Государственное образовательное учреждение высшего профессионального образования
Санкт-Петербургский национальный исследовательский университет
Информационных Технологий, Механики и Оптики
Факультет инфокоммуникационных технологий

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

По Мультимедиа Технонолиям
На тему: «Звук»

Проверил(а): Хлопотов М.В.

_____

Дата: «___»_____201__г.

Оценка: _____

Работу выполнил(а):

Никончук А.П.
Студент(ка) группы К3242
Дневного отделения

_____

2019

Цель: Изучить

Задачи:

1. Создать музыкальный фрагмент продолжительностью около 30 секунд неслучайного набора звуков, генерируя как сумму синусоид на разной частоте и с разной амплитуой
1. изменить громкость: с начала увеличение, в конце затухание
2. наложить фрагмент чужой записи
3. сохранить в формате wav
4. изобразить и прокомментировать спектрограмму

Выполнение работы:

1. Генерация последовательности.

В первую очередь заполняется массив creation нулями размером, в два раза превышающем записанную композицию. В качестве которой был взят отрывок интерпретации для начинающих изучения игры на фортепиано (упрощенная версия) Ed Sheron Perfect только для правой руки – ноты малой октавы. Ноты записаны соответственно частотам нот:

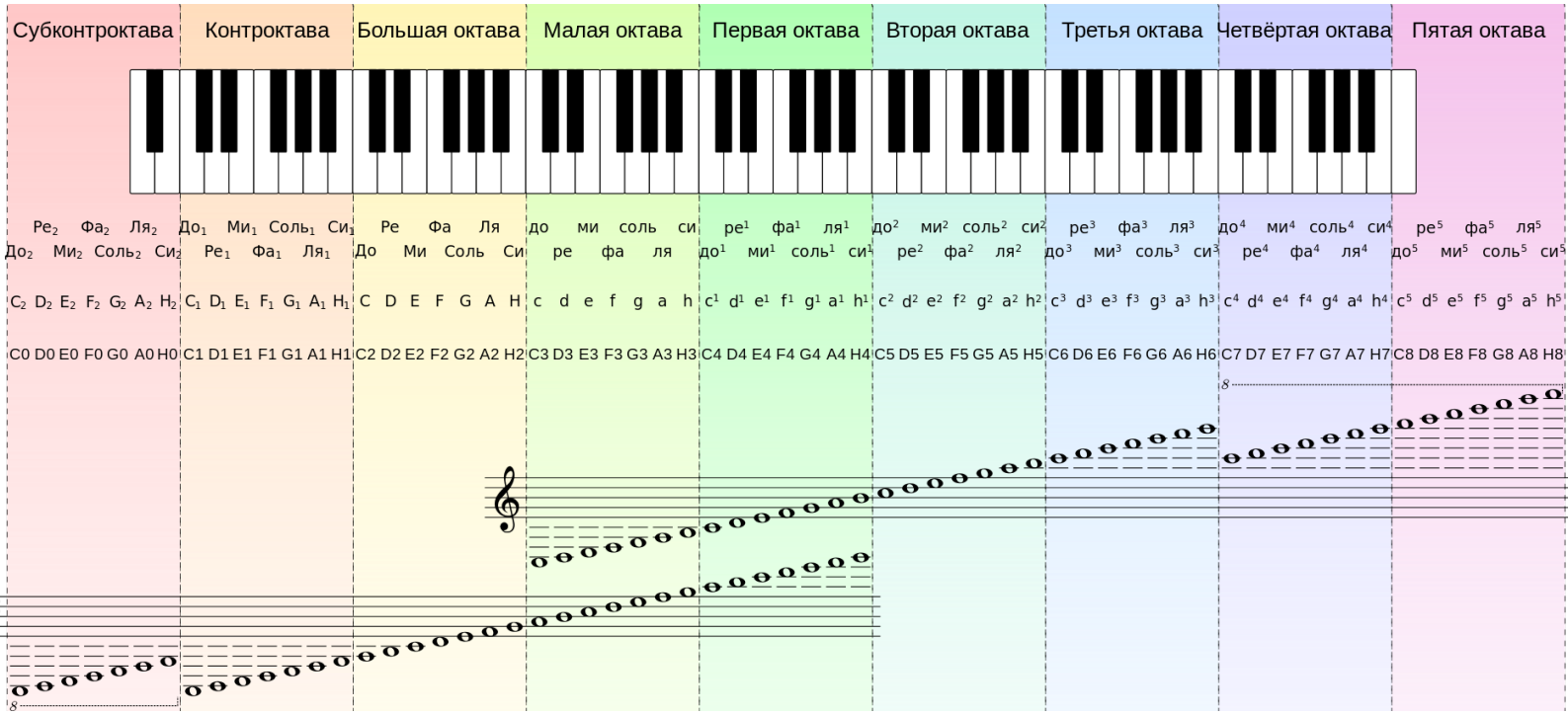


Рисунок 1. Схема клавиатуры фортепиано, позволяющая находить частоту звука

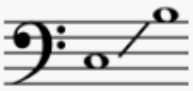
Номер ступени	Частота, Гц	Слоговое обозначение по Гельмгольцу	Буквенное обозначение по Гельмгольцу	Научная нотация	Современная музыкальная нотация
1	130,81	до	c	C3	
2	146,83	ре	d	D3	
3	164,81	ми	e	E3	
4	174,61	фа	f	F3	
5	196,00	соль	g	G3	
6	220,00	ля	a	A3	
7	246,94	си	h	H3	

Рисунок 2. Таблиа соответствия частот клавишам малой октавы фортепиано, найденной на рис.1

Так решен вопрос генерации последовательности – нобор звуков не случаен, есть изменение частот и амплитуды, предположительно автор (исходя из звучности композиции) заложил определенную синусоидальную последовательность. Вариант выбран, учитывая отсутсвие музыкального образования и элементарных знаний нотной граммотности, так как зачастую композиции изображены в виде листа с нотами.

Result = (32765*VOL*math.sin(6.28*FREQ*i/44100)), где
32765 — Фрейм у нас двухбайтовый, поэтому максимальное значение амплитуды ровно 32765.
VOL — переменная, задающая громкость. Изменяется в диапазоне от 0 до 1.
6.28 — это всего-навсего 2*Pi. Можно каждый раз высчитывать.
FREQ — А это то, ради чего все и затевалось — нужная нам частота.
i/44100 — время, относительно начала отсчета. Частота дискретизации выходного файла (Можно и меньше, но качество будет ниже). За секунду проходит 44100 отсчетов, поэтому делим. Один отсчет (при данной частоте дискретизации) это 1/44100 секунды.

```
141  # частота дискретизации
142  SAMPLE_RATE = 44100
143  # 16-ти битный звук (2 ** 16 -- максимальное значение для int16)
144  S_16BIT = 2 ** 16
145
146  def generate_sample(creation, freq, place, duration, volume):
147      #print('generate_sample call', freq, place, duration, volume)
148      # амплитуда
149      amplitude = np.round(S_16BIT * volume)
150      # длительность генерируемого звука в сэмплах
151      total_samples = np.round(SAMPLE_RATE * duration)
152      # частоте дискретизации (пересчитанная)
153      w = 2.0 * np.pi * freq / SAMPLE_RATE
154      # массив сэмплов
155      k = np.arange(0, total_samples)
156
157      for i in range(place-len(creation) + duration + 1):
158          creation.append(0)
159
160      for x in range(int(duration  * (SAMPLE_RATE / 1000.0))):
161          creation.append(volume * np.sin(2 * np.pi * freq * ( x / SAMPLE_RATE)))
162      return creation
```

Рисунок 3. Вариант кода генерации семпла по заданным и переданным парамерам

2. Регулировка громкости звука

Изменение громкости происходит в зависимости от передаваемого параметра attenuation – количества делений затухания. Первые значения в размере от 0 до заданного числа увеличивают громкость от нуля с каждым шагом цикла на step равный частному 1 (как максимума громкости) и переданного параметра. То же самое но уже в обратную сторону с уменьшением до 0 происходит с концом копозиции в ходе её генерации.

```
172  def generate_tones(creation, attitude, duration, attenuation, max_volume):
173      print('generate_tones call', len(attitude), duration, attenuation)
174      step = max_volume/ attenuation
175      volume = 0
176
177      for i in range(len(attitude)):
178          if i < attenuation:
179              volume += step
180          if i > duration - attenuation:
181              volume -= step
182
183          creation = generic_note(creation, attitude[i][2], \
184              (attitude[i][1]-attitude[i][0])*50, volume)
185
186      return creation
```

Рисунок 4. Вариант создания композиции по заданному массиву нот с продолжительностью с простейшим вариантом увеличения и уменьшения громкости

3. Наложение фрагмента чужой записи

В качестве чужой накладываемой записи выбран wav файл звука с сайта noiize.com C_BowedSteel_SP_304_01.wav, наиболее складно звучащий уху на одновременное проигровнаие. Сам код комбинирования двух аудиозаписей не предполагает обязательного соответствия длин файлов. Составление происходит путем накладывания первого сегмента на второй. Отрываются неизменяемые объекты, представляющие собой сегменты аудио, с помощью pydub.AudioSegment.from_file() с переданными параметрами наименования файла и его расширения.
overlay(seg , position = 0 , loop = False , times = None) - наложение предоставленного сегмента на другой сегмент, начиная с указанной позиции и используя указанное поведение зацикливания.

- seg (AudioSegment) - Второй накладываемый аудио сегмент, чтобы наложить на тот к которому применяется метод как к объекту AudioSegment.
- position (необязательно int) - Позиция начала наложения предоставленного сегмента.
- loop (необязательный bool) - Петление, цикл seg выполнится столько раз, сколько необходимо для соответствия длине этого сегмента. Переопределяет циклы param.
- времена (необязательно int) - Цикл seg указывающий количество раз или момент когда отрывок не будет соответствовать длине этого сегмента. «1» означает «один раз», «2» означает «дважды» и тд.

export(out_f=None, format="mp3", codec=None, bitrate=None, parameters=None, tags=None, id3v2_version="4") - экспорт AudioSegment в файл с заданными параметрами

- out_f (строка) - путь к аудиофайлу назначения

- format (строка) - формат аудио файла назначения. ('mp3', 'wav', 'ogg' или другие файлы, поддерживаемые ffmpeg / avconv)
- codec (строка) - кодек используется для кодирования по назначению.
- bitrate (строка) - битрейт, используемый при кодировании файла назначения. (128, 256, 312k…)
- parameters (строка) - дополнительные параметры ffmpeg / avconv
- tags (dict) - установка метаданных файлов назначения, обычно используемых в качестве тегов. ({title = 'Song Title', artist = 'Song Artist'})
- id3v2_version (строка) - установка версии ID3v2 для тегов. (по умолчанию: «4»)

```
282 if flg==0:
283     # Audio segment creation
284     creation = [0 for i in range(len(composition1)*2)]
285     creation = generate_tones(creation, composition1, len(composition1)*2, 8, 0.3)
286     print('generated')
287     print(creation)
288     write_wave('generated1.wav', creation)
289     print('written')
290 
291 elif flg==1:
292     # Audio segments combination
293     sound1 = AudioSegment.from_file('generated1.wav')
294     sound2 = AudioSegment.from_file('C_BowedSteel_SP_304_01.wav')
295     combined = sound1.overlay(sound2)
296     combined.export('combined1.wav', format='wav')
297 else:
298     # Getting grafics and spectrogramms to all files
299     get_grafics('generated1.wav')
300     get_spectrogramm('generated1.wav')
301     get_grafics('combined1.wav')
302     get_spectrogramm('combined1.wav')
303     get_grafics('05-Fall-Field-Cricket.wav')
304     get_spectrogramm('05-Fall-Field-Cricket.wav')
```

Рисунок 5. Вариант кода с вариацией выбора действий в зависимости от установленного флага

4. Сохранение в формате wav

Можно видеть на рис. 5. последовательность действий записи новой генерации звука по массиву композиции – задание переменных, вызов функции generate_toes к общей композиции, внутри которой производится генерация отдельной ноты, и сохранение в новом файле. Можно расскрыть отдельно функцию сохранения файла в формате wav – write_wave на следующем рисунке.

```
188 def write_wave(Name, Frames):
189     file = wave.open(Name, 'w')
190     file.setparams((1, 2, SAMPLE_RATE, len(Frames), 'NONE', 'not compressed'))
191     #file.setparams((1, 2, SAMPLE_RATE, len(Frames)*2, 'NONE', 'not compressed'))
192     result = []
193     for frame in Frames:
194         file.writeframes(struct.pack('h', int(frame)))
195     file.close()
```

Рисунок 6. Вариант кода сохранения файла формата wav

5. Визуальное отображение звука

Можно видеть на рис. 5 часть кода отвечающую за генерацию графиков и спектрограмм – последняя. Соответственно можно сразу указать на первое разделение – графики и спектрограммы. Используется несколько вариантов отораженния графиков: первый – отображение частоты длительностей частот, второй – отображение амплитуд фреймов. Для построения графиков потребовались библиотеки matplotlib и librosa, позволяющая выделить частоты и спектры также. Отсюда можно заметить что можно при помощи той же библиотеки librosa построить спектрограмму. Двушие два построения спектрограмм осуществляется с помощью библиотек scipy и matplotlib, чуть более стандартны не по отношению к звуку, как librosa.

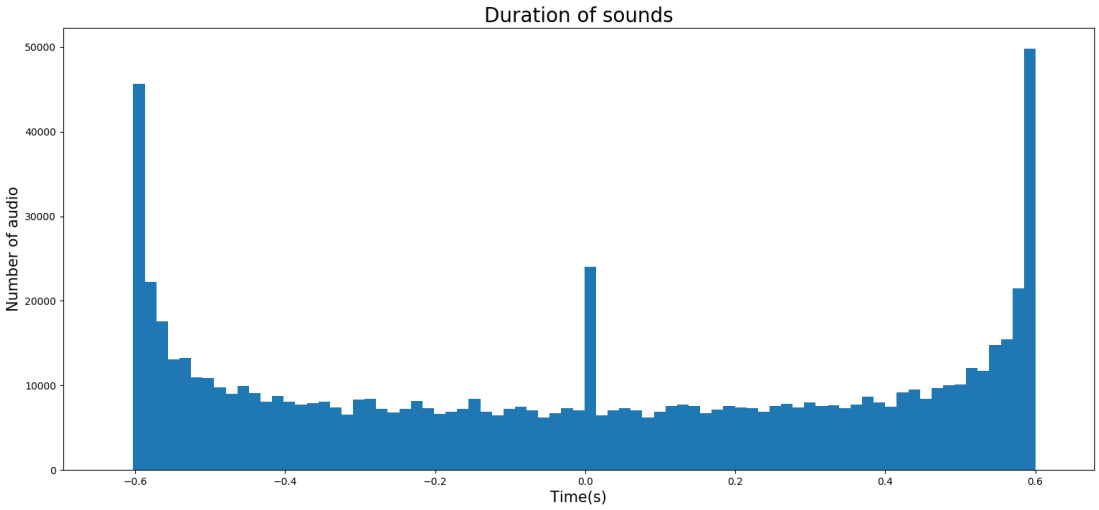


Рисунок 7. Гистограмма длительностей звуков (воспроизводимых частот) сгенерированного файла

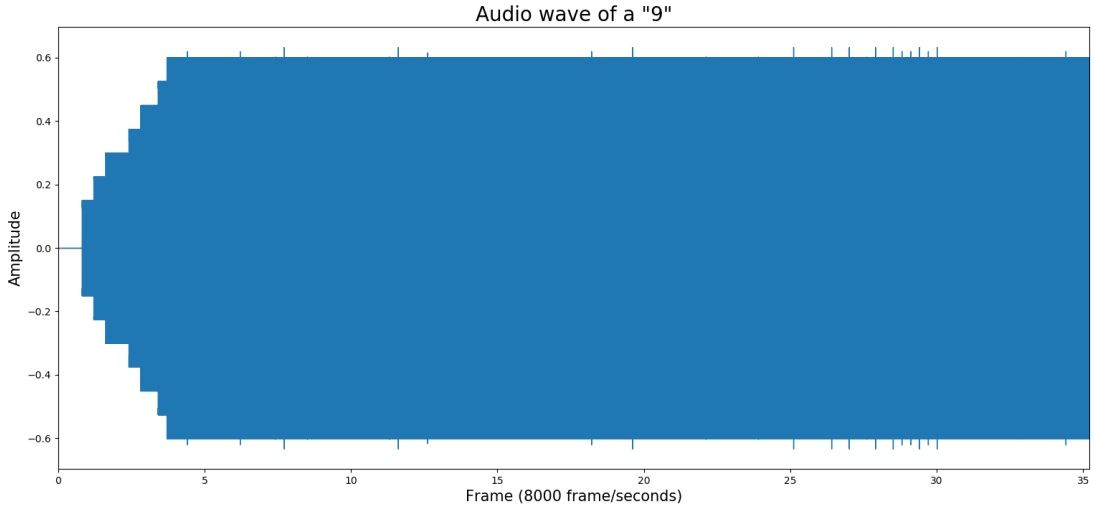


Рисунок 8. График амплитуд фреймов сгенерированного файла

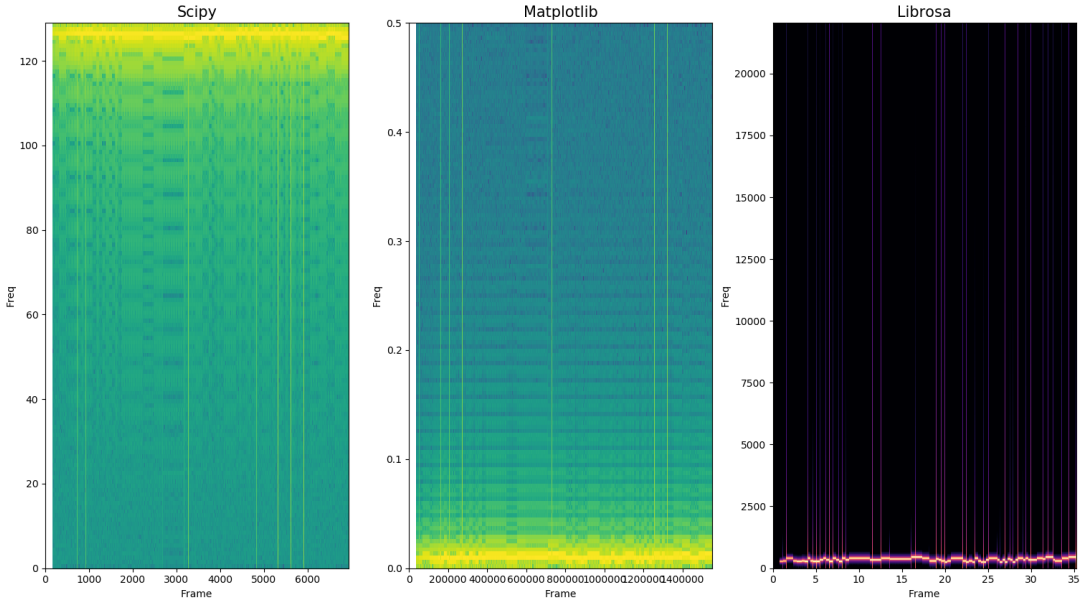


Рисунок 9. Спектрограммы сгенерированного файла

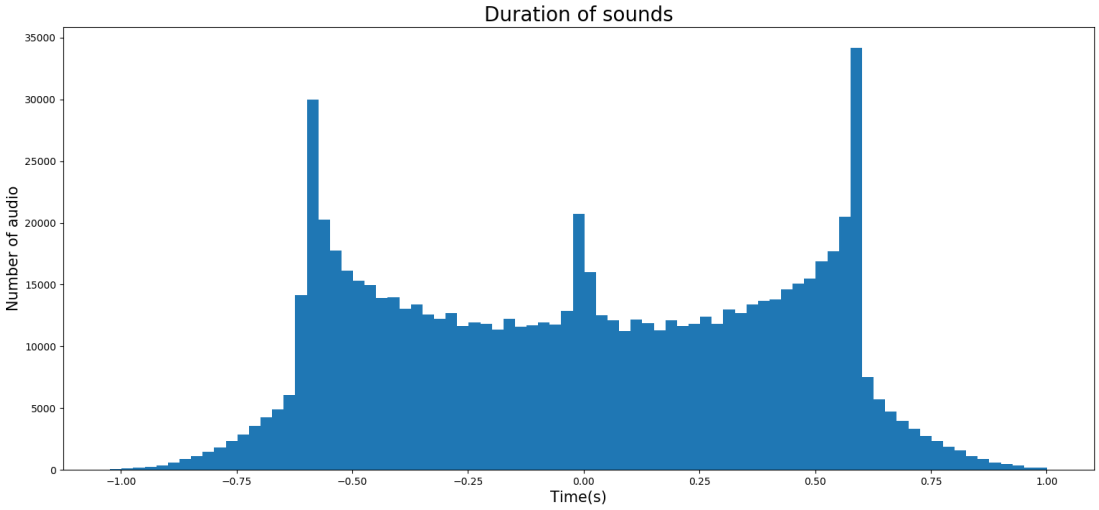


Рисунок 10. Гистограмма длительностей звуков (воспроизводимых частот) скомбинированного файла

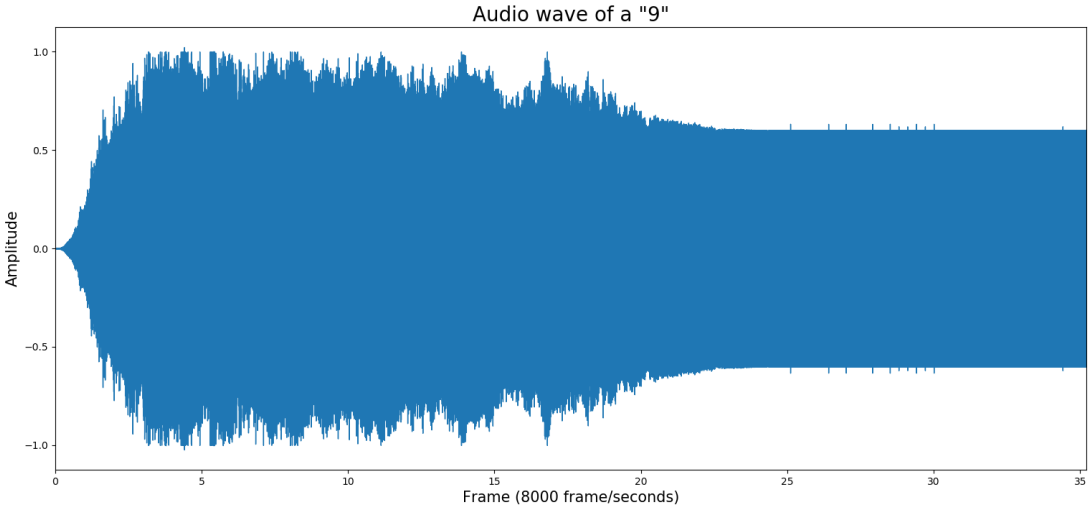


Рисунок 11. График амплитуд фреймов скомбинированного файла

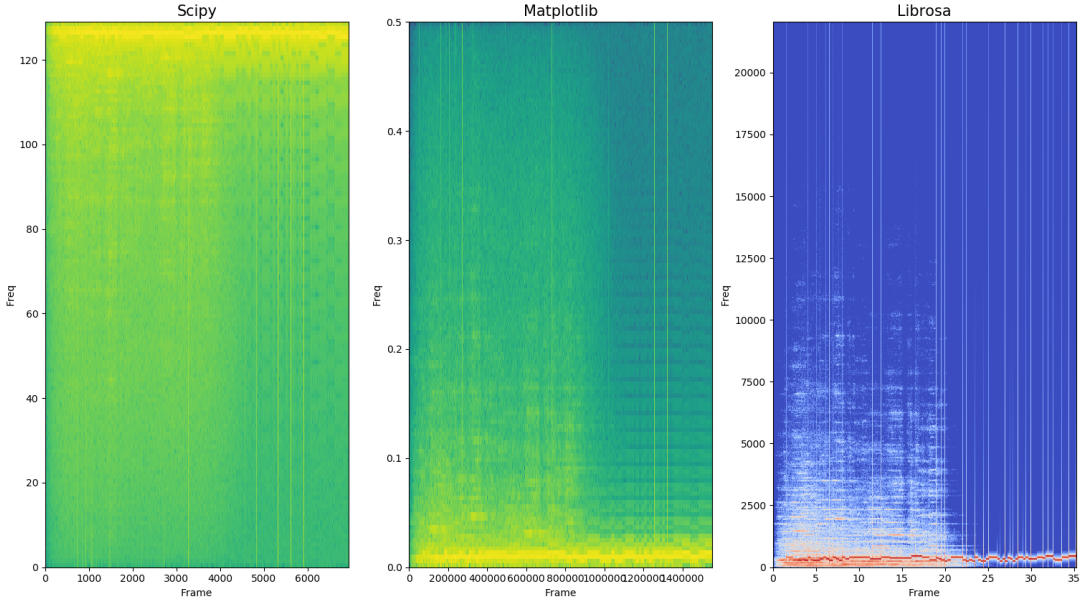


Рисунок 12. Спектрограммы скомбинированного файла

Выводы

Графики скомбинированного файла показывает, что аудио фрагменты достаточно схожи, как и было воспринято на слух. Так как наложение внешнего фрагмента на сгенерированный мало отражает именения, добавляя по частоте большую сгаживаемость – мягкость воспроизведения, большую амплитудную шероховатость – характер аудио.