МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

ОТЧЕТ

о РЕЗУЛЬТАТАХ ИНДИВИДУАЛЬНОЙ

ПРЕДДИПЛОМНОЙ практикИ

ФГАОУ ВО «Тюменский государственный университет», Кафедра программного обеспечения

(наименование организации)

|  |  |  |
| --- | --- | --- |
| Выполнил  обучающийся 4 курса,  22МОиАИС174-1 группы |  | Андреев Александр Алексеевич |
|  | (подпись) | (ФИО полностью) |
| Руководитель практики  от института / школы  к.ф-м.н., доцент |  | Ступников Андрей Анатольевич |
|  | (подпись) | (ФИО полностью) |
| Руководитель практики  от профильной организации  к.ф-м.н., доцент |  | Гаврилова Наталия Михайловна |
|  | (подпись) | (ФИО полностью) |

Защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021

Результаты

экзамена / зачета \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись руководителя практики от института / школы)

Содержание

[​ Введение 3](#__RefHeading___Toc536_1648470170)

[​ Раздел 1. 4](#__RefHeading___Toc870_1648470170)

[​ 1.1. Постановка задачи 4](#__RefHeading___Toc872_1648470170)

[​ 1.2. Анализ существующих подходов 5](#__RefHeading___Toc874_1648470170)

[​ 1.3. Преобразование Фурье 6](#__RefHeading___Toc214_2859147548)

[​ 1.4. Описание набора данных 8](#__RefHeading___Toc560_948483403)

[​ Раздел 2. 10](#__RefHeading___Toc3125_2859147548)

[​ 2.1. Описание Подхода 10](#__RefHeading___Toc3127_2859147548)

[​ 2.2. Обнаружение появлений нот 11](#__RefHeading___Toc3129_2859147548)

[​ 2.3. Определения высоты звучания 14](#__RefHeading___Toc3131_2859147548)

[​ 2.4. Алгоритм преобразования нот в табулатуры 17](#__RefHeading___Toc562_948483403)

[​ 2.5. Архитектура приложения 19](#__RefHeading___Toc564_948483403)

[​ Заключение 22](#__RefHeading___Toc2463_2785324500)

# Введение

В настоящее время с развитием информационных технологий большинство сфер начинают изменяться и расширяться в разрезе онлайн-пространства, многие предоставляемые услуги либо полноценно переходят от формат человек-человек в сторону формата человек-машина, либо добавляют взаимодействия с машиной.

Музыкальная индустрия в соответствие заданным тезисом тоже активно развивается в сторону онлайн-пространства. Описанные тенденции наблюдаются и в такой сфере как образование: появились онлайн-курсы, вебинары; в том числе в последние годы эти тенденции распространяются и на музыкальное образование [Lee, Baker, Haywood] важным аспектом которого является исполнение на музыкальном инструменте композиций.

Вследствие чего естественной оказывается необходимость в фиксации музыки и звуков, в частности, в письменной форме. Для этого используется системы музыкальных нотаций. Одним из видов такой нотации является табулатурная нотация автоматизация создания которой является целью настоящей работы.

Цель практики:

Построить модель, позволяющую построить табулатуру из аудио-сигнала

Задачи:

* Изучить существующие подходы к решению задачи извлечения структуры композиции из аудио-сигнала
* Спроектировать архитектуру приложения
* Разработать алгоритм построения табулатур из аудио-сигнала
* Программно реализовать алгоритм нахождения точек появления нот
* Программно реализовать модель извлечения высоты ноты
* Программно реализовать алгоритм переноса нотного представления в табулатуру
* Провести оценку качества работы каждого шага алгоритма

# Раздел 1. Модели цифровой обработки аудио-сигнала

## 1.1. Постановка задачи

Сигналом называется физический процесс, параметры которого изменяются в соответствии с передаваемым сообщением. Сигнал является материальным носителем информации.

Все сигналы можно разделить на четыре группы по тому, как представляются аргументы и значения сигнала: аналоговые, дискретные, квантованные и цифровые.

Для представления цифровых аудиозаписей используются дискретные сигналы, которым свойственно прерывистое (дискретное) изменение сигнала во времени. То есть изменения в сигнале происходят скачкообразно через некоторые промежутки времени, называемые интервалом дискретизации – или . Дискретизация аналогового сигнала состоит в том, что сигнал представляется в виде последовательности значений, взятых в дискретные моменты времени, которые называются отсчётами.

Исходя из этих определений получаем следующую формальную постановку задачи получения табулатуры из аудио-сигнала:

Дано:

Найти:

Однако стоит отметить, что данная работа нацелена на распознавание последовательностей, в которых в один момент времени может звучать несколько нот, например звучание аккордов. Формально это можно обозначить так:

Другое накладываемое ограничение это исключение использования мелизмов для звукоизвлечения, таких как глиссандо, легато и вибрато, как частотного, так и амплитудного.

## 1.2. Анализ существующих подходов

На сегодняшний день множество работ направлены на исследование новых подходов для экстракции внутренней структуры музыкальной композиции, в основном исследуются подходы использующие нейронные сети ввиду того, что данный подход показал большой потенциал в различных других областях.

Так Виггинс и Ким в своей работе использовали свёрточные нейронные сети на спектрограмме аудио-сигнала классифицируя частотный спектр для каждого временного отсчёта сигнала [Wiggins A. Kim Y.]. Точность этого подхода в данной работе достигает 89%, что может свидетельствовать об успешном применении подхода.

Другая похожая на эту задача, также решаемая с помощью свёрточных нейронных сетей, решалась в работе Хамфри и Белло [Humphrey, Bello]. Отличие этой работы от предыдущей в более конкретной постановке, решается проблема интерпретации конкретных аккордов, а не произвольной последовательности звучаний. В данной работе точность модели достигала 82%, что является спорным результатом в сравнении с предыдущей.

Также данный вид сетей часто применяется для разделения полифонического сигнала по разным инструментам и извлечения конкретных регистров из сигнала, в том числе для извлечения голоса из композиции [Su].

Данный подход становится всё более популярным, однако имеет значительный недостаток, заключающийся во времени обучения модели. Также стоит отметить классическую проблему, с которой приходится бороться при использовании свёрточных нейросетей это переобучение, однако при правильно построенной модели можно добиться большей точности общего решения задачи в отличии от подхода с использование преобразования Фурье. Другой проблемой этого подхода является сложность вычислений на этапе обучения нейросети соответственно требуются значительные вычислительные мощности, что налагает ограничения на используемое аппаратное обеспечение.

Небольшим ответвлением от классического подхода использующего преобразование Фурье является использование вейвлет-преобразований. Данные преобразования можно рассматривать как расширение преобразования Фурье перенимающее все его достоинства и позволяющие рассматривать только интересующие уровни разложения. Однако у данного подхода есть недостаток, заключающийся в сложности самого преобразования и естественно вытекающей из этого более низкой скорости работы, нежели в подходах с преобразованием Фурье.

Для настоящей же работы использовался классический подход к решению задач, связанных с аудио сигналами основанный на преобразовании Фурье и его модификациях. Данный подход обладает рядом преимуществ в сравнении с названными выше. Так преобразование Фурье, как было отмечено ранее, вычисляется быстрее, чем вейвлет-преобразование, благодаря оптимизированному алгоритму именуемому как быстрое преобразование Фурье (FFT). Также данный подход не требует обучения, что позволяет проводить быстрые серии экспериментов и изменять параметры конкретных шагов, не затрагивая при этом всю модель.

## 1.3. Преобразование Фурье

Наиболее распространённым вариантом представления сигнала является представление его во временной области, то есть представление его как временного ряда. Однако для большинства задач, связанных с анализом сигнала требуется его представление в частотной области, поэтому появляется необходимость в преобразовании сигнала для перехода из временной области в частотную.

Чтобы преобразовать сигнал из временной области в частотную и обратно необходимо выполнить операцию под названием прямое преобразование Фурье для дискретного, соответственно прямое дискретное преобразование Фурье (DFT).

Запишем формулу DFT для дискретной последовательности . Прямым дискретным преобразованием Фурье называется преобразование последовательности в последовательность по следующей формуле:

В этой формуле является спектральной плотностью (спектром) дискретной последовательности.

Используя формулу Эйлера , можно определить вещественную и мнимую составляющие, а также модуль и аргумент спектральной плотности, которые связаны с вещественной и мнимой частями спектра через формулы теории функции комплексного переменного.

Модуль:

Фаза:

Для преобразования из частотной области во временную используют обратное преобразование Фурье и соответственно для дискретного сигнала обратное дискретное преобразование Фурье (IDFT). IDFT есть перевод последовательности в последовательность по формуле:

где – измеренная последовательность в дискретных временных точках, значения которой являются исходными данными для прямого преобразования и выходными для обратного –последовательность комплексных амплитуд синусоидальных сигналов, образующих исходный сигнал ; значения последовательности являются выходными данными для прямого преобразования и входными для обратного

Поскольку амплитуды спектральных отсчетов - комплексные величины, то по ним можно вычислить одновременно и амплитуду, и фазу сигнала.

## 1.4. Описание набора данных

Набор данных IDMT-SMT-GUITAR — это большая база данных для автоматической транскрипции гитары. Для составления данного набора было использовано семь различных гитар со стандартной настройкой с различными настройками звукоснимателя и различными размерами струн, чтобы обеспечить достаточную диверсификацию в области электрических и акустических гитар. Записи предоставляются в одноканальном формате RIFF WAVE с частотой дискретизации 44100 Гц. Аннотации к аудио файлам сохранены в формате XML.

Набор данных состоит из четырех поднаборов. Первый содержит все введенные техники игры. Он был записан с использованием трех разных гитар и состоит из примерно 4700 нотных событий с монофонической и полифонической структурой. В частности, записанные файлы содержат реалистичные гитарные партии, от монофонических до полифонических инструментальных треков.

Второй набор данных состоит из 400 монофонических и полифонических нот, каждое из которых воспроизводится на двух разных гитарах. Здесь не применялось разграничение по стилям выразительности.

Третье подмножество состоит из пяти коротких монофонических и полифонических гитарных записей. Все пять пьес были записаны на одном и том же инструменте, и никаких особых стилей выражения не применялось.

Кроме того, включено четвертое подмножество, которое было создано для целей оценки в контексте задач распознавания аккордов и оценки стиля ритма. Этот набор содержит записи 64 коротких музыкальных произведений, сгруппированных по жанрам. Каждая пьеса была записана в двух разных темпах с тремя разными гитарами. Аннотации относительно позиций начала, аккордов, длины ритмического паттерна и текстуры включены в различные форматы файлов.

Для данной работы из всего набора данных использовалась только вторая часть, чтобы сконцентрировать работу именно на совместном определении нот и их длительностей в разрезе отрывка музыкальной композиции. В данной работе нет необходимости использовать все данные для построения объектной модели, исходя из этого выделены следующие признаки: высота ноты, время начала извлечения, время окончания извлечения для сущности ноты; номер лада, номер струны, время начала извлечения, время окончания извлечения для сущности единицы табулатуры.

# Раздел 2.

## 2.1. Описание Подхода

Для решения поставленной задачи используется следующий алгоритм:

1. Найти все точки начал звучания.
2. Для каждой точки найти высоту звучания.
3. Для каждой точки по найденной высоте звучания найти интервал звучания.
4. Сопоставить высоты звучания обозначениям используемым в MIDI.
5. Сопоставить полученным результатам табулатурное представление.

Для решения первого шага используется классический подход, использующий функции нахождения начал звучания, которые выделяют начала звучаний на фоне всего сигнала во временной области, с последующим применением алгоритма нахождения выбросов. Выброс в данном случае будет являться началом звучания ноты [Dixon].

Далее для определения высоты звучания, используется алгоритм произведения спектров гармоник (HPS), в котором для каждого временного отсчёта, производится несколько понижающих дискретизаций, полученный набор мультиплицируется и для результата применяется алгоритм нахождения выбросов.

После нахождения высоты звучания они используются для сравнения частоты для каждого последующего отсчёта до тех пор, пока частота проверяемого отсчёта окажется вне доверительного интервала. Так находятся интервалы звучания каждой ноты и соответственно длительности звучания.

На следующем этапе мы сопоставляем частоте ноты её обозначения в MIDI-формате используя небольшой доверительный интервал, так как исходно MIDI значения сопоставляется конкретным частотам, но необходимо учесть возможность небольшого отличия реальной частоты от идеальной.

Последним действием алгоритма является преобразование MIDI-подобного представления нот в табулатуры с помощью алгоритма основанного на нахождении ключа композиции.

## 2.2. Обнаружение появлений нот

Решение первого шага, то есть задачи обнаружения появлений нот, может быть представлено в виде следующего набора шагов:

1. Опционально входной сигнал предобрабатывается.
2. Далее производится редукция сигнала, при которой выделяются необходимые признаки и уменьшается общая сложность сигнала.
3. Последним шагом является нахождение выбросов в полученном сигнале, эти выбросы и будут являться временной меткой начала звучания ноты.

За всё время существования задачи определения времени появления музыкальных нот в аудиосигналах были предложены различные методы. В основном для решения данной задачи используются методы с использованием спектральных характеристик, таких как амплитуда, фаза и представление комплексной области и соответствующих им функций спектрального потока (SF), взвешенного отклонения фазы (WPD) и выпрямленная комплексная разность (RCD) [Dixon]. Также в некоторых работах используются другие функции такие как, расстояние Кульбэка-Лейблира (KLD) [Brossier], высоко частотное наполнение (HFC) [Brossier], расстояние Гото (Goto) [Goto], спектральная разность (SD) [Dixon], комплексная разность (CD) [Dixon], отклонение фазы (PD) [Dixon]. Однако существуют также вероятностные методы, которые сейчас являются предметом исследований в области обнаружения появления нот и в рамках настоящей работы не рассматриваются.

Обнаружение появления нот в данной работе реализуется с помощью классических методов, основанных на характеристиках сигнала, поэтому второй шаг решения задачи можно расписать, как применение функции обнаружения появления (ODF) к компонентам результата применения оконного преобразования Фурье (STFT) к исходному сигналу.

Функцию обнаружения появления в общем виде можно задать следующим образом:

— некоторая функция от размерности частотного спектра

— некоторая функция от номера сегмента временного спектра и номера сегмента частотного спектра.

А оконное преобразование Фурье в свою очередь определяется как:

В качестве оконной функции наиболее распространено использование окон Гаусса, Хеннинга и Хэмминга [Оппенгейм, Шаффер].

Окно Хеннинга:

Окно Хэмминга:

Окно Гаусса:

В данной работе использовались следующие функции обнаружения:

Взвешенное отклонение фазы:

Выпрямленная комплексная разность:

Модифицированное расстояние Кульбэка-Лейблира:

Расстояние Гото:

После вычисления данных функций к исходному сигналу вычисляется среднее по полученным сигналам и далее для локализации выбросов используется именно усреднённый сигнал, который формально можно обозначить как:

Следовательно задача локализации выбросов может быть обозначена как поиск множества для функции на сигнале . И для решения данной задачи каждый отсчёт сигнала проверяется на удовлетворение следующим условиям [Kehling]:

В результате экспериментов наиболее подходящими параметрами оказались: . Оценка точности работы представлена в таблице 1.

|  |  |  |
| --- | --- | --- |
| Precision | Recall | F-метрика |
| 0.96 | 0.97 | 0.95 |

Таблица 1. Оценки точности.

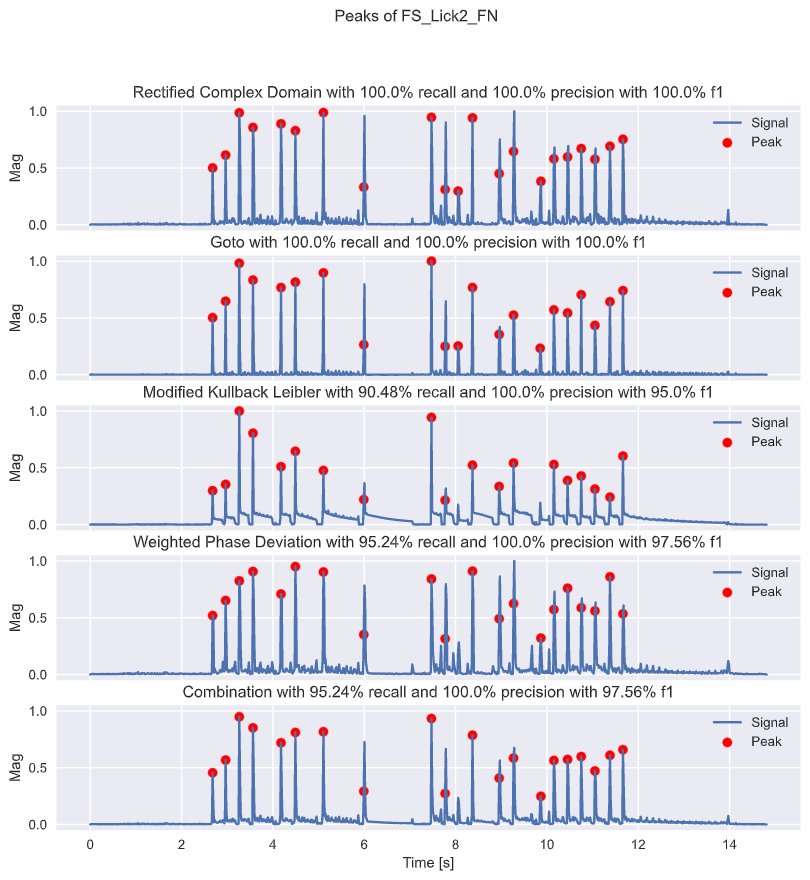


Рис.1. Результаты работы алгоритма нахождения выбросов для функций нахождения начал

## 2.3. Определения высоты звучания

Для определения высоты звучания используются подход, основанный на анализе спектрального распределения энергии и использования оконного преобразования Фурье, для этого необходимы большие размеры окна от 4096 и выше для обычного оконного преобразования Фурье (STFT), чтобы получить достаточное разрешение по частоте, которое предлагает достаточно информации для различения высоты звука в регистре основной частоты спектра гитары. В то же время большие размеры кадра значительно снижают достижимое временное разрешение, что особенно влияет на короткие ноты, и соответственно сильно влияет на конечную точность определения. Данный подход использовался в данной работе, но показал неудовлетворительные результаты. Поэтому чтобы избежать потери точности, используется следующий метод — вычисляется переназначенную спектрограмму амплитуды на основе представления мгновенной частоты (IF) в дополнение к традиционному частотно-временное преобразование. Используя информацию о фазе для частотной коррекции, IF обеспечивает высокую спектральную точность при работе с более короткими размерами кадра [Abe, Kobayashi, Imai][Kootsookos, Lovell, Boashash].

Мгновенная частота получается взятием производной от мгновенной фазы, которая определяется следующим образом:

И соответственно формула мгновенной частоты будет иметь вид:

Однако является мгновенной угловой частотой, поэтому для приведения к обыкновенной форме необходимо помножить результат дифференцирования на и итоговая формула имеет вид [Abe, Kobayashi, Imai]:

Стоит заметить, что необходимо провести развёртку фазы перед дифференцированием.

Учитывая раннее упомянутый фактор, что оконное преобразование Фурье в данном случае может помещать зафиксировать более короткие ноты, лучшим вариантом будет использование вместо преобразования Фурье – Constant-Q преобразование (CQT) [Blankertz], которое более подходит для работы с нотным представление, потому что располагает частоты логарифмически в отличии от DFT, которое располагает частоты линейно. Данное преобразование определяется следующим образом - пусть имеется оконное преобразование Фурье, тогда:

Тогда размер окна в преобразование становится переменной величиной зависимой от :

Таким образом любая оконная функция становится функцией, зависимой от длины окна, а также от номера окна. И итоговая формула CQT будет иметь вид:

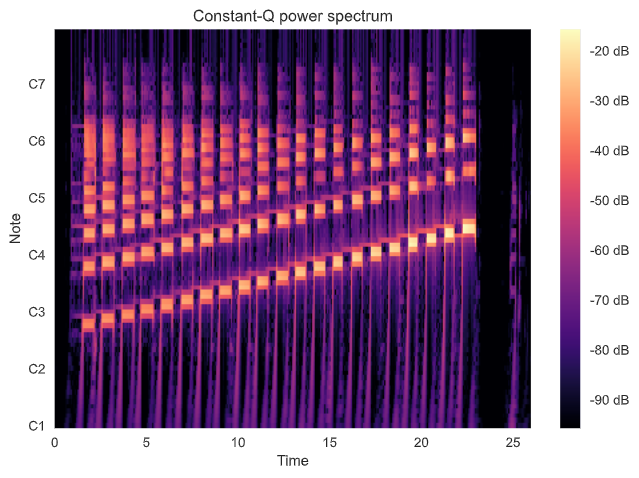


Рис. 2. Спектрограмма с использованием CQT.

Однако на практике прямая реализация данного преобразования неэффективна, поэтому для оптимизации используется тот факт, что значения оконных функций можно посчитать, не имея исходного сигнала тем самым сначала составляется матрица со значениями оконной функции. Также оптимизируется и количество памяти необходимого для хранения матрицы для этого матрицу представляют в разреженном виде [Blankertz].

К полученной спектрограмме применяется алгоритм произведения спектров гармоник, который основывается на следующем предположении - если входным сигналом является музыкальная нота, то его спектр должен состоять из серии пиков, соответствующих основной частоте с гармоническими составляющими, кратными основной частоте. Следовательно, когда мы сжимаем спектр несколько раз и сравниваем его с исходным спектром, мы можем видеть, что пики самых сильных гармоник выстраиваются в линию. Первый пик в исходном спектре совпадает со вторым пиком в спектре, сжатом в два раза, который совпадает с третьим пиком в спектре, сжатом в три раза. Следовательно, когда различные спектры умножаются вместе, в результате образуется чистый пик на основной частоте.

HPS состоит из двух этапов: понижающей дискретизации и умножения. Чтобы уменьшить частоту дискретизации, мы дважды сжимаем спектр в каждом окне путем повторной дискретизации: в первый раз мы сжимаем исходный спектр на два, а второй раз - на три. Как только это будет выполнено, мы умножаем три спектра вместе и находим частоту, соответствующую пику. Эта конкретная частота представляет собой основную частоту (F0) этого конкретного окна.

Далее найденная частота сопоставляется с соответствующим значением высоты звучания, используемой в формате MIDI. И соответственно на данном этапе данные уже готовы к сохранению в MIDI формат, что является одним из вариантов использования программы.

## 2.4. Алгоритм преобразования нот в табулатуры

После выполнения предыдущих этапов при обработке сигнала в результате получаются данные следующей структуры:

Для построения табулатуры необходимо преобразовать данную структуру к подходящему виду. Для решения данной задачи используется следующий алгоритм.

1. Ищем ключ композиции с помощью алгоритма Крумхансл-Шмуклера [Krumhansl][Schmuckler].
2. Получаем список всех нот для ключа исходя из его ладового звукоряда.
3. Для первой ноты в звукоряде получаем её расположение на струнах гитары.
4. Для каждого расположения ноты проверяем количество нот звукоряда, которые можно сыграть в интервале 4 ладов, проходя по каждой ступени звукоряда и переходя на следующую струну при выходе из интервала, запоминая их.
5. Из всех расположений первой ноты выбираем, то при котором можно сыграть больше нот, чем в других.
6. Далее для каждой ноты из исходной композиции ищем ближайшую для игры позицию проходя по каждой струне, следующим образом:
   1. Если ноту нельзя сыграть на данной струне, то пропускаем эту струну.
   2. Если данная позиция лежит в раннее рассчитанном расположении по ладовому звукоряду, то это будет искомая позиция.
   3. Иначе ищем позицию, которая будет иметь наименьшую разницу в ладах с предыдущей нотой композиции, для первого шага берётся лад тоники звукоряда.

В алгоритме поиска ключа используются два значения: одно значение координат представляет собой профиль мажорного или минорного ключа; второе общую длительность каждой высоты звучания в анализируемом музыкальном произведении.

В качестве профилей используются значения из таблицы 2 и таблицы 3.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | C# | D | D# | E | F | F# | G | G# | A | A# | B |
| 6.35 | 2.23 | 3.48 | 2.33 | 4.38 | 4.09 | 2.52 | 5.19 | 2.39 | 3.66 | 2.29 | 2.88 |

Таблица 2. Мажорный профиль.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A# | B | C | C# | D | D# | E | F | F# | G | G# |
| 6.33 | 2.68 | 3.52 | 5.38 | 2.60 | 3.53 | 2.54 | 4.75 | 3.98 | 2.69 | 3.34 | 3.17 |

Таблица 3. Минорный профиль.

Алгоритм поиска ключа вычисляет коэффициент корреляции для каждого возможного мажорного и минорного ключей, сопоставляя значения основного тона со значениями профиля для рассматриваемого ключа. Ключ с наивысшим коэффициентом корреляции используется как ключ для произведения [Krumhansl][Schmuckler].

Для набора координат (x, y) их коэффициент корреляции представляет собой число от -1 до +1, которое указывает, имеют ли координаты линейную связь. 0 означает отсутствие линейной зависимости. +1 указывает на совершенно линейную зависимость, когда при увеличении одной координаты увеличивается и другая. -1 указывает на совершенно линейную зависимость, когда одна координата увеличивается, другая уменьшается. Для координат коэффициент корреляции равен:

## 2.5. Архитектура приложения

Для того, чтобы обеспечить возможность расширения приложения в дальнейшем, при построении архитектуры использовались принципы SOLID описанные в книге Роберта Мартина «Чистая Архитектура»: принцип единственной ответственности (SRP), принцип открытости-закрытости (OCP), принцип подстановки Барбары Лисков (LSP), принцип разделения интерфейсов (ISP) и принцип инверсии зависимостей (DIP) [Мартин].

Вся архитектура разделена на 4 слоя: Domain, содержащий в себе все сущности; Model, содержащий в себе логику работы алгоритмов; I/O, инкапсулирует в себе взаимодействие с пользователем; Interaction, связывающий остальные слои инверсируя зависимости.

Слой Model подробно рассмотрен в предыдущих глава данного раздела, поэтому не нуждается в детальном рассмотрении. Слой I/O также не представляет собой особого интереса в рассмотрении, так как содержит только логику представления и преобразования считанного файла в формат пригодный для слоя Interaction.

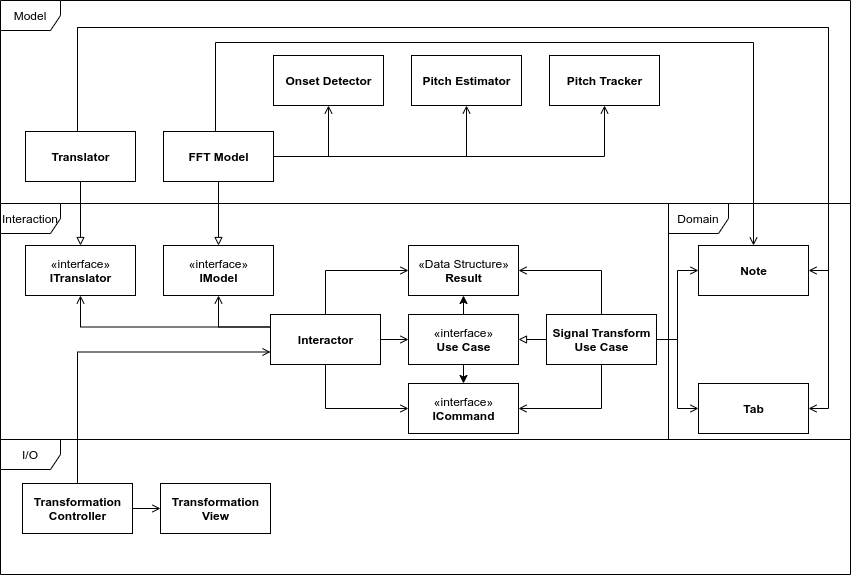


Рис 3. Архитектура приложения.

Слой Domain составлен исходя из структуры имеющегося набора данных. Из всех параметрах имеющихся в структуре аннотации, были выбраны лишь те, которые необходимы для описания предметной области и разделены на две сущности: Note и Tab. Параметры аннотаций onsetSec, offsetSec и pitch, означающие время начала, время окончания и высоту звучания соответственно, были сопоставлены сущности Note, для сущности Tab сопоставлены параметры onsetSec, offsetSec, fretNum и stringNum, означающие время начала звучания, время окончания звучания, номер лада и номер струны соответственно.

В слое Interaction определяются интерфейсы взаимодействия с компонентами, имеющими внешние зависимости, то есть для компонентов слоя Model. В данной работе определён один вариант использования пользователем, однако с помощью выделения интерфейса. Для программного описания вариантов использования используются объекты, реализующие интерфейс ICommand. Чтобы обеспечить тестируемость большинства компонентов системы при проектировании использовался подход Functional Core, Imperative Shell [Bernhardt]. Для этого при проектировании взаимодействий в работе используются монады — одна из основных концепций функциональных языков программирования, а конкретно монады Either, в данной работе Result, и Free, в данной работе ICommand и наследуемые от него классы. Это обеспечивает возможность протестировать корректность работы последовательности действий с помощью unit-тестов без уточнения деталей, таких как использование библиотек из слоя Model. Единственным компонентом архитектуры, который не может быть протестирован с помощью unit-тестов, это класс Interactor, который интерпретировать команды, получаемые от Use Case, однако он не содержит в себе сложной логики и поэтому можно без проблем протестировать его с помощью интеграционного тестирования.

# Заключение

В результате проделанной работы, были исследованы и программно реализованы функции обнаружения начал, разработан и оценён алгоритм нахождения выбросов, изучены подходы к определению высоты звучания, реализовано Constant-Q преобразование, спроектирована архитектура приложения, реализованы компоненты внутреннего взаимодействия в приложении.

# Библиографический список

1. Abe T., Kobayashi T., Imai S. Harmonics tracking and pitch extraction based on instantaneous frequency //1995 International Conference on Acoustics, Speech, and Signal Processing. – IEEE, 1995. – Т. 1. – С. 756-759.
2. Bernhardt G. Boundaries. Proceedings from SCNA 2012, Chicago, IL: Mid America Club, 2012.
3. Blankertz B. The constant Q transform //URL http://doc. ml. tu-berlin. de/bbci/material/publications/Bla\_constQ. Pdf. – 2001.
4. Brossier P. M. Automatic annotation of musical audio for interactive applications : дис. – 2006.
5. Dixon S. Onset detection revisited //Proceedings of the 9th International Conference on Digital Audio Effects. – 2006. – Т. 120. – С. 133-137.
6. Goto M. An audio-based real-time beat tracking system for music with or without drum-sounds //Journal of New Music Research. – 2001. – Т. 30. – №. 2. – С. 159-171.
7. Humphrey E. J., Bello J. P. From music audio to chord tablature: Teaching deep convolutional networks toplay guitar //2014 IEEE international conference on acoustics, speech and signal processing (ICASSP). – IEEE, 2014. – С. 6974-6978.
8. Kehling C. et al. Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score-and Instrument-Related Parameters //DAFx. – 2014. – С. 219-226.
9. Kootsookos P. J., Lovell B. C., Boashash B. A unified approach to the STFT, TFDs, and instantaneous frequency //IEEE Transactions on signal processing. – 1992. – Т. 40. – №. 8. – С. 1971-1982.
10. Krumhansl C. L. Cognitive foundations of musical pitch. – Oxford University Press, 2001 – С. 89-95.
11. Lee D. A., Baker W., Haywood N. Instrumental Teacher Education and the Incoming Tide of Information Technology: A Contemporary Guitar Perspective //Australian Journal of Teacher Education. – 2018. – Т. 43. – №. 5. – С. 17-31.
12. Naveed K., Watanabe C., Neittaanmäki P. Co-evolution between streaming and live music leads a way to the sustainable growth of music industry–Lessons from the US experiences //Technology in Society. – 2017. – Т. 50. – С. 1-19.
13. Schmuckler M. A., Tomovski R. Perceptual tests of an algorithm for musical key-finding //Journal of Experimental Psychology: Human Perception and Performance. – 2005. – Т. 31. – №. 5. – С. 1124.
14. Su L. Vocal melody extraction using patch-based CNN //2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – IEEE, 2018. – С. 371-375.
15. Wiggins A., Kim Y. Guitar Tablature Estimation with a Convolutional Neural Network //ISMIR. – 2019. – С. 284-291.
16. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – " Издательский дом"" Питер""", 2018 – С. 79-104.
17. Оппенгейм А., Шаффер Р. Цифровая обработка сигналов. Москва, Техносфера, 2012. 1048 с. ISBN 978-5-94836-329-5.

# Приложение 1. Код алгоритма нахождения начал звучания

from dataclasses import dataclass, field

from functools import reduce

from typing import Callable, Iterable, Protocol

from numpy import abs as npabs

from numpy import array, exp, log, mean, mod, ndarray, number, pi

from numpy import sum as npsum

class OnsetDetectionFunction(Protocol):

    def \_\_call\_\_(self, spectrum: ndarray, magnitude: ndarray, phase: ndarray, t: int) -> number:

        ...

@dataclass

class OnsetDetector:

    onset\_detection\_functions: Iterable[OnsetDetectionFunction] = field()

    def detect(

        self,

        peaks: ndarray,

        f\_s: int = 44100,

        t\_: float = 10e-3,

        k\_e: int = 100,

        k\_t: float = 12e-5,

        p: int = 20

    ) -> ndarray:

        tau = f\_s \* t\_

        all\_sum = sum(peaks\*\*2) / len(peaks)

        def mean\_energy(i: int) -> ndarray:

            return peaks[i]\*\*2 / tau

        def reducer(prev: tuple[list[float], int], cur: int) -> tuple[list[float], int]:

            if min(

                mean\_energy(cur - i) for i in range(1, p)

            ) < (e\_n := mean\_energy(cur)) and all\_sum < k\_e \* e\_n and k\_t \* f\_s < (cur - prev[1]):

                return [\*prev[0], cur], cur

            return prev

        empty\_list: list[float] = []

        return array(list(reduce(reducer, range(1, len(peaks)), (empty\_list, 1))[0]))

def half\_wave(arr: ndarray, norm: Callable = npsum) -> ndarray:

    return (arr + norm(arr)) / 2

def spectral\_flux(spectrum: ndarray, magnitude: ndarray, phase: ndarray, t: int) -> number:

    return npsum(half\_wave((magnitude[:, t] - magnitude[:, t - 1])\*\*2))  # type: ignore

def rectified\_complex\_domain(

    spectrum: ndarray, magnitude: ndarray, phase: ndarray, t: int

) -> number:

    return npsum(

        npabs(

            spectrum[:, t] -

            magnitude[:, t - 1] \* exp(1j \* (2 \* phase[:, t - 1] - phase[:, t - 2]))

        ) \* (magnitude[:, t] >= magnitude[:, t - 1])

    )

def weighted\_phase\_deviation(

    spectrum: ndarray, magnitude: ndarray, phase: ndarray, t: int

) -> number:

    return mean(

        npabs(

            spectrum[:, t] \*

            (mod(phase[:, t] - 2 \* phase[:, t - 1] + phase[:, t - 2] + pi, 2 \* pi) - pi)

        )

    )

def spectral\_difference(spectrum: ndarray, magnitude: ndarray, phase: ndarray, t: int) -> number:

    return npsum(npabs(magnitude[:, t]\*\*2 - magnitude[:, t - 1]\*\*2))

def high\_frequency\_content(spectrum: ndarray, magnitude: ndarray, phase: ndarray, t: int) -> number:

    return mean(magnitude[:, t]\*\*2 \* range(len(magnitude[:, t])))

def modified\_kullback\_leibler(

    spectrum: ndarray, magnitude: ndarray, phase: ndarray, t: int

) -> number:

    return npsum(magnitude[:, t] \* log(1 + magnitude[:, t] / (magnitude[:, t - 1] + 1e-7)))

def goto(spectrum: ndarray, magnitude: ndarray, phase: ndarray, t: int) -> number:

    return npsum(

        array(

            [

                max(magnitude[k, t], magnitude[k + 1, t]) \* (

                    min(magnitude[k, t], magnitude[k + 1, t]) >

                    max(magnitude[k - 1, t - 1], magnitude[k, t - 1], magnitude[k + 1, t - 1])

                ) for k in range(1,

                                 len(magnitude[:, t]) - 1)

            ]

        )

    )  # type: ignore

# Приложение 2. Код для Constant-Q преобразования

from numpy import ceil, log2, exp, pi, arange, array, sqrt

from scipy.fft import fft

from scipy.signal.windows import hamming

from scipy.sparse import coo\_matrix, vstack

def cqt\_kern(fs, bins=12, fmin=32, fmax=84\*32, window=hamming):

    K = int(ceil(bins \* log2(fmax / fmin)))

    Q = 1 / (2 \*\* (1 / bins) - 1)

    fft\_len = int(2 \*\* ceil(log2(ceil(Q \* fs / fmin))))

    res = []

    for k in range(K, 0, -1):

        N = int(ceil(Q \* fs / (fmin \* 2 \*\* ((k - 1) / bins))))

        tmp\_kernel = window(N) / N \* exp(2 \* pi \* 1j \* Q \* arange(N) / N)

        spec\_kern = fft(tmp\_kernel, fft\_len)

        spec\_kern[abs(spec\_kern) <= 0.05] = 0

        res += [coo\_matrix(spec\_kern)]

    kernel = vstack(res[::-1]).tocsc().transpose().conj() / fft\_len

    return kernel

def cqt(x, fs, hop\_size=512, bins=12, fmin=32, fmax=84\*48, window=hamming):

    kern = cqt\_kern(fs, bins, fmin, fmax, window)

    return array(range(0, len(x) - kern.shape[0], hop\_size)) / fs, array([fft(x[i:i + kern.shape[0]], kern.shape[0]) \* kern for i in range(0, len(x) - kern.shape[0], hop\_size)])