ThinkDSP. Лабораторная 1. Звуки и сигналы.

Шерепа Никита

26 апреля 2021 г.

Содержание

Список иллюстраций

Листинги

1 Упражнение 1.2

1. **Задание**

   Скачайте с сайта *http://freesound.org* образец звука, включающий музыку, речь или иные звуки, имеющие четко выраженную высоту. Выделите примерно полсекундный сегмент, в котором высота постоянна. Вычислите и распечатайте спект выбранного сегмента. Как связаны тембр звука и гармоническая структруа, видимая в спектре?

   Использвуйте *high_ pass*, *low_ pass* и *band_ stop* для фильтрации тех или иных грамоник. Затем преобразуйте спектры обратно в сигнал и прослушайте его. Как звук соотносится с изменениями, сделанными в спектре?

2. **Ход работы**

   Вместо скачивания звука я записал свой - **/res/throat_singing.wav**

```
from thinkdsp import read_wave

wave = read_wave('res/throat_singing.wav')
wave.normalize()
wave.make_audio()
```

Листинг 1: Загрузка и прослушивание звука

   Звучит как какое-нибудь тувинское горловое пение.

   Затем я визуализировал звук

```
wave.plot()
```

Листинг 2: Визуализация звука

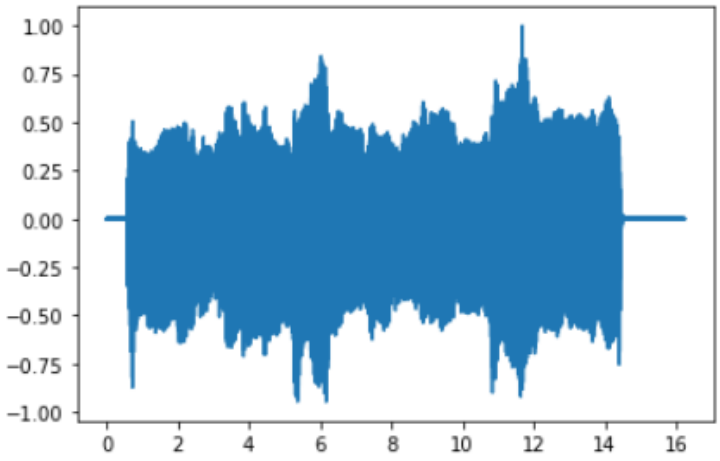

Рис. 1: Записанный звук

Затем я выделил полусекундный сегмент

```
segment = wave.segment(start=1.0, duration=0.5)
segment.make_audio()
```

Листинг 3: Сегмент и его прослушивание

И визуализировал его

```
segment.plot()
```

Листинг 4: Визуализация сегмента

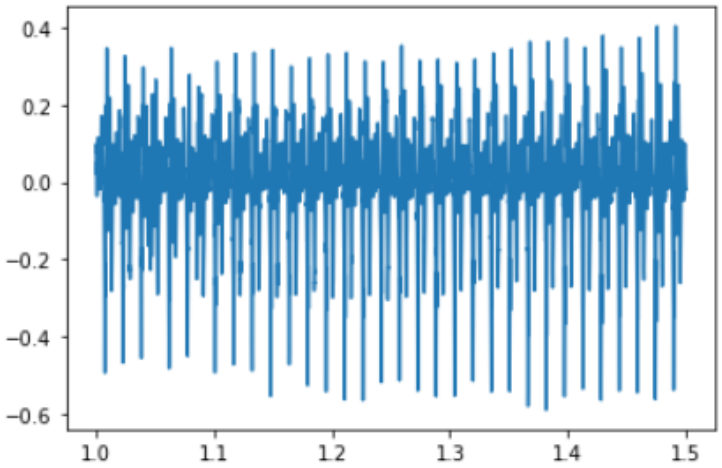

Рис. 2: Сегмент

```
segment.segment(start=1.1, duration=0.01).plot()
```

Листинг 5: Фрагмент сегмента

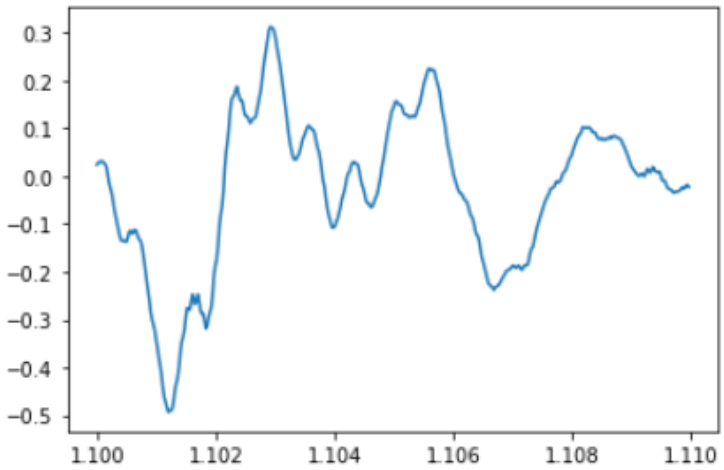

Рис. 3: Фрагмент сегмента

Затем я выделил спектр сегмента

```
spectrum = segment.make_spectrum()
spectrum.plot(high=7000)
```

Листинг 6: Спектр сегмента

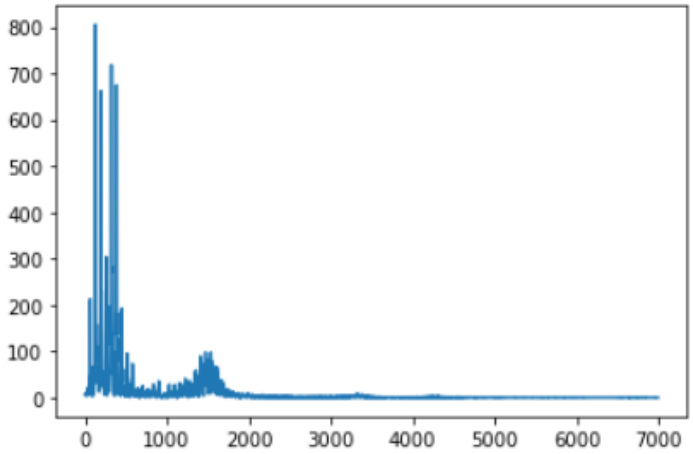

Рис. 4: Спектр сегмента

```
spectrum = segment.make_spectrum()
spectrum.plot(high=1000)
```

Листинг 7: Основные и доминирующие частоты

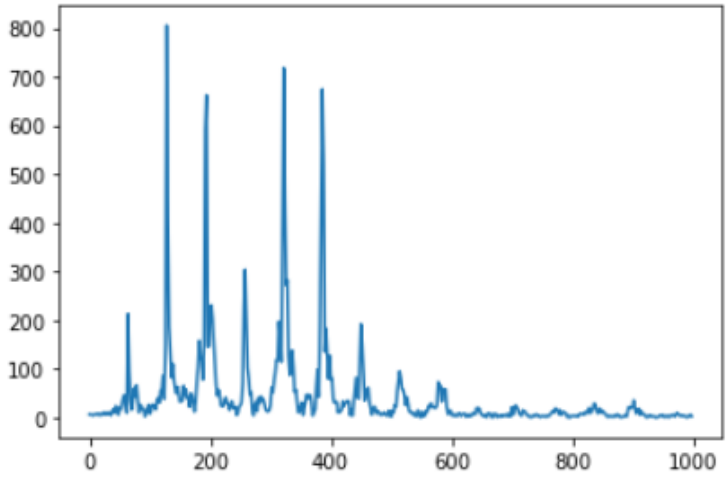

Рис. 5: Основные и доминирующие частоты

Теперь выведем самые высокие точки спектра и их частоты в порядке убывания

```
spectrum.peaks()[:30]

[(805.2011660028415, 128.0),
(717.7534137378161, 322.0),
(674.4257107121337, 386.0),
(662.1792673529742, 194.0),
(590.2975588029225, 192.0),
(522.1861101938315, 388.0),
(452.49523817628216, 324.0),
(451.2926648076347, 384.0),
(401.46804683039875, 130.0),
(386.27598232673955, 320.0),
(304.0156833122087, 258.0),
(282.4848989689322, 328.0),
(272.01525283371285, 326.0),
(230.78182369574293, 202.0),
(229.5602546915074, 256.0),
(222.41384500322584, 200.0),
```

```
(213.0448913102804, 64.0),
(203.73496641597413, 204.0),
(203.0809167531178, 260.0),
(199.3804441543911, 382.0),
(197.24287528975682, 314.0),
(192.08672045960517, 450.0),
(187.7791701290673, 132.0),
(182.9641942180879, 392.0),
(167.77207719800455, 316.0),
(160.52351987778476, 190.0),
(158.17239516488624, 182.0),
(155.35505888892268, 206.0),
(151.3961428346756, 126.0),
(150.398009639692, 452.0)]
```

Листинг 8: Пики спектра

Теперь отфильтруем гармоники

```
spectrum.low_pass(2000)
spectrum.make_wave().make_audio()

spectrum.make_wave().plot()
```

Листинг 9: Фильтрация гармоник: low_pass

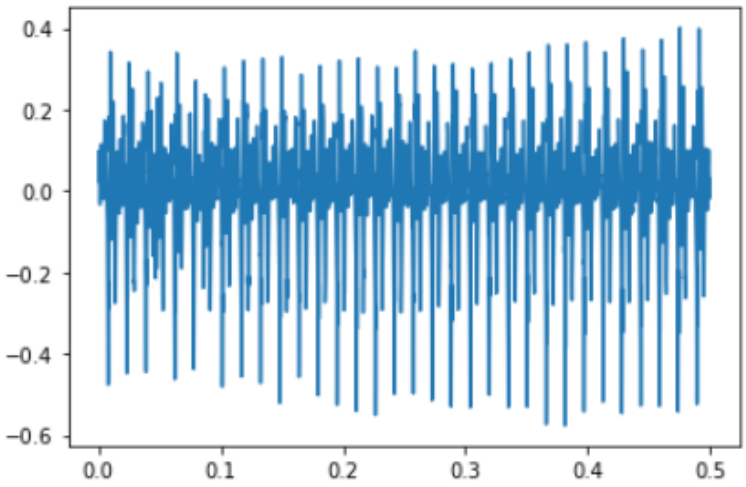

Рис. 6: Фильтрация гармоник: low_pass

Как видно из графика, звук поредел, а звучать он стал глухо, как-будто из-за стены.

```
spectrum.high_pass(2000)
spectrum.make_wave().make_audio()

spectrum.make_wave().plot()
```

Листинг 10: Фильтрация гармоник: high_pass

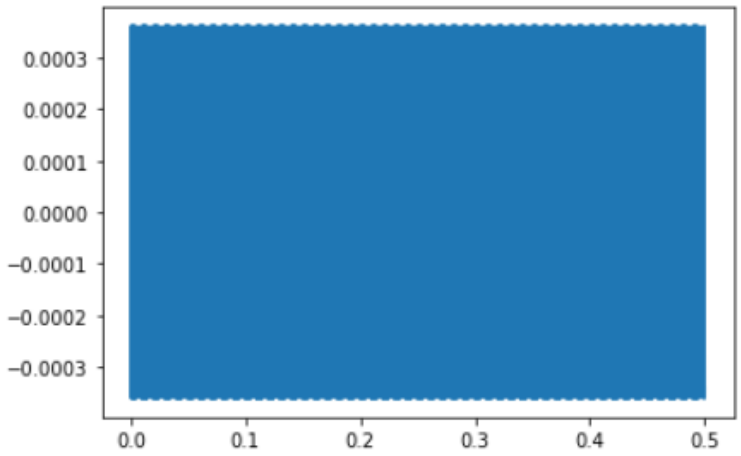

Рис. 7: Фильтрация гармоник: high_pass

Звук стал похож на высокочастотный писк, как у звукового сигнала перед записью сообщения на автоответчик на телефоне.

2 Упражнение 1.3

1. **Задание**

   Создайте сложный сигнал из объектов *SinSignal* и *CosSignal*, суммируя их. Обработайте сигнал для получения wave и прослушайте его. Вычислите *Spectrum* и распечатайте. Что произойдет при добавлении частотных компонент, не кратных основным?

2. **Ход работы**

   Создадим сложный сигнал

```
from thinkdsp import SinSignal

signal = (SinSignal(freq=300, amp=0.9) +
SinSignal(freq=520, amp=2.0) +
```

```
    SinSignal(freq=750, amp=0.3))
    signal.plot()
```

Листинг 11: Сложный сигнал

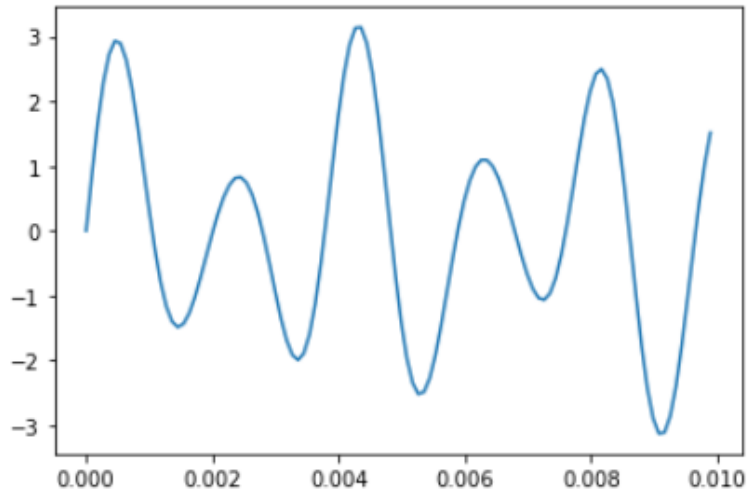

Рис. 8: Сложный сигнал

```
wave2 = signal.make_wave(duration=1)
wave2.apodize()
wave2.make_audio()
```

Листинг 12: Прослушивание сложного сигнала

Звук похож на какую-нибудь низкую ноту, прозвучавщую из терменвокса.

Теперь вычислим спектр сложного сигнала

```
spectrum = wave2.make_spectrum()
spectrum.plot(high=2000)
```

Листинг 13: Спектр сложного сигнала

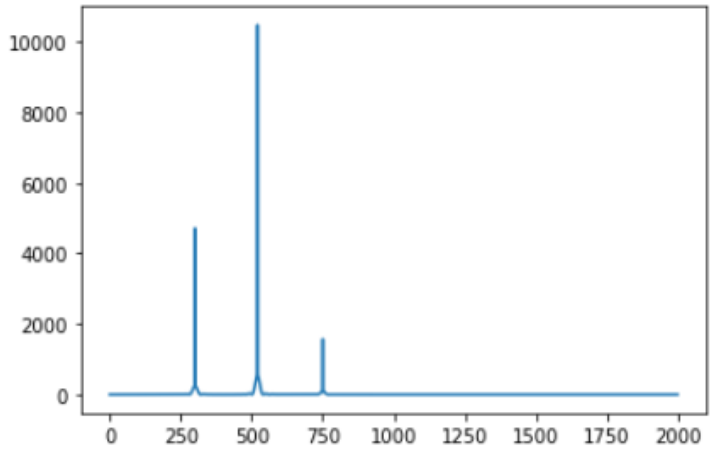

Рис. 9: Спектр сложного сигнала

Теперь добавим в звук новую частоту и прослушаем

```
signal += SinSignal(freq=850)
signal.make_wave().make_audio()
```

Листинг 14: Прослушивание спектра сложного сигнала

Звук стал выше (freq = 850).

Теперь звук похож на какую-нибудь высокую ноту, прозвучавщую из терменвокса.

3 Упражнение 1.4

1. **Задание**

   Напишите функцию *stretch*, берущую wave и коэффициент изменения. Она должна ускорять или замедлять сигнал изменением *ts* и *framerate*. Подсказка: должно получиться всего две строки кода.

2. **Ход работы**

   Напишем функцию **stretch** и прослушаем полученный результат

```
wave3 = read_wave('res/throat_singing.wav')
wave3.normalize()
wave3.make_audio()

def stretch(wave, factor):
```

```
    wave.ts *= factor
    wave.framerate /= factor
```

Листинг 15: Функция stretch

Попробуем замедлить звук

```
stretch(wave3, 1.5)
wave3.make_audio()
```

Листинг 16: Замедленние звука

Получился очень зловещий и низкий звук, как-будто какой-то огр недовольно мелодично рычит

Визуализируем результат

```
wave3.plot()
```

Листинг 17: Визуализация замедленного звука

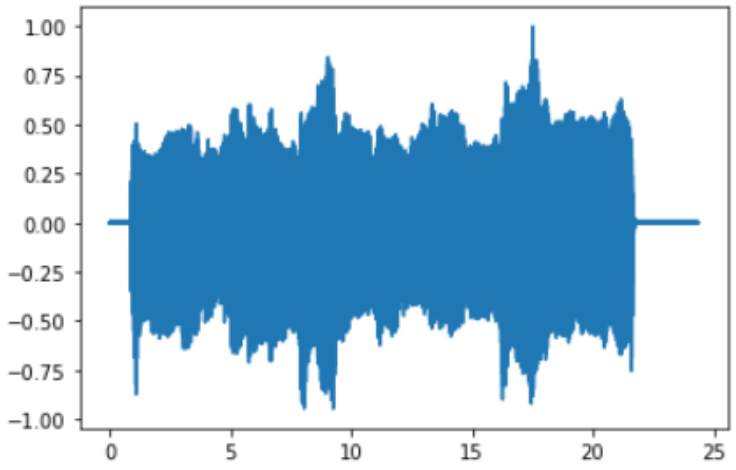

Рис. 10: Замедленный звук

Попробуем ускорить звук

```
stretch(wave3, 0.5)
wave3.make_audio()
```

Листинг 18: Ускоренние звука

Получился очень забавный высокий звук, как-будто какой-нибудь маленький тувинский мальчик осваивает горловое пение.

Визуализируем результат

```
wave3.plot()
```

Листинг 19: Визуализация ускоренного звука

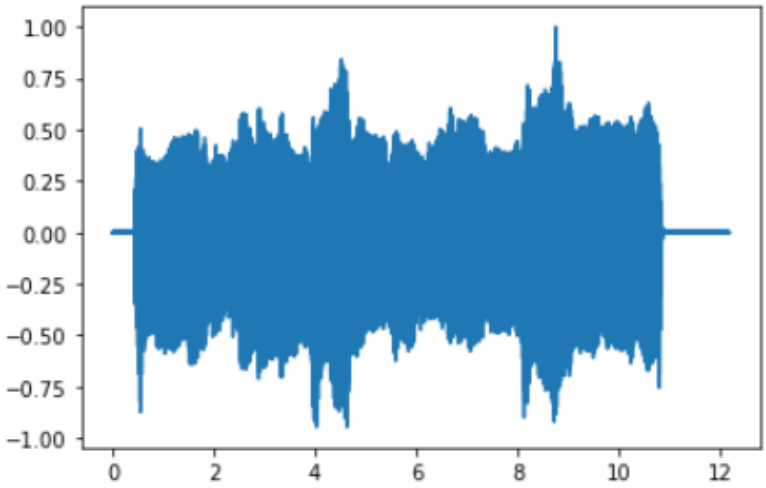

Рис. 11: Ускоренный звук

4 Вывод

В результате выполнения лабораторной работы получены навыки работы со звуками: обработка, понижение/повышение частоты, вычисление пиков. Оказывается, работа со звуками очень занимательное и в какой-то мере забавное занятие, потому что всегда интересно послушать, что получится на выходе, в результате обработки.