Professur
Allgemeine Nachrichtentechnik

Prof. Dr.-Ing. Udo Zölzer

HELMUT SCHMIDT
UNIVERSITÄT

Web Implementation of Dynamic Range Control Audio Applet

Final Presentation - Project Project

Aritra Mazumdar

29 November 2018

Outline

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Outline

HELMUT SCHMIDT
UNIVERSITÄT

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Introduction

Dynamic Range Control

Dynamic range of a signal is defined as the logarithmic ratio of maximum to minimum amplitude of a signal and is expressed in dB.

Introduction
Dynamic Range Control

Dynamic range of a signal is defined as the logarithmic ratio of maximum to minimum amplitude of a signal and is expressed in dB.

Dynamic Range of a signal can be controlled using an application named as **Dynamic Range Control**. It can amplify low level sounds or reduce the volume of high level sounds.

Introduction
Dynamic Range Control

Dynamic range of a signal is defined as the logarithmic ratio of maximum to minimum amplitude of a signal and is expressed in dB.

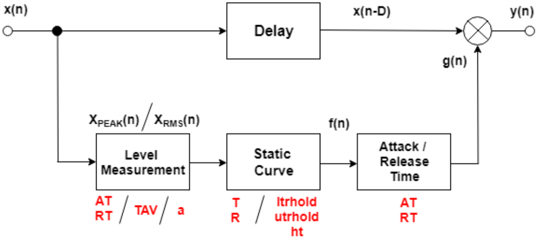
Dynamic Range of a signal can be controlled using an application named as **Dynamic Range Control**. It can amplify low level sounds or reduce the volume of high level sounds.

Application: While reproducing music or speech in a noisy environment like car or shopping mall, the dynamic range has to be adjusted to the background noise.

Outline

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Theory
Block Diagram



The figure above represents a basic Dynamic Range Control system.

$$y(n) = x(n - D) * g(n) \tag{1}$$

Theory

Working Principle - Level Measurement

- **Level Measurement:** Measures rapidity of change of input signal level.

Theory

Working Principle - Level Measurement

HELMUT SCHMIDT
UNIVERSITÄT

- **Level Measurement:** Measures rapidity of change of input signal level.
  - **Peak Measurement:** If $|x(n)| > x_{PEAK}(n-1)$,

$$x_{PEAK}(n) = (1 - AT) * x_{PEAK}(n-1) + AT * |x(n)| \tag{2}$$

If $|x(n)| < x_{PEAK}(n-1)$,

$$x_{PEAK}(n) = (1 - RT) * x_{PEAK}(n-1) \tag{3}$$

Theory

Working Principle - Level Measurement

- **Level Measurement:** Measures rapidity of change of input signal level.
  - **Peak Measurement:** If $|x(n)| > x_{PEAK}(n-1)$,

$$x_{PEAK}(n) = (1 - AT) * x_{PEAK}(n-1) + AT * |x(n)| \qquad (2)$$

If $|x(n)| < x_{PEAK}(n-1)$,

$$x_{PEAK}(n) = (1 - RT) * x_{PEAK}(n-1) \qquad (3)$$

  - **RMS Measurement:**

$$x_{RMS}^2(n) = (1 - TAV) * x_{RMS}^2(n-1) + TAV * x^2(n) \qquad (4)$$

Theory

Working Principle - Level Measurement

- **Level Measurement:** Measures rapidity of change of input signal level.
  - **Peak Measurement:** If $|x(n)| > x_{PEAK}(n-1)$,

$$x_{PEAK}(n) = (1 - AT) * x_{PEAK}(n-1) + AT * |x(n)| \tag{2}$$

If $|x(n)| < x_{PEAK}(n-1)$,

$$x_{PEAK}(n) = (1 - RT) * x_{PEAK}(n-1) \tag{3}$$

  - **RMS Measurement:**

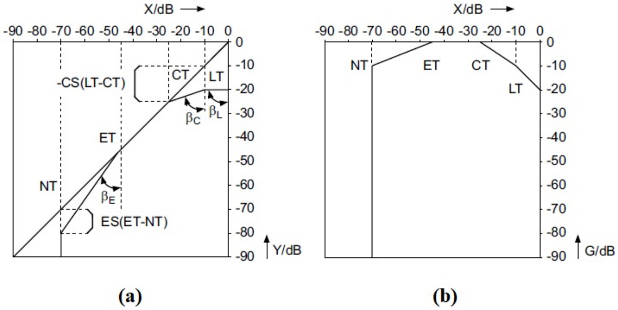
$$x^2_{RMS}(n) = (1 - TAV) * x^2_{RMS}(n-1) + TAV * x^2(n) \tag{4}$$

The measured level of signal is fed to the next block.

Theory

Working Principle - Static Curve

- **Static Curve:** Calculates the gain control factor, based on the measured level of input signal.



(a)

(b)

Theory

Working Principle - Static Curve (contd..)

- **Static Curve - Modes of Operation:**
  - LT - 0dB **(Limiter Mode)**: Limits maximum amplitude of a signal.

Theory

Working Principle - Static Curve (contd..)

- **Static Curve - Modes of Operation:**
  - LT - 0dB **(Limiter Mode)**: Limits maximum amplitude of a signal.
  - CT - LT **(Compressor Mode)**: Reduces dynamic range of a signal.

Theory

Working Principle - Static Curve (contd..)

- **Static Curve - Modes of Operation:**
  - LT - 0dB **(Limiter Mode)**: Limits maximum amplitude of a signal.
  - CT - LT **(Compressor Mode)**: Reduces dynamic range of a signal.
  - ET - NT **(Expander Mode)**: Increases dynamic range of a signal.

Theory

Working Principle - Static Curve (contd..)

- **Static Curve - Modes of Operation:**
  - LT - 0dB **(Limiter Mode)**: Limits maximum amplitude of a signal.
  - CT - LT **(Compressor Mode)**: Reduces dynamic range of a signal.
  - ET - NT **(Expander Mode)**: Increases dynamic range of a signal.
  - -90dB - NT **(Noise Gate Mode)**: Suppresses noise by attenuating quiet parts.

Theory

Working Principle - Gain Factor Calculation

- **Gain Factor Smoothing:** Smooths the previously calculated gain control factor.

$$g(n) = (1 - k) * g(n-1) + k * f(n) \tag{5}$$

The calculated gain is then applied to the input signal.

Outline

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Project Goal

Context

Dynamic Range Control audio applet is used for demo in course, Digital Audio Signal Processing, taught in TUHH.

Project Goal

Context

Dynamic Range Control audio applet is used for demo in course, Digital Audio Signal Processing, taught in TUHH.

Issue with the Applet:

- Java installation on system
- Compatibility issue between version of Java across applet and web browsers

Project Goal

Context

Dynamic Range Control audio applet is used for demo in course, Digital Audio Signal Processing, taught in TUHH.

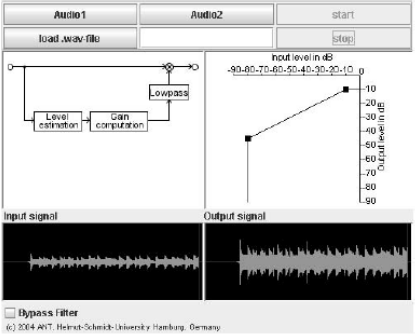
Issue with the Applet:

- Java installation on system
- Compatibility issue between version of Java across applet and web browsers

Objective: Implementation of Compressor mode of DRC audio application on web

Project Goal

Java Applet currently in use



(c) 2004 ANT, Helmut-Schmidt-University Hamburg, Germany

Outline

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Implementation

Phase 1 - Pizzicato.js

Implementation process went through 7 phases.

- **Achievement:** Compressor implementation

Implementation

Phase 1 - Pizzicato.js

Implementation process went through 7 phases.

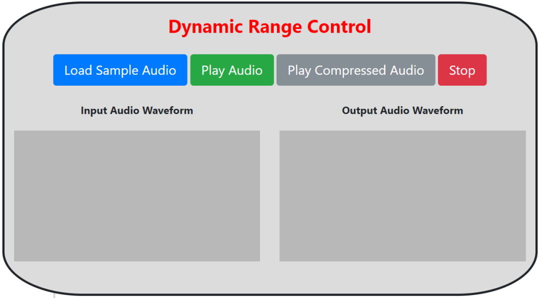
- **Achievement:** Compressor implementation
- **Bottleneck:** Waveform Visualization

Implementation

Phase 1 - Pizzicato.js

Implementation process went through 7 phases.

- **Achievement:** Compressor implementation
- **Bottleneck:** Waveform Visualization



Implementation

Phase 2 - Tone.js

- **Achievement:** Compressor with Dynamic Waveform Visualizer

Implementation

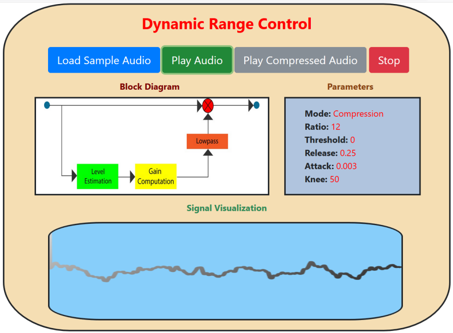
Phase 2 - Tone.js

- **Achievement:** Compressor with Dynamic Waveform Visualizer
- **Bottleneck:** Static Waveform Plot generation

Implementation

Phase 2 - Tone.js

- **Achievement:** Compressor with Dynamic Waveform Visualizer
- **Bottleneck:** Static Waveform Plot generation



Mode: Compression
Ratio: 12
Threshold: 0
Release: 0.25
Attack: 0.003
Knee: 50

Implementation

Phase 3 - Wavesurfer.js

- **Achievement:** Waveform Plot generation using Wavesurfer along with Tone

Implementation

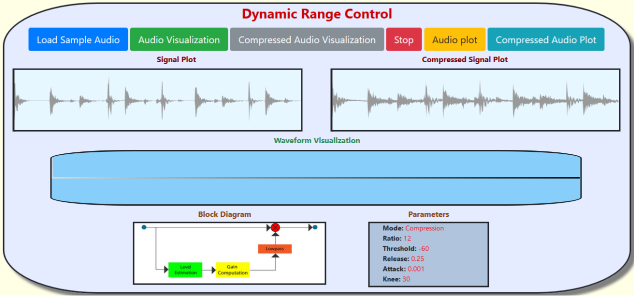
Phase 3 - Wavesurfer.js

- **Achievement:** Waveform Plot generation using Wavesurfer along with Tone
- **Bottleneck:** Incompatibility in data type

Implementation

Phase 3 - Wavesurfer.js

- **Achievement:** Waveform Plot generation using Wavesurfer along with Tone
- **Bottleneck:** Incompatibility in data type



Mode: Compression
Ratio: 12
Threshold: -60
Release: 0.25
Attack: 0.001
Knee: 30

Implementation

Phase 4 - Loopslicer.js

- **Achievement:** Compressor with Waveform Plot from already existing implementation

Implementation

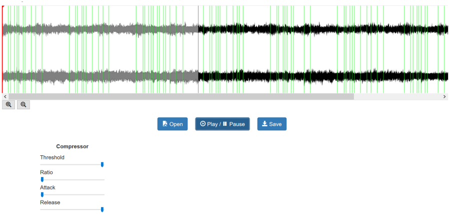
Phase 4 - Loopslicer.js

- **Achievement:** Compressor with Waveform Plot from already existing implementation
- **Bottleneck:** Output audio waveform plot generation

Implementation

Phase 4 - Loopslicer.js

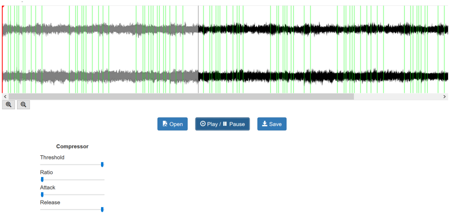
- **Achievement:** Compressor with Waveform Plot from already existing implementation
- **Bottleneck:** Output audio waveform plot generation



Implementation

Phase 4 - Loopslicer.js

- **Achievement:** Compressor with Waveform Plot from already existing implementation
- **Bottleneck:** Output audio waveform plot generation



Deduction: 1. Implementation using these APIs not possible
2. API compressed audio not same in sound level to that in MATLAB

Implementation

Phase 5 - Plain JavaScript

- **Achievement:** Vu Meter using AudioContext in plain JavaScript

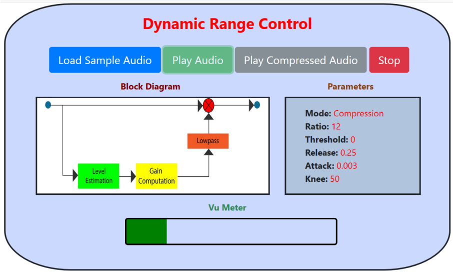
Implementation
Phase 5 - Plain JavaScript

- **Achievement:** Vu Meter using AudioContext in plain JavaScript
- **Bottleneck:** Output audio not playable

Implementation

Phase 5 - Plain JavaScript

- **Achievement:** Vu Meter using AudioContext in plain JavaScript
- **Bottleneck:** Output audio not playable



Mode: Compression
Ratio: 12
Threshold: 0
Release: 0.25
Attack: 0.003
Knee: 50

Implementation

Phase 6 - Java Web

- **Achievement:** Compressor Implementation using JavaScript in frontend and Java Web on backend

Implementation

Phase 6 - Java Web

HELMUT SCHMIDT UNIVERSITÄT

- **Achievement:** Compressor Implementation using JavaScript in frontend and Java Web on backend
- **Bottleneck:** Conversion into audio from output array

Implementation

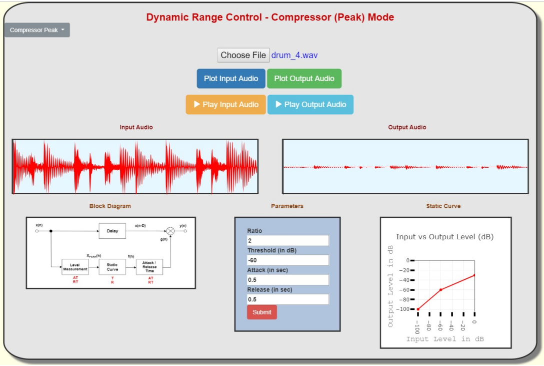
Phase 7 - JavaScript and Python

Compressor with Static Waveform Plot using JavaScript in frontend and Python in backend

Implementation

Phase 7 - JavaScript and Python

Compressor with Static Waveform Plot using JavaScript in frontend and Python in backend



Dynamic Range Control - Compressor (Peak) Mode

Choose File drum_4.wav

Parameters

Ratio: 2

Threshold (in dB): -60

Attack (in sec): 0.5

Release (in sec): 0.5

Outline

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Application Details

Technologies Used

- HTML - for structuring application

Application Details

Technologies Used

- HTML - for structuring application
- CSS - for look and feel

Application Details

Technologies Used

- HTML - for structuring application
- CSS - for look and feel
- JavaScript - for frontend logic

Application Details

Technologies Used

- HTML - for structuring application
- CSS - for look and feel
- JavaScript - for frontend logic
- Bootstrap - frontend framework for more polished look and feel

Application Details

Technologies Used

- HTML - for structuring application
- CSS - for look and feel
- JavaScript - for frontend logic
- Bootstrap - frontend framework for more polished look and feel
- Wavesurfer.js - API for plotting waveforms on canvas

Application Details

Technologies Used

- HTML - for structuring application
- CSS - for look and feel
- JavaScript - for frontend logic
- Bootstrap - frontend framework for more polished look and feel
- Wavesurfer.js - API for plotting waveforms on canvas
- Plotly.js - API for plotting Static Curve

Application Details

Technologies Used

- HTML - for structuring application
- CSS - for look and feel
- JavaScript - for frontend logic
- Bootstrap - frontend framework for more polished look and feel
- Wavesurfer.js - API for plotting waveforms on canvas
- Plotly.js - API for plotting Static Curve
- Python - for backend logic to implement the audio operations

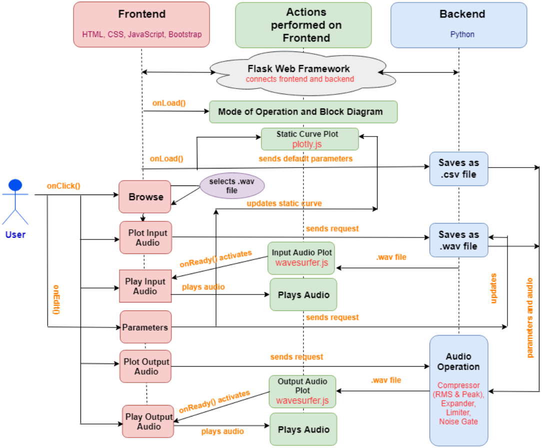
Application Details

Technologies Used

- HTML - for structuring application
- CSS - for look and feel
- JavaScript - for frontend logic
- Bootstrap - frontend framework for more polished look and feel
- Wavesurfer.js - API for plotting waveforms on canvas
- Plotly.js - API for plotting Static Curve
- Python - for backend logic to implement the audio operations
- Flask - micro web framework which facilitates to and fro communication between frontend and backend

Application Details

UML Diagram



Frontend

HTML, CSS, JavaScript, Bootstrap

Actions performed on Frontend

Backend

Python

Flask Web Framework

connects frontend and backend

onLoad()

Mode of Operation and Block Diagram

Static Curve Plot

plotly.js

onLoad()

sends default parameters

Saves as .csv file

User

onClick()

Browse

selects .wav file

updates static curve

Plot Input Audio

sends request

Saves as .wav file

onReady() activates

Input Audio Plot

wavesurfer.js

.wav file

Play Input Audio

plays audio

Plays Audio

onEdit()

updates

parameters and audio

Parameters

sends request

Plot Output Audio

sends request

Audio Operation

Compressor (RMS & Peak), Expander, Limiter, Noise Gate

Output Audio Plot

wavesurfer.js

.wav file

onReady() activates

Play Output Audio

plays audio

Plays Audio

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert
- **Play Input Audio:** Text alert

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert
- **Play Input Audio:** Text alert
- **Plot Output Audio:** 2 Text alerts

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert
- **Play Input Audio:** Text alert
- **Plot Output Audio:** 2 Text alerts
- **Play Output Audio:** Text alert

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert
- **Play Input Audio:** Text alert
- **Plot Output Audio:** 2 Text alerts
- **Play Output Audio:** Text alert
- **Parameters:**
    1. **Ratio:** Range, entry type and on empty alert

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert
- **Play Input Audio:** Text alert
- **Plot Output Audio:** 2 Text alerts
- **Play Output Audio:** Text alert
- **Parameters:**
  1. **Ratio:** Range, entry type and on empty alert
  2. **Threshold:** Same as Ratio and in range (Noise Gate) validation

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert
- **Play Input Audio:** Text alert
- **Plot Output Audio:** 2 Text alerts
- **Play Output Audio:** Text alert
- **Parameters:**
  1. **Ratio:** Range, entry type and on empty alert
  2. **Threshold:** Same as Ratio and in range (Noise Gate) validation
  3. **Attack and Release:** Entry type and on empty alert

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert
- **Play Input Audio:** Text alert
- **Plot Output Audio:** 2 Text alerts
- **Play Output Audio:** Text alert
- **Parameters:**
  1. **Ratio:** Range, entry type and on empty alert
  2. **Threshold:** Same as Ratio and in range (Noise Gate) validation
  3. **Attack and Release:** Entry type and on empty alert
  4. **Pole Placement and Holding Time:** Same as Ratio

Application Details

Frontend Validations and Functionalities

- **Browse:** Size and Name of audio file
- **Plot Input Audio:** Text alert
- **Play Input Audio:** Text alert
- **Plot Output Audio:** 2 Text alerts
- **Play Output Audio:** Text alert
- **Parameters:**
  1. **Ratio:** Range, entry type and on empty alert
  2. **Threshold:** Same as Ratio and in range (Noise Gate) validation
  3. **Attack and Release:** Entry type and on empty alert
  4. **Pole Placement and Holding Time:** Same as Ratio
- **Block Diagram:** Zoom in on hover

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate
- **Application Implementation:**
  - Connection between client and server using Flask

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate
- **Application Implementation:**
  - Connection between client and server using Flask
  - Stereo audio handling

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate
- **Application Implementation:**
  - Connection between client and server using Flask
  - Stereo audio handling
  - Enabling user selection of audio and parameters

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate
- **Application Implementation:**
  - Connection between client and server using Flask
  - Stereo audio handling
  - Enabling user selection of audio and parameters
  - Static Waveform Plot generation

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate
- **Application Implementation:**
  - Connection between client and server using Flask
  - Stereo audio handling
  - Enabling user selection of audio and parameters
  - Static Waveform Plot generation
  - Cache Control

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate
- **Application Implementation:**
  - Connection between client and server using Flask
  - Stereo audio handling
  - Enabling user selection of audio and parameters
  - Static Waveform Plot generation
  - Cache Control
  - Difference in number representation handling

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate
- **Application Implementation:**
  - Connection between client and server using Flask
  - Stereo audio handling
  - Enabling user selection of audio and parameters
  - Static Waveform Plot generation
  - Cache Control
  - Difference in number representation handling
  - Handling initialization

Application Details

Development Journey

- **Processing Implementation:**
  - Compressor (Peak and RMS)
  - Expander
  - Limiter
  - Noise Gate
- **Application Implementation:**
  - Connection between client and server using Flask
  - Stereo audio handling
  - Enabling user selection of audio and parameters
  - Static Waveform Plot generation
  - Cache Control
  - Difference in number representation handling
  - Handling initialization
  - Block Diagram and Static Curve

Application Details

Hosting

Application moved from local to web

Application Details

Hosting

Application moved from local to web

- PythonAnywhere service to host backend

Application Details

Hosting

Application moved from local to web

- PythonAnywhere service to host backend
- 000Webhost service to host frontend

Application Details
Hosting

HELMUT SCHMIDT
UNIVERSITÄT

Application moved from local to web

- PythonAnywhere service to host backend
- 000Webhost service to host frontend

Application accessible via URL

Outline

HELMUT SCHMIDT
UNIVERSITÄT

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Results

Compressor Peak Mode

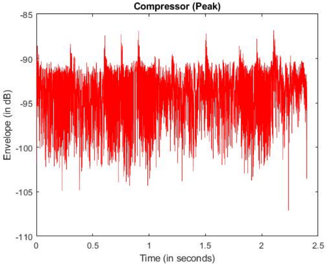
Difference of envelopes in dB vs. time

Results

Compressor Peak Mode

Difference of envelopes in dB vs. time

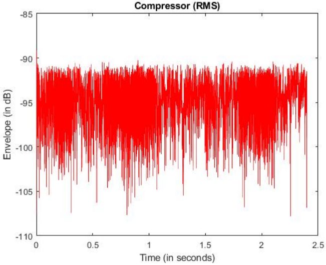
Sample Audio: `drum_4.wav`
Compressor Peak:
Ratio: 20,
Threshold: -40dB,
Attack: 0.1s,
Release: 0.001s.



Results

Compressor RMS and Expander Modes

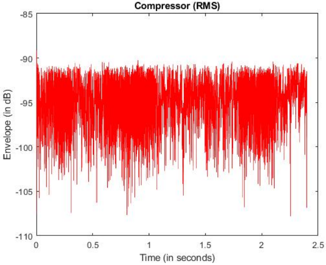
Compressor RMS:
Ratio: 10,
Threshold: -50dB,
Attack: 0.01s,
Release: 0.0001s.



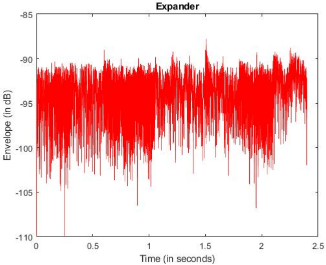
Results

Compressor RMS and Expander Modes

Compressor RMS:
Ratio: 10,
Threshold: -50dB,
Attack: 0.01s,
Release: 0.0001s.



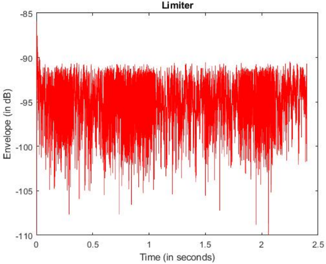
Expander:
Ratio: 0.5,
Threshold: -10dB,
Attack: 0.5s,
Release: 0.5s.



Results

Limiter and Noise Gate Modes

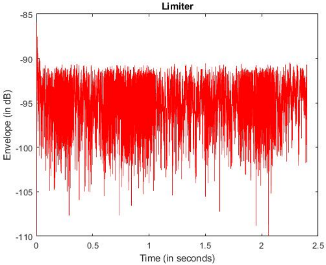
Limiter:
Threshold: -55dB,
Attack: 0.001s,
Release: 0.0005s.



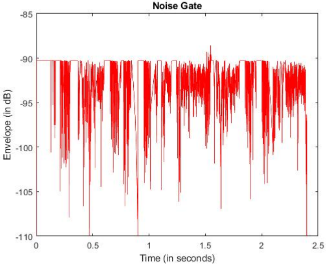
Results

Limiter and Noise Gate Modes

Limiter:
Threshold: -55dB,
Attack: 0.001s,
Release: 0.0005s.



Noise Gate:
Lower Threshold: -30dB,
Upper Threshold: -25dB,
Attack: 0.05s,
Release: 0.00001s,
Hold Time: 0.0001s,
Pole Placement: 0.3.



Outline

HELMUT SCHMIDT
UNIVERSITÄT

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Discussions

Software Development Life Cycle (SDLC)

The project followed all the steps of SDLC within the framework of student project.

Discussions

Software Development Life Cycle (SDLC)

The project followed all the steps of SDLC within the framework of student project.

- Requirements Gathering

Discussions

Software Development Life Cycle (SDLC)

The project followed all the steps of SDLC within the framework of student project.

- Requirements Gathering
- Design

Discussions

Software Development Life Cycle (SDLC)

The project followed all the steps of SDLC within the framework of student project.

- Requirements Gathering
- Design
- Development

Discussions

Software Development Life Cycle (SDLC)

The project followed all the steps of SDLC within the framework of student project.

- Requirements Gathering
- Design
- Development
- Testing

Discussions

Software Development Life Cycle (SDLC)

The project followed all the steps of SDLC within the framework of student project.

- Requirements Gathering
- Design
- Development
- Testing
- Deployment

Discussions

Software Development Life Cycle (SDLC)

The project followed all the steps of SDLC within the framework of student project.

- Requirements Gathering
- Design
- Development
- Testing
- Deployment
- Enhancement

Discussions

Pros and Cons of Implementation

Pros:

- No compatibility issue between application and browser

Discussions

Pros and Cons of Implementation

Pros:

- No compatibility issue between application and browser
- Accessible anywhere and anytime via URL

Discussions

Pros and Cons of Implementation

Pros:

- No compatibility issue between application and browser
- Accessible anywhere and anytime via URL

Cons:

- Absence of draggable operational points in Static Curve

Discussions

Pros and Cons of Implementation

Pros:

- No compatibility issue between application and browser
- Accessible anywhere and anytime via URL

Cons:

- Absence of draggable operational points in Static Curve
- Not realtime

Outline

HELMUT SCHMIDT UNIVERSITÄT

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Application Demonstration

Outline

- Introduction
- Theory
- Project Goal
- Implementation
- Application Details
- Results
- Discussions
- Demo
- Conclusion

Conclusion

Summary

- Project Goal of Compressor implementation achieved

Conclusion

Summary

- Project Goal of Compressor implementation achieved
- Rest 3 modes implemented for completion

Conclusion

Summary

- Project Goal of Compressor implementation achieved
- Rest 3 modes implemented for completion
- Hosting to facilitate ease of access

Conclusion

Summary

- Project Goal of Compressor implementation achieved
- Rest 3 modes implemented for completion
- Hosting to facilitate ease of access
- Satisfying Results

Conclusion

Future Work

- Better GUI

Conclusion

Future Work

- Better GUI
- Responsiveness in application

Conclusion

Future Work

- Better GUI
- Responsiveness in application
- Canvas with more information

Conclusion

Future Work

- Better GUI
- Responsiveness in application
- Canvas with more information
- Draggable Static Curve implementation

Conclusion

Future Work

- Better GUI
- Responsiveness in application
- Canvas with more information
- Draggable Static Curve implementation
- Realtime

Conclusion

Thank You

HELMUT SCHMIDT UNIVERSITÄT

Thanks for your time

Any Question or Suggestion?