Содержание

Введение

С повсеместным внедрением в нашу жизнь цифровых технологий, ежедневно увеличивается количество оцифрованных аудиозаписей различного содержания. На данный момент в глобальной сети Интернет содержатся миллионы файлов, содержащих записи голоса, музыки, различных звуков. Кроме того, с развитием технологий в настоящее время многие устройства поддерживают голосовое управление.

Содержимое любого аудиофайла или любую голосовую команду можно рассматривать как сигнал, обладающий некоторыми уникальными характеристиками, которые можно назвать признаками данного сигнала.

Популярной проблемой в научном сообществе является проблема формирование признакового описания звукового сигнала, обладающего как можно большей информативностью, но при этом как можно меньшей размерностью. Эффективное решение такой проблемы имеет множество применений: такую систему формирования признакового описания звукового сигнала можно использовать для идентификации человека по голосу, для классификации музыкальных произведений по жанрам, для создания рекомендательных музыкальных систем, а также систем поиска похожих сигналов на определенный сигнал (будь это голос, музыка, звук – что угодно)

В связи с этим возникла идея создания программной системы, которая представит эффективную и универсальную реализацию алгоритма для извлечения признакового описания из звукового сигнала. Такая система могла бы найти себе множество применений, за счет решения довольной общей задачи – её можно было бы применять в связке с различными классификаторами к совершенно разным по своему содержанию наборам данных.

Таким образом, целью данной работы является создание программной системы формирования признакового описания звукового сигнала. Полных функциональных аналогов данной системы нет на рынке, это обусловлено, в первую очередь, тем, что все имеющиеся системы ориентированы на извлечение признакового описания из данных, не уточняя специфику этих данных (звуковой сигнал). Кроме того, существующие системы формирования признакового описания именно звукового сигнала, являются частью систем работы со звуком, имеющих закрытый код. Задача разрабатываемой системы заключается не только в формировании признакового описания звукового сигнала, но и в получении возможности внедрения разработанной технологии в различные системы обработки и классификации звуковых сигналов для оптимизации их производительности и улучшения эффективности их работы.

1 Техническое задание

1.1 Назначение разработки и область применения

Разрабатываемая программная система предназначена для формирования признакового описания звукового сигнала, основанного на спектральном представлении сигнала, формируемым с помощью *U*-преобразования из теории активного восприятия. Для работы системы необходимо по заранее подготовленной базе аудиозаписей в формате mp3 осуществить нормирование и *U*-преобразование каждой записи, после чего будет возможным извлечение признакового описания.

Область применения разрабатываемой системы:

– Интеграция с системами распознавания и обработки звуковых сигналов.

Разрабатываемая система предназначается для использования на стационарных и портативных компьютерах в связке с модулями подготовки входных данных и принятия окончательного решения (распознавания образа)

1.2 Технические требования

Рассмотрим требования, предъявляемые разрабатываемой системой к ЭВМ:

– операционная система Microsoft Windows, начиная с версии XP и выше;

– требования к аппаратному обеспечению определяются операционной системой;

– устройства ввода: мышь, клавиатура;

– устройства вывода: монитор, динамики.

Рассмотрим, какой функциональностью должна обладать проектируемая программная система формирования признакового описания звукового сигнала:

– перед началом работы система должна получить на вход данные, формируемые в несколько этапов:

1. считывание в память .mp3-файла;
1.1. разделение его на отсчеты;
1.2. нормирование сигнала;
1.3. *Q*-преобразование сигнала;
1.4. *U*-преобразование сигнала.

Все эти подготовительные этапы реализованы в отдельном модуле, который подготавливает входные данные для работы с разрабатываемой системой:

– система должна предоставить пользователю возможность выбрать систему признаков, в соответствии с которой будет вычисляться признаковое описание звукового сигнала (*PVI*, *PVD*, и другие);

– произвести анализ полученных на вход данных и выделить из них признаковое описание в соответствии с заданной пользователем системой признаков.

2. Анализ технического задания

2.1 Выбор операционной системы

Одной из важных задач при подготовке к разработке программной системы является выбор операционной системы, под управлением которой будет происходить непосредственно сама разработка.

Для решения этой задачи необходимо произвести анализ существующих и доступных на сегодняшний день операционных систем для настольных компьютеров, каждая из которых обладает своими уникальными свойствами, которые могут оказать влияние в том числе и на процесс разработки. Результатом этого анализа должно быть определение операционной системы, в наибольшей степени удовлетворяющей требованиям, предъявляемым к разрабатываемой системе.

Для анализа возьмем три популярные на сегодняшний день операционные системы: это Linux, Windows, и MacOS

1) Windows – продукт компании Microsoft Corporation, существует на рынке с середины 80-х годов прошлого века. Имеет широкую распространенность среди рядовых пользователей персональных компьютеров, однако используется не только на домашних компьютерах, но встречается и на предприятиях, где ставятся повышенные требования к надежности, а также решаются более сложные задачи. Данная операционная система является самой первой системой, получившей графический интерфейс взамен консольному интерфейсу.

Все когда-либо существовавшие версии Windows можно условно разделить на 5 семейств, поэтому рассмотрим детальнее семейства этой операционной системы:

1.1. расширения системы MS-DOS (в эту группу входят версии до Windows-95);
1.2. семействоWindows-9x (все версии до Windows ME);
1.3. семейство Windows NT (спереходом на 32/64 бита, к этому семейству относятся все современные версии, включая Windows 10);
1.4. семейство Windows для смартфонов, и семейство встраиваемых ОС (Windows Embedded).

1) MacOS занимает второе место по популярности среди операционных систем для пользователей ПК. На рынке присутствует с 1984 года – именно тогда она появилась как единственная система для компьютера Macintosh. Основным и важнейшим отличием от Windows является принадлежность MacOS к семейству операционных систем Unix.

2) Linux – это обобщенное название Unix-подобных операционных систем, которые основа на ядре Linux Kernel. Первая версия ядра была разработана и выпущена в 1991 году. Одним из важнейших отличительных качеств LinuxKernel является открытость его кода. То есть эта операционная система создаётся и распространяется в соответствии с моделью разработки свободного и открытого программного обеспечения. Linux Kernel - это лишь основа, ядро операционной системы, поэтому не существует никакого официального дистрибутива этой ОС, и, как правило, Linux-системы распространяются как бесплатные готовые дистрибутивы, каждый из которых направлен на определенную специфику пользовательских задач, и комплектуется своим определенным набором утилит и программ прикладного характера. Несмотря на то, что ОС Linux вышла на рынке позднее своих основных конкурентов, её популярность с каждым днем становится всё больше – причиной тому является как бесплатное распространение, так и гибкость системы, позволяющая наилучшим образом использовать все ресурсы машины для решения специфичных задач пользователей.

Проанализируем основные различия между операционными системами Windows, MacOS и Linux, а также сравним их преимущества и недостатки. Отбирать наибольшим образом удовлетворяющую операционной систему для разработки нашей программы будем по следующим критериям: защищенность, стабильность работы, требования к аппаратному обеспечению, наличие прикладного программного обеспечения, удобство разработки и применения программного обеспечения.

1) Защищённость.

Если проанализировать критерий защищённости системы от каких-либо внешних воздействий и возможных нарушений работы, то по данному критерию операционные системы Linux и MacOS опережают Windows. Причинами этому являются популярность и специфика программной архитектуры операционной системы Windows. Следствием из совокупности этих факторов стало порождение довольно большого количества вредоносных программ и различного рода вирусов, направленных на нарушение корректного функционирования данной системы.

2) Стабильность работы.

По данному критерию наилучшим образом себя показывает операционная система Linux. Различного рода сбои, нарушающие корректную работу, происходят на ней значительно реже, чем на Windows или MacOS.

3) Требования к аппаратному обеспечению.

Системой, проигрывающей по данному критерию всем остальным, безусловно, является MacOS. Обусловлено это тем, что Windows и Linux предоставляется возможным установить практически на любое оборудование, предоставленное различными производителями, существующими на рынке (ASUS, HP, Acer, Dell, и т.д.), а вот для установки MacOS, напротив, подойдёт только лишь оборудование от фирмы Apple.

4) Наличие прикладного программного обеспечения.

Ввиду большой популярности операционной системы Windows, данная ОС имеет наиболее богатую библиотеку прикладного программного обеспечения, по сравнению с Linux и MacOS, и, как следствие, большинство программ, существующих в данный момент, написаны для Windows, и далеко не все из них имеют портированные версии или аналоги, которые предназначенные для установки на другие операционные системы.

5) Удобство разработки и применения программного обеспечения.

Большинство программ, разработанных для операционной системы Windows, написаны на таких высокоуровневых языках программирования как C++, C#, Java и Visual Basic. Некоторые из этих языков программирования имеют возможность компиляции исполняемых модулей для их последующего применения на операционных системах MacOS и Linux. Для разработки программной системы формирования признакового описания звукового сигнала была выбрана ОС Windows. Это обусловлено тем, что, несмотря на относительную незащищённость, эта операционная система обладает наибольшим выбором программного обеспечения для разработки, а для разработки нашей системы это очень важно.

В целом ОС Windows полностью соответствует необходимым технологическим и функциональным требованиям для разработки данного проекта.

2.2 Выбор языка программирования

Следующая важная задача – выбор языка программирования, с помощью которого будет осуществляться разработка всего приложения. Самыми распространенными высокоуровневыми языками программирования для операционной системы Windows являются C++, C#, и Java.

Рассмотрим их более подробно:

– C++ - это высокоуровневый, объектно-ориентированный язык, относящийся к типу компилируемых языков с явной и статической типизацией. Кроме того, к нему применима и процедурная парадигма программирования, унаследованная от его предка - языка C. Также этот язык позволяет использовать такие средства и приёмы программирования, как: использование обработки исключений, обобщенное программирование и абстрагирование данных. C++ известен выраженной универсальностью своего применения: его используют и для написания компонентов операционных систем, и для создания драйверов устройств, и для разработки прикладного ПО. Имеет полную совместимость с языком C – таким образом, программа, написанная на C, может быть собрана компилятором C++. Этот язык оказал влияние на разработку идеологии таких языков программирования, как C# и Java.

– C# - это объектно-ориентированный язык, который был разработан компанией Microsoft Corporation специально для их широко известной платформы - Microsoft .NET Framework. C# сочетает в себе довольно многие хорошие решения, заимствованные из других языков программирования. Таким примером может является множественное наследование интерфейсов. Однако, есть и исключения слабых мест, например, множественное наследование классов из языка C++ - его нельзя использовать в C#. Синтаксис этого языка похож на синтаксис языков, послуживших ему предшественниками - он близок к C++ и Java. C# позволяет использовать такие средства и приёмы программирования, как: перегрузка операторов, статическая типизация, анонимные функции, полиморфизм, обобщенные методы и типы, а так же исключения.

– Java – изначально данный объектно-ориентированный язык программирования разрабатывался для применения в области бытовой электроники. Основной причиной появления такого языка как Java стало нежелание производителей различного рода электроники писать код отдельно под каждую платформу или архитектуру оборудования с полного нуля. Таким образом, был создан язык программирования Java, для осуществляется трансляция исходного кода в байт код, который затем выполняется на виртуальной java-машине (JVM). Эта виртуальная машина пишется отдельно под каждую

архитектуру, и обрабатывает байт-код, передавая его в виде инструкций непосредственно на аппаратное обеспечение и, таким образом, работает как интерпретатор, но значительно быстрее. В настоящее время для всех популярных платформ существуют виртуальные java-машины, позволяющие выполнять на них исходный код на языке Java.

Для разработки данной программной системы был выбран язык программирования C++, и решающим фактором в этом выборе стала так называемая нативность этого языка (англ. Native – родной, предусмотренный самой платформой), благодаря которой можно добиться существенной производительности реализуемых алгоритмов по сравнению с интерпретируемыми, не являющимися нативными языками программирования.

2.3 Выбор среды разработки

Следующим этапом является этап выбора интегрированной среды разработки, с помощью которой будет осуществляться непосредственно написание исходного кода программы, его дальнейшая отладка и тестирование. Проанализируем и сравним три лидирующие среды разработки, совместимые с языком C++: Code::Blocks, QtCreator и Microsoft Visual Studio.

– Code::Blocks – это среда разработки, которая является свободной и кроссплатформенной. Эта IDE была разработана на C++. Имеется открытая архитектура, которая может быть масштабирована за счёт различных внешних модулей. Поддерживается работа с такими языками программирования, как C/C++, Fortran, D, и другие.

За счет кроссплатформенности данной среды, возможно её использование как на Windows, так и на Linux и на MacOS. Обладает следующими достоинствами: поддерживается множество разных компиляторов (MinGW/GCC C/C++, Borland C++, Intel C++, MSVC++), есть возможнось использования решений, содержащих несколько профилей, поддержка не только одного рабочего пространства, но и нескольких, поддержка интегрированного в отладчик механизма визуализации результатов работы методов, функций и значений переменных, профилировщик.

– QtCreator - представляет из себя кроссплатформенную и свободную среду, предназначенную для разработки на таких языках, как C/C++ и языке QML. Изначально проектировалась как среда для непосредственной работы с многофункциональным фреймворком Qt. В QtCreator есть наличие графического интерфейса для взаимодействия с отладчиком, также набор всех необходимых инструментов для проектирования пользовательского интерфейса, причем здесь есть возможность использовать и QtWidgets, и QML. В этой среде поддерживаются следующие компиляторы: GCC, Clang, MinGW, Microsoft Visual C++, и другие.

Основным назначением среды является упрощение разработки приложений на фреймворке Qt под различные платформы. В Qt Creator обеспечены интегрированные системы сборки: qmake, cmake, autotools. Также эта среда нативно поддерживает работу с различными системами контроля версий (Git, Mercurial, Subversion, и др.).

– Microsoft Visual Studio - семейство программ для разработки от Microsoft Corporation, включающих в себя как интегрированную среду разработки ПО, так и некоторые другие полезные инструментальные средства. С их помощью возможная разработка как приложений с GUI (для этого интегрирована поддержка технологии

Windows Forms), так и простых консольных приложений, а также создавать веб-ориентированные приложения. Разработка кода возможна сразу для всех существующих платформ, поддерживаемых ОС Windows, таких как Windows Mobile, Windows CE, .NET Framework, Xbox, и других.

В Visual Studio имеется встроенный редактор исходного кода, который поддерживает технологию IntelliSense и имеет набор возможностей для рефакторинга исходного кода. Интегрированным отладчиком можно пользоваться как для исходного кода, так и для инструкций машинного уровня. Другие встраиваемые модули представляют из себя редактор форм, используемый для упрощения создания графического интерфейса, средство проектирования классов классов и утилиты для построения схемы базы данных. Visual Studio позволяет разрабатывать, выкладывать, устанавливать и использовать сторонние модули (плагины), предназначенные для расширения функциональности на каждом этапе разработки, включая взаимодействие с системами контроля версий, интегрирование разных полезных при разработке проекта утилит и инструментов для различных этапов процесса разработки ПО

QtCreator не был выбран в связи с отсутствием необходимости тесной интеграции с фреймворком Qt для реализации проектируемой системы формирования признакового описания звукового сигнала. В Code::Blocks отсутствуют встроенные механизмы рефакторинга кода, которые могли бы существенно упростить отладку и сократить время, затраченное на разработку программного кода. Поэтому для реализации системы была выбрана среда разработки Microsoft Visual Studio. Она позволяет гибко организовать работу с исходным кодом, имеет удобный отладчик, поддерживает рефакторинг кода, и отвечает всем требованиям, предъявляемым к среде разработки.

2.4 Выбор подхода к решению задачи формирования признакового описания звукового сигнала

Проблема формирования системы признаков связанна с выбором конечного множества признаков, обеспечивающих однозначность решения задачи классификации на этапе распознавания и отвечающая требованиям необходимости и достаточности. Этап выбора системы признаков необходим для сокращения размерности входного описания. Учитывая, что задача сокращения размерности – оптимизационная задача, то для её решения необходимо использование критерия информативности. Отсутствие модели априорной неопределённости и модели её раскрытия породило большое количество методов в выборе критерия информативности, что, в свою очередь, порождает большое число возможных вариантов признаков.

Рассмотрим три различных подхода для получения признакового описания звукового сигнала:

1) Преобразование Фурье (обозначается символом $\mathscr{F}$) — это метод, позволяющий сопоставить одной функции вещественной переменной другую функцию вещественной переменной.

Функция, которая ставится в сопоставление имеющейся, описывает коэффициенты («амплитуды») при разложении исходной функции на элементарные составляющие — гармонические колебания с разными частотами.

Преобразование Фурье функции f вещественной переменной является интегральным и задаётся следующей формулой:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ix\omega}\, dx.$$

Преобразование Фурье используется во многих областях науки — в физике, теории чисел, комбинаторике, обработке сигналов, теории вероятностей, статистике, криптографии, акустике, океанологии, оптике, геометрии и многих других. В обработке сигналов и связанных областях преобразование Фурье обычно рассматривается как декомпозиция сигнала на частоты и амплитуды, то есть обратимый переход от временного пространства (time domain) в частотное пространство (frequency domain).

Пример Фурье-преобразования изображен на рис.1.

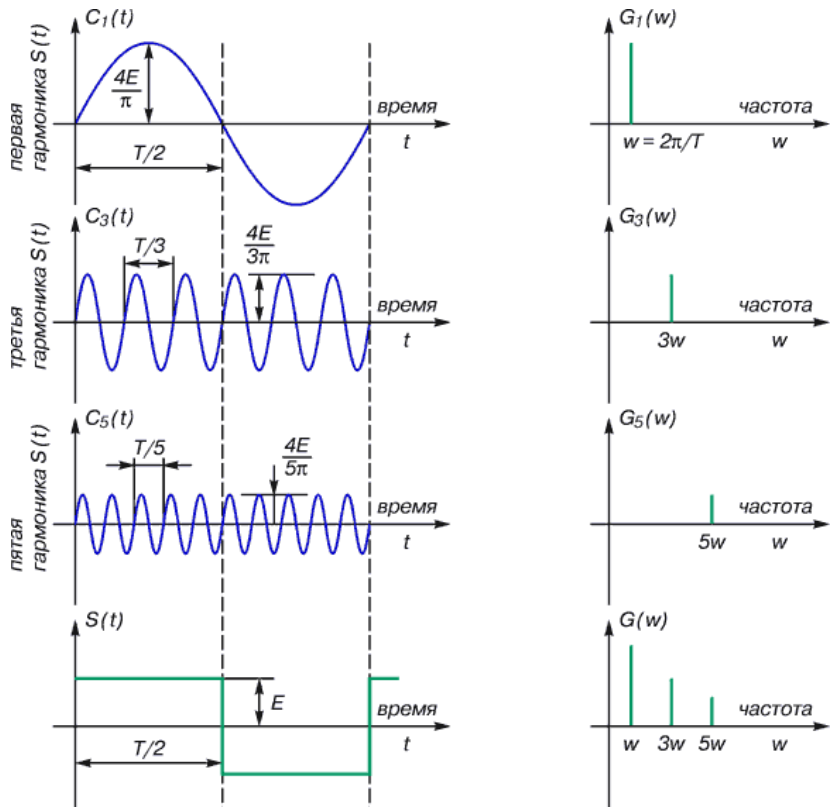


Рис. 1. Фурье-преобразование.

2) Вейвлет-функция — математическая функция, предназначенная для анализа различных частотных компонент данных. График функции выглядит как волнообразные колебания с амплитудой, уменьшающейся до нуля вдали от начала координат. Однако это частное определение — в общем случае анализ сигналов производится в плоскости вейвлет-коэффициентов (масштаб — время — уровень) (Scale-Time-Amplitude). Вейвлет-коэффициенты определяются интегральным преобразованием сигнала. Полученные вейвлет-спектрограммы принципиально отличаются от обычных спектров Фурье тем, что дают чёткую привязку спектра различных особенностей сигналов ко времени.

Вейвлет-преобразование — интегральное преобразование, которое представляет собой свертку вейвлет-функции с сигналом. Вейвлет-преобразование переводит сигнал из временного представления в частотно-временное.

Способ преобразования функции (или сигнала) в форму, которая или делает некоторые величины исходного сигнала более поддающимися изучению, или позволяет сжать исходный набор данных. Вейвлетное преобразование сигналов является обобщением спектрального анализа. Термин (англ. *wavelet*) в переводе с английского означает «маленькая волна». Вейвлеты — это обобщённое название математических

функций определенной формы, которые локальны во времени и по частоте и в которых все функции получаются из одной базовой, изменяя её (сдвигая, растягивая).

Пример непрерывного вейвлет-преобразование сигнала, содержащего смену частоты, приведен на рис.2.

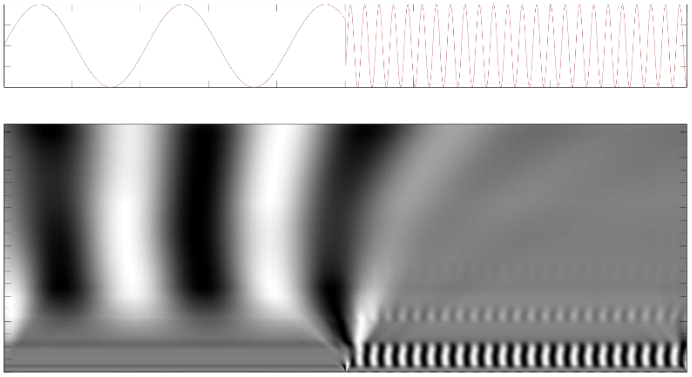


Рис. 2. Непрерывное вейвлет-преобразование сигнала, содержащего смену частоты

3) Предлагаемый подход основан на использовании теории активного восприятия [1]. Рассматривая теорию активного восприятия как систему распознавания образов, в ней можно выделить три этапа обработки информации: предварительная обработка, вычисление признаков и принятие решения.

Предварительная обработка заключается в выполнении *Q*-преобразования, заключающегося в применении к сегментам исходного сигнала операции сложения:

$$g(t) = \sum_{k=(t-1)L+1}^{t\cdot L} f(k), t = \overline{1, N},$$

где L – число отсчётов, входящих в сегмент, N – число сегментов сигнала, g – результат применения *Q*-преобразования к сигналу f, $f(k)$ – k-ый отсчёт сигнала f, $g(t)$ – t-ый отсчёт сигнала g.

Формирование признакового описания исходного сигнала заключается в применении к сигналу g множества фильтров $F_i \in \{F_i\} \equiv \boldsymbol{F}$ Уолша системы Хармута:

$$\mu(k, c(t)) = \sum_{i=0}^{M-1} F_k(i) \cdot g(((t-1)\cdot M + 1) : (t \cdot M)),$$

где $\mu(k, c(t))$ – результат применения множества фильтров Уолша системы Хармута к сигналу g, $k = \overline{0, M-1}$, $t = \overline{0, |c|-1}$, $c = \{1, P, 2\cdot P, 3\cdot P, ..., N - T\cdot P\}$ – множество значений смещений по сигналу g, $|c|$ – мощность множества c, P – величина смещения по сигналу g ($1 \le P \le M$), M – число используемых фильтров. Таким образом, признаковое описание сигнала представляет собой матрицу размером $M \times |c|$, причём каждая строка признакового описания представляет собой результат *U*-преобразования сегмента сигнала.

Последовательное применение к сигналу Q-преобразования и системы фильтров реализуют U-преобразование, являющееся базовым в теории активного восприятия. U-преобразование имеет минимально возможную вычислительную сложность, поскольку при его реализации используются простейшие операции – сложение и вычитание. Стандартные преобразования, требуют реализации свертки, а на уровне весовых коэффициентов – операции арифметического умножения.

Пусть каждому фильтру $F_i \in \{F_i\} \equiv \boldsymbol{F}$ соответствует бинарный оператор $V_i \in \{V_i\} \equiv \boldsymbol{V}$; тогда компоненте $\mu_i \neq 0$ вектора μ допустимо поставить в соответствие оператор V_i либо $\overline{V_i}$ в зависимости от знака компоненты. В результате вектору μ ставится в соответствие подмножество операторов из $\{V_i\}$, имеющих аналогичную фильтрам конструкцию, но разное значение элементов матрицы ($+1 \leftrightarrow 1$; $-1 \leftrightarrow 0$). Задавая на множестве $\{V_i\}$ операции теоретико-множественного умножения и сложения, имеем алгебру описания сигнала в одномерных булевых функциях. С учётом инверсий всего существует 15 операторов, которые могут использоваться при формировании признакового описания, так как оператор V_0 принимает только прямое значение.

На множестве операторов формируется алгебра групп (этап синтеза) анализируемого сигнала:

1) семейство алгебраических структур (названных полными группами) $\{P_{ni}\}$ вида $P_{ni} = \{V_i, V_j, V_k\}$ мощности 35;

2) семейство алгебраических структур (названных замкнутыми группами) $\{P_{si}\}$ вида $P_{si} = \{V_i, V_j, V_k, V_r\}$ мощности 105, где каждая группа образована из пары определенным образом связанных полных групп.

Среди полных групп выделяют полные группы на операции сложения и на операции умножения, среди замкнутых групп – замкнутые группы и замкнутые множества.

Используя спектральное представление сигнала μ, формируется множество операторов описывающий данный сигнал, а затем множества полных и замкнутых групп:

$V = GV[\mu]$, $P_{na} = GP_{na}[\mu, V]$, $P_{nm} = GP_{nm}[\mu, V]$,

$P_s = GP_s[\mu, V, P_{na}, P_{nm}]$, $P_c = GP_c[\mu, V, P_{na}, P_{nm}]$,

где GV – оператор вычисления по спектральному представлению сигнала признакового описания V на основе операторов, GP_{na} (GP_{nm}) – на основе полных групп на операции сложения P_{na} (умножения, P_{nm}), GP_c (GP_s) – на основе замкнутых групп P_s (замкнутых множеств, P_c).

3.Разработка структуры системы формирования признакового описания

3.1 Разработка общей структуры системы

Для построения общей структурной схемы разрабатываемой программной системы, проанализируем диаграмму преобразования данных внутри системы, приведенную на рис. 3.

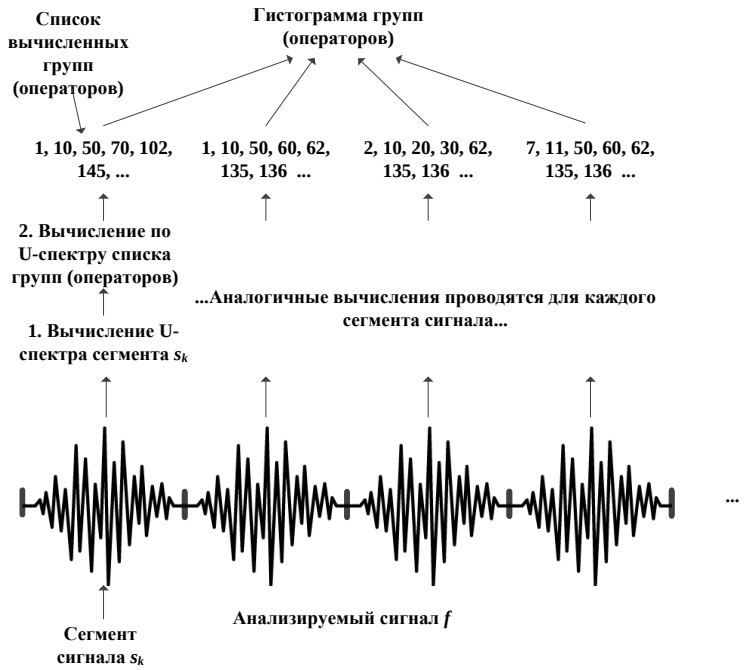


Рис. 3. Диаграмма преобразования данных

Из диаграммы видно, что сигнал S выступает в роли входных данных для системы, после чего происходит его разбиение на сегменты S_k. Далее, каждый сегмент подвергается процессу вычисления спектральных коэффициентов, который осуществляется с помощью U-преобразования. Затем, полученные спектральные коэффициенты используются для создания полных групп. Полные группы, в свою очередь, используются для создания замкнутых групп, а замкнутые группы – для создания замкнутых множеств. Далее полученные структуры данных используются вместе с заранее вычисленными списками групп (операторов) для формирования гистограмм групп, являющихся признаковым описанием звукового сигнала, полученного на входе.

Для подготовки общей схемы структуры программной системы, синтезируем блок-схему работы основного алгоритма (см. рис. 4).

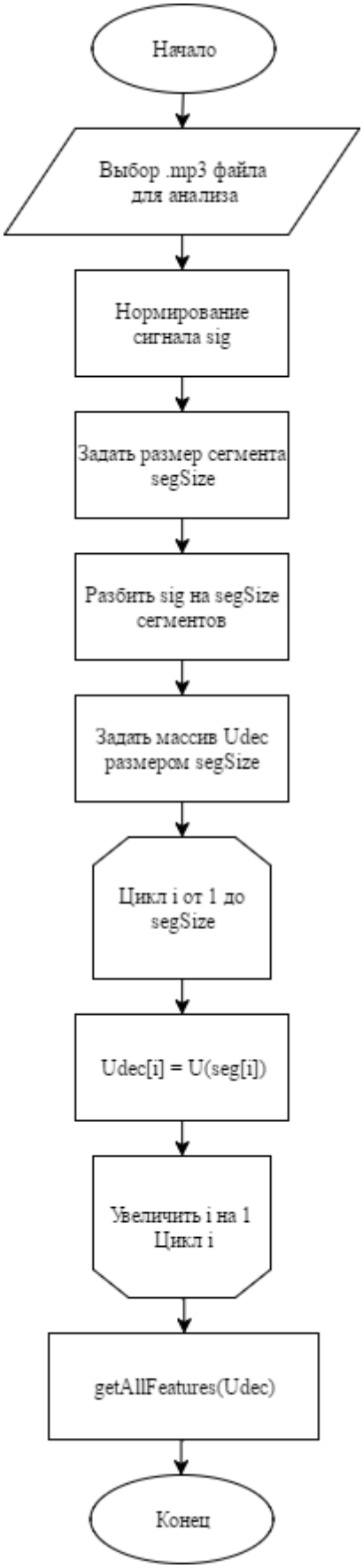


Рис. 4. Блок схема основного алгоритма программы

Поясним некоторые элементы приведенной блок-схемы. Нормирование сигнала осуществляется приведением всех значений каждого сегмента к диапазону [0,1]. В основном цикле происходит непосредственно *U*-преобразование для каждого выделенного из сигнала сегмента. После цикла все полученные спектральные коэффициенты, содержащиеся в массиве *Udec*, передаются в функцию getAllFeatures для формирования признакового описания сигнала.

Перейдем к синтезу общей структурной схемы программной системы. Схема приведена на рис. 5.

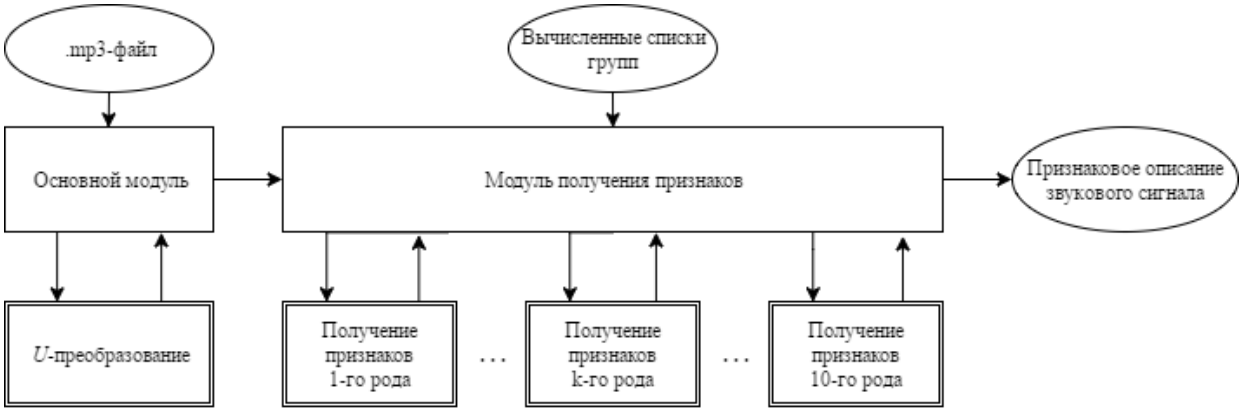


Рис.5. Общая структурная схема

Данная схема отражает разработанные модули, работу содержащихся в них функций, их взаимодействие между собой и с внешними данными, а также направление потоков данных.

Таким образом, необходимо реализовать два модуля: основной модуль, отвечающий за получение *U*-преобразования из входного сигнала, и модуль получения признаков, содержащий 10 субмодулей, каждый из которых вычисляет определенную систему признаков для всех полученных спектральных коэффициентов.

3.2 Разработка алгоритмов формирования признакового описания

Предлагается модель признакового описания [6], в которой учитываются связи между соседними сегментами звукового сигнала [2]. Метод вычисления признакового описания в таком случае состоит в формировании матрицы вероятностей переходов между описаниями соседних сегментов.

Предлагаются следующие системы признаков, основанные на матрицах вероятностей переходов:

1) система признаков, описывающая вероятности переходов между значениями операторов (оператор может принимать три возможных значения: прямое, инверсное и равное нулю), вычисленными по соседним сегментам сигнала, без учёта связей между различными операторами, размерность пространства признаков – 3 × 3 × 15 (см. рис. 6);

2) система признаков, описывающая вероятности переходов между значениями операторов, вычисленными по соседним сегментам сигнала, с учётом связи между операторами, размерность пространства признаков – 45 × 45 (см. рис. 7);

3) система признаков, описывающая вероятности переходов между описаниями сегментов, представленных в виде полных групп, размерность пространства признаков – 140×140; при использовании полных групп допустимо использовать только несколько максимальных по массе групп (граф переходов для полных групп подобен графу переходов для операторов, см. рис. 7);

4) система признаков, описывающая вероятности переходов между описаниями сегментов, представленных в виде замкнутых групп, размерность пространства признаков – 840×840; при использовании замкнутых групп допустимо использовать только несколько максимальных по массе группы.

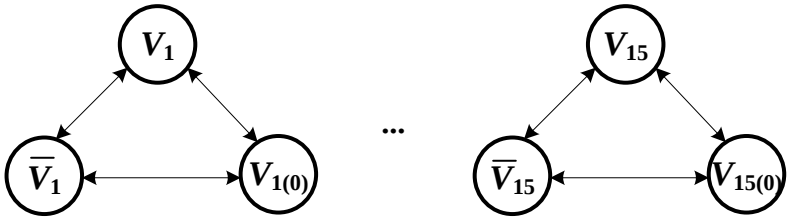


Рис 6. Графы переходов между операторами (без учёта связей между операторами)

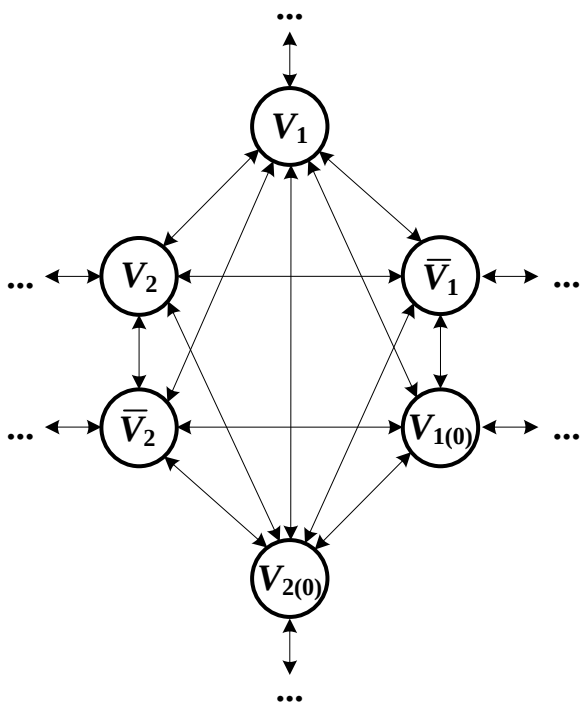


Рис 7. Графы переходов между операторами (с учётом связей между операторами)

Алгоритм формирования системы признаков *PVI* на основе операторов без учёта связей (V_i – описание *i*-го сегмента сигнала в виде операторов):

$\forall i = \overline{1, N}$
  $\forall k = \overline{1,15}$
    если $V_i(k) == -1$
      $p = 1$
    если $V_i(k) == 0$
      $p = 2$
    если $V_i(k) == +1$
      $p = 3$
    $p = p + (k-1)\cdot 3$
    если $V_{i+1}(k) == -1$
      $q = 1$
    если $V_{i+1}(k) == 0$
      $q = 2$
    если $V_{i+1}(k) == +1$
      $q = 3$
    $q = q + (k-1)\cdot 3$
    $PVI_{Desc,k}(p,q) = PVI_{Desc,k}(p,q).$

Алгоритм формирования системы признаков *PVD* на основе операторов с учётом связей между операторами:

$\forall i = \overline{1, N}$
  $\forall k = \overline{1,15}$
    если $V_i(k) == -1$
      $p = 1$
    если $V_i(k) == 0$
      $p = 2$
    если $V_i(k) == +1$
      $p = 3$
    $p = p + (k-1)\cdot 3$
    $\forall l = \overline{1,15}$
      если $V_{i+1}(k) == -1$
        $q = 1$
      если $V_{i+1}(k) == 0$
        $q = 2$
      если $V_{i+1}(k) == +1$
        $q = 3$
      $q = q + (k-1)\cdot 3$
    $PVD_{Desc}(p,q) = PVD_{Desc}(p,q).$

Алгоритм формирования системы признаков на основе полных (замкнутых групп, $Desc_i$ – описание i-го сегмента сигнала в виде полных или замкнутых групп):

$$\forall i = \overline{1, N}$$
$$\forall k = \overline{1, |Desc_i|}$$
$$\forall l = \overline{1, |Desc_{i+1}|}$$
$$MPG_{Desc_i}(Desc_i(k), Desc_{i+1}(l)) = MPG_{Desc_i}(Desc_i(k), Desc_{i+1}(l)) + 1.$$

Далее будут использоваться следующие обозначения PP_{na} (PP_{nm}) – матрица вероятностей переходов между полными группами на операции сложения (умножения), PP_c – на основе замкнутых групп.

При использовании в качестве систем признаков матриц вероятностей переходов между описаниями сегментов можно учитывать вероятности переходов не только между i и (i+1) сегментом сигнала, но и учитывать связи между большим числом сегментов. Для i-го сегмента возможен учёт не только (i+1), (i+2) и дальнейших сегментов, но и (i - 1), (i - 2) сегментов, т.е. не только «будущего», но и «прошлого».

Пример формирования признакового описания сигнала в виде матрицы вероятностей переходов размером 140×140 элементов для полных групп показан на рис. 8. При вычислении значений матрицы рассматривались связи между описания только пары соседних сегментов.

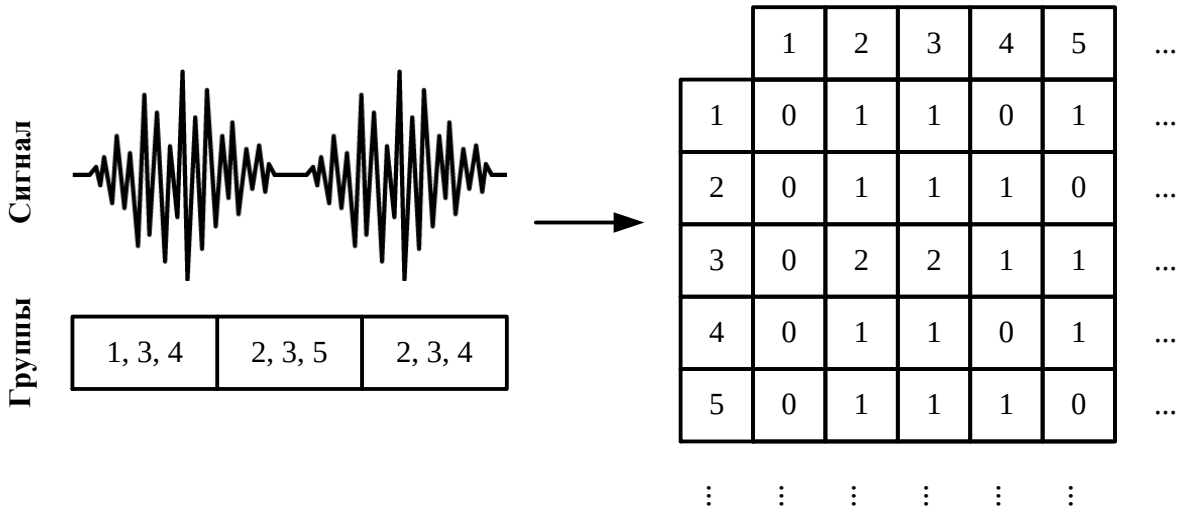
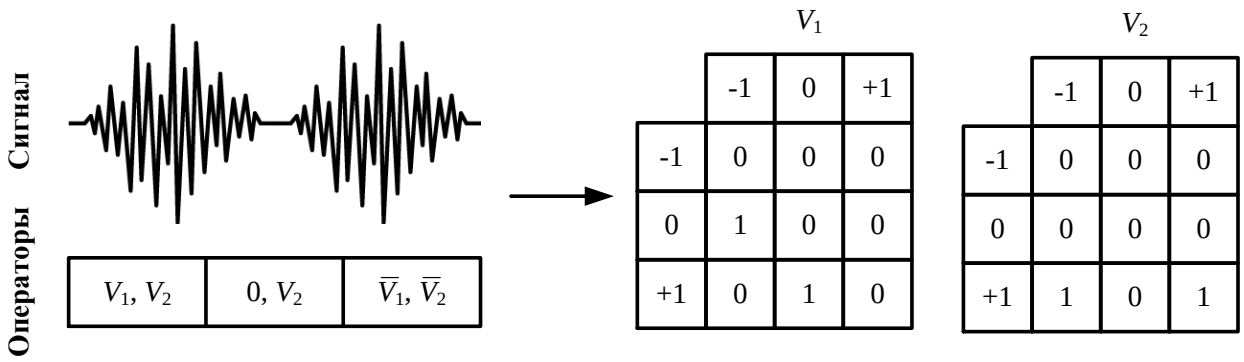


Рис. 8. Матрица вероятностей переходов между группами

Пример формирования признакового описания сигнала в виде 15 независимых матриц вероятностей переходов для операторов на показан рис. 9. При вычислении значений матрицы рассматривались связи между описания только пары соседних сегментов и учитываются вероятности переходов только для одного оператора.

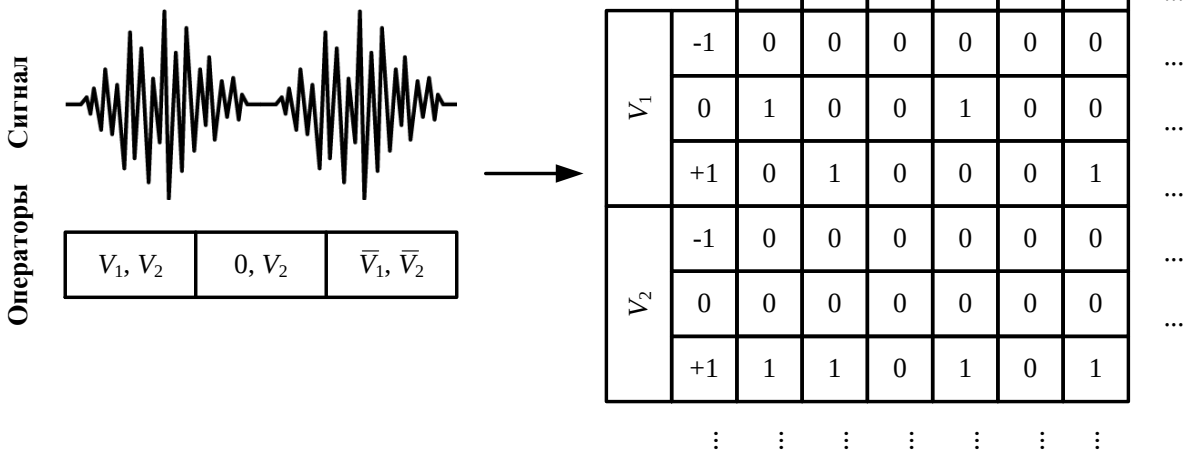


	-1	0	+1
-1	0	0	0
0	1	0	0
+1	0	1	0

	-1	0	+1
-1	0	0	0
0	0	0	0
+1	1	0	1

Рис. 9. Матрицы вероятностей переходов между операторами

Пример формирования признакового описания сигнала в виде матрицы вероятностей переходов для операторов размером 45×45 показан на рис. 10. При вычислении значений матрицы рассматривались связи между описания только пары соседних сегментов. В описаниях сегментов приведены значения только операторов V_1 и V_2.



		V_1			V_2			
		-1	0	+1	-1	0	+1	...
V_1	-1	0	0	0	0	0	0	...
	0	1	0	0	1	0	0	...
	+1	0	1	0	0	0	1	...
V_2	-1	0	0	0	0	0	0	...
	0	0	0	0	0	0	0	...
	+1	1	1	0	1	0	1	...
⋮		⋮	⋮	⋮	⋮	⋮	⋮	

Рис. 10. Матрица вероятностей переходов между операторами

В табл. 1 приведена размерность систем признаков, построенных на основе гистограмм.

Табл. 1. Число элементов в системах признаков, построенных на основе гистограмм

	Количество операторов	1*d*	2*d*	3*d*
h_V	8	14^1	14^2	14^3
	16	30^1	30^2	30^3
	32	62^1	62^2	62^3
h_{na}, h_{nm}	8	21^1	21^2	21^3
	16	140^1	140^2	140^3
	32	620^1	620^2	620^3
h_s, h_c	8	56^1	56^2	56^3
	16	840^1	840^2	840^3
	32	8680^1	8680^2	8680^3
h_{nam}	8	42^1	42^2	42^3
	16	280^1	280^2	280^3
	32	1240^1	1240^2	1240^3
h_{sc}	8	112^1	112^2	112^3
	16	1680^1	1680^2	1680^3
	32	17360^1	17360^2	17360^3

Размерность приведённых в табл. 1 систем признаков ограничена тремя измерениями, т.к. с увеличением количества используемых измерений рост числа элементов в признаковом описании сигнала описывается показательной функцией.

4 Разработка программных средств

4.1 Разработка программного интерфейса системы

Рассмотрим модули системы и их интерфейсы.

Всего было разработано 7 модулей, каждый из который представляет из себя самостоятельный .cpp-файл:

1) tap_sound.cpp – основной модуль, руководящий работой остальных модулей. Его роль заключается в подготовке входных данных (чтение .mp3-файла, выполнение *U*-преобразования) и выполнении операции извлечения признакового описания звукового сигнала из полученных после *U*-преобразования спектральных коэффициентов. Интерфейс модуля ожидает на вход путь к .mp3 файлу, для которого необходимо сформировать признаковое описание.

2) mpg123_data.cpp – служебный модуль, предназначенный для реализации чтения .mp3 файла, его нормирования, получения из него отсчетов, и предоставляющий интерфейс, с помощью которого можно извлекать из прочитанного файла сегменты сигнала, задавая итератор начала и итератор конца нужного сегмента. Модуль основан на свободной библиотеке mpg123, предназначенной для работы с аудиофайлами формата .mp3 на языке C++ [5]. Именно этот модуль используется для разбиения сигнала на заданное количество сегментов, для каждого из которых осуществляется *U*-преобразование с предварительным нормированием.

3) variables.cpp – модуль, содержащий список вычисленных групп (операторов), необходимых для вычисления признакового описания сигнала. Его интерфейс предоставляет доступ к тестовым данным, с помощью которых производилось тестирование системы на корректную работоспособность и точность вычисления признаков.

4) getfullGroup.cpp – модуль, реализующий вычисление полных групп из полученных спектральных коэффициентов. Интерфейс принимает на вход список полных групп, очередной элемент *U*-преобразования и список операторов.

5) getclosedGroup.cpp – модуль, реализующий вычисление замкнутых групп из полученных спектральных коэффициентов. Интерфейс принимает на вход список замкнутых групп, очередной элемент *U*-преобразования и список операторов.

6) getclosedGroupSets.cpp – модуль, реализующий вычисление замкнутых множеств из полученных спектральных коэффициентов. Интерфейс принимает на вход список замкнутых групп, очередной элемент *U*-преобразования и список операторов.

7) getFeatures.cpp – модуль, реализующий вычисление признакового описания звукового сигнала. Содержит в себе 10 субмодулей для вычисления 10 различных систем признаков.

В качестве хранилища для всех числовых данных будет использоваться std::vector. std::vector в языке C++ – это контейнер, элементы которого укладываются в соседние ячейки памяти, из-за чего его использование в вычислительных алгоритмах является эффективным [3,4].

4.2 Программная реализация модулей системы

Детальнее поясним особенности реализации модулей.

tap_sound.cpp содержит главную функцию main. Отметим основные этапы её работы:

```
mpg123_data mp3("D:\\file.mp3");
tap::normalize(mp3.begin(), mp3.end());
std::vector< std::vector< std::vector<data_t> > > udec_all;

auto segmCount = 287532;
auto size = mp3.get_raw_storage().size();
auto slen = size / segmCount;
```

В начале функции задается входной файл, осуществляется его нормализация, после чего подготавливается трехмерный вектор udec_all, который будет содержать все элементы *U*-преобразования.

Затем осуществляется параметризация для *U*-преобразования, которая заключается в выборе количества сегментов, на которые будет разбит входной сигнал – в итоге, количество элементов в трехмерном векторе udec_all будет совпадать с этим числом. Число 287532 было выбрано не случайно: дело в том, что длина сегмента, для которого необходимо провести *U*-преобразование, должна быть кратна 16 (это объясняется тем, что почти любую длительность с достаточной степенью точности можно аппроксимировать отрезками длиной 16). Таким образом, необходимо подобрать такое число, которое будучи делителем для количества отсчетов в сигнале давало бы частное, кратное числу 16. Для данного входного файла число отсчетов равно 9201024. Следовательно, можно вычислить ряд доступных значений для переменной segmCount – это могли быть значения 196, 10269, 63896, 191688 и так далее.

Далее следует основной цикл, получающий для каждого сегмента сигнала его *U*-преобразование:

```
auto first = mp3.begin();
for (auto it = mp3.begin() + slen; it != mp3.end(); it += slen)
{
    auto Udec_i = tap::U(first, it, filter);
    udec_all.push_back({ Udec_i.get()->responses, Udec_i.get()->part_sums
});
    first = it;
}
```

Видно, что каждый получаемый элемент заносится в вектор udec_all.

Далее этот вектор передается в модуль вычисления признакового описания звукового сигнала:

```
getAllFeatures(udec_all);
```

Далее рассмотрим модули, вычисляющие замкнутые и полные группы, замкнутые множества, и 10 субмодулей для вычисления признакового описания звукового сигнала. Для упрощения их описания введем следующие обозначения, а само описание сведем в таблицу 2:

udec – трехмерный вектор (элементы *U*-преобразования исходного сигнала)
udec[i] – очередной элемент массива udec
oper – массив из 16 булевых операторов
clsGrp – список замкнутых групп
fullGrp – список полных групп
goper2d - список совместимых операторов для матрицы
goper3d - список совместимых операторов для куба
gfullm2d - список совместимых полных групп на операции умножения
gfulls2d - список совместимых полных групп на операции сложения
gpfullm2d – список двумерных гистограмм полных групп на операции умножения
gpfulls2d - список двумерных гистограмм полных групп на операции сложения

Табл. 2. Описание вычислительных модулей

Файл	Описание модуля	Интерфейс	Возвращаемое значение
getfullGroup.cpp	вычисление полных групп	`getfullGroup(udec[i], fullGrp, oper)`	Трехмерный вектор полных групп
getclosedGroup.cpp	вычисление замкнутых групп	`getclosedGroup(udec[i] clsGrp, oper)`	Двумерный вектор замкнутых групп
getclosedGroupSets.cpp	вычисление замкнутых множеств	`getclosedGroupSets(udec[i], clsGrp, oper)`	Трехмерный вектор замкнутых множеств
getFeatures.cpp	формирование признакового описания звукового сигнала	`getFeatures_cls1d(udec, clsGrp, oper)`	Вектор признаков cls1d
		`getFeatures_full1d(udec, fullGrp, oper)`	Двумерный вектор признаков full1d
		`getFeatures_set1d(udec, clsGrp, oper)`	Вектор признаков set1d
		`getFeatures_oper2d(udec, goper2d)`	Вектор признаков oper2d
		`getFeatures_poper2d(udec)`	Вектор признаков poper2d
		`getFeatures_full2d(udec, fullGrp, oper, gfullm2d, gfulls2d)`	Двумерный вектор признаков full2d
		`getFeatures_pfull2d(udec, fullGrp, oper, gpfullm2d, gpfulls2d)`	Двумерный вектор признаков pfull2d
		`getFeatures_pcls2d(udec, clsGrp, oper)`	Вектор признаков pcls2d
		`getFeatures_poper2di(udec)`	Вектор признаков poper2di
		`getFeatures_oper3d(udec, goper3d)`	Вектор признаков oper3d

Прокомментируем также специфику модуля variables.cpp.

Он представляет из себя контейнер данных, содержащий списки вычисленных групп, интерфейс которого оптимизирован для сокращения времени компиляции: конкретный набор данных включается либо выключается из сборки системы простым определением макроса в значение «истина» либо «ложь» соответственно. Для иллюстрации этого подхода приведем часть заголовочного файла, регулирующего эти макросы:

```
#define CLS_GRP              1
#define FULL_GRP             1
#define OPER                 1
#define G_OPER_2D            0
#define G_OPER_3D            0
#define G_FULL_M_2D          0
#define G_FULL_S_2D          0
#define GP_FULL_M_2D         0
#define GP_FULL_S_2D         0
#define UDEC                 1
#define SDATA                0
#define I_MASS               0
#define FILTER               1

#if CLS_GRP
extern std::vector< std::vector<int> > clsGrp;
#endif

#if FULL_GRP
extern std::vector< std::vector<int> > fullGrp;
#endif

#if OPER
extern std::vector< std::vector<bool> > oper;
#endif

#if G_OPER_2D
extern std::vector< std::vector<int> > goper2d;
#endif

#if G_OPER_3D
extern std::vector< std::vector<int> > goper3d;
#endif
…
```

Видно, что последние два контейнера не будут включены в сборку, потому что макросы, определяющие их включение, выставлены в значение 0 («ложь»). Первые же три контейнера будут включены, потому что соответствующие им макросы выставлены в значение 1 («истина»).

Такой селективный подход позволяет сократить время компиляции, если заранее известно, какого рода систему признаков необходимо вычислить для данного сигнала.

5. Тестирование системы

5.1 Описание набора данных

Для тестирования производительности системы подготовим два набора данных: один для первичного тестирования малой размерности, и второй для заключительного тестирования – большой размерности.

Для ускорения процесса тестирования, в качестве первого набора не будет использоваться реальный сигнал. Вместо этого, создадим модель сигнала: сгенерируем массив случайных чисел, лежащих в диапазоне [0,1].

Таким образом, получим следующие характеристики для первичного набора данных:

1. Длина сигнала (количество отсчетов): 2048;
2. Длина сегмента: 32;
3. Количество элементов вектора *U*-преобразования: 32.

В качестве второго набора будем использовать реальный сигнал. Пусть это будет .mp3 файл с аудиозаписью (песней). Каждая секунда аудиозаписи содержит 44100 отсчетов.

Таким образом, получим следующие характеристики для первичного набора данных:

1. Длина сигнала (количество отсчетов): 9201024;
2. Длина сегмента: 287532;
3. Количество элементов вектора *U*-преобразования: 287532.

5.2 Описание методики тестирования

Для тестирования производительности созданной системы будем сравнивать время расчета каждой системы признаков для каждого набора данных.
К сравнению предлагаются результаты, полученные из разработанной системы, и результаты, полученные из аналогичной системы, вычисляющей те же системы признаков, но реализованной на языке программирования R.

Результаты вычисления признаков каждого рода, полученные из ранее созданной системы на языке R служили эталонными при разработке настоящей системы. Ввиду полного совпадения точности результатов вычисления систем признаков в каждой из двух имеющихся программных реализаций, нецелесообразной методикой тестирования является оценка точности вычислений результирующих данных. Поэтому выбранная методика тестирования предполагает сравнение на основе оценки производительности имеющихся реализаций – именно на эту методику мы делаем акцент в настоящей работе.

5.3 Результаты вычислительного эксперимента

Для наглядности, сведем результаты тестирования в две таблицы (табл. 3 и 4), для первого и второго набора данных соответственно.

Ввиду сильного отличия во времени вычисления признаков, полученном для разных наборов данных, для первой таблицы единицей измерения времени была выбрана миллисекунда, а для второй – секунда.

Для каждой системы признаков было произведено 10 измерений, результаты которых были усреднены и приведены в таблицах.

Табл. 3. Результаты тестирования для первого набора данных

Функция	Среднее время выполнения в R, мс.	Среднее время выполнения в C++, мс.	Коэффициент ускорения
getFeatures_set1d	10.4	2.2	5
getFeatures_oper2d	6.3	1.1	6
getFeatures_full1d	4.3	0.9	5
getFeatures_full2d	8.1	4.2	2
getFeatures_cls1d	8.5	2.8	2
getFeatures_oper3d	5.8	4.9	>2
getFeatures_poper2d	4.2	0.9	5
getFeatures_pfull2d	6.7	1.6	4
getFeatures_pcls2d	4.8	1.2	4
getFeatures_poper2di	4.6	0.2	23

Табл. 4. Результаты тестирования для второго набора данных

Функция	Среднее время выполнения в R, с.	Среднее время выполнения в C++, с.	Коэффициент ускорения
getFeatures_set1d	880.5	11.5	76
getFeatures_oper2d	61.7	8.7	7
getFeatures_full1d	243	3.8	61
getFeatures_full2d	408.5	25.6	16
getFeatures_cls1d	649.3	11.6	60
getFeatures_oper3d	205.5	35	6
getFeatures_poper2d	350	0.4	875
getFeatures_pfull2d	505.7	5.3	101
getFeatures_pcls2d	2400	17.7	135
getFeatures_poper2di	36.1	0.1	360

Как видно из полученных результатов, для первого набора данных прирост производительности был незначительным в сравнении со вторым набором данных. Такой результат обусловлен теми качествами языка C++, из-за которых он и был выбран в качестве языка для реализации данной системы.

Таким образом, для набора данных, соответствующего реальному сигналу, удалось добиться значительного прироста производительности вычислений практически для всех систем признаков.

Заключение

В результате выполнения выпускной квалификационной работы было спроектирована и программно реализована система формирования признакового описания звукового сигнала. Для реализации был выбран язык C++, а за основу алгоритма был взят подход обработки сигнала, предлагаемый в теории активного восприятия.

Таким образом, созданная программная система представляет собой эффективную и универсальную реализацию алгоритма для извлечения признакового описания из звукового сигнала. Тестирование системы подтвердило её работоспособность, возможность её использования для решения поставленной задачи, а также эффективность выбранных средств для её реализации.

Такая система может найти себе множество применений, за счет решения довольной общей задачи – её можно применять в связке с различными классификаторами к совершенно разным по своему содержанию наборам данных.

Дальнейшая работа с системой может включать в себя два пути развития, которые могут осуществляться совместно. Первый из них - это интеграция с существующими системами распознавания и обработки звуковых сигналов. Второй – рефакторинг кода, с целью оптимизации и еще большого улучшения производительности. Кроме рефакторинга, существует возможность применения к разработанной системе технологий гетерогенных вычислений, подразумевающих в использовании вычислительной мощности не только CPU, но и GPU. Такими технологиями являются CUDA и OpenCL.

Список литературы

1. "Информационный подход к описанию звукового сигнала" - Гай В.Е. -Труды МФТИ, Том 6, № 2, 2014, С. 167-173.
2. "Оценка эмоционального состояния человека по голосу с позиций теории активного восприятия" - В. Е. Гай, В. А. Утробин, П. А. Родионов, М. О. Дербасов - Системы управления и информационные технологии, №1.1(59), 2015, С. 118-122.
3. “Эффективное использование STL” – Скотт Мейерс, - “Питер”, Москва, 2002 г.
4. “Язык программирования C++. Базовый курс” - Стенли Б. Липпман, Жози Лажойе, - “Вильямс”, Москва, 2017 г.
5. mpg123 - открытая библиотека для работы с аудофайлами на C++ - https://www.mpg123.de/
6. Признаковое описание объекта - http://www.machinelearning.ru/wiki/index.php?title=ПО