Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа киберфизических систем и управления

Работа допущена к защите
Руководитель ОП
_____ А.А. Ефремов
«___»_____2020 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ЗВУЧАНИЯ МУЗЫКАЛЬНЫХ ИНСТРУМЕНТОВ

по направлению подготовки 27.03.03 Системный анализ и управление
по образовательной программе
27.03.03_01 Теория и математические методы системного анализа и управления в технических, экономических и социальных системах

Выполнил:
студент гр.3532703/60101 А.Э. Аматов

Руководитель:
Профессор, д.т.н. А.Н. Фирсов

Консультант
по нормоконтролю
доцент, к.т.н. В.Е. Магер

Санкт-Петербург
2020

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студенту Аматову Амантуру Эшмамбетовичу, гр. 3532703/60101

1. Тема работы: «Математическая модель звучания музыкальных инструментов»

2. Срок сдачи студентом законченной работы: 29 мая 2020 г.

3. Исходные данные по работе:
Meinard Müller «Fundamentals of Music Processing»
Jeremy F. Alm, James S. Walker «Time-Frequency Analysis of Musical Instruments»
Emmanuel Amiot, «Music Through Fourier Space»

4. Содержание работы (перечень подлежащих разработке вопросов): обзор задачи разделения музыкальной композиции на две составляющие – гармоническую и перкуссионную. Знакомство с различными алгоритмами гармонического-перкуссионного разделения звука и их реализация. Сравнение алгоритмов с использованием различных методов сравнения, оценка эффективности данных алгоритмов.

5. Перечень графического материала (с указанием обязательных чертежей): Графики спектрограмм, получаемых на различных этапах реализации алгоритмов, графики, полученные в результате эксперимента.

6. Дата выдачи задания 27.04.2020

Руководитель ВКР _____ А.Н. Фирсов
(подпись)

Задание принял к исполнению 21.04.2020

Студент _____ А.Э. Аматов
(подпись)

РЕФЕРАТ

На 63 с., 31 рисунок, 3 приложения.

КЛЮЧЕВЫЕ СЛОВА: ГАРМОНИЧЕСКОЕ-ПЕРКУССИОННОЕ РАЗДЕЛЕНИЕ ЗВУКОВОГО СИГНАЛА, ОКОННОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ, МЕДИАННАЯ ФИЛЬТРАЦИЯ, ВСПОМОГАТЕЛЬНАЯ ФУНКЦИЯ, БИНАРНАЯ МАСКА, «МЯГКАЯ» МАСКА, LIBROSA, BSS_EVAL, F-МЕРА

Тема выпускной квалификационной работы: «Математическая модель звучания музыкальных инструментов».

В выпускной квалификационной работе рассмотрена задача разделения звукового сигнала на гармоническую и перкуссионную составляющую. Исследованы различные алгоритмы, позволяющие решать данную задачу. В ходе исследования был проведен сравнительный анализ предложенных алгоритмов с целью выявления более эффективного алгоритма. Задачи, решаемые в ходе исследования:

1. Изучение структуры музыкального сигнала с целью его дальнейшего разделения.

2. Изучение алгоритмов разделения звукового сигнала на гармоническую и перкуссионную составляющие.

3. Исследование задачи оконного преобразования Фурье и задачи обратного оконного преобразования Фурье.

4. Анализ и сравнение алгоритмов.

В результате были реализованы алгоритмы гармонического-перкуссионного разделения звукового сигнала с целью применения на ряде музыкальных композиций. Данные алгоритмы включают в себя алгоритм разделения звукового с помощью медианной фильтрации и с помощью создания вспомогательной функции разделения сигнала. Синтез алгоритмов проводился на базе математического моделирования с помощью программного обеспечения PyCharm на языке Python. Данные алгоритмы были протестированы с целью сравнения на выборке из 10 различных музыкальных композиций, заранее агрегированных из 10 гармонических и 10 перкуссионных частей. На основании

проведенных исследований было принято решение о большей эффективности алгоритма разделения звукового сигнала с помощью медианной фильтрации, а также были предложен вариант по улучшению алгоритма, использующего вспомогательную функцию разделения.

ABSTRACT

63 pages, 31 figures, 3 appendices.

KEYWORDS: HARMONIC-PERCUSSION SOUND SEPARATION, SHORT-TIME FOURIER TRANSFORM, MEDIAN FILTRATION, AUXILIARY FUNCTION, BINARY MASK, «SOFT» MASK, , LIBROSA, BSS_EVAL, F-MEASURE

The subject of the graduate qualification work is «Mathematical model of the sound of musical instruments»

In the graduate qualification work, the problem of dividing an audio signal into a harmonic and percussion component is considered. Various algorithms studied that allow solving this problem. During the study, a comparative analysis of the proposed algorithms carried out in order to identify a more efficient algorithm. Tasks solved during the study:

1. Study of the structure of a musical signal with a view to its further separation.
2. Study of algorithms for separating the audio signal into harmonic and percussion components.
3. Research of Short-time Fourier transform and inverse Short-time Fourier transform.
4. Analysis and comparison of algorithms.

As a result, algorithms for harmonic-percussion separation of the audio signal implemented with the aim of using them on a number of musical compositions. These algorithms include an audio separation algorithm using median filtering and by creating an auxiliary signal separation function. Algorithms synthesized based on mathematical modeling using PyCharm software in Python. These algorithms tested in order to compare on a sample of 10 different musical compositions, pre-aggregated

from 10 harmonic and 10 percussion parts. Based on the studies, it was decided that the algorithm for splitting the audio signal using median filtering is more effective, and an option was proposed to improve the algorithm using the auxiliary separation function.

СОДЕРЖАНИЕ

Введение

Что такое музыка? Для некоторых это просто способ расслабиться, для других это основной способ заработка, но в целом это нечто большее, чем просто досуг или работа. Музыка – это не только ноты на бумагах, которые воспроизводят музыканты на своих инструментах, это создание и изменение в первую очередь самих звуков.

Но что же из себя представляет музыка с физической и математической точки зрения? Музыка складывается из акустических волн, которые распространяются в воздухе вследствие колебаний давления. Термин «аудио» используется для обозначения передачи, приема или воспроизведения звуков, находящихся в пределах восприятия человеческим ухом. Физическим представлением музыки мы будем называть такое представление, в котором музыка раскладывается в понятные нам физические явления. Оно будет включать в себя временные, динамические и тональные микроотклонения, которые накладываются на основную теорию музыки, включающую такие элементы, как ритм, тон, тембр, высота звука, длительности нот.

В результате мы получаем аудио-сигнал, содержащий все эти элементы, которые не заданы явно, что усложняет обработку и анализ музыкальных произведений. Помимо всего этого имеется и другая проблема - в музыкальных произведениях чаще всего имеются больше одного инструмента, которые в итоге накладываются друг на друга. И чтобы получить каждый инструмент, требуется использовать определенные алгоритмы, чтобы определить класс инструментов и выделить каждый класс в отдельную звуковую дорожку для дальнейшего анализа. Что из себя представляет этот анализ? Что угодно – от определения темпа композиции по ударной составляющей сигнала до извлечения аккордов и нот из мелодичной составляющей сигнала. Требуется определенная предобработка данных, чтобы их анализ вышел гораздо качественнее, чем мог быть без предобработки.

Актуальность работы. Прежде чем проводить анализ музыкальных сигналов, всегда имеется шаг предобработки данных сигналов. Любая музыкальная композиция состоит из мелодичной (гармонической) части и ударной (перкуссионной) части. В работе предлагается один из методов предобработки, называемый гармоническим-перкуссионным разделением звукового сигнала. Он предполагает разделение звукового сигнала на две базовые составляющие, которые облегчают процесс анализа сигнала в виде: определение темпа композиции, определение начала звучания нот, определение основной частоты (гармоники) композиции, выявление мелодии. В ходе работы рассматриваются два алгоритма [7], [12] гармонического-перкуссионного разделения звукового сигнала. В результате исследования мы сравниваем данные два алгоритма с целью выявить лучший, а также предложить варианты по улучшению худшего алгоритма. Экспериментальное исследование выполнено путем математического моделирования данных алгоритмов на основе программного обеспечения PyCharm на языке Python.

Цель работы. Целью данной выпускной квалификационной работы является изучение метода гармонического-перкуссионного разделения звукового сигнала, изучение различных алгоритмов, предлагающих совершенно разные подходы к данной задаче. В первой главе мы рассматриваем общие понятия, связанные с обработкой музыкальных сигналов, даем определение гармоническому-перкуссионному разделению сигнала, а также формализуем постановку задачи. Во второй главе мы детально рассматриваем каждый из алгоритмов, при этом помечая проблемы, которые могут возникать на этапах алгоритмов. В третьей главе производится сравнение алгоритмов гармонического-перкуссионного разделения звукового сигнала, включая экспериментальное применение полученных результатов.

Глава 1. Обзор теоретических понятий и постановка задачи

В первой главе данной работы мы рассмотрим основные теоретические понятия, которые будут использоваться на протяжении всей работы, а также сформулируем постановку задачи и кратко опишем используемые алгоритмы.

1.1 Основные понятия о разделении музыкальных сигналов

Звуковые сигналы, как правило, представляют из себя сложные соединения различных источников звука.

Источниками звука могут быть несколько человек, говорящих одновременно в комнате, разные инструменты, играющие вместе, или человек, говорящий во время того, как играет музыка на фоне.

Разложение сложного соединения звуков на его составляющие компоненты являются одной из главных тем исследований в области науки об извлечении и обработке музыкальной информации (Music Information Retrieval). Часто эту задачу называют разделением источников звука (Source Separation). Классическая проблема данной задачи – это проблема коктейльной вечеринки (см. рисунок 1.1.1), где требуется отделить определенный звук (звук говорящего человека, определенного инструмента и т.п.) от всеобщего шума.

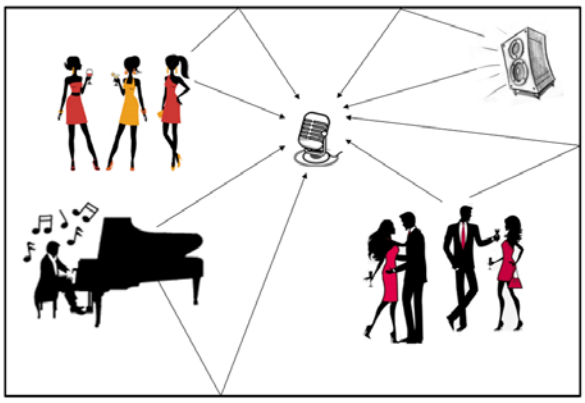


Рис.1.1.1 – Проблема коктейльной вечеринки

Если говорить конкретно про музыку, источниками звуков могут быть: мелодия, басовая или ударная партия, или голос поющего. Например, при

создании песни певец, гитарист, клавишник и барабанщик могут записывать свои партии раздельно. Уже в дальнейшем все партии сводятся в один трек, из которого уже выходит определенная песня. Задачу разделения источников звука можно охарактеризовать как обратный процесс. Имея уже готовую песню, требуется воспроизвести отдельные треки, как если бы они воспроизводились раздельно (см. рисунок 1.1.2).

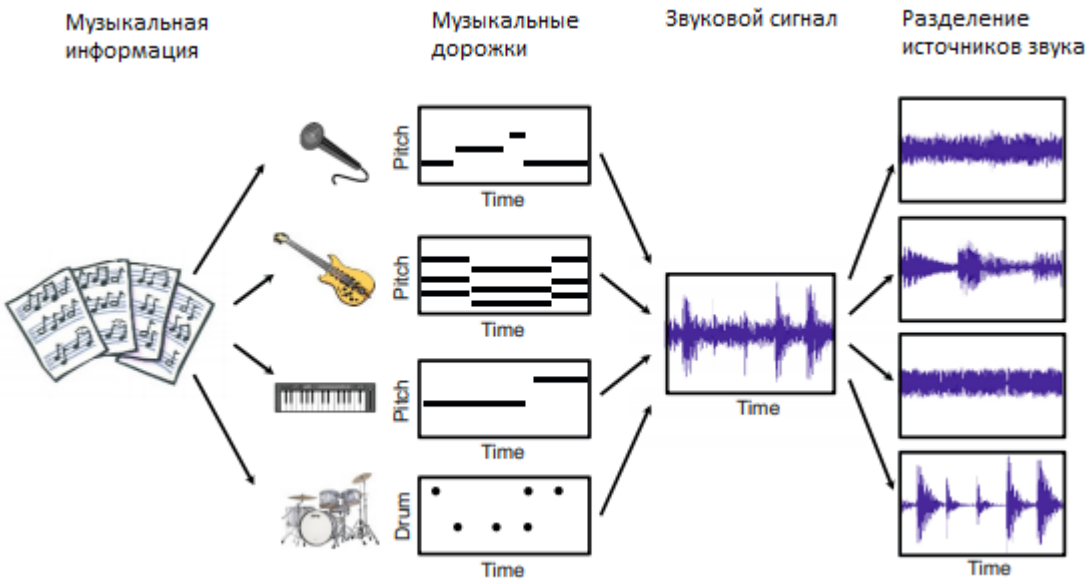


Рис.1.1.2 – Сценарий разделения звукового сигнала песни на отдельные звуковые сигналы каждого инструмента

В музыке имеются определенные техники, позволяющие добиться эффекта разделения музыкальной композиции на несколько источников звука.

При разложении музыкального сигнала требуется использовать определенные свойства, характерные только для музыкальных звуков. Для этого требуется наличие базовых знаний о музыкальной теории и как она пересекается с математикой.

Например, проблема разделения музыкальных звуков может быть смягчена тем фактом, что мелодия является ведущей партией в музыкальной композиции, характеризующейся доминирующей динамикой и временной

непрерывностью. Партию бас-гитары можно выделить, как нижнюю часть частотного спектра. Голос поющего человека может быть отделен от других источников звука благодаря определенным временно-частотным характеристикам, таким как вибрато. Партию барабанов можно выделить, используя тот факт, что большинство ее компонентов имеют перкуссионную природу, когда как остальные инструменты имеют более гармонический характер.

Основную часть работы занимает проблема разделение источников звуков на гармонические и перкуссионные составляющие. Ключевое наблюдение заключается в том, что гармонические звуки кардинально отличаются от перкуссионных на спектрограмме. Зная этот факт, возможно достаточно четко разделить одну спектрограмму смешанных звуков на пару спектрограмм, с перкуссионной и гармонической составляющей.

1.2 Понятие гармонического-перкуссионного разделения звука

Музыкальные звуки включают в себя широкий спектр звуковых компонентов с различными акустическими свойствами. Для примера стоит рассмотреть, что из себя представляет звук, издаваемый фортепиано. Для этого можно использовать спектрограмму данного звука (см. рисунок 1.2.1).

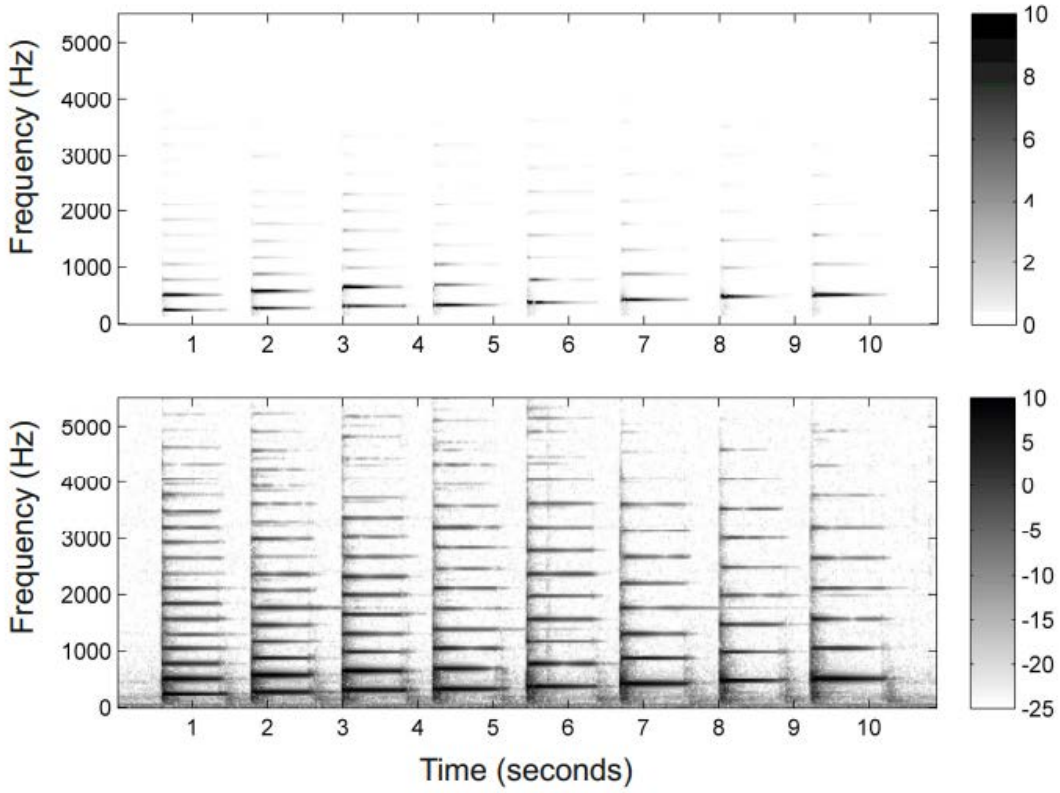


Рис. 1.2.1 – Спектрограмма аккорда Cm (до-минор), сыгранного на фортепиано в обычной и логарифмической форме, приведенной к децибелам

На данной спектрограмме можно наблюдать горизонтальные линии, которые «сложены» друг на друга. Данные линии соответствуют гармоникам, кратным основной частоте воспроизводимого аккорда Cm. Кроме того, можно наблюдать вертикальные линии в самом начале звучания ноты. Эти вертикальные линии являются частью так называемой ADSR-огибающей характеристики звука, издаваемого музыкальным инструментом, в частности пианино. Именно эта характеристика позволяет определять, какой именно инструмент играет. На огибающей выделяют четыре основных участка [14]:

1. Атака (Attack) – начальная фаза, подъем. Определяет время, нужное для того, чтобы громкость ноты достигла своего максимального уровня.

2. Спад (Decay) – фаза перехода звука в установившееся состояние.

3. Задержка (Sustain) описывает уровень звука, играемый во время удержания клавиши (после того как другие составляющие: Атака и Спад уже отыграли).

4. Затухание (Release) определяет время нужное для окончательного спада уровня ноты до нуля, после того как клавиша отпущена.

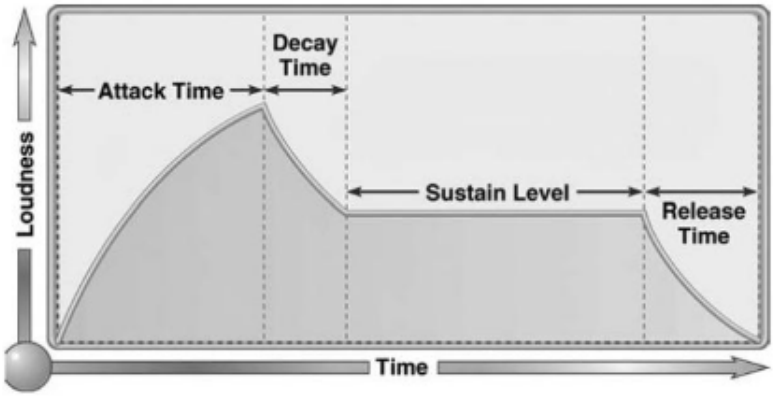
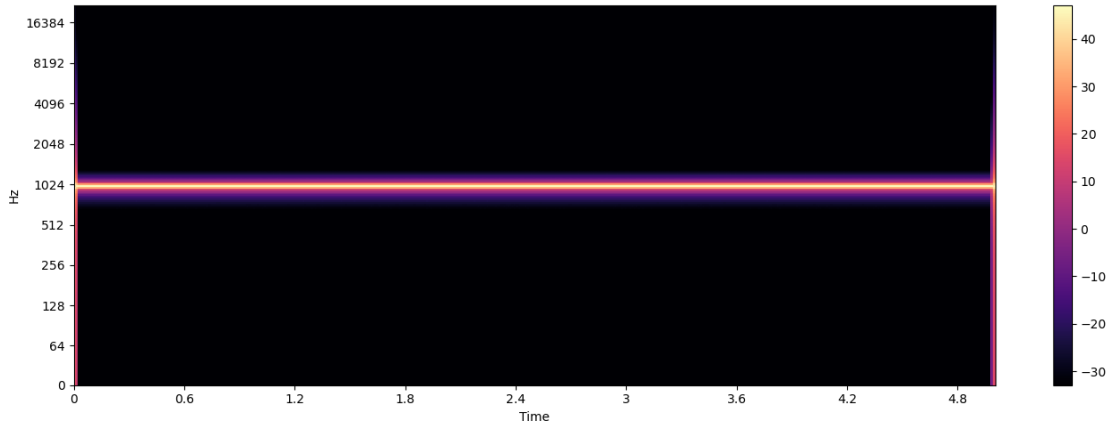


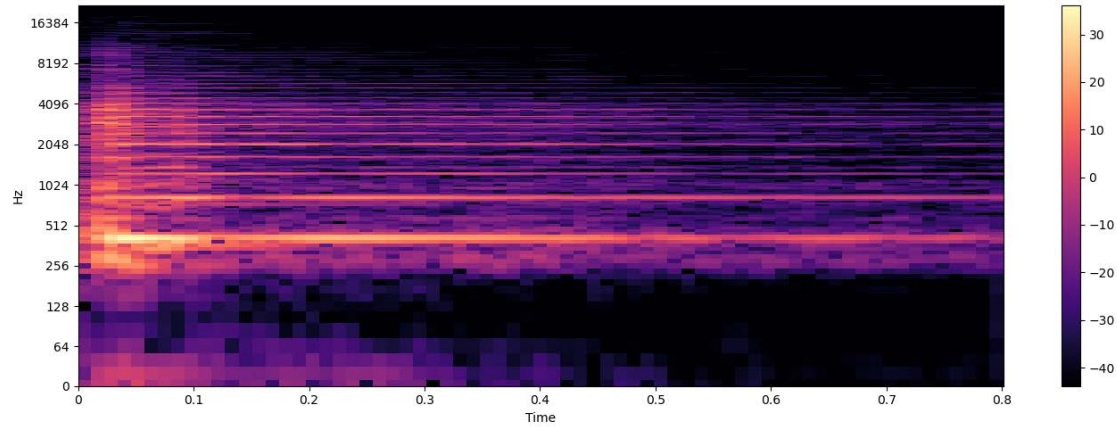
Рис. 1.2.2 ADSR-огибающая, определяющая зависимость громкости от времени звучания

Именно фаза атаки отвечает за характерные вертикальные линии на рисунке 1.2.2. Их можно трактовать как эффект, возникающий при нажатии кем-либо на клавишу фортепиано.

На основании этих наблюдений рассмотрим два типа звуковых компонентов: гармонический и перкуссионный (ударный). Если говорить простым языком, то гармонические звуки – это тональные звуки, которые позволяют нам слышать мелодию и аккорды [3]. Прототипом гармонического звука является акустическая реализация синусоиды, которая соответствует горизонтальной линии на спектрограмме (см. рисунок 1.2.3 а). Звук скрипки без вибрато – это тоже гармонический звук. Как и в случае с пианино, звук скрипки имеет свои шумы в виде обертонов над основной частотой, но тем не менее, так же представляется горизонтальными линиями на спектрограмме (см. рисунок 1.2.3 б).



(а)

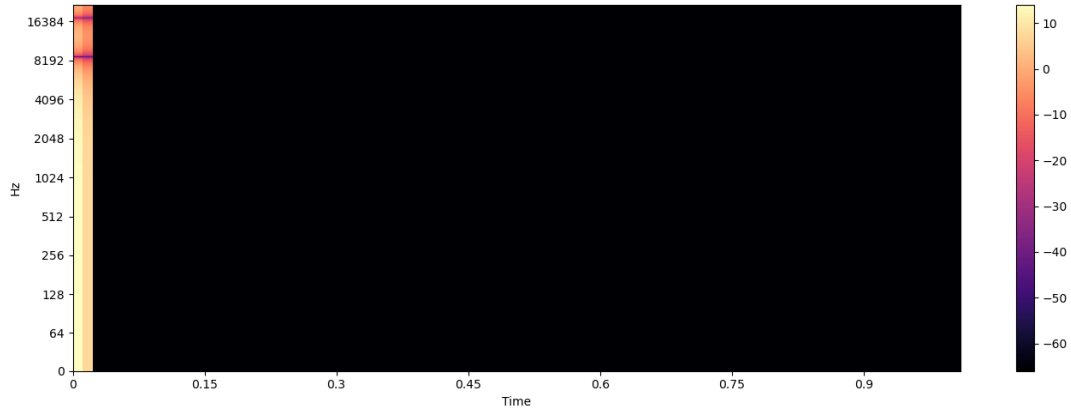


(б)

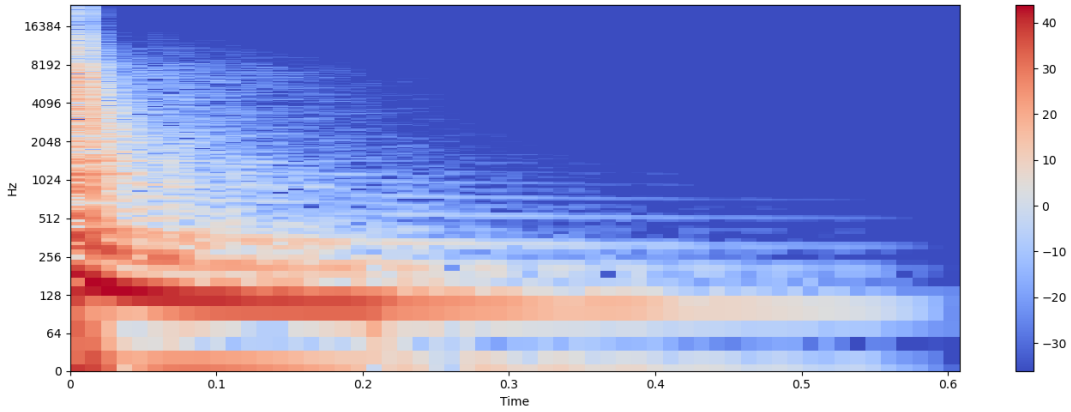
Рисунок 1.2.3 – Идеальная гармоническая синусоида(а), записанный аккорд Fm(фа-минор), сыгранная на скрипке

Перкуссионный (ударный) звук – это звук, который воспринимается человеческим ухом, как удар, стук или хлопок. Прототип перкуссионного звука – это акустическая реализация импульса, которая соответствует вертикальной линии на спектрограмме (см. рисунок 1.2.4а). В случае реального перкуссионного инструмента, как например с барабанами том-том (см. рисунок 1.2.4б), спектрограмма его удара имеет явную вертикальную линию, которая объясняется импульсом, создаваемым ударом барабана, но также помимо самого импульса имеются распределения гармонических звуков, которые объясняются тем, что в фазе атаки удар приводит к появлению музыкального тона, т.е. появлению ненулевых коэффициентов Фурье, которые распределяются на весь

частотный спектр. Важной характеристикой таких ударных звуков является то, что они не имеют высоты звука и локализованы во времени [3].



(а)



(б)

Рисунок 1.2.4 – Идеальный перкуссионный сигнал (импульс) (а), записанный удар по барабану том-том(б).

Цель гармонического-перкуссионного разделения (ГПР) звука заключается в том, чтобы разложить исходный сигнал на две части – одна состоит из гармонической составляющей, а другая из перкуссионной составляющей [11]. На самом деле задача данной разделения может быть определена немного расплывчато, так как не совсем ясна природа таких звуков, как белый шум или аплодисменты, их нельзя отнести ни к гармоническим звукам, ни к перкуссионным. Но тем не менее поставив задачу таким образом и

опустив некоторые неопределенные явления, можно рассмотреть более технический подход к решению данной задачи.

1.3 Постановка задачи

Таким образом, после определения основных понятий по данной работе, мы можем сформулировать постановку задачи:

Имеется исходный музыкальный сигнал x – сигнал с частотой дискретизации F_s, который требуется разделить на два музыкальных сигнала h и p такие, что

$$x = h + p \tag{1.3.1}$$

В общем случае процесс разделения можно описать следующим образом:

1. Расчет оконного преобразования Фурье для сигнала x с получением матрицы X размерностью m на k, где элемент матрицы $X(m,k) \in \mathbb{C}$ определяет к-ый коэффициент ряда Фурье для m-го временного шага.

2. Создание спектрограммы, исходя из полученной матрицы коэффициентов Фурье.

3. Применение выбранного алгоритма с получением двух матриц H и P размерностью m на k, где элемент матрицы $H(m,k), P(m,k) \in \mathbb{C}$ определяет к-ый коэффициент ряда Фурье для m-го временного шага для гармонической и перкуссионной составляющей соответственно.

4. Расчет обратного преобразования Фурье для матриц $H(m,k), P(m,k)$ с получением двух разделенных музыкальных сигналов h и p, удовлетворяющих формуле (1.3.1).

1.4 Используемые алгоритмы

В данной работе будут рассмотрены, сравнены и проанализированы три алгоритма гармонического перкуссионного разделения звукового сигнала.

Первый алгоритм описан в работе Дерри Фитцджеральда [7] и является основным алгоритмом, используемым библиотекой librosa для гармонического-перкуссионного разделения звука.

Мы введем процедуру ГПР, общий алгоритм которой показан на рисунке 1.4.1.

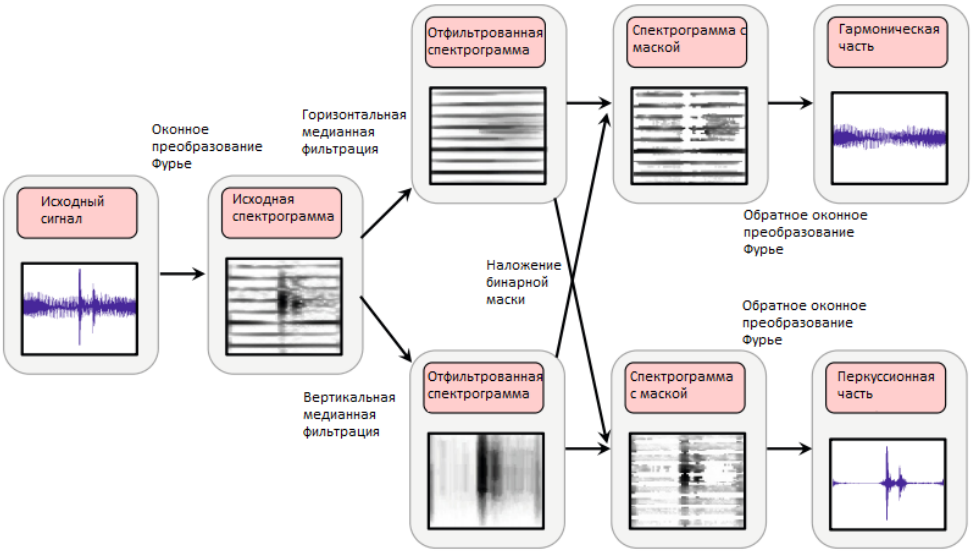


Рис. 1.4.1 – Алгоритм гармонического/перкуссионного разделения, предложенного Фитцджеральдом

Идея заключается в том, чтобы отфильтровать спектрограмму исходного сигнала в горизонтальном направлении (вдоль оси времени) для усиления гармонической части и подавлении перкуссионной части исходного сигнала. Аналогично, спектрограмма фильтруется в вертикальном направлении (по частоте) для усиления перкуссионной части и подавления гармонической части исходного сигнала. Две полученные спектрограммы используются для создания частотно-временных масок, которые применяются к исходной спектрограмме. И уже из спектрограмм с наложенными масками идет обратное оконное преобразование Фурье, которое позволяет воспроизвести уже отдельные друг от друга части исходного сигнала.

Исходя из поставленного алгоритма, у нас имеются следующие задачи, которые можно разделить на этапы:

1. Оконное преобразование Фурье исходного звукового сигнала для получения спектрограммы

2. Применение медианного фильтра для разделения исходного сигнала на гармоническую и перкуссионную составляющую

3. Создание частотно-временных масок для наложения на спектрограмму исходного сигнала

4. Восстановление разделенных сигналов из полученных спектрограмм.

Второй алгоритм описан в работе Нобутаки Оно [12]. Он первым предложил рассмотреть интуитивно спектрограмму музыкального сигнала, чтобы отнести горизонтальные линии на ней к гармонической составляющей, а вертикальные к перкуссионной. Данный алгоритм предполагает итеративную модель, которую можно описать следующим образом:

1. Оконное преобразование Фурье исходного звукового сигнала для получения спектрограммы

2. Получение амплитудной спектрограммы с сжатым диапазоном

3. Представление градиентов гармонической и перкуссионной компонент в виде нормальных распределений вдоль частотных или временных направлений

4. Минимизация нормы L_2 для амплитудной спектрограммы с учетом 3 пункта для нахождения компонент

5. Итерационное решение оптимизационной задачи с помощью добавления вспомогательной функции и параметров (используется в задачах гармонического-временного кластеризации [9] и задаче неотрицательного матричного разложения[10]

6. Создание частотно-временных масок для наложения на исходную амплитудную спектрограмму

7. Восстановление разделенных сигналов из полученных спектрограмм.

Глава 2. Реализация используемых алгоритмов

В данной главе будут рассмотрены поэтапно предложенные в первой главе алгоритмы с реализацией на языке программирования Python (в приложениях) и результатами работы данных алгоритмов.

2.1 Применение медианной фильтрации

2.1.1 Оконное преобразование Фурье исходного звукового сигнала

Преобразование Фурье несет информацию о частотах на всем временном промежутке сигнала. Однако, информация о том, когда эти частоты возникают, скрыта в самом преобразовании. Для восстановления этих данных существует метод оконного преобразования Фурье (short-time Fourier transform - STFT). Вместо анализа всего сигнала, рассматривается только его малая часть. Для этого используется оконная функция, которая не равна нулю только малый промежуток времени. После этого начальный сигнал умножают на эту функцию для получения оконного сигнала. Для получения частотной информации на разных промежутках времени оконная функция сдвигается и значение преобразования Фурье пересчитывается для каждого оконного сигнала.

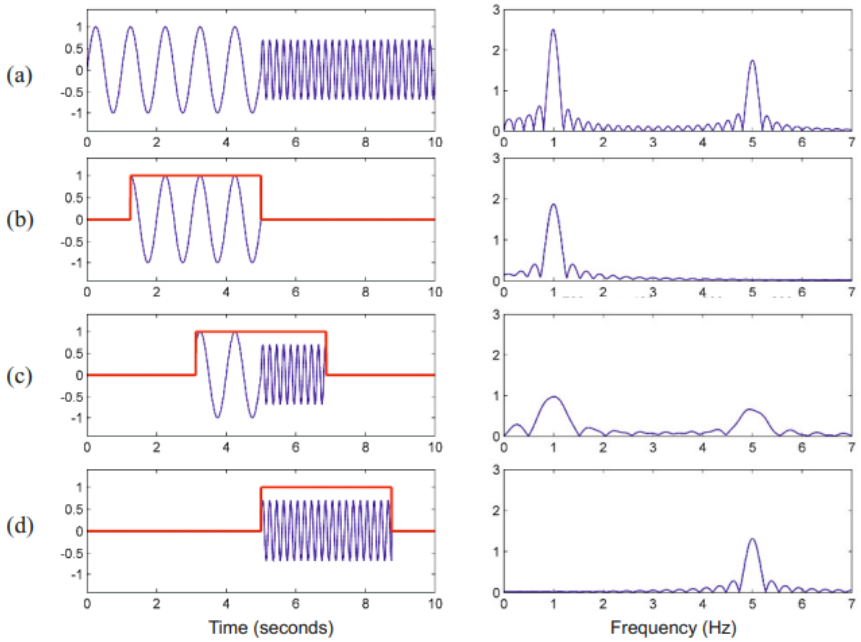


Рисунок 2.1.1.1– Сигнал и преобразование Фурье для синусоид частотой 1 Гц и 5 Гц. (a) – первоначальный сигнал, (b) – оконный сигнал для t = 3, (c) – оконный сигнал для t = 5, (d) – оконный сигнал для t=7

Пусть x – сигнал с частотой дискретизации F_s, w: [0: N-1] – оконная функция длины N. В случае прямоугольного окна w(n) = 1 для всех n ∈ [0: N-1] и w(n) = 0 во всех остальных случаях. Величина H – размер прыжка, определяющая размер шага, на который сдвигается окно. Тогда дискретное значение оконного преобразования Фурье X сигнала x имеет вид:

$$X(m,k) = \sum_{n=0}^{N-1} x(n+mH)w(n) \times e^{\frac{-2\pi ikn}{N}},$$

где m ∈ Z и k ∈ [0..K]. Число K = N/2 (предполагая, что N – четное) является частотой Найквиста.

Комплексное число $X(m,k)$ определяет к-ый коэффициент ряда Фурье для m-го временного шага. При этом каждый коэффициент связан с его физическим положением во времени:

$$T_{\text{коэф}}(m) = \frac{m \times H}{F_s},$$

при этом обычно выбирается оптимальный шаг прыжка, равный H=N/2.

Для частотного диапазона к-ый индекс оконного преобразования Фурье соответствует своей физической частоте:

$$F_{\text{коэф}}(k) = \frac{k \times F_s}{N}$$

Введем понятие спектрограммы:

$$Y(m,k) = |X(m,k)|^2$$

Она может быть иллюстрирована двухмерной картиной, где по горизонтали находится время, а по вертикали – частота. Значение спектрограммы в заданной точке характеризуется интенсивностью цвета (см. рисунок 2.1.1.2).

Рис.2.1.1.2 Пример спектрограммы отрывка песни American Football – Never Meant

2.1.2 Медианная фильтрация

В процессе разделения исходного сигнала гармоническая компонента сопоставляется горизонтальным структурам спектрограммы Y, а перкуссионная – вертикальным структурам. Пусть мы имеем частотный индекс k_0 и определим функцию $Y^{k_0}(n) = Y(n, k_0), n \in$ Z. Аналогично, возьмем временной индекс n_0 и определим функцию $Y_{n_0}(k) = Y(n_0, k), k \in Z$. Получим, что гармоническое событие некоторой частоты, соответствующей индексу k_0, приводит к всплеску функции $Y_{n_0}(k)$ на этой частоте, т.е. созданию выброса, и наоборот, перкуссионное событие в определенный момент времени создает всплеск функции $Y^{k_0}(n)$ или же созданию аналогичного выброса (см. рисунок 2.1.2.1).

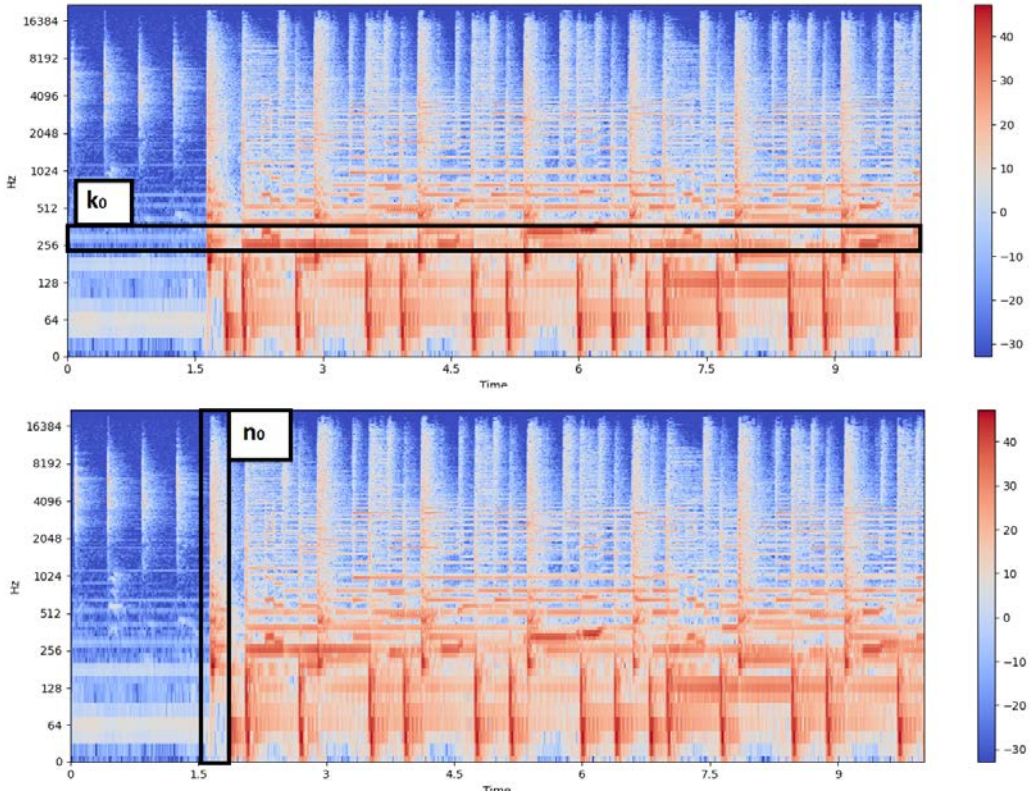


Рис. 2.1.2.1 Спектрограмма отрывка песни American Football – Never Meant с учетом функций Y^{k_0} и Y_{n_0} для фиксированного значения частотного и временного индекса k_0 и n_0

Определение всплесков, как выбросов, приводит нас к концепции медианной фильтрации, которая используется для уменьшения влияния выбросов в последовательности действительных чисел. Этот фильтр будет применяться в горизонтальных и вертикальных направлениях для уменьшения влияния гармонических и перкуссионных событий.

Медианная фильтрация – достаточно часто применяемый метод предварительной обработки сигналов. Специфической особенностью медианных фильтров является избирательность по отношению к элементам массива, представляющим собой немонотонную составляющую последовательности чисел в пределах окна (апертуры) фильтра, и резко выделяющихся на фоне соседних отсчетов. В то же время на монотонную составляющую последовательности медианный фильтр не действует, оставляя её без изменений. Благодаря этой особенности, медианные фильтры при

оптимально выбранной апертуре могут, например, сохранять без искажений резкие границы объектов, эффективно подавляя некоррелированные или слабо коррелированные помехи и малоразмерные детали. Это свойство позволяет применять медианную фильтрацию для устранения аномальных значений в массивах данных, уменьшения выбросов и импульсных помех [2].

В нашем случае с звуковым сигналом возьмем медианный фильтр длиной $L \in N$ (L мы также будем называть шириной окна фильтрации). Пусть $A = (a_n \mid n \in Z)$ – это последовательность действительных чисел $a_n \in R$, а также L нечетная. Тогда последовательность $\mu_{1/2}^{L}[A]$ определяется, как

$$\mu_{\frac{1}{2}}^{L}[A](n) = \mu_{\frac{1}{2}}(a_{n-\frac{L-1}{2}}, \dots, a_{n+\frac{L-1}{2}}),$$

где $\mu_{\frac{1}{2}}(A) = \begin{cases} \tilde{a}_{\frac{L+1}{2}}, \text{если } L \text{ нечетная} \\ \tilde{a}_{\frac{L}{2}} + \tilde{a}_{\frac{L}{2}+1}, \text{если } L \text{ четная} \end{cases}$

и $\tilde{a}$ – это элемент отфильтрованной по возрастанию последовательности A.

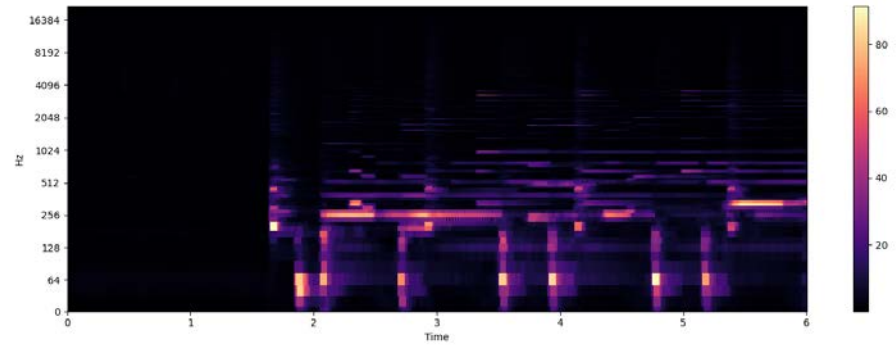
Применяя данный фильтр к спектрограмме исходного звукового сигнала Y мы получим две новые спектрограммы $\tilde{Y}^h$ и $\tilde{Y}^p$. Определим также размеры окна фильтрации для каждой спектрограммы L^h и L^p.

Получим следующие формулы для отфильтрованных спектрограмм:

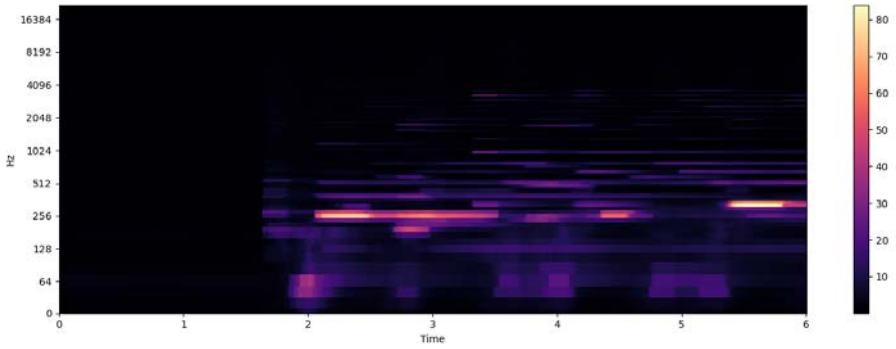
$$\tilde{Y}^h(n,k) = \mu_{\frac{1}{2}}(Y(n - \frac{L^h - 1}{2}, k), \dots, Y(n + \frac{L^h - 1}{2}, k)),$$

$$\tilde{Y}^p(n,k) = \mu_{\frac{1}{2}}(Y(n, k - \frac{L^p - 1}{2}), \dots, Y(n, k + \frac{L^p - 1}{2})),$$

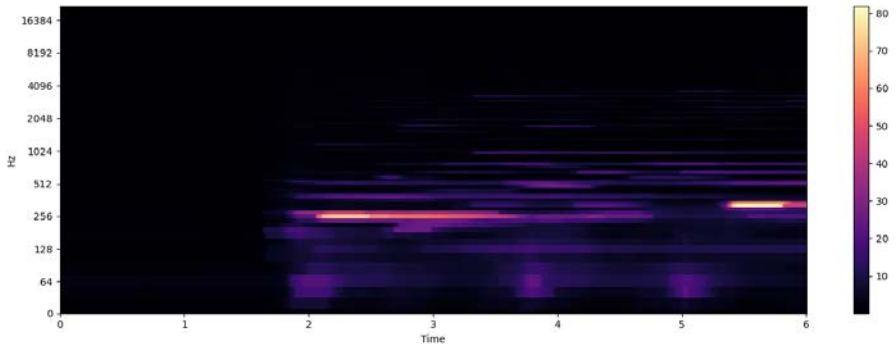
Для примера можно снова обратиться к спектрограмме из рисунка 3.1.2 и применить медианный фильтр с различными настройками размера окна фильтрации. Применяя фильтр в горизонтальном направлении, горизонтальные структуры становятся более видимыми по сравнению с вертикальными, что позволяет нам добиться эффекта вывода гармонической составляющей из исходного звукового сигнала (см. рисунок 3.2.2). Аналогично, применяя фильтр в вертикальном направлении, мы добиваемся эффекта вывода перкуссионной составляющей исходного звукового сигнала (см. рисунок 2.1.2.3).



$L^h = 9$

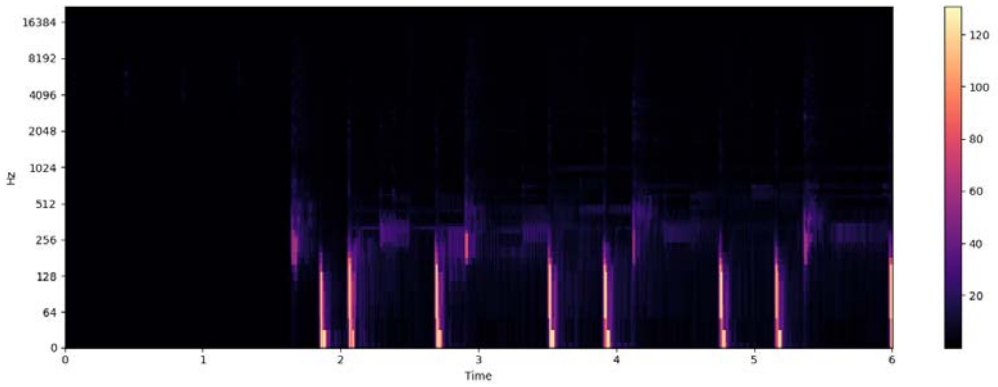


$L^h = 31$

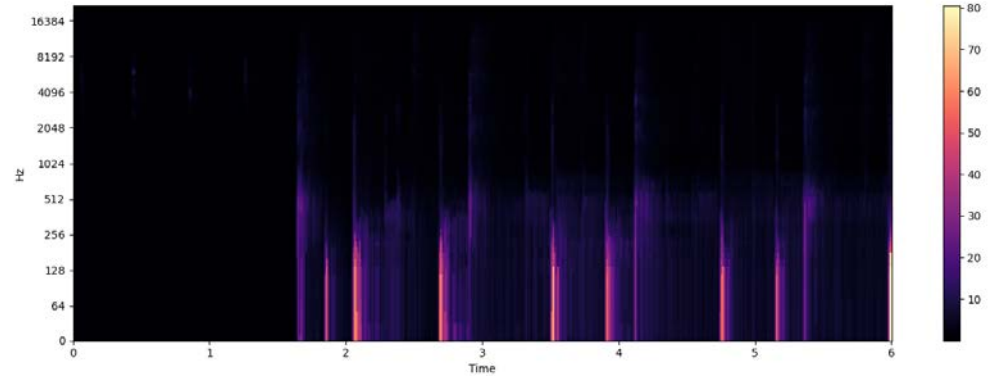


$L^h = 51$

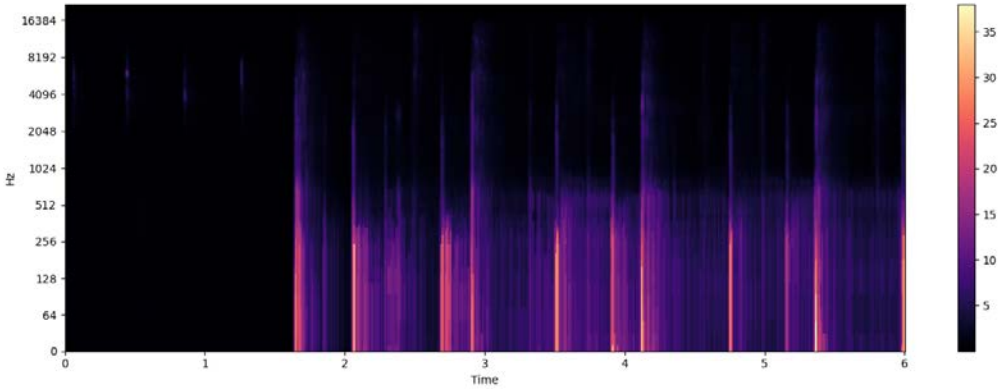
Рис. 2.1.2.2 Спектрограмма с гармонической компонентой после фильтрации с различными значениями размера окна фильтрации L^h



$L^p = 9$



$L^p = 31$



$L^p = 51$

Рис. 2.1.2.3 Спектрограмма с перкуссионной компонентой после фильтрации с различными значениями размера окна фильтрации L^p

2.1.3 Наложение частотно-временных масок

Спектрограммы $\tilde{Y}^h$ и $\tilde{Y}^p$, представленные на рисунках 2.1.2.2 и 2.1.2.3, не являются итоговыми, так как из них невозможно создать гармонические и перкуссионные составляющие исходного сигнала. Поэтому они будут использоваться для создания масок, которые уже в свою очередь будут

накладываться на исходную спектрограмму. Существует множество различных частотно-временных масок, которые можно получить из спектрограмм $\tilde{Y}^h$ и $\tilde{Y}^p$.

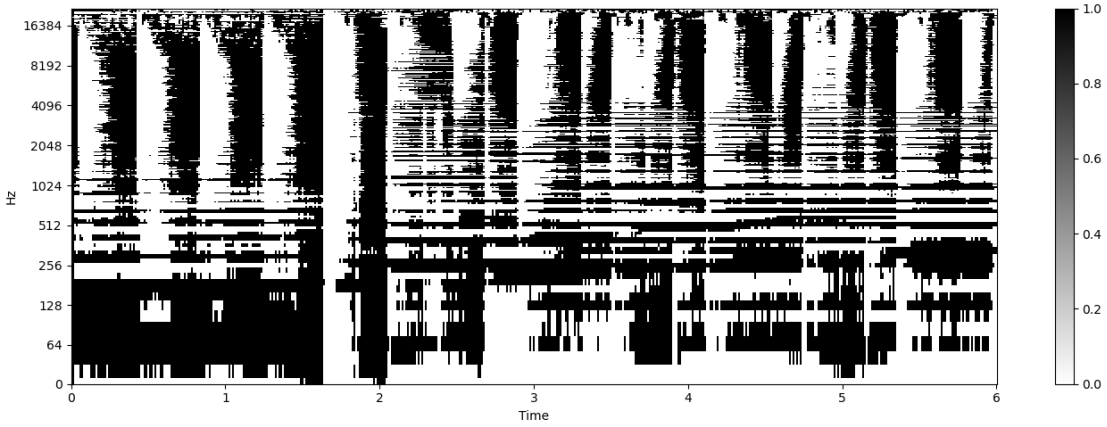
Рассмотрим первый тип частотно-временных масок, называемых бинарными масками, где каждому частотно-временному участку спектрограммы присваивается либо 1, либо 0 согласно определенным условиям. В нашем случае формула бинарной маски определяется следующим образом:

$$M^h(n,k) = \begin{cases} 1, \text{если } \tilde{Y}^h(n,k) \geq \tilde{Y}^p(n,k) \\ 0, \text{если } \tilde{Y}^h(n,k) < \tilde{Y}^p(n,k) \end{cases},$$

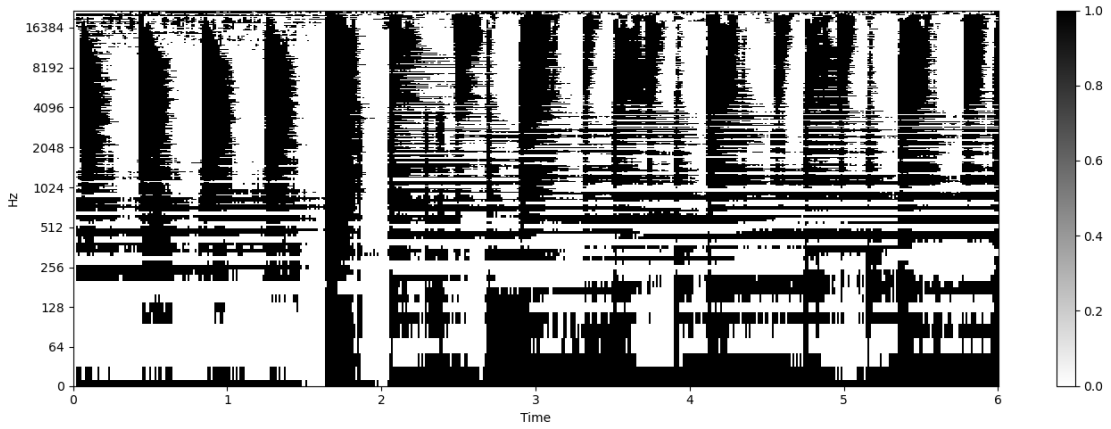
$$M^p(n,k) = \begin{cases} 1, \text{если } \tilde{Y}^h(n,k) < \tilde{Y}^p(n,k) \\ 0, \text{если } \tilde{Y}^h(n,k) \geq \tilde{Y}^p(n,k) \end{cases},$$

где n, k $\in Z$

По полученной формуле построим бинарные маски из спектрограмм гармонической и перкуссионной составляющих, полученных на предыдущем шаге (см. рисунок 2.1.3.1).



(а)



(б)

Рис.2.1.3.1 Бинарная маски, полученные из спектрограмм гармонических (а) и перкуссионных(б) составляющих.

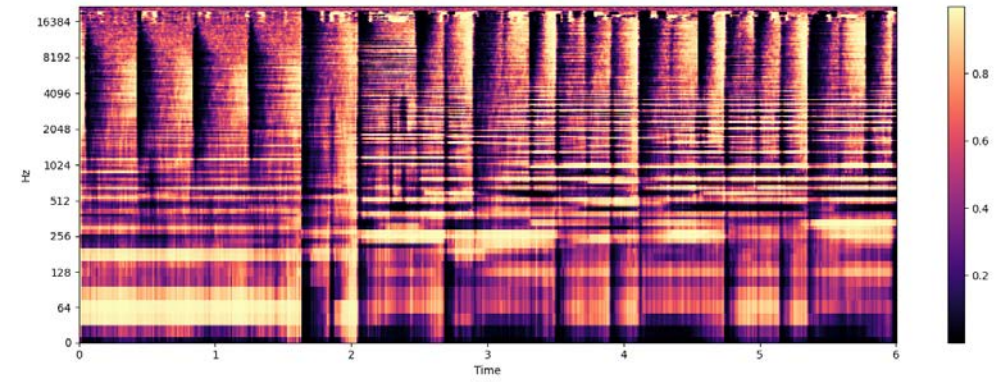
Так как данная маска относится к типу «жестких» масок, так как присваивают каждому частотно-временному участку либо 1, либо 0. Поэтому введем понятие второго типа масок – «мягкие маски». Их суть в том, чтобы сравнивать величины спектральных коэффициентов, присваивая каждому частотно-временному участку спектрограммы относительный вес. В данном случае формула «мягкой» маски будет определяться следующим образом:

$$M^h(n,k) = \frac{\tilde{Y}^h(n,k) + \frac{\varepsilon}{2}}{\tilde{Y}^h(n,k) + \tilde{Y}^p(n,k) + \varepsilon},$$

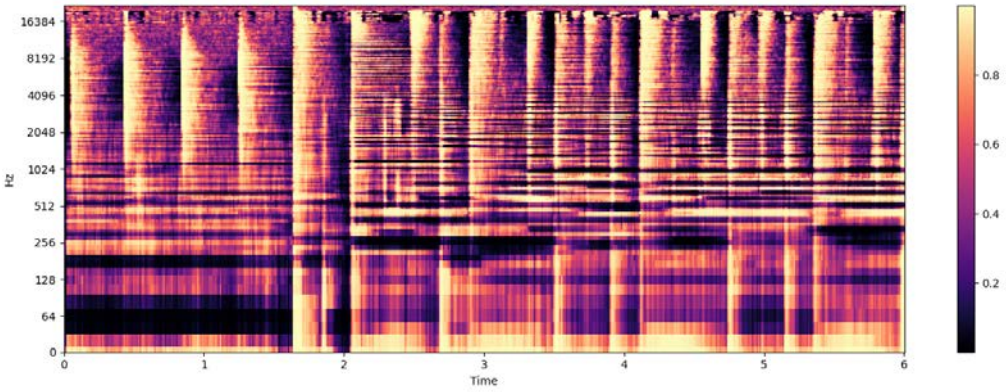
$$M^p(n,k) = \frac{\tilde{Y}^p(n,k) + \frac{\varepsilon}{2}}{\tilde{Y}^h(n,k) + \tilde{Y}^p(n,k) + \varepsilon},$$

где n, k $\in Z$ и при любом малом положительном $\varepsilon > 0$, чтобы избежать деления на ноль.

По полученной формуле построим «мягкие» маски из спектрограмм гармонической и перкуссионной составляющих, полученных на предыдущем шаге (см. рисунок 2.1.3.2).



(а)



(б)

Рис.2.1.3.2 «Мягкие» маски, полученные из спектрограмм гармонических (а) и перкуссионных(б) составляющих

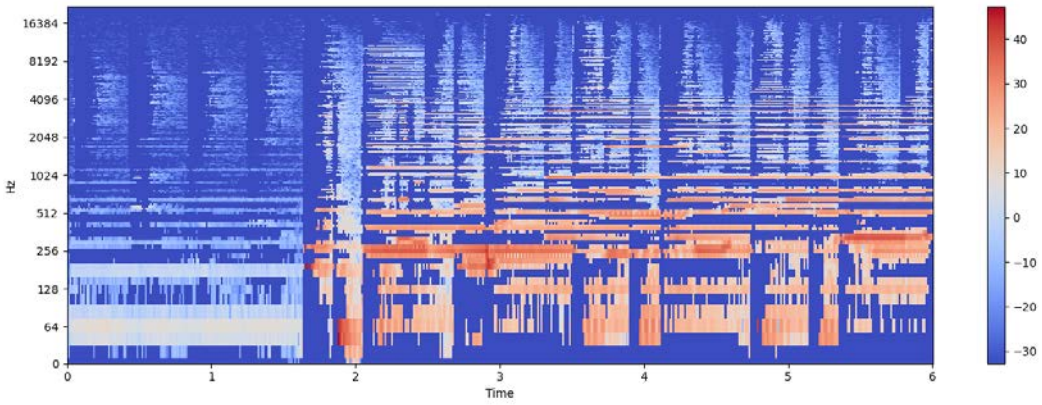
После получения данных частотно-временных масок можно накладывать их на спектрограмму исходного звукового сигнала. Суть в том, что после их поточечного наложения каждый частотно-временной участок спектрограммы будет относиться к определенной компоненте, гармонической или перкуссионной. Таким образом, мы имеем следующие формулы для получения итоговых спектрограмм.

$$Y^h = M^h(n,k) \times Y(n,k),$$
$$Y^p = M^p(n,k) \times Y(n,k),$$

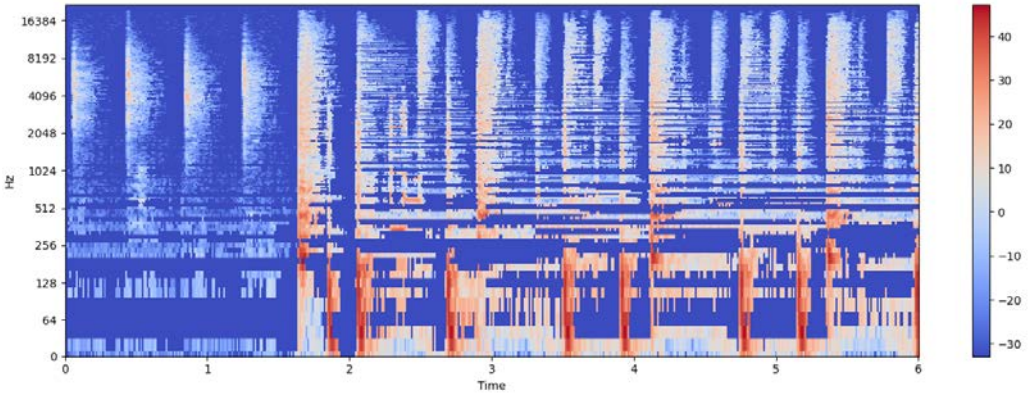
при n, k $\in Z$.

Таким образом, мы получим две спектрограммы с гармонической и перкуссионной компонентой, соответственно. Спектрограммы с наложенной бинарной маской можно увидеть на рисунке 2.1.3.3, с «мягкой» маской на

рисунке 2.1.3.4. В дальнейшем мы будем работать именно с результатом наложения «мягкой» маски на спектрограмму.

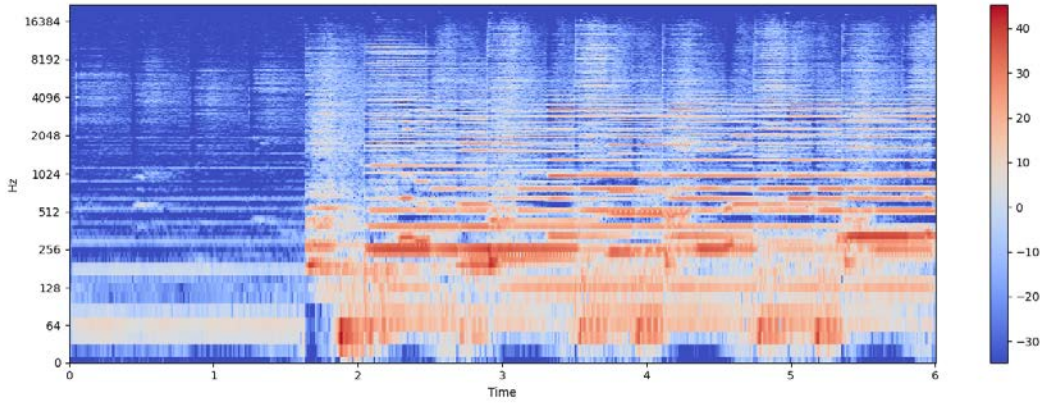


(а)

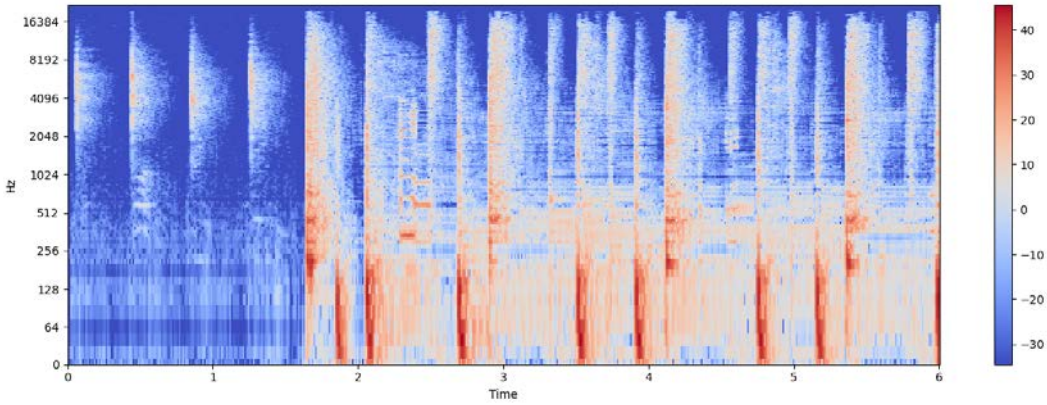


(б)

Рис. 2.1.3.3 Спектрограммы гармонической (а) и перкуссионной (б) компоненты с наложенной бинарной маской



(а)



(б)

Рис. 2.1.3.4 Спектрограммы гармонической (а) и перкуссионной (б) компоненты с наложенной «мягкой» маской.

В дальнейшем можно применять эти маски на само оконное преобразование Фурье исходного сигнала по следующим формула, чтобы в дальнейшем восстановить сигналы гармонической и перкуссионной компоненты во временной области:

$$X^h = M^h(n,k) \times X(n,k)$$
$$X^p = M^p(n,k) \times X(n,k),$$

при n, k ∈ Z.

2.1.4 Восстановление сигнала из измененной спектрограммы

Пусть x:Z → R– это дискретный сигнал, который требуется восстановить, а X – это измененное оконное преобразование Фурье, полученное на прошлых шагах. Также объявим оконную функцию w(r) длиной N ∈ N и r ∈ Z\ [0 : N −1] и

величину H – размер прыжка, определяющую размер шага, на который сдвигается окно.

Каждый коэффициент X определяется с помощью применение дискретного преобразования Фурье (DFT) на оконный сектор дискретного сигнала x. В таком случае оконный дискретный сигнал x_n будет определяться следующим образом:

$$x_n(r) = x(r + nH)w(r) \tag{3.4.1}$$

Тогда коэффициенты оконного преобразования Фурье X(n, k) при $k \in [0:N-1]$ будут определяться следующим образом:

$$(X(n,0),\dots,X(n,N-1))^T = DFT_n \cdot (x_n(0),\dots,x_n(N-1))^T \tag{3.4.2}$$

Так как матрица DFT обратима, мы можем восстановить оконный дискретный сигнал x_n следующим образом:

$$(x_n(0),\dots,x_n(N-1))^T = DFT_n^{-1}(X(n,0),\dots,X(n,N-1))^T \tag{3.4.3}$$

Определим суперпозицию всех оконных секторов сигнала:

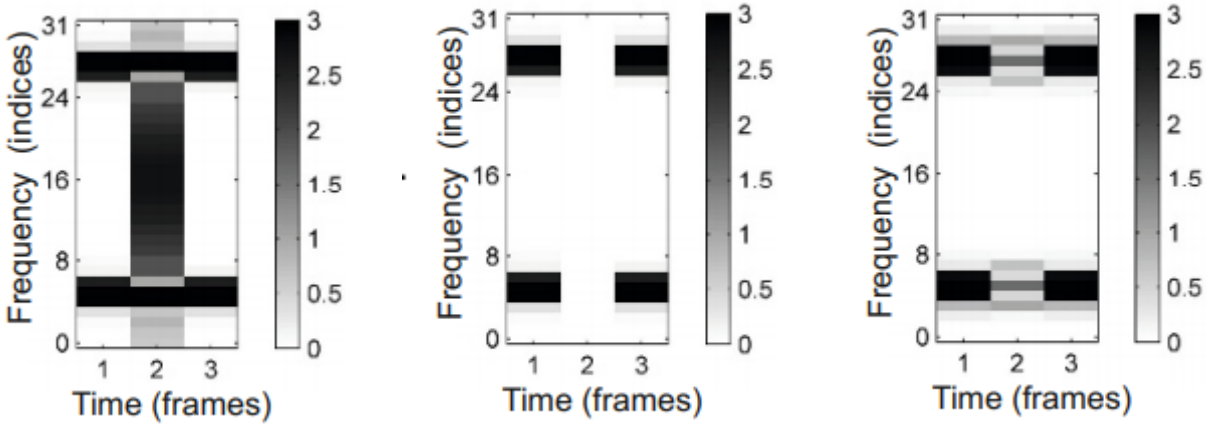
$$\begin{aligned}\sum_{n\in Z} x_n(r-nH) &= \sum_{n\in Z} x_n(r-nH+nH)w(r-nH) = \\ &= x(r)\sum_{n\in Z} w(r-nH)\end{aligned} \tag{3.4.4}$$

Таким образом, мы можем получить участок сигнала на промежутке r:

$$x(r) = \frac{\sum_{n\in Z} x_n(r-nH)}{\sum_{n\in Z} w(r-nH)}, \tag{3.4.5}$$

при условии, что $\sum_{n\in Z} w(r-nH) \neq 0$.

Но в таком случае возникает ситуация, что сигнал не будет восстанавливаться точно из измененного оконного преобразования Фурье, так как после применения масок в спектрограмме остаются пустые участки на перкуссионной части и на гармонической. Пример можно увидеть на следующем рисунке 2.1.4.1.



(а) (б) (в)

Рис. 2.1.4.1 – Пример ошибки, которая возникает при прямом использовании формулы 3.4.1.5

Как можно заметить на рисунке 3.4.1(а) показано оконное преобразование Фурье, где видна гармоническая и перкуссионная составляющие сигнала. После применения маски гармоническая часть принимает вид как на рисунке 3.4.1(б). Как можно заметить, из такой спектрограммы невозможно получить непрерывный сигнал, так как на участке 2 в принципе отсутствует информация о сигнале.

Таким образом, требуется восстановить сигнал из измененного оконного преобразования Фурье с учетом неизмененного оконного преобразования Фурье (см. рисунок 3.4.1в). Для этого определим следующие понятия:

$X^{мод}$ – измененное оконное преобразование Фурье, $X^{исх}$ – исходное оконное преобразование Фурье, $\Delta(X^{мод},X^{исх})$ – среднеквадратичная ошибка, определяемая по формуле:

$$\Delta(X^{\text{мод}}, X^{\text{исх}}) = \sum_{n \in Z} \sum_{k \in [0:N-1]} |X^{\text{мод}}(n,k) - X^{\text{исх}}(n,k)|^2 \qquad (3.4.6)$$

Требуемая задача – это найти такой сигнал x^*, чье оконное преобразование Фурье минимизировала ошибку между измененным и исходным оконным преобразованием Фурье, т.е.:

$$x^*(r) = argmin_x \Delta(X^{\text{мод}}, X^{\text{исх}}) \qquad (3.4.7)$$

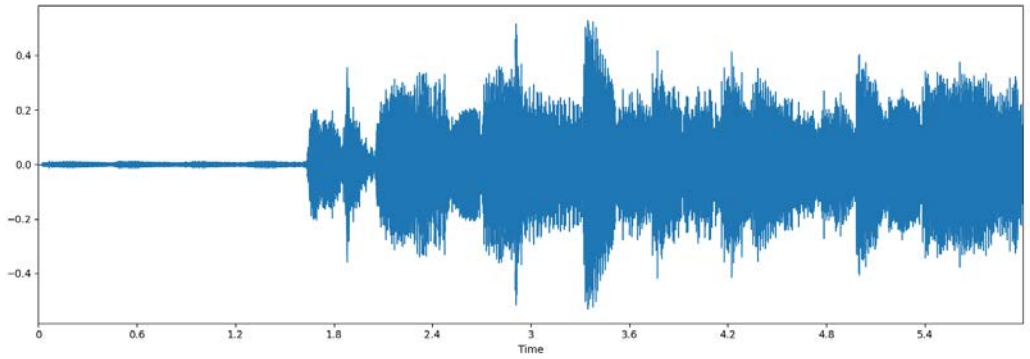
Решение данной оптимизационной задачи имеет вид:

$$x^*(r) = \frac{\sum_{n \in Z} w(r - nH) x_n(r - nH)}{\sum_{n \in Z} w(r - nH)^2} \quad (3.4.8)$$

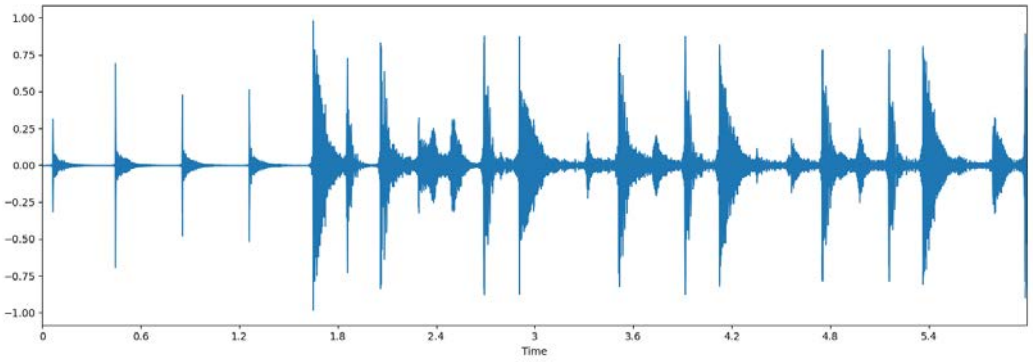
Полное решение данной задачи описано в [8].

Таким образом, мы смогли получить гармоническую и перкуссионную компоненты из измененного преобразования Фурье. Непрерывные формы волны перкуссионной и гармонической составляющей спектрограммы уже знакомой музыкальной композиции представлены на рисунке 2.1.4.2.

Полная реализация данного алгоритма представлена в Приложении 1.



(а)



(б)

Рис.2.1.4.2 – Формы волны гармонической (а) и перкуссионной (б) составляющих, восстановленных из измененного оконного преобразования Фурье

2.2 Применение вспомогательной функции разделения

В данной части мы рассмотрим алгоритм, предложенный Нобутакой Оно в [12]. Его работа является одной из первых, в которой Оно обнаруживает тот факт, что гармоническая составляющая исходного музыкального сигнала представляет из себя горизонтальные линии на спектрограмме, когда как перкуссионная часть представляет из себя вертикальные.

В своей работе он предлагает минимизировать функцию потерь (которая исходит из L_2 нормы для амплитудной спектрограммы исходного сигнала) для нахождения гармонической и перкуссионной компонент $H_{h,i}, P_{h,i}$:

$$J(H,P) = \frac{1}{2\sigma_H^2}\sum_{h,i}\left(H_{h,i-1} - H_{h,i}\right)^2 + \frac{1}{2\sigma_p^2}\sum_{h,i}\left(P_{h-1,i} - P_{h,i}\right)^2, \qquad (2.2.1)$$

с ограничениями вида:

$$P_{h,i} + H_{h,i} = W_{h,i}$$

$$H_{h,i} \geq 0, P_{h,i} \geq 0$$

где H, P это матрицы, состоящие из компонент $H_{h,i}, P_{h,i}$, соответственно, σ_H, σ_p - это параметры, контролирующие гладкость получаемых вертикальных и горизонтальных компонент, а $W_{h,i}$ это амплитудная спектрограмма исходного сигнала. Минимизация (2.2.1) эквивалентна методу максимального правдоподобия с предположением, что градиенты спектрограммы $\left(H_{h,i-1} - H_{h,i}\right)$ и $\left(P_{h-1,i} - P_{h,i}\right)$ подчиняются нормальному распределению. Но опытным путем было выяснено, что градиенты не совсем подчиняются этому правилу, вследствие чего стоит изменить амплитудную спектрограмму на ее версию со сжатым диапазоном $\widetilde{W_{h,i}} = |F_{h,i}|^{2\gamma}$ ($0 < \gamma < 1$), чтобы заполнить разрыв между предположением и реальной ситуацией.

2.2.1 Создание вспомогательной функции

Так как функция из формулы (2.2.1) является квадратичной для всех переменных, она имеет один глобальный минимум, который может быть получен путем решения системы дифференциальных уравнений:

$$\begin{cases} \dfrac{\partial J}{\partial H_{h,i}} = 0 \\ \dfrac{\partial J}{\partial P_{h,i}} = 0 \end{cases} \tag{2.2.1.1}$$

Но так как учитывается каждый частотно-временной участок спектрограммы (h,i), и количество уравнений будет равно количеству данных участков, требуется найти более простое решение, например, с применением вспомогательной функции.

Прежде чем создавать вспомогательную функцию, примем то, что

$$(A-B)^2 \leq 2(A-X)^2 + 2(B-X)^2 \tag{2.2.1.2}$$

для любых A,B и X, т.к.

$$(A-X)^2 + 2(B-X)^2 - (A-B)^2 = 4(X - \frac{A+B}{2})^2 \geq 0$$

Применяя формулу (2.2.1.2) к формуле (2.2.1), получим:

$$(H_{h,i-1} - H_{h,i})^2 \leq 2\left(H_{h,i-1} - U_{h,i}\right)^2 + 2(H_{h,i} - U_{h,i})^2$$

$$(P_{h-1,i} - P_{h,i})^2 \leq 2\left(P_{h-1,i} - V_{h,i}\right)^2 + 2\left(P_{h,i} - V_{h,i}\right)^2,$$

для любых $V_{h,i}$ и $U_{h,i}$

Равенство выполняется при $U_{h,i} = \frac{H_{h,i-1}+H_{h,i}}{2}$ и $V_{h,i} = \frac{P_{h-1,i}+P_{h,i}}{2}$. Таким образом, можно определить вспомогательную функцию:

$$\begin{aligned} Q(H,P,U,V) = \\ = \frac{1}{2\sigma_H^2}\sum_{h,i}((H_{h,i-1} - U_{h,i})^2 + (H_{h,i} - U_{h,i})^2) + \\ + \frac{1}{2\sigma_p^2}\sum_{h,i}((P_{h-1,i} - V_{h,i})^2 + (P_{h,i} - V_{h,i})^2), \end{aligned} \tag{2.2.1.3}$$

удовлетворяющее условиям:

$$J(H,P) \leq Q(H,P,U,V)$$

$$J(H,P) = \min_{U,V} Q(H,P,U,V)$$

Таким образом, мы получим:

$$\left\{U^{(k+1)}, V^{(k+1)}\right\} = \min_{U,V} Q\left(H^{(k)}, P^{(k)}, U, V\right), \tag{2.2.1.4}$$

$$\left\{H^{(k+1)}, P^{(k+1)}\right\} = \min_{H,P} Q\left(H, P, U^{(k+1)}, V^{(k+1)}\right), \quad (2.2.1.5)$$

где k – это количество итераций алгоритма.

2.2.2 Вывод гармонической и перкуссионной составляющей

Требуется решить уравнение из формулы (2.2.1.5). Для этого будем использовать метод множителей Лагранжа, так как требуется найти условный экстремум функции $Q(H, P)$. Получим:

$$\hat{Q}(H, P) = Q\left(H, P, U^{(k+1)}, V^{(k+1)}\right) + \sum_{h,i} \lambda_{h,i}(H_{h,i} + P_{h,i} - W_{h,i})$$

Дифференцируя данное выражение по $\lambda_{h,i}, H_{h,i}, P_{h,i}$, получим:

$$\begin{cases} H_{h,i} + P_{h,i} - W_{h,i} = 0 \\ \frac{2}{\sigma_H^2}\left(2H_{h,i} - U_{h,i+1}^{(k+1)} - U_{h,i}^{(k+1)}\right) + \lambda_{h,i} = 0 \\ \frac{2}{\sigma_P^2}\left(2P_{h,i} - V_{h,i+1}^{(k+1)} - V_{h,i}^{(k+1)}\right) + \lambda_{h,i} = 0 \end{cases}$$

Решая данную систему уравнений, получим

$$H_{h,i}^{(k+1)} = \frac{\alpha}{2}\left(U_{h,i+1}^{(k+1)} + U_{h,i}^{(k+1)}\right) + \frac{1-\alpha}{2}(2W_{h,i} - V_{h-1,i}^{(k+1)} - V_{h,i}^{(k+1)})$$

$$P_{h,i}^{(k+1)} = \frac{1-\alpha}{2}\left(V_{h-1,i}^{(k+1)} + V_{h,i}^{(k+1)}\right) + \frac{\alpha}{2}\left(2W_{h,i} - U_{h,i+1}^{(k+1)} - U_{h,i}^{(k+1)}\right),$$

где $\alpha = \frac{\sigma_P^2}{\sigma_P^2 + \sigma_H^2}$

Так как вспомогательные параметры $V^{(k+1)}, U^{(k+1)}$ удовлетворяют условиям (2.2.1.4), их можно выразить через H и P.

$$U_{h,i}^{(k+1)} = \frac{(H_{h,i-1}^{(k)} + H_{h,i}^{(k)})}{2}, V_{h,i}^{(k+1)} = \frac{(P_{h-1,i}^{(k)} + P_{h,i}^{(k)})}{2}$$

Таким образом, после замены мы можем убрать вспомогательные параметры для нахождения гармонической и перкуссионной компонент.

$$H_{h,i}^{(k+1)} = H_{h,i}^{(k)} + \Delta^{(k)} \quad (2.2.2.1)$$

$$P_{h,i}^{(k+1)} = P_{h,i}^{(k)} - \Delta^{(k)} \quad (2.2.2.2)$$

где $\Delta^{(k)} = \alpha(\frac{(H_{h,i-1}^{(k)} - 2H_{h,i}^{(k)} + H_{h,i+1}^{(k)}}{4}) - (1-\alpha)(\frac{P_{h-1,i}^{(k)} - 2P_{h,i}^{(k)} + P_{h+1,i}^{(k)}}{4})$ (2.2.2.3)

2.2.3 Пошаговая реализация алгоритма

Используя информацию, полученную в предыдущих уравнениях, распишем реализацию данного алгоритма по шагам (код, реализующий данный алгоритм имеется в Приложении 2, основан на программе из [12]).

1. Рассчитать $F_{h,I}$ – оконное преобразование Фурье для входного сигнала f(t)

2. Рассчитать амплитудную спектрограмму со сжатым диапазоном $\widetilde{W_{h,\iota}} = |F_{h,i}|^{2\gamma}$ ($0 < \gamma < 1$)

3. Обозначить начальные значения для гармонической и перкуссионной компонент $H_{h,i}^{(0)} = P_{h,i}^{(0)} = \frac{1}{2}\widetilde{W_{h,\iota}}$ для всех h,i, k=0.

4. Рассчитать $\Delta^{(k)}$ согласно формуле (2.2.2.3)

5. Рассчитать $H_{h,i}^{(k+1)}$ и $P_{h,i}^{(k+1)}$ как

$$H_{h,i}^{(k+1)} = min(max(H_{h,i}^{(k)} + \Delta^{(k)}, 0), \widetilde{W_{h,\iota}})$$

$$P_{h,i}^{(k+1)} = \widetilde{W_{h,\iota}} - H_{h,i}^{(k+1)}$$

6. Увеличивать k, пока $k < k_{max}$. k_{max} – максимальное число итераций.

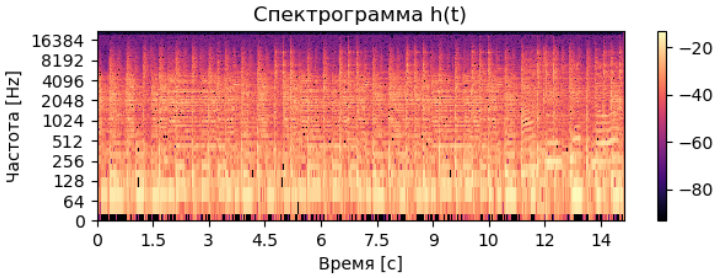
7. Разделить амплитудную спектрограмму с помощью полученных матриц (бинарная маска)

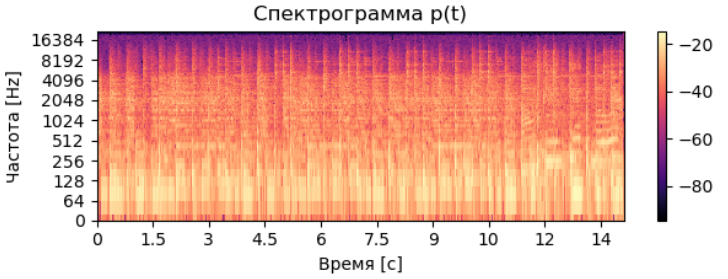
$$H_{h,i} = \text{Если } (H_{h,i} < P_{h,i}), \text{ТО } 0, \text{ИНАЧЕ } \widetilde{W_{h,\iota}}$$

$$P_{h,i} = \text{Если } (H_{h,i} \geq P_{h,i}), \text{ТО } \widetilde{W_{h,\iota}}, \text{ИНАЧЕ } 0$$

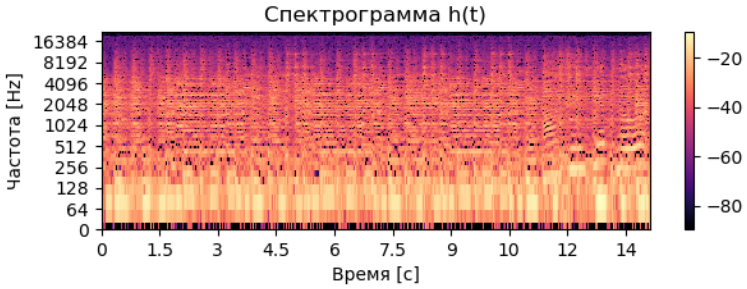
8. Рассчитать обратное оконное преобразование Фурье для $H_{h,i}, P_{h,i}$ с получением двух сигналов h(t) и p(t).

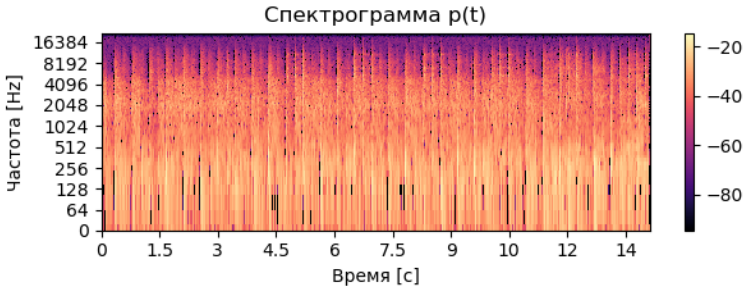
Ниже на рисунке 2.2.3.1 можно ознакомиться с результатами работы программы при различных значениях k=0,50,75 и после окончания работы алгоритма.


Спектрограмма h(t)
Частота [Hz]
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
12
14
Время [c]
−20
−40
−60
−80

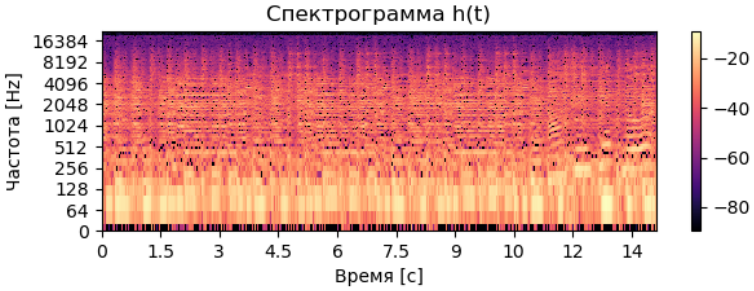

Спектрограмма p(t)
Частота [Hz]
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
12
14
Время [c]
−20
−40
−60
−80

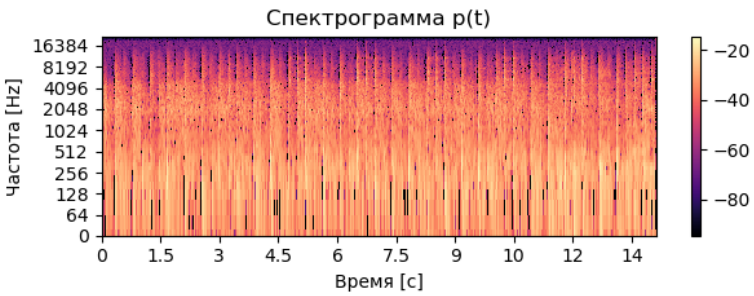
k=0


Спектрограмма h(t)
Частота [Hz]
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
12
14
Время [c]
−20
−40
−60
−80

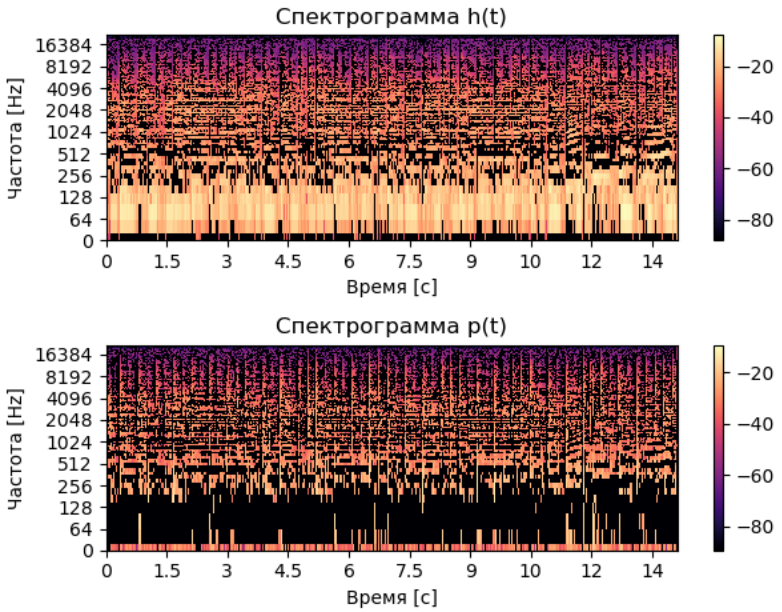

Спектрограмма p(t)
Частота [Hz]
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
12
14
Время [c]
−20
−40
−60
−80

k=50


Спектрограмма h(t)
Частота [Hz]
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
12
14
Время [c]
−20
−40
−60
−80


Спектрограмма p(t)
Частота [Hz]
16384
8192
4096
2048
1024
512
256
128
64
0
0
1.5
3
4.5
6
7.5
9
10
12
14
Время [c]
−20
−40
−60
−80

k=75



После наложения на $\widetilde{W_{h,\iota}}$

Рис. 2.2.3.1 – Спектрограммы компонент $H_{h,i}^{(k)}$(сверху) и $P_{h,i}^{(k)}$(снизу) на разных итерациях алгоритма k и после окончания работы программы

Таким образом, после получения разделенной амплитудной спектрограммы, можно получить итоговые спектрограммы разделенных компонент, а также их форму волны. Они представлены на рисунках 2.2.3.2 и 2.2.3.3, соответственно.

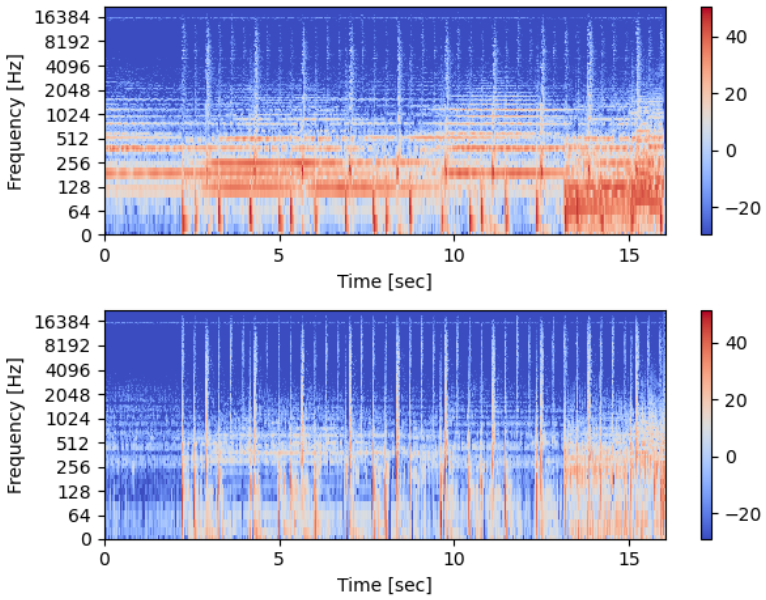


Рис. 2.2.3.2 Спектрограммы гармонической и перкуссионной компонент композиции Radiohead – All I Need

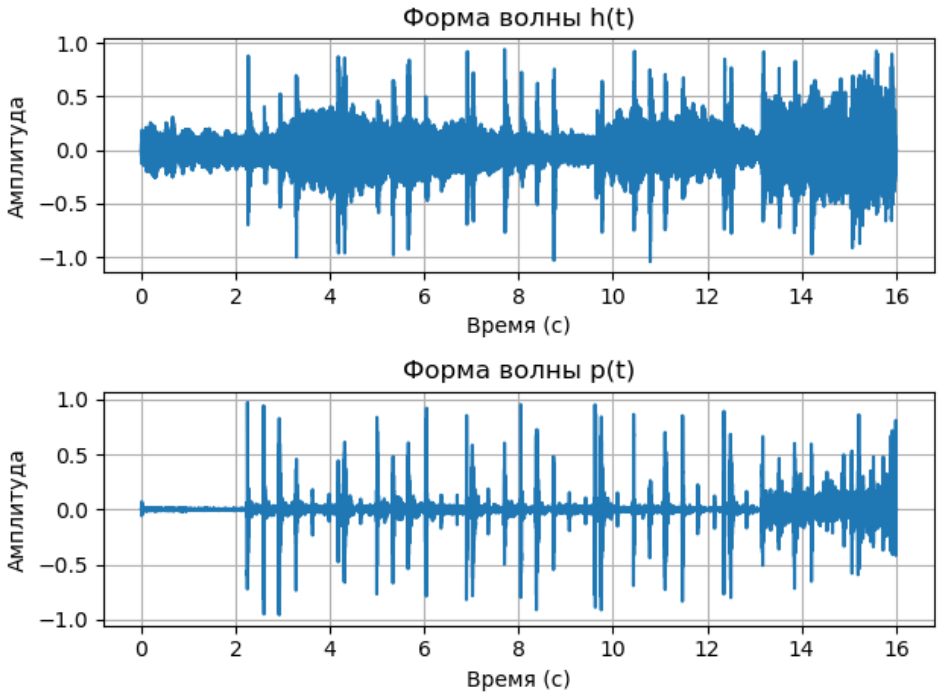


Рис. 2.2.3.3 Форма волны гармонической и перкуссионной компонент композиции Radiohead – All I Need

Глава 3. Сравнение алгоритмов

В данной главе мы сравним предложенные в прошлой главе два алгоритма по гармоническому-перкуссионному разделению звука. Для простого обозначения объявим, что алгоритм с медианной фильтрацией будет называться Median, а с использованием вспомогательной функции по разделению Ono (в честь Набутаки Оно, придумавшего алгоритм). Программная реализация примеров из данной главы приведена в Приложении 3.

Сначала мы оценим их спектрограммы, потом с использованием численных метрик сравним данные алгоритмы на основе выборки из 10 музыкальных композиций, которые были созданы из уже имеющихся гармонических и перкуссионных частей.

3.1 Сравнение спектрограмм

Для начала оценим спектрограммы двух алгоритмов. Для этого нам потребуется музыкальная композиция, которая изначально представляла из себя две разделенных музыкальных композиции, одна с гармонической составляющей, другая с перкуссионной. Условно будем называть в дальнейшем эти композиции «чистыми». Мы объединим данные две композиции, чтобы получить одну общую, которая в свою очередь будет использоваться в алгоритмах. Полученные разделенные части после работы алгоритма будем называть «рассчитанными».

Таким образом, на рисунке 3.1.1 мы можем увидеть спектрограмму исходной композиции, которая уже совмещает в себе гармоническую и перкуссионную часть.

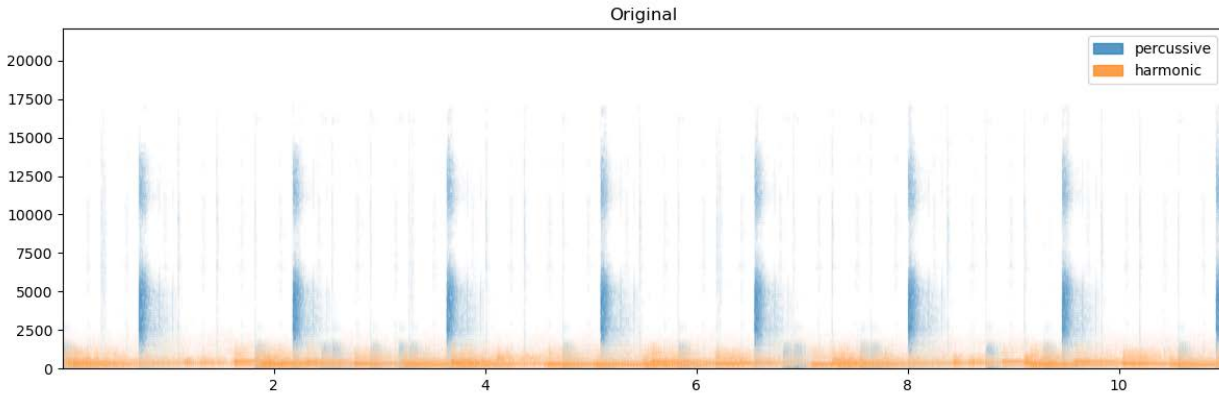
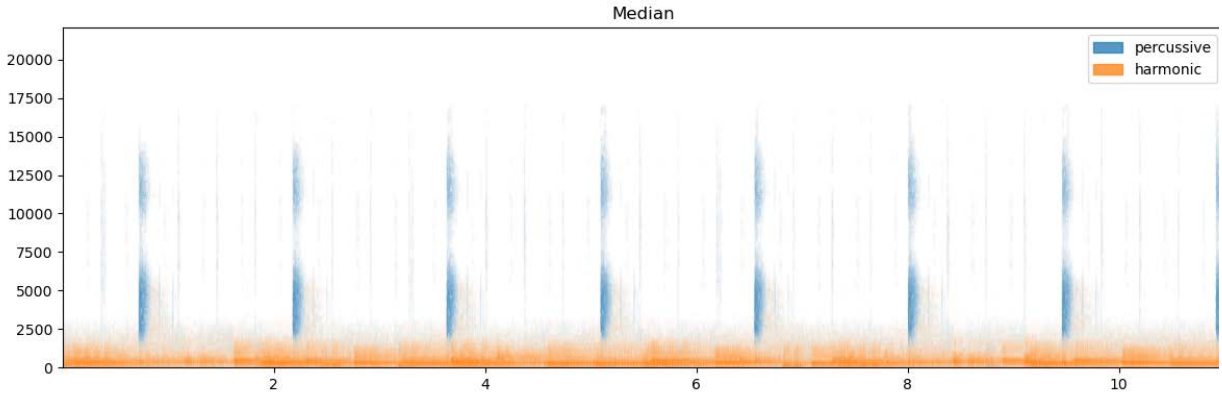


Рис.3.1.1 – Исходная спектрограмма

Как можно увидеть, на данной спектрограмме четко выделена перкуссионная часть в виде вертикальных линий, и гармоническая в виде основной гармоники. Как можно заметить, между «чистыми» композициями пересечения не наблюдается.

На рисунке 3.1.2 представим аналогичные спектрограммы, после разделения с помощью алгоритмов, и в дальнейшем слияния по аналогии с исходной композицией.



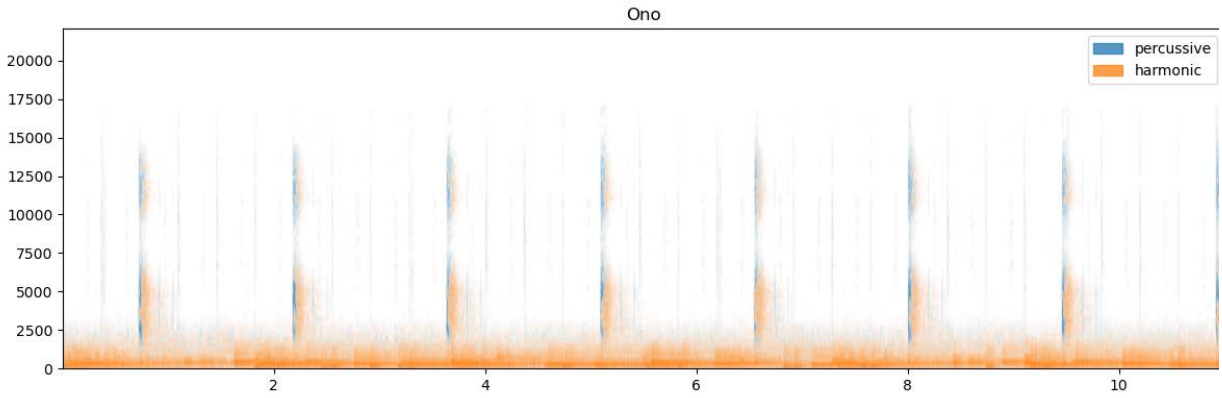


Рис.3.1.2 – Спектрограммы по алгоритму Median (сверху) и Ono (снизу)

Как можно заметить из данных спектрограмм, в случае алгоритма Median, гармонические и перкуссионные части не смешиваются, когда как в случае алгоритма Ono видно четкое наложение гармонической составляющей на перкуссионную, что может говорить о качестве алгоритма.

Пока говорить, что алгоритм Median лучше Ono, рано, так как не были использованы никакие численные метрики для сравнения, поэтому перейдем к следующему разделу.

3.2 Метрики BSS_eval

В данном разделе мы рассмотрим метрики из пакета bss-eval, которые полностью описаны в [6] и над которыми проводилось сравнение с другими метриками PEASS Toolkit [5] в [4]. К сожалению, не удалось воспользоваться метриками PEASS Toolkit в связи с тем, что они реализованы на языке Matlab. Для метрик bss-eval имеется же своя имплементация на языке Python под названием MIR_eval [13], которым мы и воспользовались в ходе работы.

BSS Eval представляет из себя набор из четырех показателей эффективности, которые оценивают качество извлеченного сигнала $\widetilde{s_j}$ с помощью соотношений энергий между различными компонентами данного сигнала. Эти метрики - первая попытка разложить сигнал на различные искажения сигнала: помехи от внешних источников, шумы и различные

артефакты (музыкальный шум). Извлеченный источник разлагается следующим образом:

$$\widetilde{s_j} = s_{target} + e_{interf} + e_{noise} + e_{artif},$$

где s_{target} это «чистый сигнал», относительно которого сравнивается извлеченный сигнал. $e_{interf}, e_{noise}, e_{artif}$ это ошибки искажения сигнала, шума и музыкальных артефактов, соответственно.

Численные метрики эффективности рассчитываются как отношения энергий, выраженных в дБ. А именно, отношение «чистого» сигнала к его искажению (SDR), отношение «чистого» сигнала к помехам (SIR), отношение «чистого» сигнала к шуму (SNR) и отношение «чистого» сигнала к артефактам (SAR). Если говорить проще, то SDR оценивает качество разделения сигнала в целом, SIR оценивает, насколько сильны помехи других сигналов, SNR оценивает, насколько высок уровень шума, SAR оценивает количество артефактов, которые создаются в ходе разделения сигнала.

$$SDR = 10\ log_{10} \frac{||s_{target}||^2}{||e_{interf} + e_{noise} + e_{artif}||^2}$$

$$SIR = 10\ log_{10} \frac{||s_{target}||^2}{||e_{interf}||^2}$$

$$SNR = 10\ log_{10} \frac{||s_{target} + e_{interf}||^2}{||e_{noise}||^2}$$

$$SAR = 10\ log_{10} \frac{||s_{target} + e_{interf} + e_{noise}||^2}{||e_{artif}||^2}$$

Важной характеристикой этого набора объективных показателей является то, что они присваивают одинаковые веса различным ошибкам. Предполагается, что с точки зрения качества, все типы искажений вносят одинаковый вклад в общее качество извлеченного источника. Еще одной важной характеристикой этих наборов мер является то, что они не учитывают перцептивные аспекты слуха для своих расчетов (в отличии от PEASS Toolkit).

В нашем случае $\widetilde{s_j}$ будет извлеченная или «рассчитанная» гармоническая или перкуссионная часть, а s_{target} исходная «чистая» гармоническая часть сигнала. Для сравнения были выбраны 10 музыкальных композиций, которые изначально были разделены (итого получаем 20), которые будут входными данными для двух алгоритмов Median и Ono. После обработки алгоритмов мы используем возможности библиотеки mir_eval для Python для расчета метрик по выходным данным каждого алгоритма, усредним их и с помощью специального типа графика «ящик с усами» сравним данные алгоритмы.

На рисунке 3.2.1 мы можем увидеть сравнение алгоритмов по гармонической части по данным метрикам . Как можно заметить, алгоритм Median превосходит алгоритм Ono практически по всем метрикам, в частности, по SDR, SAR, SNR среднее значение находится выше, но при этом метрика SIR по среднему значению хоть и совпадает у обоих алгоритмов, но при этом у алгоритма Ono имеется большее распределение данного значения, что может говорить о нестабильности избавления от помех в случае алгоритма Ono.

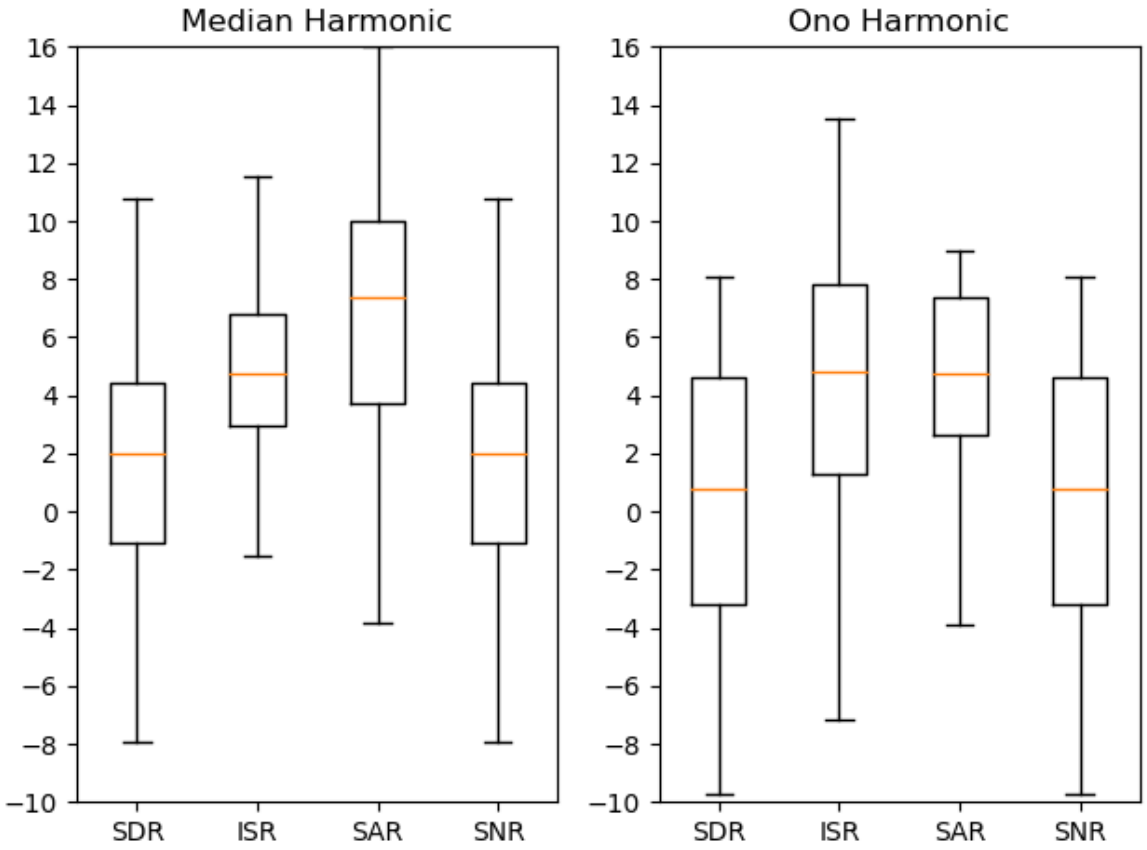


Рис.3.2.1 – Метрики bss_eval для сравнения гармонической составляющей

Аналогично, на рисунке 3.2.2 мы можем увидеть сравнение алгоритмов по перкуссионной части по данным метрикам. Как можно снова заметить, алгоритм Median снова превосходит алгоритм Ono по средним значениям. Но при этом алгоритм Ono показал устойчивость к выбросам данных в виде меньшего распределения метрик кроме SAR, что говорит о том, что в ходе разделения у перкуссионной части появляется множество музыкальных артефактов.

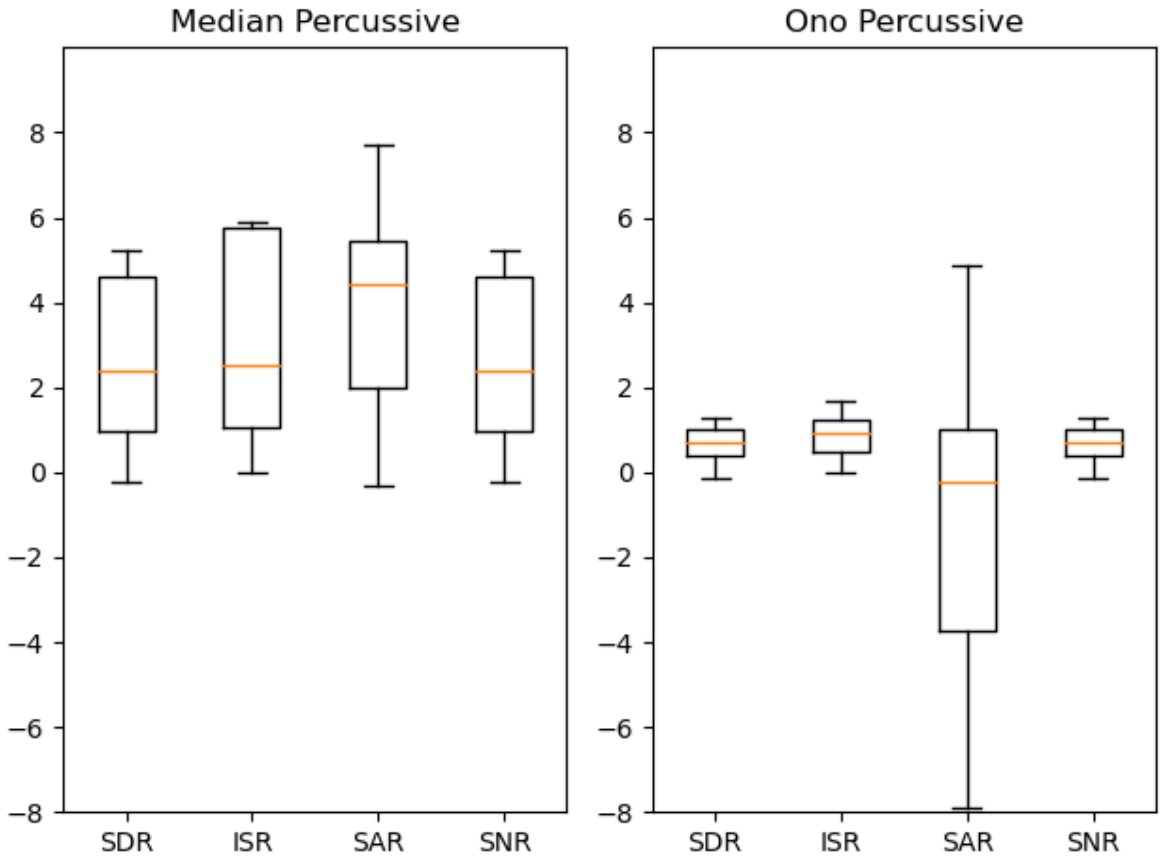


Рис.3.2.2 – Метрики bss_eval для сравнения перкуссионной составляющей

Таким образом, можно утверждать, что алгоритм Median превосходит алгоритм Ono по данным метрикам.

3.3 Сравнение начала звучания нот и ударных

В данном разделе мы сравним на той же выборке из прошлого раздела алгоритмы путем определения начала звучания нот или ударных у «чистых» гармонических и перкуссионных частей и «рассчитанных». Для условности будем считать, что определенные начала звучания нот или ударных у чистых композиций являются верными. Чтобы определить начала звучания воспользуемся библиотеками librosa [16] и для сравнения mir_eval.

Получив временные участки, где начинаются ударные или ноты, мы можем воспользоваться F-мерой Ван Ризбергена и мерой точности и полноты.

Точность (precision) и полнота (recall) являются метриками, которые используются при оценке большей части алгоритмов извлечения информации.

Иногда они используются сами по себе, иногда в качестве базиса для производных метрик, таких как F-мера Ван Ризбергена. Полнота показывает, как много начал звучания нот или ударных «рассчитанных» гармонических или перкуссионных частей музыкального произведения совпадают с общим количеством всех начал «чистых» частей. Точность показывает, как много начал звучания нот или ударных «чистых» гармонических или перкуссионных частей музыкального произведения совпадают с общим количеством всех начал «рассчитанных» частей. F-мера объединяет эти два понятия, чтобы определить в целом эффективность алгоритма.

На рисунке 3.3.1 можно увидеть сравнение F-меры двух алгоритмов для гармонической и перкуссионной частей. Как можно заметить, алгоритм Median снова превосходит алгоритм Ono в части определения начала звучания нот и ударных. Аналогичную ситуацию можно увидеть на рисунке 3.3.2 и 3.3.3. Также можно отметить, насколько хорошо определяются начала ударных у обоих алгоритмов. Данное преимущество можно использовать при предобработке музыкального сигнала с целью определения начала звучания нот или же темпа.

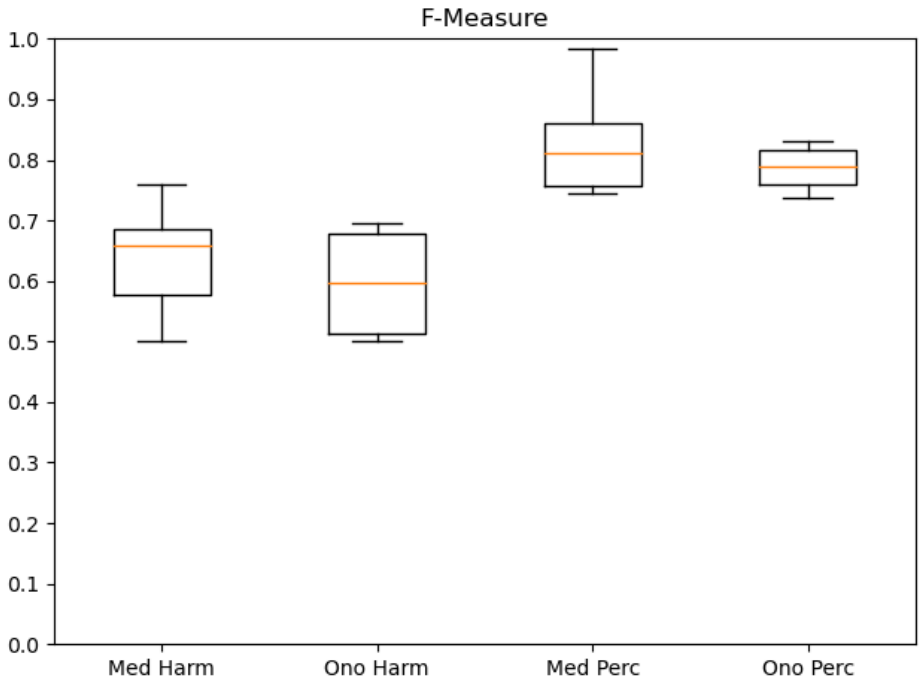


Рис. 3.3.1 – Сравнение F-мер алгоритмов Median и Ono

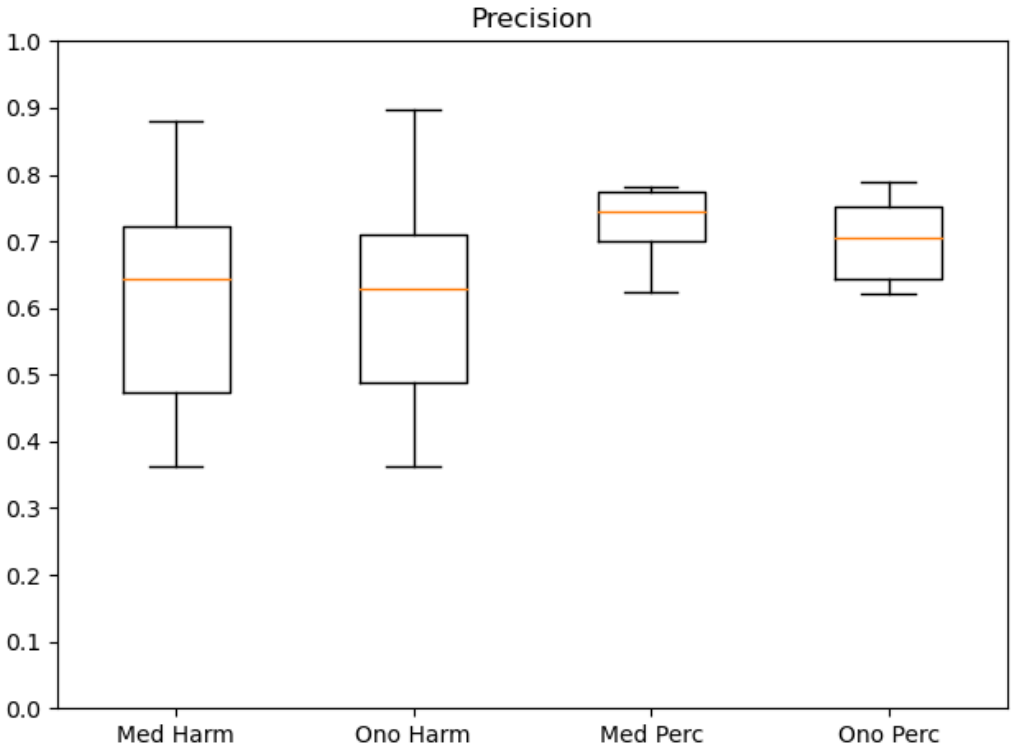


Рис. 3.3.2 – Сравнение мер точности алгоритмов Median и Ono

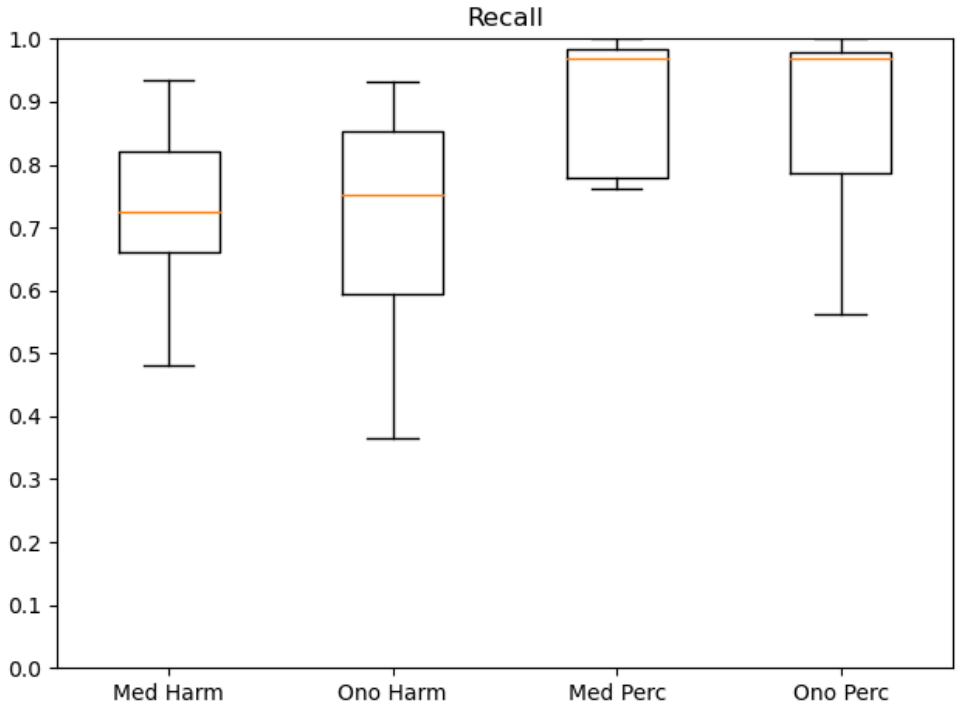


Рис. 3.3.3 – Сравнение мер полноты алгоритмов Median и Ono

3.4 Сравнение нормы и среднеквадратичной ошибки

В данном разделе мы сравним данные алгоритмы с помощью расчета евклидовой нормы и среднеквадратичной ошибки между «чистыми» и «рассчитанными» гармоническими и перкуссионными частями. В качестве выборки будем использовать те же 10 композиций.

Согласно [1] норма евклидового пространства определяется как:

$$||x|| = \sqrt{(x \cdot x)} \tag{3.4.1}$$

Использование евклидовой нормы и среднеквадратичного отклонения является довольно популярной мерой для измерения «расстояния» между сигналами [11] [15].

На рисунке 3.4.1 и 3.4.2 представлены сравнения норм и среднеквадратичных ошибок для гармонических и перкуссионных частей алгоритмов. Данные графики можно трактовать следующим образом. Чем меньше норма или среднеквадратичное отклонение, тем меньше разница между «чистым» сигналом и «рассчитанным». Как можно заметить, алгоритм Median показывает лучшие результаты по сравнению с алгоритмом Ono.

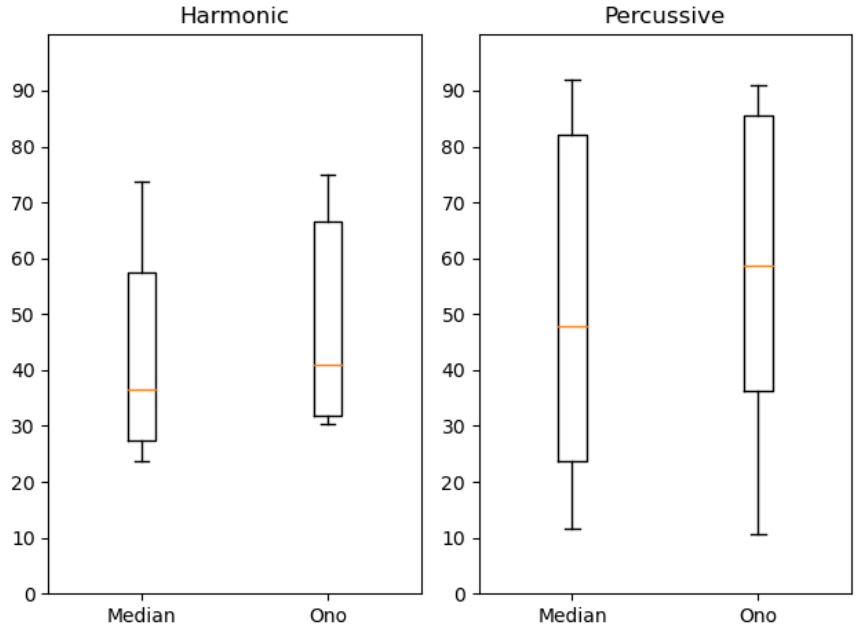


Рис.3.4.1 – Евклидовы нормы алгоритмов Median и Ono для гармонической и перкуссионной частей

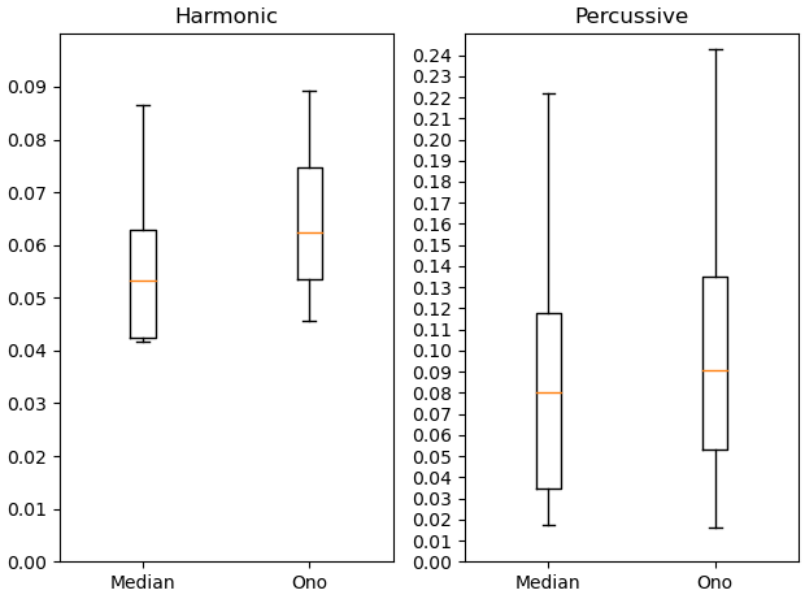


Рис.3.4.2 – Среднеквадратичное отклонение алгоритмов Median и Ono для гармонической и перкуссионной частей

Заключение

В ходе данной работы мы исследовали проблему разделения музыкального сигнала, обозначили основные термины (спектрограмма, оконное преобразование Фурье, понятие ADSR-кривой), связанные с данной проблемой, а также исследовали конкретный метод по разделению музыкального сигнала на гармоническую и перкуссионную части, сформулировав на ее основе постановку задачи. В дальнейшем мы детально рассмотрели два алгоритма, которые реализуют данную методику, а также рассмотрели все сложности, которые возникают в ходе работы данных алгоритмом. Первый алгоритм использует в своей основе медианную фильтрацию на спектрограмме, и в дальнейшем использование маски для разделения спектрограммы на две различные спектрограммы двух компонент – гармонической и перкуссионной. Второй алгоритм использует вспомогательную функцию, которая помогает решить задачу оптимизации (нахождению максимума по частотной и временной оси) для выведенного дифференциального уравнения. Далее мы сравнили эти два алгоритма с помощью различных методик, в частности, сравнили спектрограммы, определили понятие метрик BSS_eval и использовали их для сравнения алгоритмов на основе выборки из 10 композиций. Далее мы сравнивали начала звучания нот или ударных и определяли, какой из алгоритмов лучше определяет эти начала. В конце мы рассчитали евклидову норму и среднеквадратичное отклонение между «чистыми» композициями и полученными из алгоритмов на основе той же выборки. Итог оказался таков, что алгоритм, использующий медианную фильтрацию, оказался лучше, и поэтому для второго алгоритма было выдвинуто предложение использовать «мягкие» маски вместо бинарных с целью повышения эффективности алгоритма.

В ходе данной работы была изучена очень малая наука, которая активно развивается последние двадцать лет в мире, но при этом в России данная наука остается практически незамеченной. В этом заключается инновативность данной работы – показать, каким образом уже имеющиеся математическое методы

используются в прикладной сфере. Существует множество способов прикладного использования методик из данной науки, в том числе и методы, рассмотренной в данной работе. Гармоническое-перкуссионное разделение исходного сигнала активно используется на этапе предобработки сигнала, чтобы в дальнейшем легче обрабатывать сигналы, например, с помощью гармонической составляющей по сравнению с неразделенной композицией гораздо легче извлечь ноты или аккорды для их транскрипции или извлечь основную мелодию, с помощью перкуссионной части гораздо проще находить музыкальный темп музыки. Данные методики можно использовать при создании автоматического караоке, поиску похожих песен, составлению плейлистов по настроению и многое другое. На данный момент только Яндекс использует в своей сфере методики из данной науки в своем сервисе Яндекс.Музыка. Каждая композиция проходит этап предобработки перед тем, как использоваться моделью машинного обучения.

Список использованной литературы

1. Козлов В.Н. Системный анализ, оптимизация и принятие решений: учебное пособие. – СПб: Издательство Политехнического университета, 2011. – 245 с.

2. Фисенко В.Т., Фисенко Т.Ю. Компьютерная обработка и распознавание изображений: учебное пособие, СПб: СПбГУ ИТМО, 2008. – 195 с.

3. Alm J.F., Walker, J.S. Time-frequency analysis of musical instruments. SIAM Rev. 2002, 44, 457–476.

4. Cano E., FitzGerald D. and Brandenburg K. Evaluation of quality of sound source separation algorithms: Human perception vs quantitative metrics, 2016 24th European Signal Processing Conference (EUSIPCO), Budapest, 2016, pp. 1758-1762.

5. Emiya V., Vincent E., Harlander N., Hohmann V. Subjective and Objective Quality Assessment of Audio Source Separation in IEEE Transactions on Audio, Speech, and Language Processing, vol. 19, no. 7, Sept. 2011, pp. 2046-2057.

6. Févotte C., Gribonval R., Vincent E. BSS_EVAL Toolbox User Guide – Revision 2.0.:Technical Report, 2005, pp.19 .

7. Fitzgerald D. Harmonic/Percussive Separation using Median Filtering. 13th International Conference on Digital Audio Effects (DAFx-10). 2010.

8. Griffin, DW & Lim, Jae. Signal estimation from modified short-time Fourier transform. Acoustics, Speech and Signal Processing, IEEE Transactions on. 32. 1984, pp. 236-243.

9. Kameoka, Hirokazu & Nishimoto, Takuya & Sagayama, Shigeki. A Multipitch Analyzer Based on Harmonic Temporal Structured Clustering. Audio, Speech, and Language Processing, IEEE Transactions on. 15. 2007, pp. 982 – 994.

10. D. D. Lee and H. S. Seung, “Algorithms for NonNegative Matrix Factorization” Proc. NIPS, 2000, pp. 556– 562.

11. Müller, Meinard. Fundamentals of Music Processing, 2015, 487 p.

12. Ono, Nobutaka & Miyamoto, Kenichi & Le Roux, Jonathan & Kameoka, Hirokazu & Sagayama, Shigeki. Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram, 2008.

13. Raffel, Colin & Mcfee, Brian & Humphrey, Eric & Salamon, Justin & Nieto, Oriol & Liang, Dawen & Ellis, Daniel. mir_eval: A Transparent Implementation of Common MIR Metrics. Proceedings - 15th International Society for Music Information Retrieval Conference (ISMIR 2014), 2014.

14. Vail M. The Synthesizer: A Comprehensive Guide to Understanding, Programming, Playing, and Recording the Ultimate Electronic Music Instrument, Oxford University Press, 2014, 432 p.

15. Asvial M. Digital Communication Signal Space and Average Error Probability,CICER,
http://staff.ui.ac.id/system/files/users/ir.muhammad/material/komdig45.pdf

16. LibROSA, a python package for music and audio analysis https://librosa.github.io/librosa/

Приложение 1

Harm_perc_ideal_and_not.py:

```
import matplotlib.pyplot as plt
import librosa.display
import numpy as np
from scipy.ndimage import median_filter
from librosa import core
from librosa import util
import os


def main():
    plt.rcParams['figure.figsize'] = (14, 5)

    filename = './audio/classic_rock_RS.wav'
    x, sr = librosa.load(filename, duration=6, sr=None)

    # information about wav
    print(len(x))

    # short-time fourier transform
    X = librosa.stft(x)
    #  log-amplitude
    Xmag = librosa.amplitude_to_db(X)

    # show harm-perc spectrogram
    librosa.display.specshow(Xmag, sr=sr, x_axis='time', y_axis='log')
    plt.colorbar()
    plt.show()
    ##############
    S = X
    kernel_size = 31
    power = 2.0
    mask = False
    margin = 1.0

    if np.iscomplexobj(S):
        S, phase = core.magphase(S)
    else:
        phase = 1

    if np.isscalar(kernel_size):
        win_harm = kernel_size
        win_perc = kernel_size
    else:
        win_harm = kernel_size[0]
        win_perc = kernel_size[1]

    if np.isscalar(margin):
        margin_harm = margin
        margin_perc = margin
    else:
        margin_harm = margin[0]
        margin_perc = margin[1]

    split_zeros = (margin_harm == 1 and margin_perc == 1)
    # Compute median filters. Pre-allocation here preserves memory layout.
    harm = np.empty_like(S)
```

```
    harm[:] = median_filter(S, size=(1, win_harm), mode='reflect')

    perc = np.empty_like(S)
    perc[:] = median_filter(S, size=(win_perc, 1), mode='reflect')

    mask_harm_soft = util.softmask(harm, perc * margin_harm,
                                   power=power,
                                   split_zeros=split_zeros)
    mask_perc_soft = util.softmask(perc, harm * margin_perc,
                                   power=power,
                                   split_zeros=split_zeros)
    soft_mask_X_harm = (S * mask_harm_soft) * phase
    Xmag_harm_soft = librosa.amplitude_to_db(soft_mask_X_harm)
    soft_mask_X_perc = (S * mask_perc_soft) * phase
    Xmag_perc_soft = librosa.amplitude_to_db(soft_mask_X_perc)

    # mask_harm_hard = harm > perc * margin_harm
    # mask_perc_hard = perc > harm * margin_perc
    # hard_mask_X_harm = (S * mask_harm_hard) * phase
    # Xmag_harm_hard = librosa.amplitude_to_db(hard_mask_X_harm)
    # hard_mask_X_perc = (S * mask_perc_hard) * phase
    # Xmag_perc_hard = librosa.amplitude_to_db(hard_mask_X_perc)


    # x_h, sr_h = librosa.load('my_audio_mod/01_AF_NM_h.wav', duration=6, sr=None)
    # x_p, sr_p = librosa.load('my_audio_mod/01_AF_NM_p.wav', duration=6, sr=None)
    # librosa.display.waveplot( x_h, sr=sr_h)
    # plt.show()
    # librosa.display.waveplot( x_p, sr=sr_p)
    # plt.show()
    H = (S * mask_harm_soft) * phase
    P = (S * mask_perc_soft) * phase

    Hmag = librosa.amplitude_to_db(H)
    Pmag = librosa.amplitude_to_db(P)

    librosa.display.specshow(Hmag, sr=sr, x_axis='time', y_axis='log')
    plt.colorbar()
    plt.show()

    librosa.display.specshow(Pmag, sr=sr, x_axis='time', y_axis='log')
    plt.colorbar()
    plt.show()

    h = librosa.istft(H)
    p = librosa.istft(P)

    librosa.output.write_wav(os.path.splitext(filename)[0] + '_H_libr.wav', h, sr)
    librosa.output.write_wav(os.path.splitext(filename)[0] + '_P_libr.wav', p, sr)

main()
```

Приложение 2

Complement_diff.py:

```
      import numpy as np
import os
import matplotlib.pyplot as plt
from scipy.signal import stft, istft, spectrogram
import librosa.display
from scipy.io.wavfile import read, write


def separate_instruments(file_path):
    # Считывание файла

    # fs, x = read("./audio/" + file_path)
    x, fs = librosa.load(file_path, duration=16, sr=None)
    # f, t, Sxx = spectrogram(x, fs)
    X = librosa.stft(x)
    #  log-amplitude
    Xmag = librosa.amplitude_to_db(X)

    # показать изначальную спектрограмму
    librosa.display.specshow(Xmag, sr=fs, x_axis='time', y_axis='log')
    plt.colorbar()
    plt.title('Spectrogram of x(t)')
    plt.ylabel('Frequency [Hz]')
    plt.xlabel('Time [sec]')
    plt.show()

    winlen = 1024

    # Шаг 1. Создать оконное преобразование Фурье
    h, i, F = stft(x=x, fs=fs, window='hann', nperseg=winlen, noverlap=int(winlen / 2),
                   nfft=winlen, detrend=False, return_onesided=True, padded=True,
axis=-1)

    # Шаг 2: Амплитудная спектрограмма
    gamma = 0.3
    W = np.power(np.abs(F), 2 * gamma)

    # Шаг 3: Инициализация
    k_max = 100
    H = 0.5 * W
    P = 0.5 * W
    alpha = 0.3

    for k in range(k_max):
        # Шаг 4. Рассчитать дельту
        term_1 = np.zeros_like(H)
        term_2 = np.zeros_like(H)

        for i_iter in range(1, np.shape(H)[1] - 1):
            term_1[:, i_iter] = alpha * ((H[:, i_iter - 1] + H[:, i_iter + 1] - (2 *
H[:, i_iter])) / 4)

        term_1[:, 0] = alpha * ((H[:, 1] - H[:, 0]) / 2)
        term_1[:, -1] = alpha * ((H[:, -2] - H[:, -1]) / 2)
```

```
        for h_iter in range(1, np.shape(H)[0] - 1):
            term_2[h_iter, :] = (1 - alpha) * ((P[h_iter - 1, :] + P[h_iter + 1, :] -
(2 * P[h_iter, :])) / 4)

        term_2[0, :] = (1 - alpha) * ((P[1, :] - P[0, :]) / 2)
        term_2[-1, :] = (1 - alpha) * ((P[-2, :] - P[-1, :]) / 2)

        delta = term_1 - term_2

        # уменьшить шаг
        delta = delta * 0.9

        # Шаг 5. Обновить матрицы компонент
        H = np.minimum(np.maximum(H + delta, 0), W)
        P = W - H

        # Шаг 6: Автоматически увеличивать к

    # Шаг 7: Разделить амлпитудную спектрограмму
    H = np.where(np.less(H, P), 0, W)
    P = np.where(np.greater_equal(H, P), 0, W)

    # Шаг 8. Обратное оконное преобразование Фурье
    H_temp = np.power(H, (1 / (2 * gamma))) * np.exp(1j * np.angle(F))  # ISTFT is
taken first on this, with H
    P_temp = np.power(P, (1 / (2 * gamma))) * np.exp(1j * np.angle(F))  # ISTFT is
taken second on this, with P
    _, h = istft(H_temp, fs=fs, window='hann', nperseg=winlen,
                 noverlap=int(winlen / 2), nfft=winlen, input_onesided=True)
    _, p = istft(P_temp, fs=fs, window='hann', nperseg=winlen,
                 noverlap=int(winlen / 2), nfft=winlen, input_onesided=True)

          # saving
    librosa.output.write_wav(os.path.splitext(file_path)[0] + '_H_cd.wav', h, fs)
    librosa.output.write_wav(os.path.splitext(file_path)[0] + '_P_cd.wav', p, fs)


if __name__ == '__main__':
    print("Начало")

    separate_instruments(file_path = './audio/classic_rock_RS.wav')

    print("Конец")
```

Приложение 3

Comparison.py:

```
import mir_eval
from librosa import display
import librosa
import scipy
import matplotlib.pyplot as plt
import mass_ts as mts
import numpy as np
from scipy.spatial import distance
import matplotlib.axes as ax

def get_wavs(file_path_true, file_path_est):
    x_est, sr = librosa.load(file_path_est, sr=None)
    x_true, sr = librosa.load(file_path_true, sr=None)
    return x_true,x_est

def comparison(x_true,x_est,sr=44100):
    hop_length = 1024
    mfcc_est = librosa.feature.mfcc(y=x_est, sr=sr, hop_length=hop_length)
    mfcc_true = librosa.feature.mfcc(y=x_true, sr=sr, hop_length=hop_length)
    xsim = librosa.segment.cross_similarity(mfcc_est, mfcc_true)
    xsim_cosine    =    librosa.segment.cross_similarity(mfcc_est,    mfcc_true,
metric='cosine')
    xsim_aff     =     librosa.segment.cross_similarity(mfcc_est,     mfcc_true,
mode='affinity')
    # P = simple(A=x_est, B= x_true,m=100)
    # corr = scipy.signal.correlate(x_true,x_est)
    norm = np.linalg.norm(x_true-x_est)
    mse = (np.square(x_true - x_est)).mean()
    corr  =  distance.cdist(abs(librosa.stft(x_true)),  abs(librosa.stft(x_est)),
'chebyshev')
    time_static = np.max(x_true*x_est)
    print(time_static)
    print(mse)
    print(norm)
    # print(distances)
    plt.figure(figsize=(8, 4))
    plt.subplot(1, 2, 1)
    librosa.display.specshow(xsim_aff,cmap='magma_r',              x_axis='time',
y_axis='time', hop_length=hop_length)
    plt.title('Binary recurrence (symmetric)')
    plt.subplot(1, 2, 2)
    plt.plot(corr)
    plt.title('Cross-correlated')
    plt.tight_layout()
    plt.show()

def bss_eval(x_true,x_est):
    (sdr, isr, sir, sar, perm) = mir_eval.separation.bss_eval_images(x_true,x_est)
    original_power = 10 * np.log10(np.sum(np.power(x_true, 2)))
    error = x_true - x_est[0:len(x_true)]
    noise_power = 10 * np.log10(np.sum(np.power(error, 2)))
    snr = original_power - noise_power
    return sdr,isr,snr,sar

def norm_euc_and_mse(x_true,x_est):
    norm = np.linalg.norm(x_true - x_est)
```

```
            rmse = np.sqrt((np.square(x_true - x_est)).mean())

            return norm,rmse

        def onset_eval(x_true,x_est,sr = 44100):
            est_onsets = librosa.onset.onset_detect(y=x_est, sr=sr, units='time')
            true_onsets = librosa.onset.onset_detect(y=x_true, sr=sr, units='time')
            # print(est_onsets)
            # print(true_onsets)
            dict = mir_eval.onset.evaluate(true_onsets, est_onsets)
            return dict['F-measure'],dict['Precision'],dict['Recall']

        if __name__ == '__main__':
            print("Начало")
            filepath = 'samples_for_3_glava/'
            alg_path = ('original','ono','median')
            type_path = ('_harmonic/','_percussive/')
            number_path = 1
            end_path = '.wav'
            sdr,isr,snr,sar = (0,0,0,0)
            rmse,norm = (0,0)
            f,Prec,Rec = (0,0,0)

            for i in range(1,3):
                for j in range (0,2):
                    for m in range (1,9):
                        file_path_true                                                  =
filepath+alg_path[0]+type_path[j]+str(m)+end_path
                        file_path_est = filepath + alg_path[i] + type_path[j] + str(m) +
end_path
                        x_true, x_est = get_wavs(file_path_true = file_path_true,
                                                 file_path_est = file_path_est)
                        # comparison(x_true,x_est)
                        # sdr,isr,snr,sar = bss_eval(x_true, x_est)
                        # norm,rmse = norm_euc_and_mse(x_true,x_est)
                        f,Prec,Rec = onset_eval(x_true,x_est)
                        if alg_path[i]== 'ono':
                            if type_path[j] == '_harmonic/':
                                ono_h_sdr = np.append(ono_h_sdr,sdr)
                                ono_h_isr = np.append(ono_h_isr, isr)
                                ono_h_sar = np.append(ono_h_sar, sar)
                                ono_h_snr = np.append(ono_h_snr,snr)
                                ono_h_norm = np.append(ono_h_norm, norm)
                                ono_h_rmse = np.append(ono_h_rmse, rmse)
                                ono_h_f = np.append(ono_h_f,f)
                                ono_h_pr = np.append(ono_h_pr, Prec)
                                ono_h_rec = np.append(ono_h_rec, Rec)
                            if type_path[j] == '_percussive/':
                                ono_p_sdr = np.append(ono_p_sdr,sdr)
                                ono_p_isr = np.append(ono_p_isr,isr)
                                ono_p_sar = np.append(ono_p_sar,sar)
                                ono_p_snr = np.append(ono_p_snr, snr)
                                ono_p_norm = np.append(ono_p_norm, norm)
                                ono_p_rmse = np.append(ono_p_rmse, rmse)
                                ono_p_f = np.append(ono_p_f,f)
                                ono_p_pr = np.append(ono_p_pr, Prec)
                                ono_p_rec = np.append(ono_p_rec, Rec)
                        if alg_path[i]== 'median':
                            if type_path[j] == '_harmonic/':
                                md_h_sdr = np.append(md_h_sdr,sdr)
                                md_h_isr = np.append(md_h_isr, isr)
```

```
                    md_h_sar = np.append(md_h_sar, sar)
                    md_h_snr = np.append(md_h_snr, snr)
                    md_h_norm = np.append(md_h_norm, norm)
                    md_h_rmse = np.append(md_h_rmse, rmse)
                    md_h_f = np.append(md_h_f,f)
                    md_h_pr = np.append(md_h_pr, Prec)
                    md_h_rec = np.append(md_h_rec, Rec)
                if type_path[j] == '_percussive/':
                    md_p_sdr = np.append(md_p_sdr,sdr)
                    md_p_isr = np.append(md_p_isr,isr)
                    md_p_sar = np.append(md_p_sar,sar)
                    md_p_snr = np.append(md_p_snr, snr)
                    md_p_norm = np.append(md_p_norm, norm)
                    md_p_rmse = np.append(md_p_rmse, rmse)
                    md_p_f = np.append(md_p_f,f)
                    md_p_pr = np.append(md_p_pr, Prec)
                    md_p_rec = np.append(md_p_rec, Rec)
data_md_h = [ md_h_sdr, md_h_isr, md_h_sar,md_h_snr]
data_md_p = [ md_p_sdr, md_p_isr, md_p_sar,md_p_snr]
data_ono_h = [ ono_h_sdr, ono_h_isr, ono_h_sar,ono_h_snr]
data_ono_p = [ ono_p_sdr, ono_p_isr, ono_p_sar,ono_p_snr]
data_h_norm = [md_h_norm,ono_h_norm]
data_p_norm = [md_p_norm,ono_p_norm]
data_h_rmse = [md_h_rmse,ono_h_rmse]
data_p_rmse = [md_p_rmse,ono_p_rmse]
data_f = [md_h_f,ono_h_f,md_p_f,ono_p_f]
data_pr = [md_h_pr,ono_h_pr,md_p_pr,ono_p_pr]
data_rec = [md_h_rec,ono_h_rec,md_p_rec,ono_p_rec]
print("Конец")
```