

VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMACINIŲ SISTEMŲ INŽINERIJA

**Darbas Nr. 2**

**GARSO SIGNALŲ ANALIZĖ LAIKO SRITYJE**

Garso signalų apdorojimas

Atliko: Paulius Zaranka

Vilnius

2022

1. DARBO TIKSLAS IR UŽDAVINIAI

**Darbo tikslas.** Garso signalų analizė laiko srityje, analizės pritaikymas signalui, signalo segmentams įraše aptikti.

**Darbo uždaviniai.** Sukurti priemonę garso failams nuskaityti, nuskaitytiesiems signalams (ar jų atkarpoms) grafiškai atvaizduoti. Įvertinti signalo energijos bei nulio kirtimų skaičiaus (NKS) kitimą laike. Pritaikius slenksčio principą pabandyti aptikti signalo atkarpas, segmentus ar kitus vienetus įrašuose.

1. DARBO REZULTATAI

Darbe analizuojama 3 sekundžių *mono* muzikos įrašo iškarpa naudojant Python programavimo kalbą.

Žemiau pateikta signalo laiko diagrama.

Chart, line chart

Description automatically generated

pav. Signalo laiko diagrama

Tuomet buvo apskaičiuota signalo energijos diagrama. Slenkstinė riba nustatyta 0,02. Slenkstį peržengiančios atkarpos gali būti laikomos naujo signalo dalimis, o likusios – tylos atkarpomis. Žemiau pateikta signalo energijos diagrama.

Chart, line chart, histogram

Description automatically generated

pav. Signalo energijos diagrama

Galiausiai atlikta nulio kirtimų skaičiaus (NKS) analizė. Slenkstinė riba nustatyta sąlyginai didelė – 0,1. Žemiau pateiktoje diagramoje galima įžvelgti, kad signale gana tolygiai per visą signalą yra pasiskirsčiusios atkarpos, kurios peržengia slenkstinę ribą, t.y. pasižymi neįprastai aukštu dažniu.

Chart, line chart, histogram

Description automatically generated

pav. Signalo NKS diagrama

1. APIBENDRINIMAS IR IŠVADOS

Garso signalo analizei atlikti sukurta *Python* programa, leidžianti pasirinkti „wav“ tipo failą sistemoje ir apskaičiuojanti bei atvaizduojanti *mono* bei *stereo* failų laiko, energijos bei NKS diagramas. Energijos ir NKS diagramoms nustatytos skirtingos, konkrečiai diagramai reikšmingos slenkstinės ribos ir jos taip pat atvaizduojamos diagramose.

Atlikus analizę galima teigti, kad analizuojamas garso signalas yra gana tolygiai pasiskirstęs visomis analizuojamomis reikšmėmis.

Signalo energijos reikšmės iki maždaug 0,5s neperžengia slenkstinės ribos, todėl atlikus apdorojimą pritaikant 0,2 slenkstinę ribą signalas sutrumpėtų puse sekundės.

Dažnis pakyla maždaug nuo 0,25s ir yra gana tolygus visame likusiame signale išskyrus kelias per visa signalą pasiskirsčiusias trumpas atkarpas, kuriose dažnis pakyla neįprastai aukštai ir peržengią aukštą slenkstinę ribą.

1. Priedai

import wave

import tkinter as tk

from tkinter import filedialog

from scipy.io import wavfile

import matplotlib.pyplot as plt

import numpy as np

ENERGY\_THRESHOLD = 0.02

NKS\_THRESHHOLD = 0.1

def chooseFile():

return filedialog.askopenfilename(filetypes=(

("wav files", "\*.wav"), ("all files", "\*.\*")))

def readWav(path):

return wave.open(path, 'rb')

def processFile(file):

framerate = file.getframerate()

n\_frames = file.getnframes()

n\_channels = file.getnchannels()

l\_audio = n\_frames / framerate

b\_signal = file.readframes(n\_frames)

v\_signal = np.frombuffer(b\_signal, dtype=np.int16)

norm\_values = v\_signal / v\_signal.max()

times = np.linspace(0, (n\_frames - 1) / framerate, num=n\_frames)

visualizeChannels(norm\_values, n\_channels, times, l\_audio)

l\_frame = int(framerate / 1000 \* 25)

l\_hop = int(l\_frame / 2)

energy = getEnergy(norm\_values, l\_frame, l\_hop)

visualize(energy, l\_audio, 'T(s)', 'Energy',

'Sound Energy', ENERGY\_THRESHOLD)

zcr = getZeroCrossRate(norm\_values, l\_frame, l\_hop)

visualize(zcr, l\_audio, 'T(s)', 'ZCR',

'Zero Cross Rate', NKS\_THRESHHOLD)

def visualizeChannels(norm\_values, n\_channels, times, length):

if n\_channels > 1:

visualizeStereoChannels(norm\_values, n\_channels, times, length)

return

visualizeMonoChannel(norm\_values, times, length)

def visualizeMonoChannel(norm\_values, times, length):

plt.figure(figsize=(13, 4))

plt.plot(times, norm\_values, zorder=1)

plt.title('Channel')

plt.ylabel('A')

plt.xlabel('T(s)')

plt.xlim(0, length)

plt.show()

def visualizeStereoChannels(norm\_values, n\_channels, times, length):

figure, plotChannel = plt.subplots(n\_channels, figsize=(13, 6))

for i in range(n\_channels):

plotChannel[i].set\_title(f"Channel #{i}")

plotChannel[i].set\_ylabel('A')

plotChannel[i].set\_xlabel('T(s)')

plotChannel[i].plot(times, norm\_values[i::n\_channels], zorder=1)

plotChannel[i].set\_xlim(0, length)

print('hey')

figure.tight\_layout()

plt.show()

def visualize(data, length, x\_label, y\_label, title, threshold):

time = np.linspace(0, length, num=len(data))

plt.figure(figsize=(13, 4))

plt.plot(time, data)

plt.title(title)

plt.xlabel(x\_label)

plt.ylabel(y\_label)

plt.axhline(threshold, color='r')

plt.show()

def getEnergy(signal\_values, l\_frame, l\_hop):

energy = []

for i in range(0, len(signal\_values), l\_hop):

energy.append(

np.sum(signal\_values[i: i + l\_frame] \*\* 2) / l\_frame)

return energy

def sign(s):

if s >= 0:

return 1

return -1

def getZeroCrossRate(signal\_values, l\_frame, l\_hop):

zcr = []

for i in range(0, len(signal\_values), l\_hop):

frame\_zcr = 0

for f in range(i, i + l\_hop):

if f < len(signal\_values):

frame\_zcr += abs(sign(signal\_values[f]) -

sign(signal\_values[f - 1]))

zcr.append(frame\_zcr / (2 \* l\_frame))

return zcr

def main():

root = tk.Tk()

root.withdraw()

file\_name = chooseFile()

root.update()

file = readWav(file\_name)

processFile(file)

main()