



федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий
Кафедра управления и интеллектуальных технологий

Отчёт по лабораторной работе №2
По дисциплине «Методы и алгоритмы обработки данных и изображений»

Выполнили студенты: Михайловский М. Ю., Озеров С. Д.

Группа: А-02м-25

Бригада: 2

Проверил: Бородкин А. А.

Содержание

1	Выполнение работы	3
1.1	Выявление голосовой активности	3
1.2	Фильтрация шумов	3
1.3	Запись и сегментация голосовых команд	5
2	Выводы	6
A	Листинги	7

1 ВЫПОЛНЕНИЕ РАБОТЫ

1.1 Выявление голосовой активности

Имеется звуковой файл в формате .wav, где диктор на протяжении минуты зачитывает текст. В данной записи имеются значительные шумы, которые громче самого диктора. График первых 1,5 секунд этого сигнала представлен на рис. 1.1.

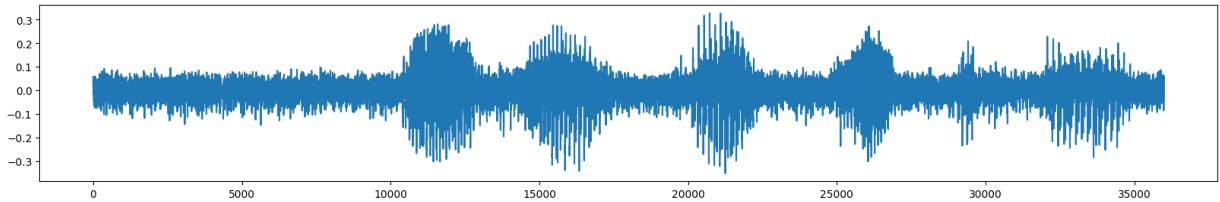


Рис. 1.1. Исходный речевой сигнал

Для выявления голосовой активности была написана программа, представленная на листинге 3. Первые 0.4 секунды были вырезаны в качестве фрагмента с примером шума. Этот фрагмент был разбит на кадры с перекрытием, и для каждого кадра рассчитана кратковременная энергия речевого сигнала:

$$E = \frac{1}{N} \sum_{k=1}^N s_k^2, \quad (1)$$

где s_k – отсчёты речевого сигнала.

Для шума было получено пороговое значение $E_{\text{пор}} = \max E = \max (E_1 \dots E_M)$, где значение M – количество кадров, на которое был разбит сигнал

Это значение будет использоваться для выявления речевой активности. Кадры, для которых $E < E_{\text{пор}}$ будут считаться участками молчания. Для рассматриваемого сигнала было получено значение $E_{\text{пор}} = 1,63 \cdot 10^{-3}$. Результаты выявления участков речи представлены на рис. 1.2.

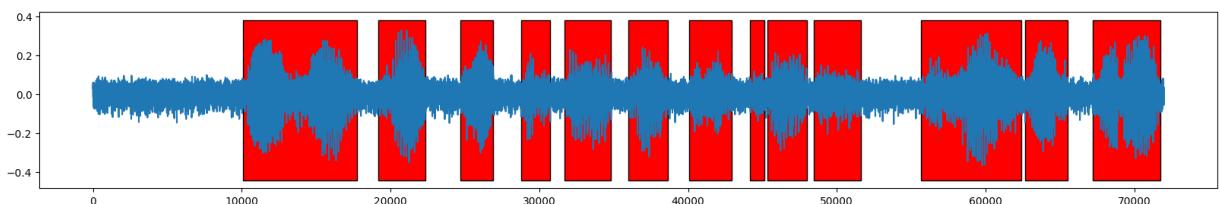


Рис. 1.2. Результат работы алгоритма выявления участков речи

Также этот алгоритм будет использован далее в работе.

1.2 Фильтрация шумов

Для фильтрации шумов было рассмотрено два подхода. Был реализован алгоритм, представленный в листинге 4.

Первый подход основан на функции `reduce_noise` из библиотеки `noisereduce`. Это модификация идеи метода спектральных вычитаний. Для разных параметров, этот алгоритм позволяет по-разному устранить шум, в том числе полностью его заглушить. Однако, для параметров, полностью устраниющих шум, рассматриваемый речевой сигнал становится

сильно заглушённым, в следствие чего малоразборчивым. Это связано с большим отношением шум/сигнал в данном случае.

Если подбирать параметры так, чтобы шум снижался не полностью, получается добиться хорошей слышимости при низком значении шума.

В этой модификации на фильтрацию влияют следующие параметры:

- **prop_decrease=1** – представляет собой коэффициент пропорциональности, определяющий степень подавления спектра шума. По своей сути это аналог параметра A из классического метода спектрального вычитания;
- **thresh_n_mult_nonstationary=1.5** – пороговое значение, в количестве среднеквадратических отклонений (СКО) сигнала от математического ожидания шума, для того, чтобы в сигнале было принято наличие речи. Косвенный аналог B из классического метода, представляющего собой минимальный уровень спектра.

В результате фильтрации этим методом один из полученных сигналов имел вид, представленный на рис. 1.3.

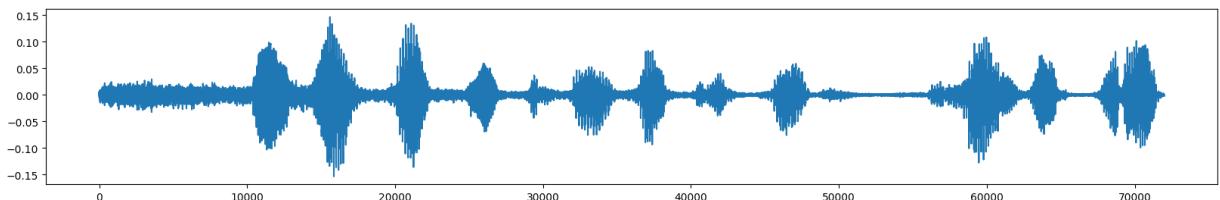


Рис. 1.3. Фильтрация шумов через `reduce_noise`

Второй подход основан на использовании цифровых фильтров низких частот. Если шум является высокочастотной аддитивной составляющей, то он таким фильтром может быть снижен.

Для фильтрации было использовано следующее разностное уравнение. $y[0] = \frac{s[0]}{\sqrt{1-\alpha^2}}$:

$$y[n] = \alpha \cdot y[n-1] + \sqrt{1-\alpha^2} \cdot s[n] \quad (2)$$

Здесь полученный вектор $y[n]$ – представляет собой отфильтрованный сигнал $s[n]$. Полученный результат для $\alpha = 0,99$ представлен на рис. 1.4. В отфильтрованном сигнале всё ещё слышен шум, однако он стал значительно тише, по сравнению с голосом диктора. Сам голос диктора сохранился разборчивым.

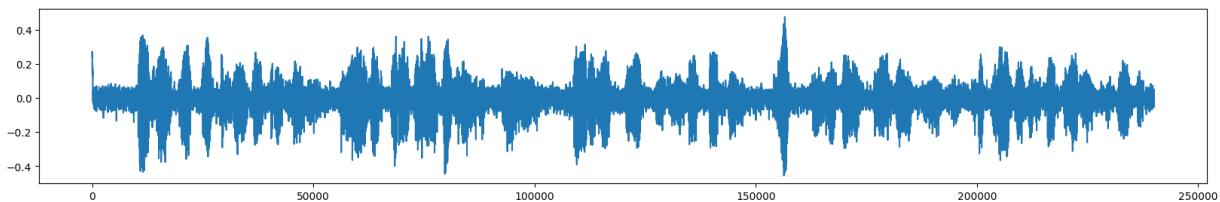


Рис. 1.4. Фильтрация шумов через цифровой низкочастотный фильтр

1.3 Запись и сегментация голосовых команд

Была написана программа для записи реализаций голосовых команд, представленная на листинге 2. С помощью неё были записаны выборки команд, по выбору направлений и действий роботом, в ходе исследования территории.

В данном алгоритме первая секунда используется для определения $E_{\text{пор}}$ шума. Все последующие фрагменты, на которых выявлена голосовая активность, и которые не являются слишком короткими, вырезаются в отдельные .wav файлы.

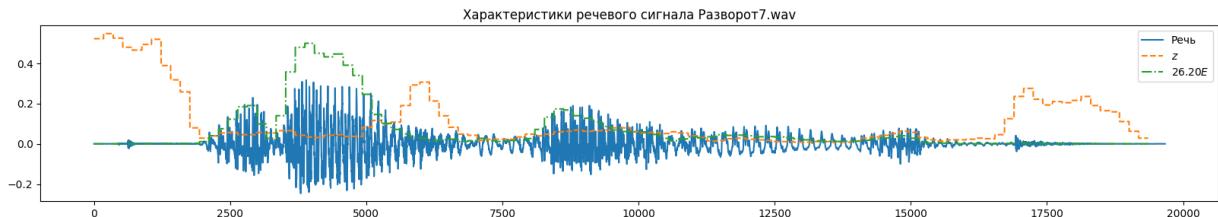


Рис. 1.5. Фильтрация шумов через цифровой низкочастотный фильтр

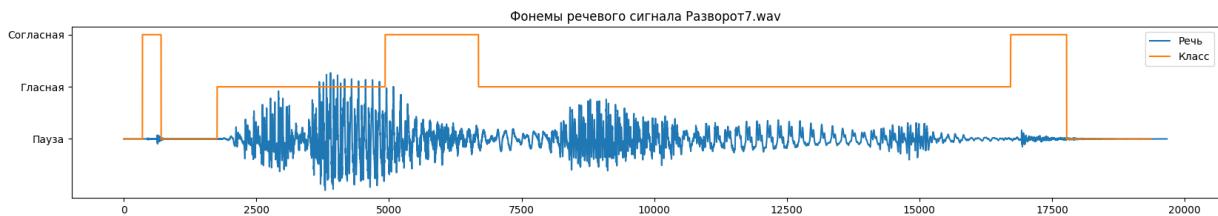


Рис. 1.6. Фильтрация шумов через цифровой низкочастотный фильтр

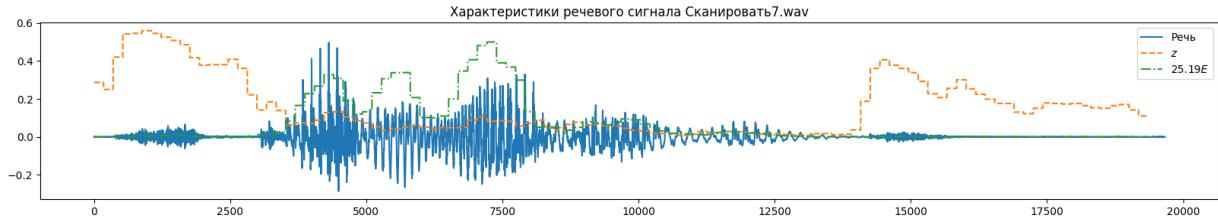


Рис. 1.7. Фильтрация шумов через цифровой низкочастотный фильтр

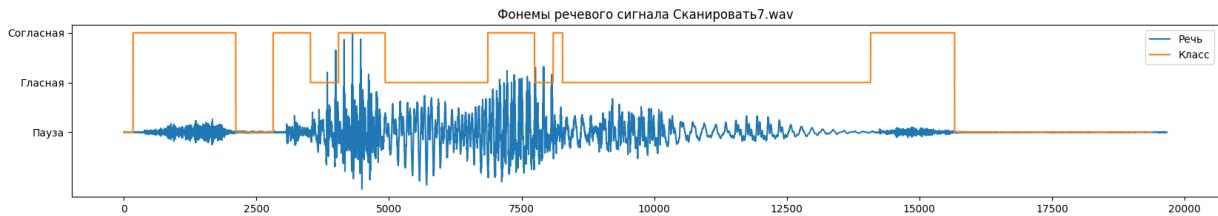


Рис. 1.8. Фильтрация шумов через цифровой низкочастотный фильтр

Для классификации фонем был написан алгоритм, описанный в листинге 5. Здесь сформирована функция, которая по данным пороговым значениям кратковременной энергии речевого сигнала $E_{\text{пор}}$ и числу нулей интенсивности $z_{\text{пор}}$ классифицирует кадры речевого сигнала на классы: пауза, гласный, согласный. Пороговые значения были подобраны вручную.

Рассчитанные характеристики для кадров, представлены на рис. 1.5, 1.7. А классифицированные фонемы, приведены на рис. 1.6, 1.8. Были взяты два слова, из ранее записанного словаря.

Можно видеть, что чётко определяются гласные. Некоторые согласные приняты за гласные, что может быть связано с их малым временем произношения в середине слова, а также с особенностями произношения конкретного диктора.

2 ВЫВОДЫ

В данной работе были рассмотрены 3 темы по обработке речевых сигналов.

Был рассмотрен алгоритм выявления голосовой активности. Он успешно выделил участки речи, в анализируемом сигнале.

Были рассмотрены два алгоритма фильтрации шумов. Метод, основанный на анализе спектров, позволяет снизить шум до требуемого уровня, за счёт чёткости речи. Метод, основанный на цифровой фильтрации, частично снижает наличие шума в сигнале.

Был реализован алгоритм записи голосовых команд, и автоматической вырезки их из одной записи. Для команд был реализован алгоритм сегментации, который проводит классификацию кадров на паузы, гласные и согласные. Этот алгоритм хорошо определяет наличие пауз, но склонен пропускать некоторые согласные в угоду гласным.

Приложение А. Листинги

Листинг 1. Файл dataprocessing.py

```

1 import numpy as np
2 import soundfile as sf
3 import librosa.feature as ft
4
5 def dataToEnergy(data, sr, frame_time, frame_shift, do_print = True):
6     frameWidth = int(frame_time * sr)
7     frameShift = int(frame_shift * frameWidth)
8
9     frameCount = int(data.size / (frameWidth - frameShift)) - 1
10    if do_print:
11        print(f'{frameWidth=}')
12        print(f'{frame_shift=}')
13        print(f'{frameCount=}')
14
15    E = [0.0] * frameCount
16    sh = 0
17    df = frameWidth - frameShift
18    for i in range(frameCount):
19        En = 0
20        for j in range(frameWidth):
21            sn = float(data[j + sh] ** 2)
22            En = En + sn
23
24        E[i] = En / frameWidth
25        sh = sh + df
26
27    return E
28
29 def getEMax(file_name, seconds_from_start, frame_time, frame_shift):
30     data, samplerate = sf.read(file_name)
31
32     noise_indices = int(seconds_from_start * samplerate)
33     noise_data = data[:noise_indices]
34
35     E = dataToEnergy(noise_data, samplerate, frame_time, frame_shift, do_print=False)
36     return np.max(E)
37
38 def VAD(data, sr, frame_time, frame_shift, noise_frame_end, eTh):
39     frameWidth = int(frame_time * sr)
40     frameShift = int(frame_shift * frameWidth)
41
42     frameCount = int(data.size / (frameWidth - frameShift)) - 1
43
44     maxY = np.max(data)
45     df = frameWidth - frameShift
46     E = [0] * frameCount
47     sh = 0
48     for i in range(frameCount):
49         En = 0
50

```

```

51     for j in range(frameWidth):
52         sn = float(data[j + sh] ** 2)
53         En = En + sn
54
55     E[i] = En / frameWidth
56     sh = sh + df
57
58     if noise_frame_end > 0:
59         ePorog = 0
60         for j in range(0, noise_frame_end):
61             if ePorog < E[j]:

```

Листинг 2. Файл recording.py

```

1 import wave
2 import pyaudio as py
3 import keyboard
4 from array import array
5 import os
6 import dataprocessing as dp
7 import numpy as np
8 import soundfile as sf
9
10 def record_voice(file_to_save, descriptive = True):
11     CHUNK = 1024
12     FORMAT = py.paInt16
13     CHANNELS = 1
14     RATE = 22050
15
16     p = py.PyAudio()
17     stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True,
18                      frames_per_buffer=CHUNK)
19
20     data_all = array('h')
21
22     recording = True
23
24     def key_press(e):
25         nonlocal recording
26         if e.name == 'enter':
27             recording = False
28             keyboard.unhook_all()
29
30             keyboard.hook(key_press)
31
32             start_promted = False
33             while recording:
34                 if stream.is_active():
35                     if not start_promted:
36                         start_promted = True
37                         if descriptive:
38                             print("Началась запись")

```

```

39         data = stream.read(CHUNK, exception_on_overflow=False)
40         data_all.extend(array('h', data))
41
42     if descriptive:
43         print('Вышли из цикла')
44
45     stream.stop_stream()
46     stream.close()
47     p.terminate()
48     if descriptive:
49         print('Запись остановлена')
50
51     wav = wave.open(file_to_save, "wb")
52     wav.setnchannels(CHANNELES)
53     wav.setframerate(RATE)
54     wav.setsampwidth(p.get_sample_size(FORMAT))
55     wav.writeframes(data_all)
56     wav.close()
57     if descriptive:
58         print(f"Запись сохранена в {file_to_save}")
59
60 def main():
61     FRAME_TIME = 20E-3

```

Листинг 3. Выявление участков голосовой активности

```

1 import numpy as np
2 import soundfile as sf
3 import matplotlib.pyplot as plt
4 from matplotlib.patches import Rectangle
5 import noisereduce as nr
6 import dataprocessing as dp
7 import librosa.feature as ft
8 from enum import Enum
9
10 # Считывание файла с речью
11 data, samplerate = sf.read('Y.wav')
12 seconds_to_plot = 1.5
13 indices_to_plot = int(seconds_to_plot * samplerate)
14 plt.rcParams["figure.figsize"] = (20,3)
15 plt.plot(data[:indices_to_plot])
16 plt.show()
17
18 # Сохранение шума в начале
19 noise_seconds = 0.4
20 noise_indices = int(noise_seconds * samplerate)
21 noise_data = data[:noise_indices]
22 dp.resave('Y.wav', 0, noise_indices, 'Y_noise.wav')
23
24 # Получение выборки кратковременных энергий кадров речевого сигнала
25 frame_time = 20e-3
26 frame_shift = 0.5
27

```

```

28
29 E = dp.dataToEnergy(noise_data, samplerate, frame_time, frame_shift, do_print=False)
30 E_noise_max = np.max(E)
31 print(f'E_noise_max={:.2e}')
32 plt.rcParams["figure.figsize"] = (6,4)
33 plt.plot(E)
34 plt.show()
35
36 # Выявление голосовой активности
37 vadData, tickData = dp.VAD(data, samplerate, frame_time, frame_shift, noise_frame_end=0, eTh=
38   ↪ E_noise_max)
39 seconds_to_plot = 3
40 indices_to_plot = int(seconds_to_plot * samplerate)
41 ticks = []
42 for tick in tickData:
43     if tick > indices_to_plot:
44         break
45
46     ticks.append(tick)
47
48 plt.rcParams["figure.figsize"] = (20,3)
49 fig, ax = plt.subplots()
50 for i in range(int(len(ticks) / 2)):
51     left_tick = ticks[2*i]
52     right_tick = ticks[2*i + 1]
53     rect = Rectangle((left_tick, np.min(data)), width=right_tick - left_tick, height =
54       ↪ np.max(data) - np.min(data), facecolor='r', linewidth=1, edgecolor='k')
55     ax.add_patch(rect)
56 ax.plot(data[:indices_to_plot])
57 plt.show()

```

Листинг 4. Фильтрация шума

```

1 # Метод recude_noise (Модификация метода спектральных вычислений)
2 filtered_data = nr.reduce_noise(y=data, sr=samplerate, y_noise=noise_data, prop_decrease=1,
3   ↪ thresh_n_mult_nonstationary=1.5, stationary=False)
4 plt.rcParams["figure.figsize"] = (20,3)
5 plt.plot(filtered_data[:indices_to_plot])
6 plt.show()
7 sf.write('Y_filtered12.wav', filtered_data, samplerate)
8
9 # Цифровая фильтрация
10 def digital_filter(data, alpha):
11     N = len(data)
12     y = np.zeros(N)
13     y[0] = data[0] / np.sqrt(1 - alpha**2)
14     for n in range(1, N):
15         y[n] = alpha * y[n-1] + np.sqrt(1 - alpha**2) * data[n]
16
17     return y
18
19 seconds_to_plot = 10

```

```

20 indices_to_plot = int(seconds_to_plot * samplerate)
21
22 filtered_data2 = digital_filter(data, 0.99)
23 plt.rcParams["figure.figsize"] = (20,3)
24 plt.plot(filtered_data2[:indices_to_plot])
25 plt.show()
26
27 sf.write('Y_filtered2.wav', filtered_data2, samplerate)

```

Листинг 5. Определение типов фонем

```

1 word1_path = './notebook/Разворот/Разворот7.wav'
2 word2_path = './notebook/Сканировать/Сканировать7.wav'
3
4 data1, samplerate1 = sf.read(word1_path)
5 data2, samplerate2 = sf.read(word2_path)
6
7 frame_time = 16e-3
8 frame_length = int(frame_time * samplerate1)
9 frame_shift_ratio = 0.5
10 frame_shift = int(frame_shift_ratio * frame_length)
11
12 class Phoneme(Enum):
13     Pause = 0
14     Vowel = 1
15     Consonant = 2
16
17     def __int__(self):
18         return self.value
19
20 def scale_to_plot(data):
21     to_plot_data = []
22     for zero_crossing_rate in data:
23         to_plot_data += [zero_crossing_rate] * frame_shift
24
25     return to_plot_data
26
27 def calc_plot_features(data, samplerate, frame_length, frame_shift, E_max, z_max, data_name:str =
28     ''):
29     # Рассчитаем характеристики
30     E = dp.dataToEnergy(data, samplerate, frame_time, frame_shift_ratio, do_print=False)
31     E = np.array(E)
32     E_max_calc = np.max(E)
33     scaler = 0.5/E_max_calc
34     E *= scaler
35     zero_crossing_rates = ft.zero_crossing_rate(data, frame_length=frame_length,
36         hop_length=frame_shift, center=False)
37
38     # Классификация фонем для кадров
39     phoneme_classes = np.empty(len(E), dtype=object)
40     pause_mask = E < E_max
41     vowel_mask = (~pause_mask) & (zero_crossing_rates[0] < z_max)
42     consonant_mask = ~(pause_mask | vowel_mask)
43
44     phoneme_classes[pause_mask] = Phoneme.Pause

```

```
43     phoneme_classes[vowel_mask] = Phoneme.Vowel
44     phoneme_classes[consonant_mask] = Phoneme.Consonant
45
46     # Преобразование для построения графиков
47     to_plot_crossing_rate = scale_to_plot(zero_crossing_rates.flatten())
48     to_plot_E = scale_to_plot(E)
49     to_plot_classes = scale_to_plot(phoneme_classes)
50
51     plt.title(f'Характеристики речевого сигнала {data_name}')
52     plt.plot(data, label='Речь')
53     plt.plot(to_plot_crossing_rate, '--', label='$z$')
54     plt.plot(to_plot_E, '-.', label=f'${scaler:.2f}E$')
55     plt.legend()
56     plt.show()
57
58     plt.title(f'Фонемы речевого сигнала {data_name}')
59     plt.plot(data, label='Речь')
60     plt.plot(0.25*np.int16(to_plot_classes), label='Класс')
61     plt.yticks(
62         [0, 0.25, 0.5],
63         ['Пауза', 'Гласная', 'Согласная']
64     )
65     plt.legend()
66     plt.show()
67
68     return E, zero_crossing_rates
69
70 E_max = 1e-4
71 z_max = 0.08
72
73 calc_plot_features(data1, samplerate1, frame_length, frame_shift, E_max, z_max,
74     ↵ word1_path.split('/')[-1])
75 calc_plot_features(data2, samplerate2, frame_length, frame_shift, E_max, z_max,
76     ↵ word2_path.split('/')[-1])
```