 – Соотношение разрешения изображения типу исследования

Разрешение (пикс)	**Тип исследования**
256x256	Магнитно-резонансная томография (МРТ)
512x512/560x560	Компьютерная томография (КТ)
1728x2328	Компьютерная рентгенография

Время обработки с помощью библиотеки DicomDroid (Java) представлено в таблице 2.1.2 (выборка – 1000 снимков для каждого из разрешений).

Таблица 2.1.2 – Время обработки DicomDroid

#	**160x160**	**256x256**	**288x288**	**320x320**	**384x384**	**560x560**
Среднее (мс)	13	37	35	44	63	135
Медианное (мс)	11	28	31	39	56	131

Обработка с помощью библиотеки Imebra (C++) представлена в таблице 2.1.3 (выборка – 1000 снимков для каждого из разрешений).

Таблица 2.1.3 – Время обработки Imebra

#	160x160	256x256	288x288	320x320	384x384	560x560
Среднее (мс)	8	17	11	16	18	25
Медианное (мс)	6	15	8	12	14	20

Далее рассматривается только библиотека Imebra.

Измерим время, необходимое для получения одного DICOM снимка в локальной сети (802.11n). Полученные данные приведены в таблице 2.1.4.

Таблица 2.1.4 – Время загрузки изображений формата DICOM

Разрешение (пикс)	Время (мс)	
	Минимально	Максимально
256x256	553	19882
512x512	6548	49056
1728x2328	38774	83459

В сумме время получения и обработки одного изображения формата DICOM составляет от 0,5 секунд до 1 минуты 20 секунд.

2.1.2 Исследование архитектуры системы мобильного доступа к сведениям по пациенту с использованием сервера предобработки данных

Дополнительно формализуем процесс получения и отображения DICOM файлов на мобильном устройстве. Диаграмма последовательности получения и обработки исследований изображена на рисунке 2.1.5.

Рисунок 2.1.5 – Получение и обработка исследований с помощью мобильного устройства

Устройство производит конвертацию полученного DICOM файла (п. 5) и далее, передавая данные на обработку в SDK, отображает его на экране мобильного устройства (п. 6, 7).

В таблице 2.1.5 представлено время обработки одного снимка разных (выборка – 1000 снимков для каждого из разрешений).

Таблица 2.1.5 – Время обработки одного изображения

#	**160x160**	**256x256**	**288x288**	**320x320**	**384x384**	**560x560**
Среднее (мс)	0,138057	0,123932	0,129468	0,197961	0,231982	0,145672
Медианное (мс)	0,12724	0,119662	0,124141	0,181797	0,202448	0,131771

На рисунке 2.1.6 представлена сводная диаграмма, из которой видно сколько времени, относительно всего процесса отображения DICOM снимка, который включает в себя конвертацию исследования, занимает время его обработки на финальном этапе системными средствами SDK.

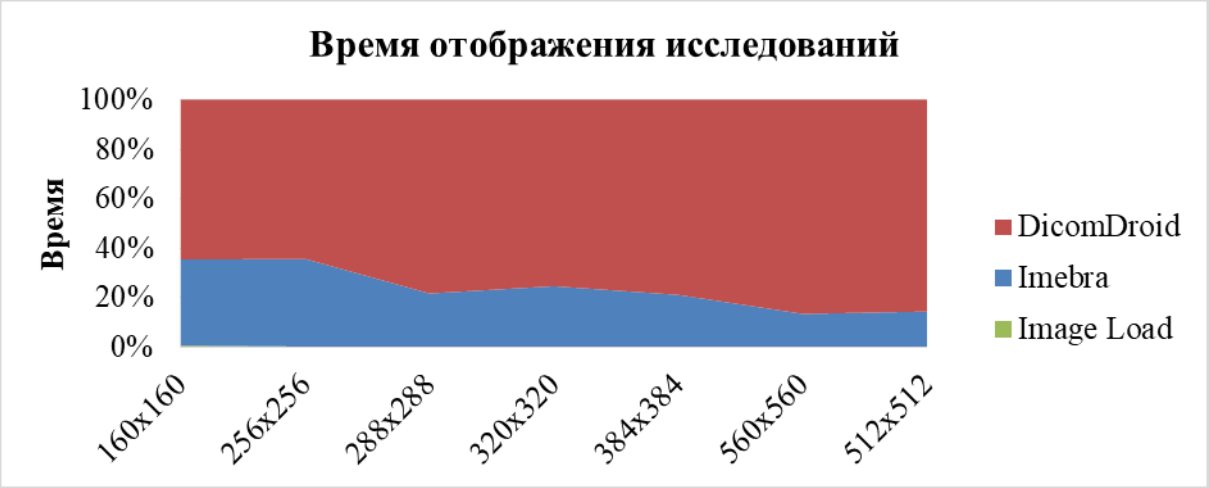

Рисунок 2.1.6 – Диаграмма относительного времени отображения исследований

Как видно, время непосредственной обработки финального изображения (Image Load), по сравнению со временем, необходимым для конвертации DICOM, ничтожно мало. Таким образом, если исключить из алгоритма, представленного на рисунке 2.1.7, передать шаги по работе с протоколом DICOM (2, 3, 4 и 5) серверу предобработки и буферизации данных, возможно, получить выигрыш в скорости получения исследований.

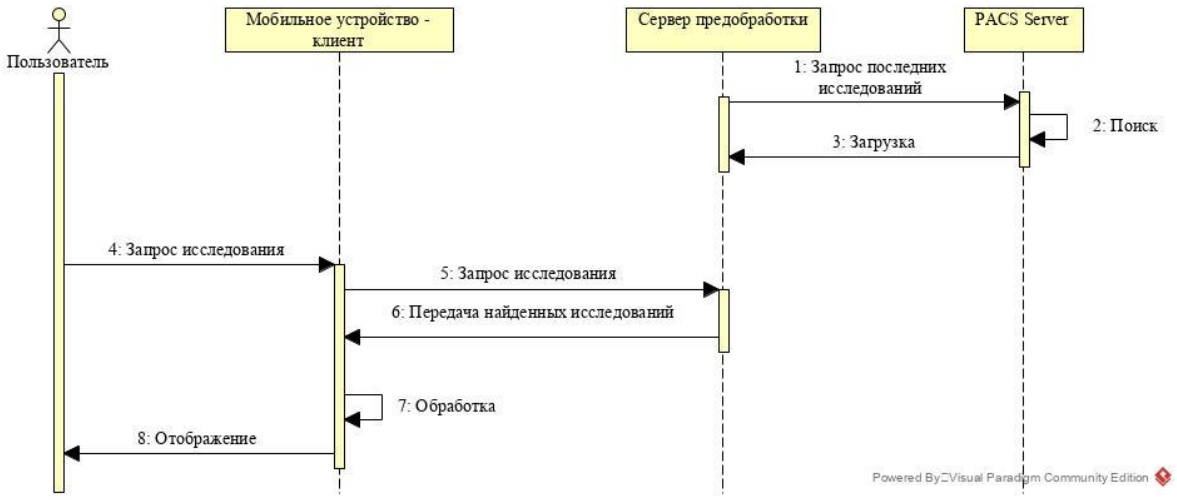

Рисунок 2.1.7 - Получение и обработка исследований с помощью мобильного устройства с использованием сервера предобработки данных

Необходимо разработать архитектуру, с учётом новых требований. Диаграмма компонентов представлена на рисунке 2.1.8.

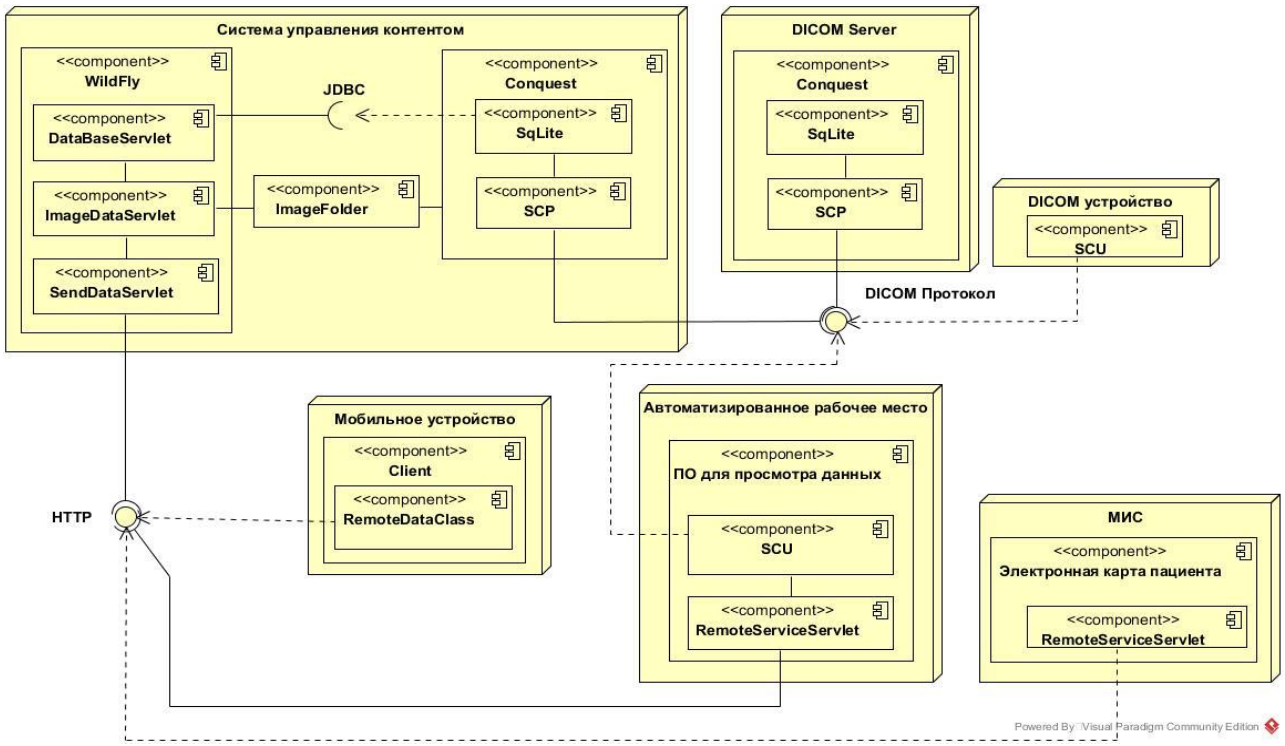

Рисунок 2.1.8 – Диаграмма компонентов системы на основе архитектуры с применением промежуточного сервера

Методика проведения исследования аналогична пункту 2.1.1.

Время загрузки одного изображения JPG при подключении по локальной беспроводной сети (802.11n) приведено в таблице 2.1.6.

Таблица 2.1.6 – Время загрузки изображений формата JPG

Разрешение (пикс)	Время (мс)	
	Минимально	Максимально
256x256	101	1802
512x512	1190	4467
1728x2328	7050	7587

В сумме время получения и обработки одного изображения формата JPG составляет от 0,1 до 8 секунд.

Таким образом, получены следующие данные:

В сумме время получения и обработки одного изображения формата DICOM составляет от 0,6 секунд до 1 минуты 25 секунд.

В сумме время получения и обработки одного изображения формата JPG составляет от 0,1 до 8 секунд.

2.2 Проблема регулировки окна денситометрических показателей при использовании растрового формата изображений

Использование мобильного устройства в качестве медицинской техники имеет свои особенности. Так, одними из основных параметров, при просмотре снимков исследований, являются параметры Центр и Ширина окна денситометрических показателей. Правильная регулировка и применение данных параметров качественным образом влияет на опыт работы пользователя, помогает правильно считать информацию со снимка.

На рисунке 2.2.1 показано влияние изменения параметра ширины окна плотностей тканей. Когда ширина окна увеличивается, становится доступно больше информации, так как отображается более широкий диапазон плотностей.

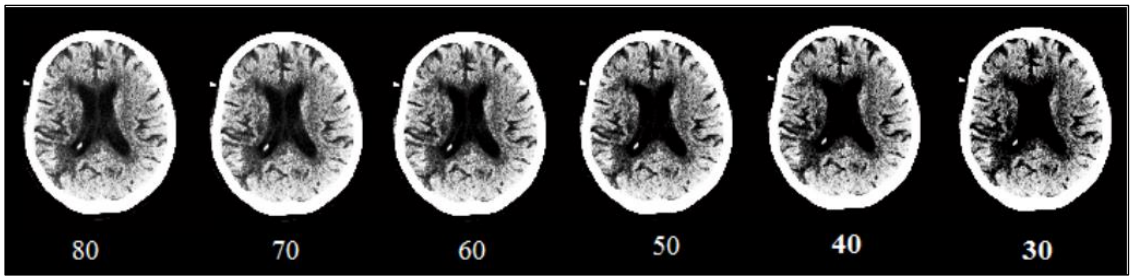

Рисунок 2.2.1 – Изменение значение параметра ширины окна

На рисунке 2.2.2 показано влияние изменение параметра центра окна. То есть, при неизменной ширине, происходит перемещение окна в целом. В

данном примере использованы значения (слева на право) – 800, 1065 и 1100 HU. При ширине окна в 100 HU и центре окна 800 HU окно будет иметь диапазон от 700 до 900 HU.

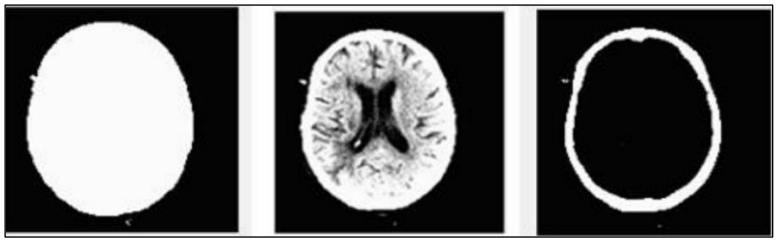

Рисунок 2.2.2 – Изменение значение параметра центра окна

Установка правильных, то есть наиболее информативных в этой ситуации, значений ширины и центра окна имеет решающее значение для выявления патологий. Контрольное значение для определенной части тела определяется эмпирически и не обязательно подходит для конкретного исследования. «Центр и ширина окна регулируются один за другим и наблюдаются человеческим глазом с целью наилучшего значения для конкретного исследования» [32].

2.3 Исследование структуры хранения данных в формате DICOM

Для реализации возможности управления параметрами регулировки атрибутов, отвечающих за отображение плотностей тканей, необходимо проанализировать структуру формата хранения данных медицинских исследований формата DICOM.

DICOM, как протокол, позволяет хранить результаты передачи данных в одноименном формате. Стандартная структура такого файла представлена на рисунке 2.3.1.

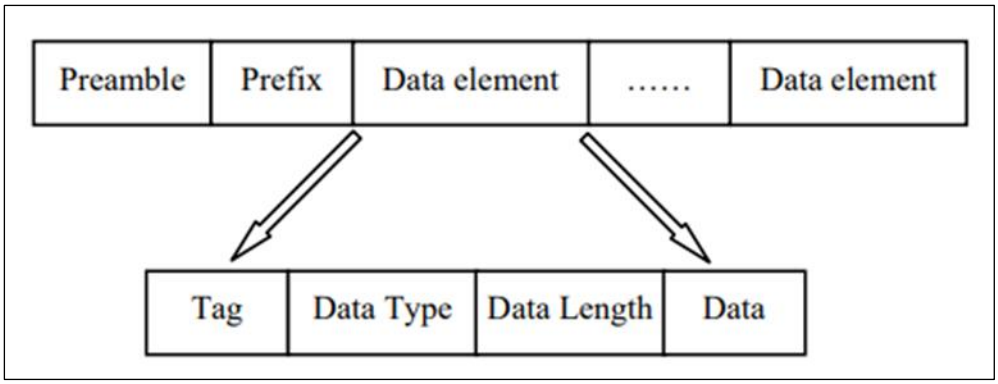

Рисунок 2.3.1 – Файловая структура формата DICOM

1) заголовок файла – состоит из вступительной части и префикса;

1.1) вступительная часть (англ. preamble) – 128 байт в длину (используется для доступа к изображениям и другим данным в файле DICOM, обеспечивая совместимость с обычно используемыми форматами файлов изображений);

1.2) префикс (англ. prefix) – 4 байта в длину, содержит строку "DICM" в верхнем регистре;

2) данные (англ. data element) – «обычно имеет несколько наборов элементов данных (каждый элемент данных, соответствующий атрибуту IOD, имеет четыре домена, а именно: тег, представление значения, длина значения и поле значения, в котором представление значения является необязательным)».

Данные пикселя (7FE0, 0010) содержат «необходимые данные для медицинского монитора». Другие элементы данных, тесно связанные с элементом пиксельных данных, представлены ниже.

(0028, 0008): количество кадров;

(0028, 0010): количество линий изображения;

(0028, 0011): количество столбцов изображения;

(0028, 0100): номер бита распределения;

(0028, 0101): номер бита хранения;

(0028, 0102): старший бит числа.

Код пиксельных данных определяется номером бита распределения, номером бита памяти и наивысшим номером бита, а номер бита распределения должен быть больше, чем номер бита памяти. Данные пикселя DICOM изображения часто являются 16-битными или 12-битными. Если выбран 16-битный формат, каждый пиксель имеет два байта; если 12-бит, распределение байтов на пиксель более сложное, и можно определить номер бита распределения, номер бита памяти и наивысшее число бит, различая значения элементов (0028,0100), (0028,0101) и (0028,0102).

Когда 16 распределенных бит, 12 бит памяти и старший бит составляют по 12 в каждом пикселе, то пиксель занимает всего 2 байта и использует младшие 12 бит. Когда самый старший бит – 12-й, то каждый пиксел использует старшие 12 бит из 2-х байт. Когда 12 распределенных бит, 12 бит хранения и самый старший бит – 11-й в каждом пикселе, пиксель занимает только 3 байта, содержание среднего байта разделено на 2 части, соответственно, принадлежа к первому и последнему байту, после этого каждый пиксель имеет по 12 бит. «Данные изображения могут быть сжаты или оставлены без сжатия. Когда данные передаются в сжатом формате, Value Representation принимает значение OB, иначе, значение OW. Для несжатых данных, последовательность обычно записывается сверху вниз, слева направо, данные кодируются и сохраняются как непрерывный битовый поток. Сжатые данные могут храниться сегментами и разделяться серией связанных элементов для поддержки процесса сжатия изображения неопределённой длины».

Dicom Standard Image – «это специальный формат, который имеет сложные типы и различные комбинированные форматы». Для его отображения и обработки необходим специально разработанный процессор обработки изображений, однако большинство современных приложений его не поддерживают. «Испытания показали, что врачи обычно выбирают

специфические части изображения как основу для заключений в диагнозе заболевания. Составляющие характеристики данных изображения DICOM имеют много общего с распространёнными форматами изображений, такими как BMP. Формат BMP, использующий сжатие без потерь, подходит для анализа изображений, обнаружения признаков заболевания и другой необходимой информации».

Таким образом, проведён анализ структуры данных формата DICOM, в результате которого определены особенности структуры файла, изучены атрибуты IOD, а также атрибуты при передаче и обработке изображений. Это позволило сделать вывод о возможности использования растрового формата изображений для обнаружения признаков заболевания и другой необходимой информации.

2.4 Исследование математических моделей представления визуальных данных

С математической точки зрения изображение может быть представлено в виде вещественной функции *img* от двух переменных *x* и *y*, *I (x, y)*.

В статье Штанчаева Х.Б. изображение представлено в матричном виде. «Функция $img = I\ (x, y)$, обычно определяется в прямоугольной форме, но для удобства можно представить квадратными областями, т.е. $x \in [0; W]$, а $y \in [0; H]$, где W - ширина изображения, H - высота изображения и $W = H$, т.е. изображение имеет размер $W \times H$. Значение, стоящее на пересечении x и y, называется пикселем. Не трудно заметить, что изображение схоже с прямоугольной системой координат. Началом координат *(0, 0)* применительно к изображению, нужно считать *(x, y) = (1, 1)*. Пара *(x, y) = (1, 2)* относится ко второму пикселю».

Также изображение можно представить в матричном виде:

$$img = I(x, y) = \begin{bmatrix} I(0,0) & I(0,1) & \cdots & I(0,W-1) \\ I(1,0) & I(1,1) & \cdots & I(1,W-1) \\ \vdots & \vdots & & \vdots \\ I(H-1,0) & I(H-1,1) & \cdots & I(H-1,W-1) \end{bmatrix}, \quad (2.1)$$

где *H*-высота изображения, *W*-ширина изображения.

Особенностью этого представления цифрового изображения состоит в том, что первый элемент в выражении (1) имеет координаты *(x, y) = (0,0)*, а не *(x, y) = (1, 1)* как указывалось выше [3].

В работе Грузмана И.С., Киричука В.С., Косых В.П., Перетягина Г.И., Спектора А.А. изображение представляется в виде дискретного представления непрерывного изображения. Редко, изображения, получаемые в информационных системах, имеют цифровую форму, поэтому их преобразование к цифровой форме – «обязательная операция, если это предполагается в поставленной задаче». Такое преобразование включает в себя две процедуры. «Первая процедура состоит в замене непрерывного кадра на дискретный (дискретизация), а вторая заменяет непрерывный набор значений яркости на набор квантованных значений (квантование)». «В цифровом представлении каждое из квантованных значений яркости связано с двоичным числом, чем и достигается возможность передачи изображения в электронно-вычислительную машину».

«Существуют различные способы замены непрерывного изображения на дискретное. Например, вы можете выбрать любую систему ортогональных функций и, вычислив коэффициенты представления изображения по этой системе (базису), заменить ими изображение. Большой выбор базисов дает возможность образования различных дискретных представлений непрерывного изображения. Однако наиболее используемой является периодическая дискретизация, в частности, дискретизация с прямоугольным растром. Данный способ дискретизации возможно рассматривать как вариант применения

ортогонального базиса, в котором в качестве элементов используются сдвинутые δ-функции» [7]. Далее рассмотрим основные особенности прямоугольной дискретизации.

Пусть $x_{\text{н}}(t_1, t_2)$ – непрерывное изображение, а $x(i_1, i_2)$ соответствующее ему дискретное, полученное из непрерывного изображения путем прямоугольной дискретизации. Таким образом, связь между ними определяется выражением:

$$x(i_1, i_2) = x_{н}(i_1\Delta t_1, i_2\Delta t_2), \qquad (2.2)$$

где $\Delta t_1, \Delta t_2$ – вертикальный и горизонтальный интервалы дискретизации, соответственно. Рисунок 2.4.1 иллюстрирует расположение отсчетов на плоскости (t_1, t_2) при прямоугольной дискретизации.

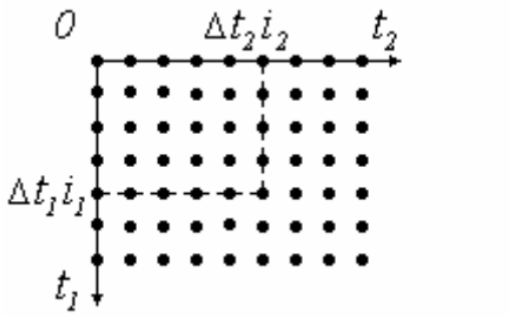

Рисунок 2.4.1 - Расположение отсчетов при прямоугольной дискретизации

Основной вопрос при замене непрерывного изображения на дискретное состоит в том, чтобы определить условия, при которых эта замена является полноценной, то есть без потери данных, содержащихся в непрерывном сигнале. «Потери отсутствуют, если, возможна обратная операция – восстановление непрерывного сигнала из дискретного. С точки зрения

математики, вопрос состоит в том, чтобы провести реконструкцию непрерывного сигнала в двумерных промежутках между узлами, в которых его значения известны или, иными словами, в осуществлении двумерной интерполяции». Ответить на данный вопрос можно, проведя анализ спектральных свойств непрерывного и дискретного изображений.

Двумерный непрерывный частотный спектр $X_н(\Omega_1, \Omega_2)$ непрерывного сигнала $x_н(t_1, t_2)$ определяется двумерным прямым преобразованием Фурье:

$$X_н(\Omega_1, \Omega_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_н(t_1, t_2) \exp(-j\Omega_1 t_1 - j\Omega_2 t_2) dt_1 dt_2, \qquad (2.3)$$

которому отвечает двумерное обратное непрерывное преобразование Фурье:

$$x_н(t_1, t_2) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} X_н(\Omega_1, \Omega_2) exp(j\Omega_1 t_1 + j\Omega_2 t_2) d\Omega_1 d\Omega_2. \qquad (2.4)$$

«Двумерный характер изображения по сравнению с обычными сигналами включает дополнительные возможности оптимизации цифрового представления для уменьшения объема получаемых цифровых данных. В связи с этим изучался вопрос о наилучшем размещении уровней квантования и об использовании различных растров.

На практике, в большинстве случаев, используют дискретизацию, основанную на применении прямоугольного растра, и равномерное квантование яркости. Это связано с небольшими преимуществами от использования оптимальных преобразований и простотой выполнения соответствующих операций» [17].

«При использовании прямоугольного растра в окончательном виде цифровое изображение обычно представляет собой матрицу, строки и столбцы которой соответствуют строкам и столбцам изображения».

«При цифровой обработке изображений непрерывный динамический диапазон значений яркости делится на ряд дискретных уровней. Эта процедура называется квантованием». Квантователь преобразует непрерывную переменную x в дискретную переменную $x_{кв}$, принимающую конечное множество значений $\{r_1, \dots, r_L\}$. Эти значения называются «уровнями квантования». В общем случае преобразование выражается ступенчатой функцией, изображенной на рисунке 2.4.2. Если яркость x отсчета изображения принадлежит интервалу $(d_j, d_{j+1}]$ (т.е., когда $d_j < x \leq d_j$), то исходный отсчет заменяется на уровень квантования r_j, где $d_j, j = \overline{1, L+1}$ - пороги квантования. При этом полагается, что динамический диапазон значений яркости ограничен и равен $[d_1, d_{L+1}]$.

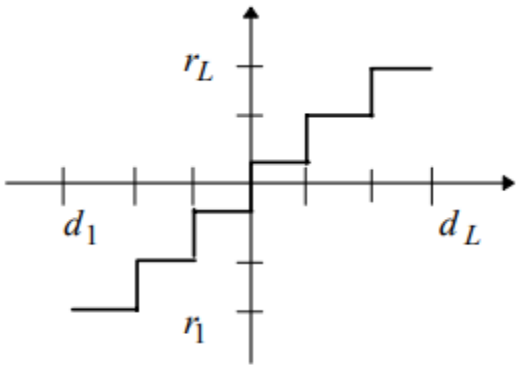

Рисунок 2.4.2 - Ступенчатая функция квантования

Красильников Н.Н. описывает растровую и векторную модели двумерного изображения следующим образом. «В основе растровой модели лежит растр – матрица пикселей, которые представляют интенсивность соответствующих участков изображения. Растровое изображение

характеризуется разрешением, которое определяется количеством пикселей на единицу длины. Чем больше пикселей приходится на единицу длины, тем выше разрешение». Для обработки изображения посредством ЭВМ, его представляют в цифровой форме. «В случае чёрно-белого изображения это означает, что интенсивность (яркость) каждого пикселя представляется числом от 0 до 255, то есть изображение представляется в виде двумерной матрицы, состоящей обычно из 8-разрядных двоичных чисел». Для представления цветного изображения в цифровой форме используют три матрицы, каждая из которых состоит из 8-разрядных двоичных чисел, реже из 16-разрядных. При этом элементы каждой из этих матриц представляют интенсивность красного, зелёного и синего компонентов цвета пикселя (модель RGB). В случае векторной модели, в отличие от растрового изображения, основой которых являются пиксели, основой векторных изображений являются контуры, представляемые кривыми, называемыми векторами. «Каждый контур векторного изображения представляет собой отдельный объект, который можно независимо от других редактировать. В соответствии с этим векторную графику иногда называют объектно-ориентированной графикой».

Для описания отрезка прямой в векторной графике используют уравнение

$$y = ax + b, \tag{2.5}$$

где x и y – декартовы координаты, а a и b – постоянные коэффициенты. Начало и конец отрезка задаются координатами x_1 и x_2.

Для описания кривых второго порядка – окружностей, эллипсов, парабол и гипербол – используют кривые второго порядка

$$x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5 = 0, \tag{2.6}$$

где a_1, a_2, a_3, a_5 – постоянные коэффициенты. Также для данной кривой необходимо задать координаты, определяющие начало и конец - x_1 и x_2.

В отличие от кривых второго порядка, кривые третьего порядка могут иметь точки перегиба, что даёт возможность их использовать для представления в векторной форме различных природных объектов. Уравнение для описания кривых третьего порядка имеет следующий вид

$$x^3 + a_1y^2 + a_2x^2y + a_3xy^2 + a_4x^2 + a_5y^2 + a_6xy + a_7x + a_8y + a_9 = 0, \quad (2.7)$$

где $a_1, a_2, a_3, a_5, a_6, a_7, a_8, a_9$ – постоянные коэффициенты. x_1 и x_2 – координаты, задающие начало и конец [6].

Широкое применение в векторной графике получили кривые Безье третьего порядка. «Особенностью этих кривых является то, что они позволяют удобно регулировать не только положение узлов на плоскости изображение, но также величины первой производной линии (угла наклона) и её второй производной (кривизну) в этих точках». Это «обеспечивает возможность соединять отдельные сегменты без изломов в точках соединения и тем самым аппроксимировать отрезками кривых Безье контуры любой сложности».

2.5 Математическая модель растрового изображения

Растровые изображения определяются как обычная прямоугольная сетка размерность M на N, состоящая из ячеек, называемых пикселями, каждый пиксель содержит значение цвета. Они характеризуются только двумя параметрами: количеством пикселей и содержанием информации (глубиной цвета) на пиксель. Есть другие атрибуты, которые применяются к растровым изображениям, но они являются производными от этих двух фундаментальных параметров.

Ниже на рисунке 2.5.1 представлена схема матрицы пикселей в растровом изображении.

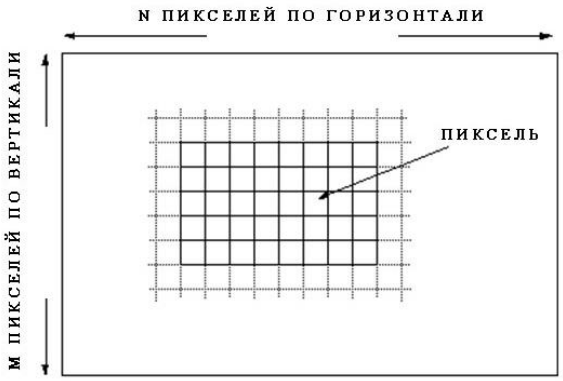

Рисунок 2.5.1 – Матрица пикселей растрового изображения

При цветовой палитре RGBA цвет – это набор компонент:

- R – красный цвет;
- G – зелёный цвет;
- B – синий цвет;
- A – альфа-канал, используется для кодирования уровня прозрачности.

При 32 битной глубине с четырьмя каналами цвета каждый канал кодируется 8 битами. Модель представления пикселя в двоичном виде изображена на рисунке 2.5.2.

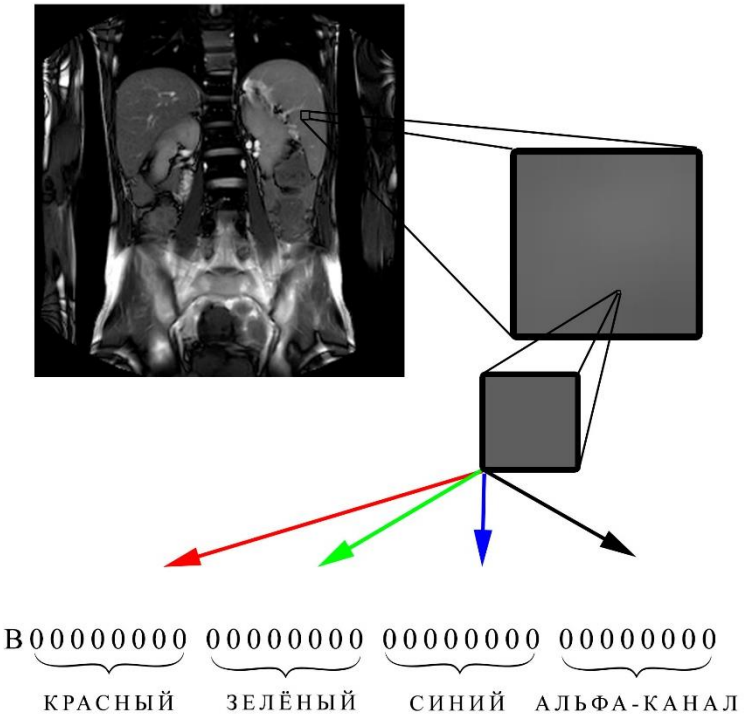

Рисунок 2.5.2 – Модель пикселя

Таким образом, один цвет может быть закодирован диапазоном значений от 0 до 255 (при 8-битном кодировании). Формат DICOM часто использует кодирование в 10, 12 или 16 битном формате. При кодировании с 16 битной глубиной цвета максимальное число значений будет равно 2^16 = 65536.

2.6 Исследование хранения и преобразования данных о плотностях тканей в формате DICOM

Стандарт DICOM позволяет выделять только те ткани, которые необходимы пользователю. В таблице 2.6.1 описана часть значений веществ по шкале Хаунсфилда (денситометрических показателей, англ. HU).

Шкала Хаунсфилда – «количественная шкала рентгеновской плотности» (радиоденсивности).

Таблица 2.6.1 – Значения вещества по шкале Хаунсфилда

Вещество	**HU**
Воздух	-1000
Жир	от -120 до -90
Мягкие ткани на контрасте КТ	от +100 до +300
Вода	0
Сталь	+20000
Золото, сталь, латунь	+30000 (верхний измерительный предел)

Для отображения определенного вещества в формате DICOM используются параметры Window Level (Window Center) и Window Width.

Ширина и центр окна - важные параметры, позволяющие выделить наиболее информативный диапазон данных по плотностям тканей. Для изменения параметров центра и ширины окна плотности тканей применяется перехват и наклон масштабирования для преобразования значений пикселей изображения в значимые для приложения значения. Применение значения масштабирования / перехвата масштабирования к значению пикселя преобразует исходные значения в оптическую плотность или другие известные единицы измерения (например, Шкала единиц Хаунсфилда – HU [2]). Когда преобразование не является линейным, применяется таблица поиска. После применения модального преобразования (перескакивает наклон / перехват или LUT) ширина окна / центр определяет, какие пиксели должны быть видимыми: все пиксели вне значений, указанных в окне, отображаются как черные или белые. Например, если центр окна равен 120, а ширина окна равна 40, то все пиксели со значением меньше 80 HU отображаются как черные, а все пиксели со значением больше 160 HU отображаются как белые. Это позволяет отображать только часть плотностей. Формулы передаточной функции представлены ниже.

$$minV = w_{centre} - \frac{w_{width}}{2}, \quad (2.8)$$

$$maxV = w_{centre} + \frac{w_{width}}{2}, \tag{2.9}$$

$$K = (\frac{HU - \left(w_{centre} - \frac{w_{width}}{2}\right)}{r} * Kmax), \tag{2.10}$$

где w_{centre} – центр окна, w_{width} – ширина окна, $minV$ – минимальное значение окна интенсивности пикселей, $maxV$ – максимальное значение окна интенсивности пикселей, K – текущая яркость пикселя (от 0 до 255), $Kmax$ – максимальный диапазон яркости (255 для формата оттенков серого), r – диапазон значений интенсивности пикселей.

Ширина окна позволяет задавать диапазон интенсивностей пикселей изображения относительно выбранного центра окна.

Величина HU определяется по формуле:

$$HU = \frac{\mu_x - \mu_{\text{воды}}}{\mu_{\text{воды}} - \mu_{\text{воздуха}}} \times 1000, \tag{2.11}$$

где $\mu_{\text{воды}}$ и $\mu_{\text{воздуха}}$ – линейные коэффициенты ослабления излучения для воды и воздуха при стандартных условиях, μ_x – линейный коэффициент ослабления излучения для материала x.

Линейные коэффициенты ослабления излучения – это «относительное изменение интенсивности направленного излучения на единице толщины среды».

«Для плотностей по шкале Хаунсфилда верно следующее утверждение: каждая плотность соответствует определённому типу ткани. Однако для МРТ это утверждение не верно, так как МР-томограф для каждой серии генерирует собственный набор плотностей. То есть для двух серий одна и та же плотность

может соответствовать разным тканям тела. В абсолютной передаточной функции аргументы соответствуют абсолютным значениям плотности».

Применение Центра и Ширины окна денситометрических показателей зависит от функции VOI LUT, которая может иметь следующие значения:

- LINEAR;
- LINEAR_EXACT;
- SIGMOID.

Если функция VOI LUT отсутствует или имеет значение LINEAR, Window Level и Window Width задают линейное преобразование из сохраненных значений пикселей (после применения любого из параметров: Modality LUT или Rescale Slope и Intercept) к значениям, которые будут отображаться.

VOI LUT позволяет преобразовывать значения пикселей модальности в значения пикселей, которые имеют значение для печати или отображения. Это преобразование применяется после любой Modality LUT. VOI LUT может содержаться в изображении или в состоянии презентации (которое ссылается на изображение) или в виде «Standalone VOI LUT» (которое ссылается на изображение). «Window Center» (0028,1050) и «Window Width» (0028,1051) используются для описания преобразования, когда оно является линейным, а VOI LUT Sequence (0028,3010) используется для описания нелинейных преобразований. В обоих случаях подразумевается, что преобразование может происходить только для данных в градациях серого (изображения со значениями «Фотометрическая интерпретация» «MONOCHROME1» и «MONOCHROME2»). Если для «Window Center» и «Window Width» имеется несколько значений, оба атрибута должны иметь одинаковое количество значений и должны рассматриваться как пары. Несколько значений указывают, что могут быть представлены несколько альтернативных представлений. Если в VOI LUT Sequence (0028,3010) присутствует несколько элементов, только один

из них может быть применен к изображению для отображения. Несколько элементов указывают, что могут быть представлены несколько альтернативных представлений.

Modality LUT позволяет преобразовывать зависящие от производителя значения пикселей в независимые от производителя значения пикселей (например, единицы Хаунсфилда для изображений CT). Modality LUT может содержаться в изображении, может содержаться в Presentation State (которое ссылается на изображение) или как Standalone Modality LUT (которое ссылается на изображение). Элементы «Rescale Slope» (0028,1053) и «Rescale Intercept» (0028,1052) используются для описания преобразования, когда оно является линейным, в то время как элемент «Modality LUT Sequence» (0028,3000) используется для описания нелинейные преобразований. В обоих случаях подразумевается, что преобразование может происходить только для данных в градациях серого, то есть для изображений со значениями «Photometric Interpretation» «MONOCHROME1» или «MONOCHROME2» [30].

Rescale Intercept (0028, 1052) и Rescale Slope (0028, 1053) являются тегами DICOM, которые задают линейное преобразование пикселей от сохраненного до исходного представления.

$$U = m \times SV + b \tag{2.12}$$

где U - в выходных единицах, m - наклон перемасштабирования (англ. rescale slope), SV – исходное сохранённое значение, b - перехват масштабирования (англ. rescale intercept). Другие теги дополнительно описывают формат сохраненного значения: Bits Allocated (0028, 0100), Bits Stored (0028, 0101), Pixel Representation (0028, 0103) и Sample per Pixel (0028, 0002).

Почему существует разница между данными на диске и данными в памяти? Такое различие возможно, если представление хранится на диске, а не

в памяти. Пример: изображения CT, значения пикселей которых измеряются в единицах Хаунсфилда, которые могут иметь отрицательные значения, обычно хранятся с целым числом без знака. Как следствие, для файлов CT характерен отрицательный перехват (англ. Intercept). Линейное масштабирование также применяется в тех случаях, когда пиксель может иметь большой диапазон значений, сохраняя при этом значения с минимальным количеством битов и избегая ошибок квантования. Это обычно применяется в ПЭТ-визуализации, где диапазон измеренных значений активности может превышать маленькие целые числа. Кроме того, максимальная активность ПЭТ может сильно варьироваться от среза к срезу; например, срез с опухолью может иметь очень высокие значения относительно среза здоровой ткани [14].

2.6.1 Функция вычисления денситометрических показателей VOI LUT – LINEAR

Если функция VOI LUT (0028,1056) отсутствует или имеет значение LINEAR, Window Center (0028,1050) и Window Width (0028,1051) задают линейное преобразование из сохраненных значений пикселей (после применения любого значения Modality LUT, Rescale Slope / Intercept) к значениям, которые будут отображаться.

Window Width (0028,1051) всегда должна быть больше или равна 1.

Когда Window Width (0028,1051) больше 1, эти атрибуты выбирают диапазон входных значений, которые должны отображаться во весь диапазон отображаемого вывода.

Когда ширина окна (0028,1051) равна 1, они определяют порог, ниже которого входные значения будут отображаться как минимальное выходное значение.

То, будет ли минимальное выходное значение отображаться в черном или белом цвете, может зависеть от значения фотометрической интерпретации

(0028 0004) (англ. Photometric Interpretation) или наличия модуля представления LUT (англ. Presentation LUT Module).

Эти атрибуты применяются в соответствии с псевдокодом, представленным в листинге 2.6.1.1.

Листинг 2.6.1.1. Псевдокод алгоритма применения атрибутов WW/WL

if (x <= c - 0.5 - (w-1) /2), then y = y_{min}

else if (x > c - 0.5 + (w-1) /2), then y = y_{max}

else y = ((x - (c - 0.5)) / (w-1) + 0.5) * (y_{max}- y_{min}) + y_{min}

где x - входное значение, y - выходное значение в диапазоне от ymin до ymax, c - Window Center (0028,1050) и w - Window Width (0028,1051).

Для целей этого определения предполагается вычисление с плавающей запятой без целочисленного усечения, хотя способ реализации может отличаться, если результат одинаков.

Функция псевдокода вычисляет непрерывное значение в выходном диапазоне без каких-либо разрывов на границах. Значение 0 для w категорически запрещено, а значение 1 для w не вызывает деления на ноль, поскольку для этого случая никогда не будет достигнут непрерывный сегмент функции.

Далее приведены примеры для диапазона вывода от 0 до 255.

При c = 2048, ш = 4096. Если (x <= 0), то y = 0. Иначе, если (x> 4095), то y = 255. Иначе y = ((x - 2047,5) / 4095 + 0,5) * (255-0) + 0.

При c = 2048, w = 1. Если (x <= 2047,5), то y = 0. Иначе, если (x> 2047,5), то y = 255. Иначе - не определено.

При c = 0, w = 100. Если (x <= -50), то y = 0. Иначе, если (x> 49), то y = 255. Иначе y = ((x + 0,5) / 99 + 0,5) * (255-0) + 0.

При c = 0, w = 1. Если (x <= -0,5), то y = 0. Иначе, если (x> -0.5), то y = 255. Иначе - не определено.

Параметр Window Center 2^{n-1} и ширина окна 2^n выбирают диапазон значений ввода от 0 до 2^{n-1}. Это представляет собой преобразование VOI LUT идентификатора в случае, когда LUT Modality не указано, и сохраненные данные пикселя представляют собой n-битные целые числа без знака.

Параметр Window Width, равный 1, обычно используется для представления «пороговой» операции, в которой эти целочисленные входные значения, меньшие, чем центр окна, представлены как минимальное отображаемое значение, а значения, которые больше или равны центру окна, представлены как максимальное отображаемое значение. Значение параметра 2 будет иметь тот же результат для интегральных входных значений.

Для применения параметров Window Center (0028,1050) и Window Width (0028,1051) можно выбирать диапазон ввода со знаком. Не подразумевается, что этот входной диапазон со знаком обрезается до нуля.

Выбранный входной диапазон может превышать фактический диапазон входных значений, таким образом, эффективно «сжимая» диапазон контрастности отображаемых данных в более узкую полосу доступного диапазона контрастности, и «выравнивая» внешний вид. Не существует ограничений на максимальное значение ширины окна или на минимальное, или максимальное значение уровня окна, которые могут превышать фактический или возможный диапазон входных значений.

Входные значения «под» окном отображаются как минимальное выходное значение, а входные значения «над» окном отображаются как максимальное выходное значение. Это обычное использование оконной операции в медицинской визуализации. Не предусмотрено альтернативного подхода, при котором все значения «вне» окна отображаются как минимальное выходное значение.

Выходные данные преобразования Window Center/Width или VOI LUT либо неявно масштабируются до полного диапазона устройства отображения,

если не определено последующее преобразование, либо неявно масштабируются до полного входного диапазона последующего шага преобразования (например, Presentation LUT), если представлено.

Допускаются дробные значения Window Center и Window Width (поскольку значение этих атрибутов является десятичной строкой), и, хотя они встречаются не часто, приложения должны быть готовы принять их.

Параметры ymin и ymax, а также цветовая палитра, устанавливаются в зависимости от значений параметров Samples per Pixel и Photometric Interpretation.

2.6.2 Функция вычисления денситометрических показателей VOI LUT – LINEAR_EXACT

Если значением функции VOI LUT (0028,1056) является LINEAR_EXACT, функция, которая будет использоваться для преобразования вывода (концептуальных) значений Modality LUT на вход (концептуального) Presentation LUT, задается псевдокодом, представленным в листинге 2.6.2.1.

Листинг 2.6.2.1. Псевдокод алгоритма преобразования значений Modality LUT

if ($x <= c - w/2$), then $y = y_{min}$

else if ($x > c + w/2$), then $y = y_{max}$

else $y = ((x - c) / w + 0.5) * (y_{max} - y_{min}) + y_{min}$

где x - входное значение, y - выходное значение в диапазоне от ymin до ymax, c - центр окна (0028,1050) и w - ширина окна (0028,1051).

Например, учитывая сохраненные значения беззнаковых пикселей от 0 до 65535, Rescale Intercept 0 и Rescale Slope 1.0/65535, Window Width 1.0 и Window Center 0.5 будут указывать весь диапазон значений (преобразование идентификатора для этих значений масштабирования).

Window Width (0028,1051) всегда должен быть больше нуля, чтобы предотвратить деление на ноль.

2.6.3 Функция вычисления денситометрических показателей VOI LUT – SIGMOID

Если значением функции VOI LUT (0028,1056) является SIGMOID, то функция, которая будет использоваться для преобразования вывода (концептуальных) значений Modality LUT во вход (концептуального) Presentation LUT, определяется как:

$$y = \frac{y_{max} - y_{min}}{1 + exp(-4\frac{x - c}{w})} + y_{min} \quad (2.13)$$

где x - является входным значением LUT (то есть, выходом (концептуального) Modality LUT), c – параметр Window Center, определяемый интерактивно пользователем или с использованием значений, указанных в (0028,1050), w- параметр Window Width, определяемый в интерактивном режиме пользователем или с использованием значений, указанных в (0028,1051), y – итоговое выходное значение, y_{min} - минимальное выходное значение, y_{max} - максимальное выходное значение.

Window Width (0028,1051) всегда должен быть больше нуля, чтобы предотвратить деление на ноль.

Атрибуты Window Center (0028,1050), Window Width (0028,1051) и Функция VOI LUT (0028,1056) должны использоваться только для изображений со значениями атрибута Photometric Interpretation (0028,0004) MONOCHROME1 и MONOCHROME2. Они не используются для других изображений.

Если присутствует несколько значений, оба атрибута должны иметь одинаковое количество значений и должны рассматриваться как пары. Несколько значений указывают, что могут быть представлены несколько альтернативных представлений.

Если какая-либо таблица VOI LUT включена в изображение, ширина окна и центр окна или таблицы VOI LUT, но не обе, могут быть применены к изображению для отображения. Включение обоих указывает на то, что могут быть представлены несколько альтернативных представлений.

«Если несколько элементов присутствуют в VOI LUT Sequence (0028,3010), только один из них может быть применен к изображению для отображения». Несколько элементов указывают, что могут быть представлены несколько альтернативных видов.

Если модуль VOI LUT определен в IOD, и если нет ни последовательности VOI LUT, ни ширины и центра окна, то этап VOI LUT конвейера в градациях серого определяется как преобразование идентичности.

Это требование определено таким образом, чтобы IOD, которые определяют конкретное выходное пространство для конвейера в градациях серого, например, P-значения, не находились в неопределенном состоянии, когда отсутствуют последовательность VOI LUT или ширина окна и центр окна.

Для последовательности и оконного центра VOI LUT, ожидается, что реализации, которые визуализируют изображения, будут реализовывать и применять эти преобразования, когда они присутствуют в изображении, если только они не отменены пользователем, состоянием представления или протоколом зависания, и позволяет пользователю выбрать, какое преобразование применять при наличии нескольких преобразований [29].

2.7 Исследование возможности параллельных вычислений модели регулировки плотностей тканей при использовании растрового изображения

Рассмотренные в пункте 2.3 формулы предполагают вычисление значений пикселей растрового изображения, модель которого рассмотрена в пункте 2.1 и 2.2, согласно которым растровое изображение представляет из себя матрицу, размерностью $x \times y$. Такое представление данных является примером

классической задачи параллельного программирования (пример: произведение двух квадратных матриц) [18].

В данном случае ситуация аналогична тем, что порядок вычисления значений пикселей не важен. В рассмотренных в пункте 2.3 формулах 2.10, 2.13 порядок вычисления значений не является однозначно определенным.

Возьмём для примера формулу 2.13, учитывая индексы для вычисления значений итогового изображения в виде матрицы.

$$y_{ij} = \frac{y_{max} - y_{min}}{1 + exp(-4\frac{x_{ij} - c}{w})} + y_{min} \qquad (2.14)$$

Явно видно, что для вычисления, к примеру, значения пикселя под индексом $(50, 50)$, изображения, размерностью 100×100, нет необходимости в последовательном вычислении всех остальных индексов, так как в формуле нет ограничения на порядок вычисления.

Таким образом, данные модели и формулы для вычисления значения пикселей и алгоритмы, построенные на базе данных формул, содержат внутренний параллелизм, который возможно реализовать существующими средствами для параллельных вычислений.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МОДЕЛИ МАТЕМАТИЧЕСКОЙ ОБРАБОТКИ СНИМКОВ ИССЛЕДОВАНИЙ

3.1 Обоснование выбора средств реализации математической обработки медицинских снимков

3.1.1 Сравнительный анализ языков программирования

Проведём сравнительный анализ языков программирования, используемых в Android разработке:

- Kotlin;
- Java;
- C#;
- C++;
- C.

Kotlin – кроссплатформенный, статически типизированный язык программирования общего назначения [31]. Kotlin спроектирован так, чтобы полностью быть совместимым с Java. Версия его стандартной библиотеки JVM зависит от библиотеки классов Java, однако вывод типов позволяет сделать его синтаксис более лаконичным [34]. Kotlin в основном нацелен на JVM, но также компилируется в Java Script или нативный код (через LLVM) [19].

Java – «язык программирования общего назначения, основанный на классах. Объектно-ориентированный и спроектированный таким образом, чтобы иметь как можно меньше зависимостей реализации» [13]. «Скомпилированный код Java может работать на всех платформах, поддерживающих Java, без необходимости перекомпиляции. Приложения Java обычно компилируются в байт-код, который может работать на любой виртуальной машине Java (JVM) независимо от базовой компьютерной архитектуры» [4]. «Синтаксис Java похож на C и C ++, но у него меньше

низкоуровневых средств, чем у любого из них. По данным GitHub, с 2019 года Java был одним из самых популярных языков программирования, особенно для веб-приложений клиент-сервер, с 9 миллионами разработчиков» [24].

C# – это универсальный многопарадигмальный язык программирования, включающий строгую типизацию, лексическую область видимости, императив, декларативный, функциональный, универсальный объект -ориентированные (на основе классов) и компонентно-ориентированные дисциплины программирования [11]. Он был разработан Microsoft примерно в 2000 году в рамках инициативы .NET, а затем утвержден в качестве международного стандарта Ecma (ECMA-334) в 2002 году и ISO (ISO / IEC 23270) в 2003 году [8].

C++ – язык программирования общего назначения, созданный Бьярном Страуструпом как расширение языка программирования C, или «C with Classes». Язык значительно расширился со временем, и современный C ++ теперь обладает объектно-ориентированными, универсальными и функциональными функциями в дополнение к средствам низкоуровневой манипуляции с памятью. Он почти всегда реализуется как скомпилированный язык, и многие поставщики предоставляют компиляторы C++, включая Free Software Foundation, LLVM, Microsoft, Intel, Oracle и IBM, поэтому он доступен на многих платформах. [9]

C – язык процедурного компьютерного программирования общего назначения, поддерживающий структурированное программирование, область действия лексических переменных и рекурсию, в то время как система статического типа предотвращает непреднамеренные операции. По своей конструкции C предоставляет конструкции, которые эффективно отображаются на типичные машинные инструкции, и нашел длительное применение в приложениях, ранее закодированных на языке ассемблера. Такие приложения включают операционные системы и различное прикладное программное

обеспечение для компьютеров, от суперкомпьютеров до встроенных систем [10].

Введём критерии оценки:

1. распространенность (5 баллов) – использование языка достаточно большим числом разработчиков, а также возможность найти ответ на возникшую проблему;
2. наличие опыта (10 баллов) – наличие знаний и навыков работы на данном языке;
3. производительность (5 баллов) – скорость выполнения скомпилированной программы;
4. наличие объектно-ориентированной парадигмы программирования (1 балл).

Сравнительный анализ языков программирования представлен в таблице 3.1.1.

Таблица 3.1.1 – Сравнительный анализ языков программирования

Альтернатива **Критерии оценки**	**Kotlin**	**Java**	**C#**	**C**	**C++**
Распространенность	4	5	3	2	3
Наличие опыта	6	9	6	5	6
Производительность	2	2	2	5	5
Наличие объектно-ориентированной парадигмы программирования	1	1	1	0	1
Итого	10	17	12	8	15

В результате сравнительного анализа, как основной, выбран объектно-ориентированный язык программирования Java. Однако, обработка изображения является ресурсоёмким процессом, для которого важна производительность. Поэтому, для сравнения, часть алгоритмов будет реализовываться на языках программирования C/C++ с использованием Android NDK.

3.1.2 Сравнительный анализ сред разработки

Проведём сравнительный анализ интегрированных сред разработки, используемых для разработки приложений на платформе Android:

- MS Visual Studio;
- Eclipse;
- IntelliJ IDEA;
- Android Studio.

MS Visual Studio - это интегрированная среда разработки (IDE) от Microsoft. Используется для разработки компьютерных программ, а также веб-сайтов, веб-приложений, веб-сервисов и мобильных приложений [1].

Eclipse - это интегрированная среда разработки (IDE), используемая в компьютерном программировании. Содержит базовое рабочее пространство и расширяемую систему плагинов для настройки среды. Eclipse написан в основном на Java, и его основное использование предназначено для разработки приложений Java, но его также можно использовать для разработки приложений на других языках программирования с помощью плагинов [20].

IntelliJ IDEA - это интегрированная среда разработки (IDE), написанная на Java для разработки компьютерного программного обеспечения. Разработана JetBrains (ранее известный как IntelliJ) и доступна в виде лицензии сообщества Apache 2 и в проприетарной коммерческой версии. Обе версии могут быть использованы для коммерческой разработки [12].

Android Studio - это «официальная интегрированная среда разработки (IDE) для операционной системы Google Android, созданная на основе программного обеспечения JetBrains IntelliJ IDEA и разработанная специально для разработки под Android. Возможна работа в операционных системах Windows, macOS и Linux». «Заменила собой Eclipse Android Development Tools (ADT) в качестве основной IDE для разработки собственных приложений Android» [25].

Введём критерии оценки:

1. распространенность (5 баллов) – использование большим числом разработчиков;
2. наличие опыта (10 баллов) – наличие знаний и навыков в представленной среде разработки;
3. возможность бесплатного использования (10 баллов) – лицензия на бесплатное использование программы;
4. поддержка (5 баллов) – наличие поддержки разработчика, регулярные обновления, актуальность;
5. кроссплатформенность (10 балл) – наличие возможности использования на разных платформах. Например, ОС на базе ядра Linux, ОС Windows и т.д.

Сравнительный анализ интегрированных сред разработки представлен в таблице 3.1.2.

Таблица 3.1.2 – Сравнительный анализ сред разработки

Альтернатива Критерии оценки	**MS Visual Studio**	**Eclipse**	**IntelliJ IDEA**	**Android Studio**
Распространенность	2	6	8	10
Наличие опыта	4	5	8	8
Бесплатность	5	10	5	10
Поддержка	10	8	10	10
Кроссплатформенность	0	10	10	10
Итого	21	39	41	48

В результате данного анализа выбрана интегрированная среда разработки Android Studio.

3.1.3 Сравнительный анализ фреймворков и библиотек

Проведём сравнительный анализ библиотек для использования в разрабатываемом техническом решении.

Введём критерии оценки:

1. документация (5 баллов) – наличие полной документации;

2. наличие опыта (10 баллов) – наличие знаний и навыков работы с библиотекой;
3. производительность (5 баллов) - система должна иметь минимальную по времени реакцию для ответа на действия пользователя;
4. поддержка (5 баллов) – наличие поддержки библиотеки разработчиком, регулярные обновления, актуальность.

Рассмотрим библиотеки для работы с изображениями:

- Picasso;
- Glide;
- Universal Image Loader;
- Fresco.

Picasso – «библиотека, предназначенная для асинхронной загрузки изображений из сети, ресурсов или файловой системы, их кэширования и отображения» [27].

Glide – «библиотека, как и Picasso, предназначенная для асинхронной загрузки изображений. Также позволяет декодировать изображения напрямую из массива байт» [23].

Universal Image Loader – «библиотека для асинхронной погрузки изображений, их кэширования и отображения. Позволяет загружать изображения как из сети, так и из локальных хранилищ» [22].

Fresco – «многофункциональная библиотека для асинхронной загрузки и отображения изображений с тремя уровнями кеширования (2 в памяти, 1 в internal storage). Разработана Facebook. Поддерживает форматы: JPEG, PNG, GIF и WebP» [21].

Сравнительный анализ библиотек для работы с изображениями представлен в таблице 3.1.3.

Таблица 3.1.3 – Сравнительный анализ библиотек для работы с изображениями

Альтернатива Критерии оценки	**Picasso**	**Glide**	**Universal Image Loader**	**Fresco**
Документация	3	5	5	5
Наличие опыта	8	10	9	3
Производительность	4	5	2	5
Поддержка	5	5	0	4
Итого	20	24	17	16

В результате данного анализа выбрана библиотека работы с изображениями Glide.

Рассмотрим библиотеки для работы с сетью:

- OkHttp;
- Volley;
- Retrofit 2.

Сравнительный анализ библиотек для работы с HTTP протоколом представлен в таблице 3.1.4.

Таблица 3.1.4 – Сравнительный анализ библиотек для работы с HTTP протоколом

Альтернатива Критерии оценки	**OkHttp**	**Volley**	**Retrofit 2**
Документация	4	5	5
Наличие опыта	7	7	9
Производительность	5	2	5
Поддержка	5	5	5
Итого	21	19	24

В результате сравнительного анализа выбрана библиотека для работы с HTTP протоколом Retrofit 2.

Рассмотрим библиотеки для связывания данных:

- Butter Knife;
- Data Binding.

Добавим дополнительный критерий – гибкость.

Гибкость (5 баллов) – возможность свободно связывать любой класс Java с любым ресурсом мобильного приложения.

Сравнительный анализ библиотек для связывания данных представлен в таблице 3.1.5.

Таблица 3.1.5 – Сравнительный анализ библиотек для связывания данных

Критерии оценки	Butter Knife	Data Binding
Документация	4	5
Наличие опыта	10	9
Производительность	5	5
Поддержка	5	5
Гибкость	5	3
Итого	29	27

В результате сравнительного анализа выбрана библиотека Butter Knife. В некоторых случаях, для ускорения разработки, возможно совместное использование с Data Binding.

В результате сравнительного анализа для разработки мобильного приложения будет использовано следующее: язык программирования Java, интегрированная среда разработки Android Studio, библиотека Glide для изображений, библиотека для работы с протоколом HTTP – Retrofit 2, библиотека для привязки данных - Butter Knife (использовать с привязкой данных).

3.2 Разработка архитектуры мобильного клиента

Для разработки архитектуры мобильного клиента будет использоваться паттерн Model–View–ViewModel (MVVM). «MVVM используется для разделения модели и её представления, что необходимо для изменения их отдельно друг от друга» [33].

Для реализации мобильного клиента разработаем диаграмму пакетов и классов. Диаграмма пакетов мобильного приложения представлена на рисунке 3.2.1. Диаграмма классов представлена в Приложении А.

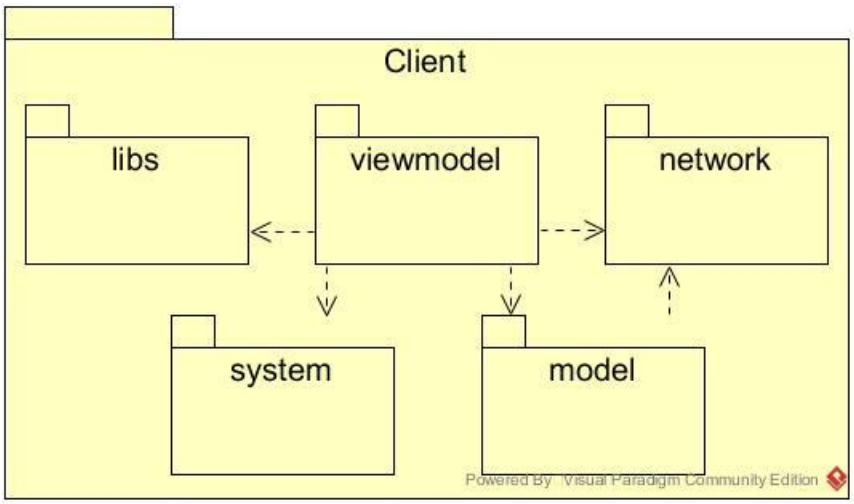

Рисунок 3.2.1 – Диаграмма пакетов мобильного приложения

Пакет Client содержит в себе классы, описание основных представлено в таблице 3.2.1.

Таблица 3.2.1 – Классы пакета Client

Объект	**Описание**
Класс StartActivity	Запускается при запуске приложения, это точка входа. В свою очередь, это заставка во время загрузки программы. Инициализирует компоненты приложения.
Класс SearchActivity	Реализует окно поиска исследований пациентов.
Класс PatientActivity	Реализует окно со списком найденных пациентов с выбором.
Класс SeriesActivity	Реализует окно со списком найденных исследовательских серий с возможностью выбора.
Класс GridViewActivity	Реализует окно с предварительным просмотром серии исследований в виде миниатюр.
Класс SlideImageActivity	Реализует полноэкранный просмотр изображений серии, а также настройку параметров центра и ширины окна плотностей тканей.
Интерфейс MedicalAPI	Интерфейс содержит API для доступа к промежуточному серверу обработки данных.

Интерфейс, определенный на стороне клиента и обеспечивающий асинхронный запрос к серверу, основан на библиотеке Retrofit 2. Для построения запросов создается интерфейс MedicalAPI, который содержит API (интерфейс) для работы с сервером. Библиотека Retrofit 2 обрабатывает ответ сервера и в случае ошибки выдает исключение. Асинхронные запросы на серию снимков исследований реализованы с использованием библиотеки Glide.

В результате была разработана архитектура, которая реализует мобильную клиентскую часть разработанной системы для мобильного доступа к данным пациента на основе шаблона MVVM.

3.3 Реализация математической модели обработки снимков базовыми инструментами Android SDK

Обработка снимков реализована с помощью языка программирования Java, версии 8.

Ниже на рисунке 3.3.1 приведён участок кода, отвечающий за выбор функции расчёта изображения.

```
switch (Integer.valueOf(dicomImage.getSamplesPer())) {
    case 1:
        switch (dicomImage.getPhotoMetri()) {
            case MONOCHROME2:
                calculateMONOCHROME2(oBitmap, bitmap, mWL, mWW,
String.valueOf(dicomImage.getVoilutFunc()));
                break;
        }
        break;
    case 3:
        calculateRGB(oBitmap, bitmap, mWL, mWW);
        break;
    default:
        calculateMONOCHROME1(oBitmap, bitmap, mWL, mWW,
String.valueOf(dicomImage.getVoilutFunc()));
        break;
}
```

Рисунок 3.3.1 – Функция выбора методов расчёта значений цвета исходя из значений плотностей тканей и других атрибутов

Для правильного выбора расчётной функции необходимо знать значения атрибутов Photometric Interpretation (0028,0004) и Samples per Pixel (0028,0002). Ниже приведена таблица 3.3.1, со значениями Photometric Interpretation и Samples per Pixel [28].

Таблица 3.3.1 – Значения Samples Per Pixel

Photometric Interpretation	**Samples Per Pixel Value**
MONOCHROME2	1
RGB	3
YBR_FULL	3
YBR_FULL_422	3
YBR_RCT	3
YBR_ICT	3
YBR_PARTIAL_420	3
PALETTE COLOR	1

mWL, mWW – задаются исходя из значений, записанных в DICOM файл во время проведении исследования. Последующие изменения значений данных параметров являются произвольными и управляются пользователем.

oBitmap – исходное изображение, не изменяется при изменении mWL/mWW, перезаписывается только при открытии нового изображения.

Bitmap – измененное (обработанное) изображение, полученное в результате работы одной из передаточных функций.

Для обеспечения правильного расчёта значений цвета пикселя после изменения значений Window Center и Window Width, в соответствии с документацией, разработаны функции для каждого из параметров атрибута VOI LUT.

В качестве параметров расчётные функции принимают:

- pixelData – текущий цвет пикселя;
- mWL – значение Window Center;
- mWW – значение Window Width;

- ymin – минимально возможное значение цвета пикселя;
- ymax – максимально возможное значение цвета пикселя;
- ymin и ymax устанавливаются в зависимости от значения атрибута фотометрической интерпретации (англ. Photometric Interpretation) (0028,0004).

Значения атрибута фотометрической интерпретации для данной программной реализации:

- MONOCHROME1: ymin = 0, ymax = 255;
- MONOCHROME2: ymin = 255, ymax = 0;
- RGB: ymin = 0, ymax = 255.

Такой диапазон получен по причине использования цветового пространства RGB с ограничением 8 бит для каждой цветовой компоненты.

3.3.1 Реализация линейной функции LINEAR

Листинг линейной функции вычисления цвета пикселя LINEAR в зависимости от параметров Window Center (0028,1050) и Window Width (0028,1051) представлен на рисунке 3.3.2.

```
private static float calculateLinearColor(int pixelData, int mWL, int mWW,
int ymin, int ymax) {
    float oColor;
    if (pixelData <= mWL - 0.5 - (mWW - 1) / 2)
        oColor = ymin;
    else if (pixelData > mWL - 0.5 + (mWW - 1) / 2)
        oColor = ymax;
    else {
        oColor = ((pixelData - (mWL - 0.5f)) / (mWW - 1) + 0.5f) *
                (ymax - ymin) + ymin;
    }
    return oColor;
}
```

Рисунок 3.3.2 – Программная реализация метода вычисления линейной функции LINEAR

В результате выполнена реализация вычисления линейной функции LINEAR.

3.3.2 Реализация линейной функции LINEAR_EXACT

Листинг линейной функции вычисления цвета пикселя LINEAR_EXACT в зависимости от параметров Window Center (0028,1050) и Window Width (0028,1051) представлен на рисунке 3.3.3.

```
private static float calculateLinearExactColor(int pixelData, int mWL, int
mWW, int ymin, int ymax) {
    float oColor;
    if (pixelData <= mWL - mWW / 2)
        oColor = ymin;
    else if (pixelData > mWL + mWW / 2)
        oColor = ymax;
    else {
        oColor = ((pixelData - mWL) / mWW + 0.5f) *
                (ymax - ymin) + ymin;
    }
    return oColor;
}
```

Рисунок 3.3.3 – Программная реализация метода вычисления линейной функции LINEAR_EXACT

В результате выполнена реализация вычисления линейной функции LINEAR_EXACT.

3.3.3 Реализация нелинейной функции SIGMOID

Ниже на рисунке 3.3.4 приведён листинг нелинейной функции вычисления цвета пикселя SIGMOID в зависимости от параметров Window Center (0028,1050) и Window Width (0028,1051).

```
private static float calculateSigmoidColor(int pixelData, int mWL, int mWW,
int ymin, int ymax) {
    return ((ymax - ymin) / (1 + (float) Math.exp(-4 * ((pixelData - mWL)
/ mWW)))) + ymin;
}
```

Рисунок 3.3.4 – Программная реализация метода вычисления линейной функции SIGMOID

В результате выполнена реализация вычисления нелинейной функции SIGMOID.

Таким образом, был реализован программный алгоритм, основанный на математической модели вычисления цвета пикселей, исходя из значений плотностей тканей, а также связанных атрибутов на языке программирования Java без применения дополнительных средств оптимизации вычислений.

3.4 Реализация математической модели обработки снимков с применением оптимизаций и Android NDK

3.4.1 Реализация методов расчёта отображения плотностей тканей

В качестве основного языка программирования был выбран Java. Наиболее ресурсоёмкие части программы реализованы на C++.

Для применения значений WW WL (окна плотностей) к растровому снимку используется метод applyDicomWWWLToBitmap. В качестве параметров метод принимает: WW, WL, samplesPerPixel, photometricInterpretation, rescaleSlope, rescaleIntercept, rescaleType, voilut, bitsStored, modality.

Далее метод валидирует полученные значения и передаёт их, в зависимости от значения параметров samplesPerPixel и photometricInterpretation, методам jniCalculateDicomMONOCHROME и jniCalculateDicomRGB, которые являются обёрточными методами для C++ функций. Методы C++ называются JniBitmapHolder_jniCalculateDicomMONOCHROME и

JniBitmapHolder_jniCalculateDicomRGB соответственно. Для удобства далее используются названия обёрточных методов, подразумевая функции C++.

Метод jniCalculateDicomMONOCHROME используется для расчёта DICOM параметров для двух типов монохромных изображений. На вход данный метод принимает следующие параметры: handler – буфер данных изображения (используется для передачи изображения из Java кода в C++), WW, WL, rescaleSlope, rescaleIntercept, rescaleType, voilut, bitsStored, type – тип монохромного изображения (1 или 2), modality.

Метод jniCalculateDicomRGB, как понятно из названия, используется для расчёта DICOM параметров для цветных RGB изображений. На вход данный метод принимает следующие параметры: handler, WW, WL.

Далее каждый метод будет рассмотрен подробнее.

3.4.2 Реализация метода applyDicomWWWLToBitmap

Для применения значений WW WL к растровому снимку используется метод applyDicomWWWLToBitmap. В качестве параметров метод принимает: WW, WL, samplesPerPixel, photometricInterpretation, rescaleSlope, rescaleIntercept, rescaleType, voilut, bitsStored, modality.

Далее метод валидирует полученные значения и передаёт их, в зависимости от значения параметров samplesPerPixel и photometricInterpretation, методам jniCalculateDicomMONOCHROME и jniCalculateDicomRGB, которые являются обёрточными методами для C++ функций.

Валидация значений происходит посредством последовательной проверки условий валидности для каждого входящего параметра изображена на рисунке 3.4.1.

```
if(_handler==null)
    return;

if (WW < 1) WW = 1;

if (samplesPerPixel==null ) samplesPerPixel = "2";
if (photometricInterpretation==null ) photometricInterpretation = MONOCHROME2;
if (rescaleType==null ) rescaleType = "NULL";
if (voilut==null ) voilut = "NULL";
if (bitsStored==null ) bitsStored = "8";
if (modality==null ) modality = "CT";

float rSlo = 1;
if (rescaleSlope!=null) {
    rSlo = Float.parseFloat(rescaleSlope);
}
float rInt = 0;
if (rescaleIntercept!=null){
    rInt = Float.parseFloat(rescaleIntercept);
}
int bStored = Integer.parseInt(bitsStored);
```

Рисунок 3.4.1 – Валидация входных параметров метода applyDicomWWWLToBitmap

Затем, согласно условиям, параметры передаются далее в методы обработки снимков. Алгоритм представлен на рисунке 3.4.2.

```
switch (Integer.parseInt(samplesPerPixel)) {
    case 1:
        switch (photometricInterpretation) {
            case MONOCHROME2:
                jniCalculateDicomMONOCHROME(_handler, WW, WL, rSlo, rInt, rescaleType, voilut, bStored, type: 2, modality);
                break;
        }
        break;
    case 3:
        jniCalculateDicomRGB(_handler, WW, WL);
        break;
    default:
        jniCalculateDicomMONOCHROME(_handler, WW, WL, rSlo, rInt, rescaleType, voilut, bStored, type: 1, modality);
        break;
}
```

Рисунок 3.4.2 – Передача параметров в методы обработки снимков

Таким образом, данный метод упрощает читаемость кода, распределяет входные DICOM параметры по методам обработки снимков в зависимости от входных DICOM параметров.

3.4.3 Реализация метода jniCalculateDicomRGB

Метод применяется, когда значение параметра Photometric Interpretation (0028,0004) – RGB. Пиксельные данные представляют собой цветное изображение, описанное красной, зеленой и синей плоскостями изображения. Минимальное значение выборки для каждой цветовой плоскости представляет минимальную интенсивность цвета. Это значение может использоваться только в том случае, если значение Samples per Pixel «Выборки на пиксель» (0028 0002) имеет значение 3. Planar Configuration Планарная конфигурация (0028 0006) может быть 0 или 1. Может использоваться для данных пикселей в собственном (несжатом) или инкапсулированном (сжатом) формате.

Программный код вычисления цвета снимка представлен на рисунке 3.4.3.

```
    ARGB pixelDataRGB;
    uint32_t *newBitmapPixels = new uint32_t[width * height];
    jniBitmap->_storedBitmapPixels = newBitmapPixels;
#pragma omp parallel for
    for (int y = height - 1; y >= 0; --y)
#pragma omp parallel for
            for (int x = width - 1; x >= 0; --x) {
                uint32_t pixelData = previousData[width * y + x];
                convertIntToAndroidArgb(pixelData, &pixelDataRGB);

                uint8_t color[3] = {pixelDataRGB.red, pixelDataRGB.green, pixelDataRGB.blue};
                jfloat oColor[3] = {};

                for (int i = 0; i < 3; i++) {
                    oColor[i] = calculateLinearColor(color[i], wl, ww, ymin, ymax);
                    if (oColor[i] > ymax) oColor[i] = ymax;
                    if (oColor[i] < ymin) oColor[i] = ymin;
                    oColor[i] = round(oColor[i]);
                }

                newBitmapPixels[width * y + x] = convertToAndroidArgb(pixelDataRGB.alpha,
                                                                      static_cast<uint8_t>(oColor[0]),
                                                                      static_cast<uint8_t>(oColor[1]),
                                                                      static_cast<uint8_t>(oColor[2]));
            }
    delete[] previousData;
```

Рисунок 3.4.3 – Программный код вычисления цвета снимка

Как видно из рисунка 3.4.4, изображение представляет собой массив беззнаковых целых чисел, величиной в 32 разряда. Обработка происходит с помощью двух вложенных циклов с предусловием в обратном порядке. Выполнение циклов распараллелено с помощью аннотации OpenMP «#pragma omp parallel for». Функция convertIntToAndroidArgb, представленная на рисунке 3.4.5, производит конвертацию числа в структуру для упрощения работы с отдельными цветовыми компонентами.

```
typedef struct {
    uint8_t alpha, red, green, blue;
} ARGB;
```

Рисунок 3.4.4 – Структура для хранения значений ARGB

```
void convertIntToArgb(uint32_t pixel, ARGB *argb) {
    argb->red = ((pixel >> 24) & 0xff);
    argb->green = ((pixel >> 16) & 0xff);
    argb->blue = ((pixel >> 8) & 0xff);
    argb->alpha = (pixel & 0xff);
}
```

Рисунок 3.4.5 – Функция преобразования числа в структуру ARGB

Далее цвета передаются в массив из трёх 8-разрядных беззнаковых целых чисел.

Так как типы данных для SDK и NDK имеют свои различия, то для обработки используются одни, а для передачи обратно в SDK другие. Таким образом после задания массива создаётся ещё один пустой массив, повторяющий смысл первого массива, за исключением формата данных – jfloat, который является аналогом типа данных Float из SDK.

Далее в цикле происходит расчёт цвета каждой цветовой компоненты при помощи функции calculateLinearColor. На вход данная функция принимает следующие параметры: finalColor – исходный цвет пикселя, WL, WW, ymin и ymax – минимальное и максимальное значение интенсивности цвета пикселя. Функция calculateLinearColor представлена на рисунке 3.4.6.

```
jfloat calculateLinearColor(jfloat finalColor, jfloat mWL, jfloat mWW, uint8_t ymin, uint8_t ymax) {
    float oColor;
    if (finalColor <= mWL - 0.5 - (mWW - 1) / 2)
        oColor = ymin;
    else if (finalColor > mWL - 0.5 + (mWW - 1) / 2)
        oColor = ymax;
    else {
        oColor = static_cast<jfloat>(((finalColor - (mWL - 0.5)) / (mWW - 1) + 0.5) *
                                     (ymax - ymin) + ymin);
    }
    return oColor;
}
```

Рисунок 3.4.6 – Программная реализация функции LINEAR

Реализация данной функции полностью соответствует алгоритму вычисления окна денситометрических показателей, описанному в пункте 2.6.1.

На вход функция получает текущее значение цвета пикселя finalColor, необходимые для вычисления атрибуты DICOM – WL, WW и ymin, ymax.

После применения функции calculateLinearColor полученное значение валидируется на предмет выхода за пределы диапазона и сохраняется во второй массив oColor.

Так как вначале исходные данные были преобразованы в структуру ARGB и массив из 8-разрядных чисел, необходимо применить обратное преобразование. Для этого используется, представленная на рисунке 3.4.7, функция конвертирования convertToAndroidArgb.

```
uint32_t convertToAndroidArgb(
        uint8_t alpha,
        uint8_t red,
        uint8_t green,
        uint8_t blue) {
    return (alpha << 24) | (red << 16)
            | (green << 8) | blue;
}
```

Рисунок 3.4.7 – Функция конвертации отдельных цветовых компонент в единый пиксель

После применения данной функции отдельные цветовые компоненты сохраняются в один пиксель, который присваивается элементу width × y + x массива newBitmapPixels, где width – ширина изображения в пикселях, y – текущее необработанное количество пикселей по высоте, x – текущее необработанное количество пикселей по ширине.

Таким образом, происходит обработка снимка со значение параметра Photometric Interpretation (0028,0004) – RGB.

3.4.4 Реализация метода jniCalculateDicomMONOCHROME

Метод применяется, когда значение параметра Photometric Interpretation (0028,0004) – MONOCHROME1. Минимальное значение выборки предназначено для отображения белым цветом после любых преобразований серой шкалы VOI. Это значение может использоваться только в том случае, если значение Samples per Pixel «Выборки на пиксель» (0028 0002) имеет значение 1. Может использоваться для данных пикселей в собственном (несжатом) или инкапсулированном (сжатом) формате.

Также метод применяется, когда значение параметра Photometric Interpretation (0028,0004) – MONOCHROME2. Данные пикселей представляют собой одну плоскость монохромного изображения. Минимальное значение выборки предназначено для отображения в виде черного после любых преобразований серой шкалы VOI. Это значение может использоваться только в том случае, если значение Samples per Pixel «Выборки на пиксель» (0028 0002) имеет значение 1. Может использоваться для данных пикселей в собственном (несжатом) или инкапсулированном (сжатом) формате.

Вначале функция, изображенная на рисунке 3.4.8, производит преобразование типов данных для использования в C++ коде.

```
JniBitmap *jniBitmap = (JniBitmap *) env->GetDirectBufferAddress(handler);
if (jniBitmap == NULL || jniBitmap->_storedBitmapPixels == NULL)
    return;
uint32_t *previousData = jniBitmap->_storedBitmapPixels;
const char *rescale_type_n = env->GetStringUTFChars(rescale_type, NULL);
const char *voilut_n = env->GetStringUTFChars(voilut, NULL);
const char *modality_n = env->GetStringUTFChars(modality, NULL);
```

Рисунок 3.4.8 – Конвертация входящих параметров

Далее необходимо определить максимально и минимальное значение интенсивности цвета пикселя в зависимости от типа Photometric Interpretation

(0028,0004) – 1 или 2. Если изображение сохранено как первый тип, то минимальное значение равно 255, что соответствует белому цвету, а максимальное – 0, что соответствует чёрному цвету. В случае если изображение сохранено как второй тип или тип не указан, то, наоборот, минимальное значение равно 0, а максимальное – 255. Алгоритм представлен на рисунке 3.4.9.

```
switch (type) {
    case 1:
        ymin = 255;
        ymax = 0;
        break;
    case 2:
        ymin = 0;
        ymax = 255;
        break;
    default:
        ymin = 0;
        ymax = 255;
        break;
}
```

Рисунок 3.4.9 – Значение минимального/максимального значения интенсивности цвета пикселя

Программный код вычисления цвета снимка представлен на рисунке 3.4.10.

```
    ARGB pixelDataRGB;
    uint32_t *newBitmapPixels = new uint32_t[width * height];
    jniBitmap->_storedBitmapPixels = newBitmapPixels;
#pragma omp parallel for
    for (int y = height - 1; y >= 0; --y)
#pragma omp parallel for
            for (int x = width - 1; x >= 0; --x) {
                uint32_t pixelData = previousData[width * y + x];
                convertIntToArgb(pixelData, &pixelDataRGB);
                jfloat finalColor = (
                        (pixelDataRGB.red + pixelDataRGB.green +
                         pixelDataRGB.blue) / 3);
                jfloat oColor;
                if (strcmp(voilut_n, "NULL") == 0) {
                    if (strcmp(modality_n, "CT") == 0) {
                        if (strcmp(rescale_type_n, "HU") == 0) {
                            finalColor = applyHounsfieldScale(finalColor, bits_stored,
                                                              rescale_slope,
                                                              rescale_intercept, ymax);
                        }
                    }
                    oColor = calculateLinearColor(finalColor, wl, ww, ymin, ymax);
                } else if (strcmp(voilut_n, "SIGMOID") == 0) {
                    oColor = calculateSigmoidColor(finalColor, wl, ww, ymin, ymax);
                } else if (strcmp(voilut_n, "LINEAR_EXACT") == 0) {
                    oColor = calculateLinearExactColor(finalColor, wl, ww, ymin, ymax);
                } else {
                    oColor = calculateLinearColor(finalColor, wl, ww, ymin, ymax);
                }

                if (oColor > ymax) oColor = ymax;
                if (oColor < ymin) oColor = ymin;
                oColor = round(oColor);

                newBitmapPixels[width * y + x] = convertToAndroidArgb(pixelDataRGB.alpha,
                                                                      static_cast<uint8_t>(oColor),
                                                                      static_cast<uint8_t>(oColor),
                                                                      static_cast<uint8_t>(oColor));
            }
    delete[] previousData;
```

Рисунок 3.4.10 – Программный код определения функции вычисления цвета снимка исследования

Аналогично функции для вычисления RGB, изображение представляет собой массив беззнаковых целых чисел, величиной в 32 разряда. Обработка происходит с помощью двух вложенных циклов с предусловием в обратном

порядке. Выполнение циклов распараллелено с помощью аннотации OpenMP «#pragma omp parallel for». Функция convertIntToAndroidArgb производит конвертацию числа в структуру с цветовыми компонентами для упрощения работы с отдельными цветовыми компонентами.

Далее вычисляется общий цвет как среднее от трёх цветовых компонент и сохраняется в переменную finalColor, также создаётся итоговая переменная oColor, в которую будет записан цвет после применения DICOM параметров.

Далее, в зависимости от значения VOI LUT Function (0028,1056), выбирается функция для расчёта значения цвета пикселя. Она может принимать три значения: VOI LUT Function – значение по-умолчанию, может задаваться в виде отсутствия какого-либо значения параметра; SIGMOID Function – задается как SIGMOID; LINEAR_EXACT Function – задаётся как LINEAR_EXACT. Алгоритм представлен на рисунке 3.4.11.

```
if (strcmp(voilut_n, "NULL") == 0) {
    if (strcmp(modality_n, "CT") == 0) {
        if (strcmp(rescale_type_n, "HU") == 0) {
            finalColor = applyHounsfieldScale(finalColor, bits_stored,
                                              rescale_slope,
                                              rescale_intercept, ymax);
        }
    }
    oColor = calculateLinearColor(finalColor, wl, ww, ymin, ymax);
} else if (strcmp(voilut_n, "SIGMOID") == 0) {
    oColor = calculateSigmoidColor(finalColor, wl, ww, ymin, ymax);
} else if (strcmp(voilut_n, "LINEAR_EXACT") == 0) {
    oColor = calculateLinearExactColor(finalColor, wl, ww, ymin, ymax);
} else {
    oColor = calculateLinearColor(finalColor, wl, ww, ymin, ymax);
}
```

Рисунок 3.4.11 – Алгоритм определения функции вычисления цвета пикселей снимков исследований

В случае отсутствия атрибута VOI LUT Function (0028,1056) программа проверяет модальность снимка, в случае, если модальность имеет значение «CT» и Rescale Type (0028,1054) имеет значение «HU», то дополнительно, перед передачей в основную функцию линейного вычисления цвета пикселя,

происходит преобразование значения пикселя по шкале Хаунсфилда. При отсутствии атрибута VOI LUT или при значении LINEAR используется функция calculateLinearColor, так же как при обработке снимка типа RGB.

В случае если значение атрибута VOI LUT равно SIGMOID, для обработки применяется функция calculateSigmoidColor, представленная на рисунке 3.4.12.

```
jfloat
calculateSigmoidColor(float finalColor, jfloat mWL, jfloat mWW, uint8_t ymin, uint8_t ymax) {
    return (((ymax - ymin) / (1 + exp(-4 * ((finalColor - mWL) / mWW)))) +
            ymin);
}
```

Рисунок 3.4.12 – Программная реализация функции SIGMOID

Реализация данной функции полностью соответствует алгоритму вычисления окна денситометрических показателей, описанному в пункте 2.6.3.

На вход функция получает текущее значение цвета пикселя finalColor, необходимые для вычисления атрибуты DICOM – WL, WW и ymin, ymax.

В случае если значение атрибута VOI LUT равно LINEAR_EXACT, для обработки применяется функция calculateLinearExactColor представленная на рисунке 3.4.13.

```
jfloat
calculateLinearExactColor(jfloat finalColor, jfloat mWL, jfloat mWW, uint8_t ymin, uint8_t ymax) {
    float oColor;
    if (finalColor <= mWL - mWW / 2)
        oColor = ymin;
    else if (finalColor > mWL + mWW / 2)
        oColor = ymax;
    else {
        oColor = static_cast<jfloat>(((finalColor - mWL) / mWW + 0.5) *
                                     (ymax - ymin) + ymin);
    }
    return oColor;
}
```

Рисунок 3.4.13 – Программная реализация функции LINEAR_EXACT

Реализация данной функции полностью соответствует алгоритму вычисления окна денситометрических показателей, описанному в пункте 3.2.2.

На вход функция получает текущее значение цвета пикселя finalColor, необходимые для вычисления атрибуты DICOM – WL, WW и ymin, ymax.

После применения одной из функций VOI LUT полученное значение валидируется на предмет выхода за пределы диапазона и сохраняется во второй массив oColor.

Далее применяется обратное преобразование цвета пикселя. Для этого используется функция конвертирования convertToAndroidArgb.

После применения данной функции отдельные цветовые компоненты сохраняются в один пиксель, который присваивается элементу width * y + x массива newBitmapPixels, где width – ширина изображения в пикселях, y – текущее необработанное количество пикселей по высоте, x – текущее необработанное количество пикселей по ширине.

Таким образом, происходит обработка снимка со значение параметра Photometric Interpretation (0028,0004) – MONOCHROME (1/2).

В результате был реализован программный алгоритм, основанный на математической модели вычисления цвета пикселей, исходя из значений плотностей тканей, а также связанных атрибутов на языке программирования Java с применения дополнительных средств оптимизации вычислений, таких как использование Android NDK для наиболее ресурсоёмких задач и открытого стандарта для распараллеливания программ на языках C, C++ - OpenMP.

4 АНАЛИЗ РЕЗУЛЬТАТОВ ПРОГРАММНОЙ РЕАЛИЗАЦИИ МАТЕМАТИЧЕСКОЙ МОДЕЛИ РЕГУЛИРОВКИ ПЛОТНОСТЕЙ ТКАНЕЙ НА СНИМКЕ

4.1 Анализ реализации математической обработки снимков

Реализация производилась с использованием Android SDK и Android NDK.

Android NDK представляет из себя набор инструментов, позволяющих реализовывать часть программного кода на языке программирования C или C++. Данный подход позволяет ускорить выполнение отдельных участков кода, наиболее требовательных аппаратным к ресурсам, таких как, к примеру, обработка изображений.

Ниже на рисунке 4.1.1 приведен график сравнения скорости обработки снимков при использовании Java (SDK) и C++ (NDK).

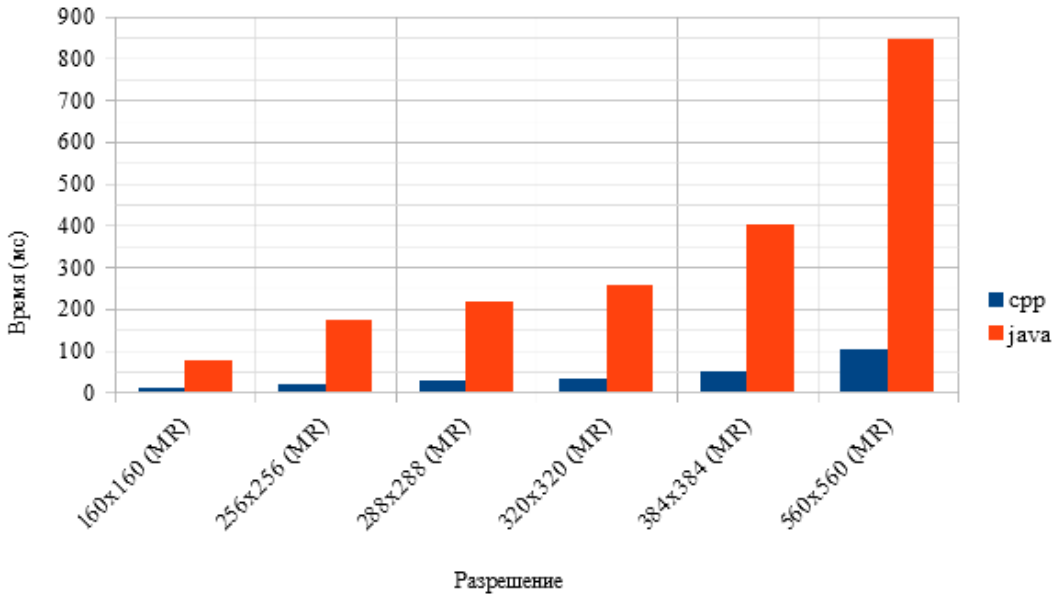

Рисунок 4.1.1 – График сравнения времени обработки снимков на языке Java и C++

Примем результат обработки снимков с помощью языка Java за единицу. Ниже приведена сравнительная таблица 4.1.1.

Таблица 4.1.1 – Сводные данные времени обработки снимка (Java и C++)

	160x160	**256x256**	**288x288**	**320x320**	**384x384**	**560x560**
Java	1	1	1	1	1	1
C++	0,1429	0,1207	0,1239	0,1202	0,1194	0,1194

Таким образом, чем выше разрешение снимка, тем эффективнее код, написанный с применением Android NDK в сравнении с кодом на Java.

В приведенной выше реализации обработка снимков производится в однопоточном режиме.

Растровое изображение представляет собой двумерную матрицу, обработка которой производится последовательно двумя циклами, которые проходят каждый пиксель снимка. Возьмём, для примера, изображение разрешением 256x256 - 65536 пикселей, которые в свою очередь хранят в себе 4 цветовых канала. Следовательно, алгоритму необходимо обработать 262144 значений. Это довольно большой объём данных. Для такой задачи было бы целесообразно применить распараллеливание на несколько потоков.

Для решения данной задачи необходимо выбрать инструмент для распараллеливания выполнения кода. Наиболее используемыми инструментами являются: OpenMP, MPI, C++ AMP и OpenCL (на графическом процессоре). Технология CUDA не рассматривалась из-за ограничения по аппаратной платформе. Данные представлены в таблице 4.1.2.

Таблица 4.1.2 – Сравнение инструментов реализации параллельных программ

	Процессор	**Ориентированность**	**Платформа**	**Синтаксис**
OpenMP	ЦП	Системы с общей памятью	Кроссплатформенная	Аннотации
MPI	ЦП	Системы с распределенной памятью	Кроссплатформенная	Встроенные функции
C++ AMP	ГП	Системы с общей памятью	Windows 7, Windows 8, Windows Server 2008 R2, or Windows Server 2012 (DirectX 11 Feature Level 11.0 or later hardware)	Синтаксическая конструкция «restrict(amp)» и встроенные функции через подключение заголовочного файла <amp.h>
OpenCL	ГП	Системы с общей памятью	Кроссплатформенная	Язык OpenCL C

Исходя из данных, представленных в таблице 4.2, видно:

C++ AMP не подходит по причине ориентированности под конкретную систему и платформу.

OpenCL потребовала бы существенную переработку существующего кода.

MPI больше ориентирована на кластеры и суперкомпьютеры.

Таким образом, наиболее оптимальным выбором является OpenMP.

Ниже на рисунке 4.1.2 приведены результаты работы алгоритма в сравнении с двумя предыдущими подходами.

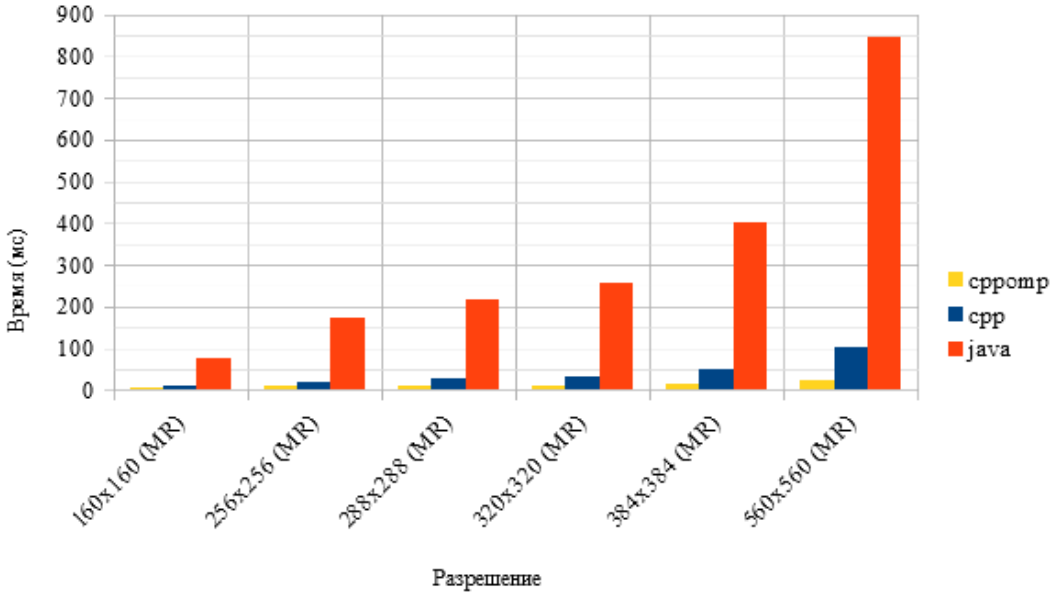

Рисунок 4.1.2 – График сравнения времени обработки снимков на языке Java, C++ и C++ с применением OpenMP

Для лучшей наглядности примем результаты Java за единицу, т.е. за 100%. Полученные результаты приведены в таблице 4.3.

Таблица 4.1.3 – Сводные данные времени обработки снимка для Java, C++ и C++ OpenMP

	160x160	**256x256**	**288x288**	**320x320**	**384x384**	**560x560**
Java	1	1	1	1	1	1
C++	0,1429	0,1207	0,1239	0,1202	0,1194	0,1194
C++ OpenMP	0,0779	0,0517	0,0413	0,0388	0,0373	0,0284

Ниже на рисунке 4.1.3 приведен график, демонстрирующий выигрыш в производительности при использовании OpenMP.

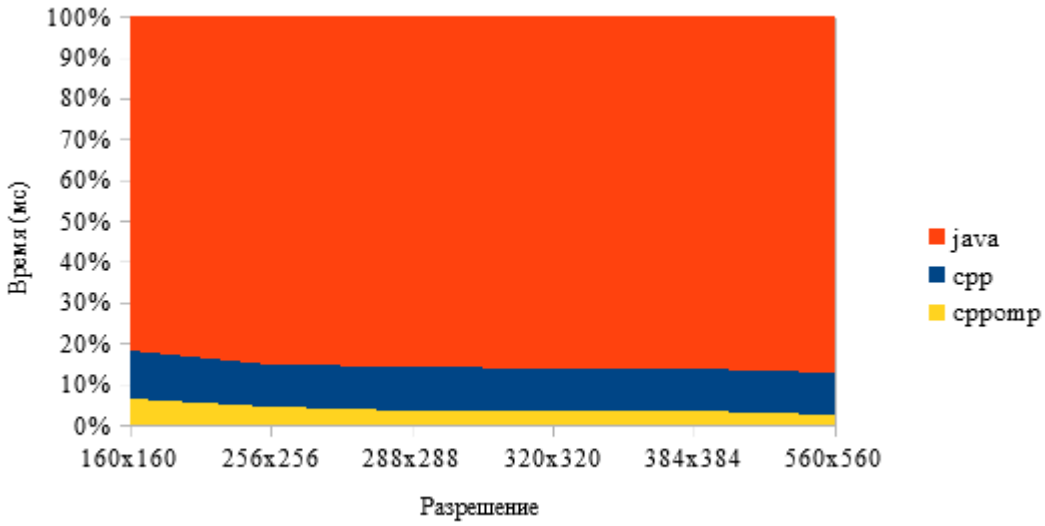

Рисунок 4.1.3 – График отношения времени обработки снимков на языке Java, C++ и C++ с применением OpenMP

Результат работы алгоритма обработки представлен на рисунке 4.1.4.

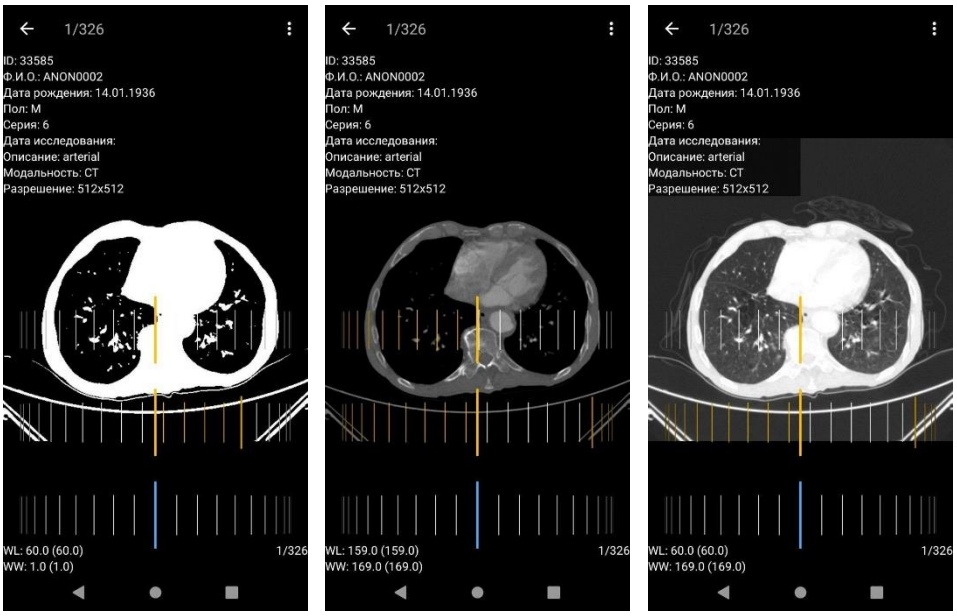

Рисунок 4.1.4 – Изменение окна денситометрических показателей

В результате проведён анализ реализованных алгоритмов обработки снимков с использованием языков программирования Java, C++ (Android NDK, JNI) и библиотеки для распараллеливания алгоритмов OpenMP.

4.2 Сравнительный анализ полученных результатов

Для оценки эффективности работы, разработанной в результате математического моделирования, системы регулировки плотности тканей на снимке исследования обобщим полученные в процессе работы данные.

Время загрузки и отображения DICOM без применения буферизации и предобработки представлены в таблице 4.2.1. Для сравнения берётся наиболее эффективная из рассмотренных в работе библиотек - Imebra.

Таблица 4.2.1 – Суммарное время для отображения одного DICOM снимка

#	**256x256**	**560x560**
Среднее (мс)	17 + 10217 = 10234	25 + 27802 = 27827

Время загрузки и отображения DICOM при применении буферизации и предобработки представлено в таблице 4.2.2.

Таблица 4.2.2 - Суммарное время для отображения одного снимка при применении буферизации и предобработки

#	**256x256**	**560x560**
Среднее (мс)	0,123932 + 951,5 = 951,62	0,145672 + 2828,5 + 2828,65

Также проведём сравнение времени обработки снимка – параметров плотности тканей.

Время обработки на мобильном устройстве при использовании формата DICOM для применения параметров плотности тканей представлено в таблице 4.2.3.

Таблица 4.2.3 – Время применения параметров WW/WL для формата DICOM

#	**160x160**	**256x256**	**288x288**	**320x320**	**384x384**	**560x560**	**1728x2328**
Среднее (мс)	3,86789	5,31135	4,9529	5,2875	5,9256	6,5889	51,131
Медианное (мс)	3,58981	5,10458	4,7459	4,9542	5,01653	6,22	51,025

Время обработки JPG для применения параметров плотности тканей представлено в таблице 4.2.4.

Таблица 4.2.4 – Время применения параметров WW/WL для формата DICOM

#	**160x160**	**256x256**	**288x288**	**320x320**	**384x384**	**560x560**	**1728x2328**
Среднее (мс)	5,3185	10,139	11,590	12,1001	17,9349	26,6615	476,9255
Медианное (мс)	6,2079	8,8823	8,9492	10,2274	15,2041	23,6161	475,9459

Сводная диаграмма медианных значений времени применения параметров плотности тканей представлена на рисунке 4.2.1.

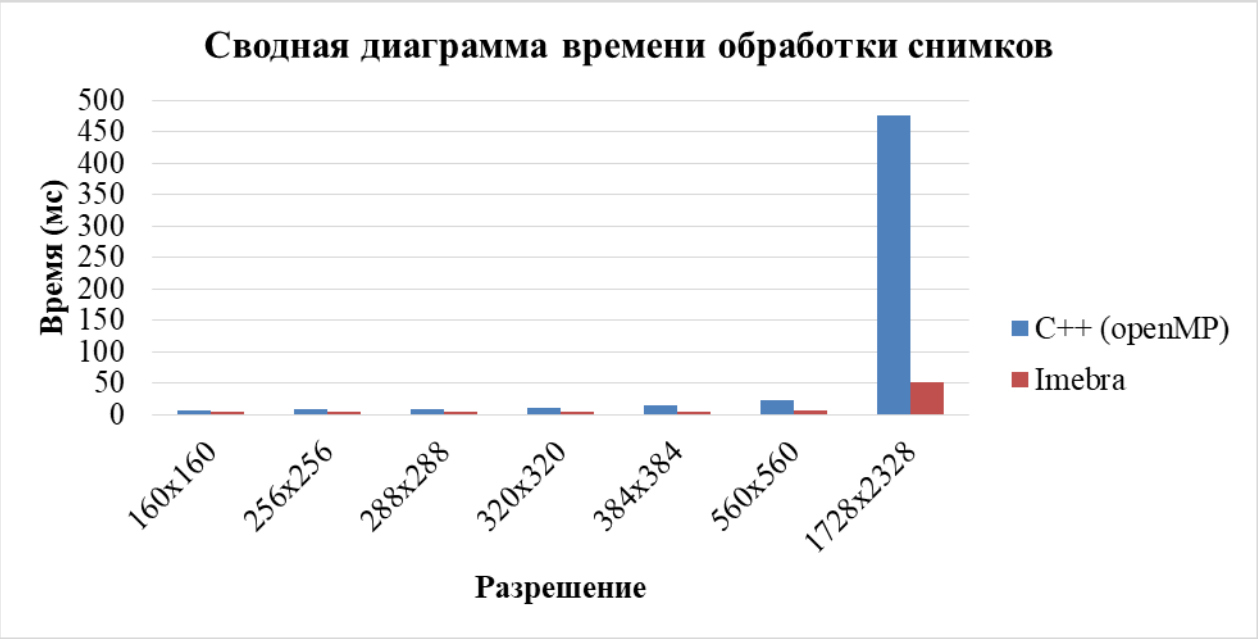

Рисунок 4.2.1 – Сводная диаграмма времени обработки снимков

В таблице 4.2.5 представлены данные по общему времени с момента отправки запроса до момента получения снимка, учитывая время применения значений плотности тканей.

Таблица 4.2.5 – Общее время получения, отображения и обработки снимков

#	**256x256**	**560x560**
С сервером предобработки	10,26368	26,80726
Напрямую	22,63414	31,61491

На рисунке 4.2.2 представлена сводная диаграмма полученных значений времени.

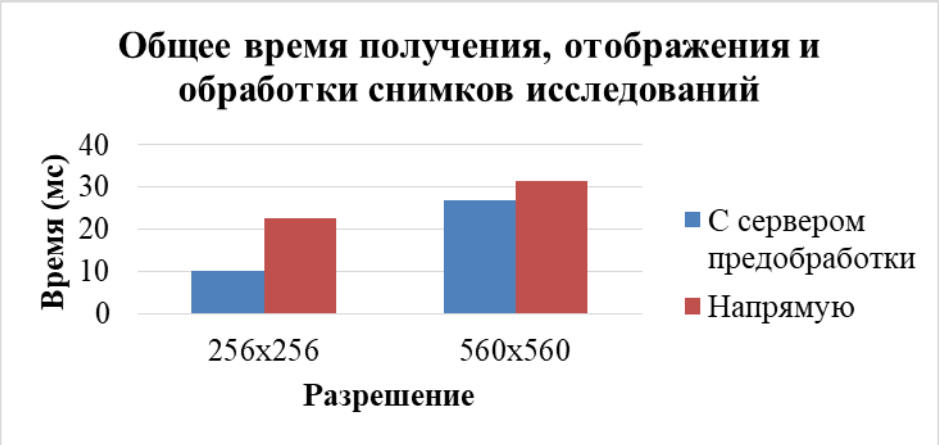

Рисунок 4.2.2 - Общее время получения, отображения и обработки снимков исследований

Построим диаграмму времени получения и обработки снимков в процентном соотношении. Построенная диаграмма представлена на рисунке 4.2.3.

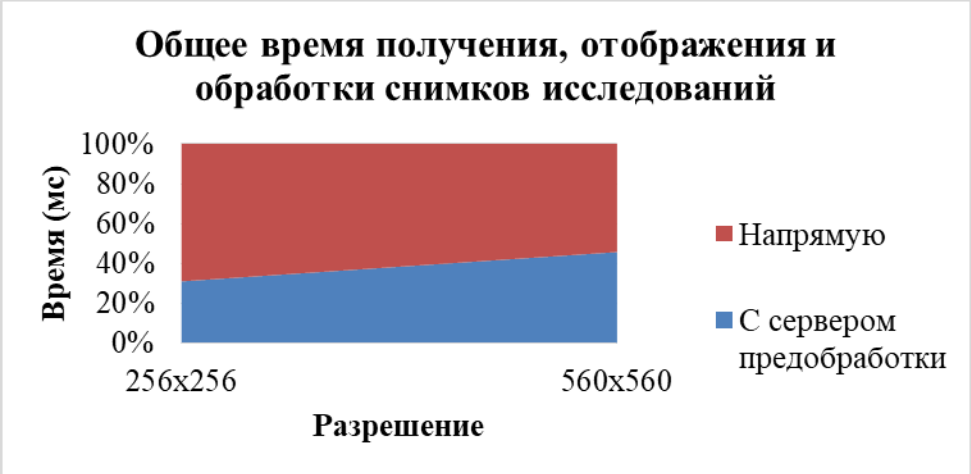

Рисунок 4.2.3 - Общее время получения, отображения и обработки снимков исследований в процентном соотношении

Таким образом, диаграммы рисунка 4.2.3 видно, что разработанная архитектура является более эффективной, нежели классический подход к получению DICOM снимков.

ЗАКЛЮЧЕНИЕ

В результате анализа мировых исследований в области использования мобильных устройств в сфере медицины были спрогнозированы результаты, представленные ниже.

По данным Росстата количество частных клиник на 2016 год составляет 23173.

Потенциально заинтересованными могут быть:

$$23173 \times 0.8 \times 0.4 = 7415 \text{ учреждений.}$$

Таким образом, около 9111 учреждений могут быть потенциально заинтересованы в использовании результатов данного исследования.

Были проанализированы основные технологические этапы, которые используются для получения исследований по пациентам.

Определены основные технологические этапы:

1. проведение исследования с помощью медицинского оборудования;

2. анализ лечащим врачом произведённых исследований с целью установления диагноза.

Были выделены основные этапы работы сотрудников по проведению и получению доступа к исследованиям с использованием медицинской автоматизированной информационной системы. Этапы рассматривались по принципу «чёрного ящика», это позволило определить последовательность работы сотрудников с системой. В этом случае можно выделить наиболее важные этапы – проведение исследования, просмотр информации о пациенте и обслуживание системы.

Исследована существующая система получения и обработки DICOM снимков, на основе анализа которой были выделены этапы, которые необходимо передать внешней системе – серверу буферизации и предобработки медицинских исследований с целью повышения эффективности работы системы в целом.

Определено, что установка правильной ширины окна и центра окна плотности тканей имеет решающее значение для обнаружения патологий.

Проведён анализ структуры данных формата DICOM, в результате которого определены особенности структуры файла, изучены атрибуты IOD, а также атрибуты при передаче и обработке изображений. Это позволило сделать вывод о возможности использования растрового формата изображений для обнаружения признаков заболевания и другой необходимой информации.

Проведено исследование математических моделей представления визуальных данных.

Исследованы функции регулировки денситометрических показателей. Составлены математические модели.

В результате проведённого анализа, как основной, выбран объектно-ориентированный язык программирования Java. Однако, обработка изображения является ресурсоёмким процессом, для которого важна производительность. Поэтому, для сравнения, часть алгоритмов будет реализовываться на языках программирования C/C++ с использованием Android NDK.

Также в результате анализа выбрана интегрированная среда разработки Android Studio.

Реализован программный алгоритм, основанный на математической модели вычисления цвета пикселей, исходя из значений плотностей тканей, а также связанных атрибутов на языке программирования Java без применения дополнительных средств оптимизации вычислений.

Реализован программный алгоритм, основанный на математической модели вычисления цвета пикселей, исходя из значений плотностей тканей, а также связанных атрибутов на языке программирования Java с применения дополнительных средств оптимизации вычислений, таких как использование

Android NDK для наиболее ресурсоёмких задач и открытого стандарта для распараллеливания программ на языках C, C++ с OpenMP.

Проведён анализ реализованных алгоритмов обработки снимков с использованием языков программирования Java, C++ (Android NDK, JNI) и библиотеки для распараллеливания алгоритмов OpenMP.

Проведён сравнительный анализ, учитывающий время передачи исследования, его конвертации и обработки. Получены данные, говорящие о том, что разработанная архитектура является более эффективной, нежели классический подход к получению DICOM снимков без применения сервера предобработки и буферизации данных.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

Научная и методическая литература

1. Алекс Макки. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов. — М.: «Вильямс», 2010. – С. 416.
2. Шимановский Н.Л. Контрастные средства: рук. по рацион. применению. – Москва: ГЭОТАР-Медиа, 2009. 463 с.
3. Штанчаев Х.Б., Математическая модель представления изображения в системах распознавания образов / Мир науки. – 2015. №2.
4. Прохоренко Н.А. Основы Java. – СПб: БХВ-Петербург, 2017. 704 с.
5. Воеводин В. В. Параллельные вычисления. – БХВ-Петербург, 2004, 608 с.
6. Красильников Н.Н. Цифровая обработка 2D и 3D изображений: Учебное пособие. – СПб.: БХВ-Петербург, 2011. – 608 с.: ил.
7. Грузман И.С., Киричук В.С. и др. Цифровая обработка изображений в информационных системах: Учебное пособие. – Новосибисрк: Изд-во НГТУ, 2000. - С. 168.
8. Либерти Д. Язык программирования C# // Программирование на C#. – Санкт-Петербург. — 2003: Символ-Плюс. — С. 26. — 688 с.
9. Бьёрн Страуструп. Язык программирования C++ / Пер. с англ. – 3-е изд. – СПб.; М.: Невский диалект — Бином, 1999. — 991 с.
10. Шилдт Г. C: полное руководство, классическое издание. – М.: Вильямс, 2010. — С. 704.
11. Герберт Шилдт. C# 4.0: полное руководство = C# 4.0 The Complete Reference. – М.: «Вильямс», 2010. — С. 1056.
12. Давыдов С., Ефимов А. IntelliJ IDEA. Профессиональное программирование на Java. – СПб.: БХВ, 2005. — 800 с.
13. Герберт Шилдт. Java. Полное руководство, 10-е издание = Java. The Complete Reference, 10th Edition. – М.: «Диалектика», 2018. — 1488 с.
14. Жемеров Д., Исакова С. Kotlin в действии. – ДМК-Пресс, 2017. — 402 с.

15. Арлоу Д. UML 2 и унифицированный процесс. Практический объектно-ориентированный анализ и проектирование. 2nd ed. – СПб: Символ-Плюс, 2014. 624 с.

Электронные ресурсы

16. Здравоохранение по Росстат: число больниц, больных и врачей [Электронный ресурс] / Режим доступа: https://rosinfostat.ru/zdravoohranenie/.
17. Исследование возможности улучшения качества изображения / КиберЛенинка [Электронный ресурс] / Режим доступа: https://cyberleninka.ru/article/n/issledovanie-vozmozhnosti-uluchsheniya-kachestva-izobrazheniya
18. Dicom Droid [Электронный ресурс] // Google Code Archive [Электронный ресурс] / Режим доступа: https://code.google.com/archive/p/dicom-droid/.
19. DICOM Rescale Intercept / Rescale Slope and ITK - Kitware Blog [Электронный ресурс] / Режим доступа: https://blog.kitware.com/dicom-rescale-intercept-rescale-slope-and-itk/
20. Enabling Open Innovation & Collaboration | The Eclipse Foundation [Электронный ресурс] / Режим доступа: https://www.eclipse.org/
21. Fresco An image management library. | Fresco [Электронный ресурс] / Режим доступа: https://frescolib.org/
22. GitHub - nostra13/Android-Universal-Image-Loader: Powerful and flexible library for loading, caching and displaying images on Android. Android [Электронный ресурс] / Режим доступа: https://github.com/nostra13/Android-Universal-Image-Loader
23. Glide v4 : Fast and efficient image loading for Android [Электронный ресурс] / Режим доступа: https://bumptech.github.io/glide/
24. "JavaOne 2013 Review: Java Takes on the Internet of Things" [Электронный ресурс] / Режим доступа:

https://www.imarslan.com/javaone-2013-review-java-takes-on-the-internet-of-things

25. Meet Android Studio | Для разработчиков Android | Android Developers [Электронный ресурс] / Режим доступа: https://developer.android.com/studio/intro
26. Open source C++ DICOM library [Электронный ресурс] // Imebra [Электронный ресурс] / Режим доступа: https://imebra.com
27. Picasso [Электронный ресурс] / Режим доступа: https://square.github.io/picasso/
28. Samples per Pixel Attribute – DICOM Standard Browser. Browser [Электронный ресурс] / Режим доступа: https://dicom.innolitics.com/ciods/us-image/us-image/00280002
29. VOI LUT Function Attribute – DICOM Standard Browser [Электронный ресурс] / Режим доступа: https://dicom.innolitics.com/ciods/nm-image/voi-lut/00281056
30. Working with DICOM LUT | Raster, Medical, Document Help [Электронный ресурс] / Режим доступа: https://www.leadtools.com/help/sdk/v20/dh/to/working-with-dicom-lut.html

Литература на иностранном языке

31. Moskala M., Wojda I. Android Development with Kotlin. – Birmingham: Packt Publishing Ltd, 2017.
32. Chin J. L., Kok S. S., Teck K. K. Contrast and Brightness Enhancement for DICOM Images and Lesions Auto Detection. – Journal of Image and Graphics, Vol. 4, No. 1, 2016.
33. Leiva A. Kotlin for Android Developers. – Antonio Leiva, 2017.
34. Meier R., Lake I. Professional Android . – John Wiley & Sons Inc, 2018.
35. The New Role of Technology in Consumer Health and Wellness. – Consumer Electronic Assocoation, 2011.

ПРИЛОЖЕНИЕ А

Диаграмма классов мобильного клиента