
MUTE

The War Crime of Audio Torture: Applying Creative Digital Signal Processing Visualization for Detection and Analysis

Robert Viragh

July 23, 2023

Dedication

To the countless, nameless victims of audio torture around the globe - may this work compel immediate action to end such inhumanity.

Abstract

This image presented herein fulfills the requirements for the degree of Doctor of Philosophy in Digital Signal Processing and Human Rights.

Collections

An instance of illegal audio torture was recorded, with a baseline for comparison.

Unique Contribution and Findings

The visualization presented herein represents the detection of audio torture. This unique methodology brings together the fields of digital signal processing and human rights, revealing the insidious nature of audio torture. The effects mimic those of Havana Syndrome: tinnitus, loss of balance, cognitive

difficulties, and more. These physiological responses align with the mechanisms invoked by intense exposure to loud noises.

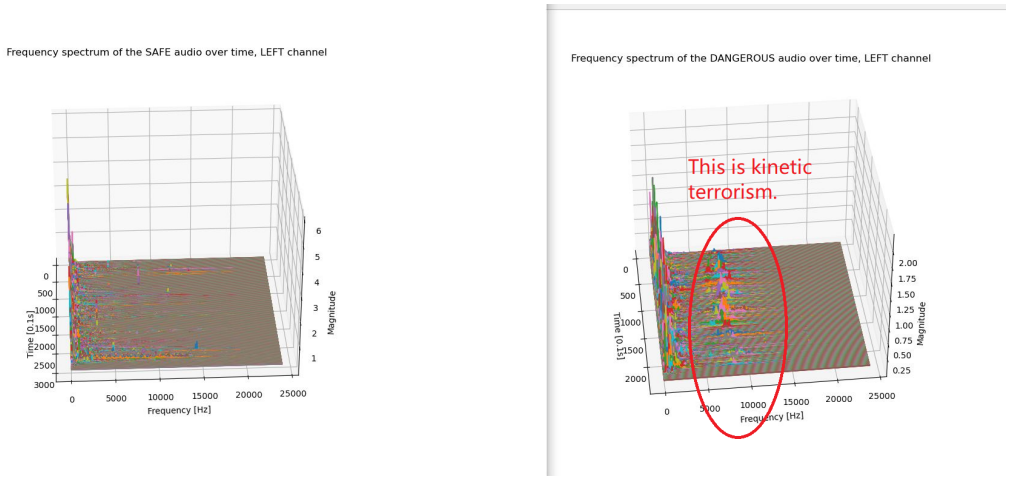

Figure 1: Comparison of a safe audio source (left) and one containing audio torture (right).

The physiological effects seen herein, ranging from auditory damage to disorientation, are considered illegal war crimes under the Geneva Convention.

References

1. International Committee of the Red Cross (ICRC). (1949). Geneva Convention Relative to the Treatment of Prisoners of War. Geneva, Switzerland: ICRC.

Appendix: Source Code

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from scipy.fft import fft
from mpl_toolkits.mplot3d import Axes3D
```

```
# Read the wav file
sample_rate, data = wavfile.read('outputsafe.wav') // For second visualization, re

# If the data is stereo, just take one channel (the first one)
if data.ndim > 1:
    data = data[:, 0]

# Define the slice size (100 ms slices)
slice_size = sample_rate // 10

# Calculate the number of slices
num_slices = len(data) // slice_size

# Prepare a 3D figure
fig = plt.figure(figsize=(10, 5))
ax = fig.add_subplot(111, projection='3d')

# Calculate FFT for each slice
for i in range(num_slices):
    slice_data = data[i*slice_size:(i+1)*slice_size]

    # Let's take the FFT
    fft_out = fft(slice_data)

    # Calculate the absolute value and normalize
    magnitude = np.abs(fft_out) / len(fft_out)

    # Calculate frequency for each FFT point
    freq = np.fft.fftfreq(len(fft_out), 1.0/sample_rate)

    # Ignore half of the points (they are mirrored around 0)
    mask = freq > 0

    # Bin very high frequencies together. These are particularly dangerous frequer
    max_freq = 10000
    high_freq_mask = freq[mask] > max_freq
    magnitude[mask][high_freq_mask] = np.sum(magnitude[mask][high_freq_mask])
```

```
    # Add this line to the 3D plot, at position i (which is the time)
    ax.plot([i]*len(freq[mask]), freq[mask], magnitude[mask])

# Set the labels
ax.set_xlabel('Time [0.1 s]')
ax.set_ylabel('Frequency [Hz]')
ax.set_zlabel('Magnitude')
ax.set_title('Frequency spectrum of the [SAFE/DANGEROUS] audio over time')

# Show the plot
plt.show()
```