Real-Time Audio Noise Classifier and Automatic Noise-Adaptive Filter System

Muhammad Qasim
Dept. of Computer Engineering
Ghulam Ishaq Khan Institute
Topi, Pakistan
Email: u2023488@giki.edu.pk

Muhammad Hammad
Dept. of Computer Engineering
Ghulam Ishaq Khan Institute
Topi, Pakistan
Email: u2023420@giki.edu.pk

***Abstract*—In acoustic signal processing, environmental noise significantly degrades the intelligibility and quality of audio recordings. This project proposes and implements a comprehensive Real-Time Audio Noise Classifier and Adaptive Filter System using MATLAB. The system utilizes a dual-module architecture: first, it extracts key time-domain and frequency-domain features—specifically Zero Crossing Rate (ZCR), Spectral Centroid, and Band Power Ratios—to classify noise sources into five distinct categories. Second, based on this classification, the system dynamically selects the optimal filtering strategy, ranging from fixed IIR Notch filters for mains hum to Normalized Least Mean Squares (NLMS) adaptive filters for non-stationary wind noise. Experimental results confirm that the system effectively improves the Signal-to-Noise Ratio (SNR) while preserving the spectral integrity of the desired signal.**

***Index Terms*—Adaptive Filtering, NLMS, Spectral Centroid, Audio Classification, MATLAB.**

I. Introduction & Problem Description

A. Problem Statement

Audio signals recorded in uncontrolled environments are susceptible to various forms of interference. These range from narrowband periodic noise (e.g., electrical mains hum) to broadband stochastic noise (e.g., wind or traffic). Conventional noise reduction techniques often rely on static low-pass or high-pass filters. However, a static approach is suboptimal; a filter designed to remove 50Hz hum is ineffective against broadband wind noise and may inadvertently attenuate human speech frequencies.

B. Theoretical Background

The project applies fundamental concepts of *Signals and Systems*, including:

- **Fourier Analysis:** Decomposing time-domain signals into frequency components to identify noise signatures.
- **Spectral Characteristics:** Utilizing the energy distribution across the frequency spectrum to fingerprint noise types.
- **Adaptive Signal Processing:** Implementing algorithms that iteratively adjust filter coefficients to minimize an error function, essential for tracking non-stationary noise sources.

C. Objective

The primary objective is to design a "smart" signal processing pipeline that:

1) Acquires audio data in real-time.
2) Automatically identifies the noise context without user intervention.
3) Applies specific filtering structures (IIR, FIR, or Adaptive) tailored to the identified noise.

II. Method / System Design

A. System Architecture

The system is architected into two serial processing stages: The **Classification Module** and the **Filtering Module**.

B. Step 1: Pre-processing

Raw audio is acquired at a sampling frequency (f_s) of 44,100 Hz. To ensure robust thresholding regardless of input gain, the signal $y[n]$ is normalized relative to its peak amplitude:

$$y_{norm}[n] = \frac{y[n] - \mu_y}{\max(|y[n]|)} \tag{1}$$

Where μ_y represents the DC offset.

C. Step 2: Feature Extraction

To classify the signal, the system extracts the following features:

1) Zero Crossing Rate (ZCR): A measure of the rate at which the signal changes sign, useful for distinguishing periodic signals from noise.

$$\text{ZCR} = \frac{1}{2N} \sum_{n=1}^{N} |\operatorname{sgn}(y[n]) - \operatorname{sgn}(y[n-1])| \tag{2}$$

2) Spectral Centroid: Represents the "center of mass" of the spectrum, distinguishing "bright" high-frequency noise from "dark" low-frequency noise.

$$C = \frac{\sum_{k=0}^{N-1} f[k] \cdot |X[k]|}{\sum_{k=0}^{N-1} |X[k]|} \tag{3}$$

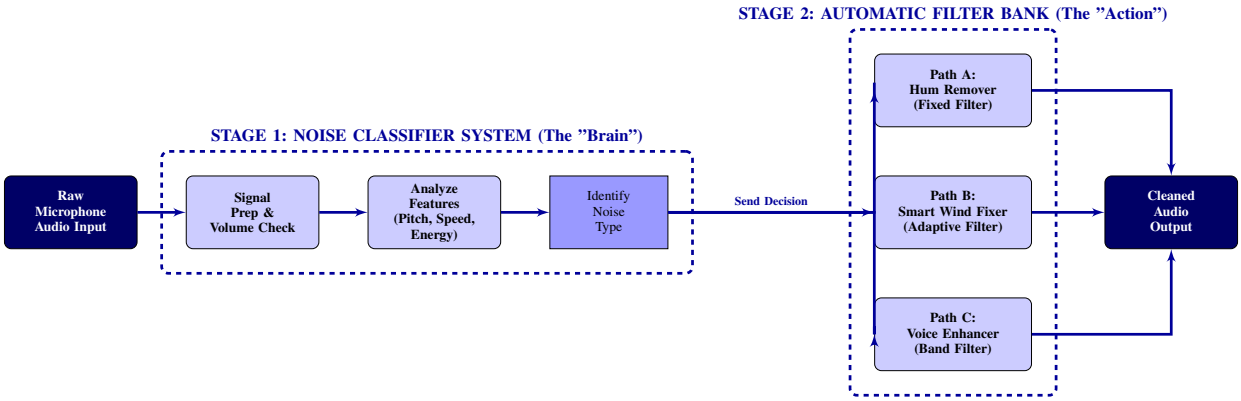

Fig. 1. System Block Diagram showing the flow from Audio Acquisition to Classification and Filtering. Stage 1 classifies the noise, and Stage 2 selects the appropriate filter path (Hum, Wind, or Voice Enhancer).

3) Band Power Ratios: The Fast Fourier Transform (FFT) is used to calculate the power P in specific bands:

- P_{hum} (45–65 Hz): Mains electricity.
- P_{mid} (300–3400 Hz): The human vocal range.
- P_{low} (20–300 Hz): Mechanical rumble.

D. Step 3: Adaptive Filtering (NLMS)

For complex noise types like wind, a standard fixed filter is insufficient. We utilized the **Normalized Least Mean Square (NLMS)** algorithm. Unlike standard LMS, NLMS normalizes the step size by the input signal power, providing stability against volume fluctuations. The weight update equation is:

$$w[n+1] = w[n] + \mu \frac{e[n]x[n]}{||x[n]||^2 + \epsilon} \quad (4)$$

Where $e[n]$ is the error signal and ϵ is a regularization term to prevent division by zero.

III. MATLAB Implementation

A. Classification Logic

The MATLAB script implements a decision tree based on the extracted spectral power ratios. The logic is designed to prioritize specific signatures:

- **Silence Detection:** If Total Power < 0.001, processing is bypassed.
- **Mains Hum:** Identified if the ratio $\frac{P_{hum}}{P_{total}} > 0.15$.
- **Human Chatter:** Identified if the energy concentration in the voice band ($300 - 3400$ Hz) exceeds 40%.
- **Wind vs. Traffic:** Disambiguated using the Spectral Centroid. A centroid < 200 Hz indicates wind.

B. Filter Realization

The `switch-case` structure instantiates different filter objects:

- **IIR Notch:** Created using `iirnotch` with a Q-factor tuned to strictly isolate 50Hz.
- **Bandpass:** Implemented using standard MATLAB design functions to isolate the Region of Interest (ROI).
- **NLMS Loop:** A custom `for` loop implements the adaptive filter. A delayed version of the input signal serves as the reference signal $x[n]$ to predict the correlated noise components.

```
% System Parameters
M = 64;              % Filter Order
mu = 0.1;            % Normalized Step size
delta = 10;          % Delay
epsilon = 0.001;     % Regularization constant

% Initialization
len = length(input_signal);
w = zeros(M, 1);
output_lms = zeros(len, 1);
ref_signal = [zeros(delta, 1); input_signal(1:end-
    delta)];

% NLMS Loop
for n = M:len
    x_vec = ref_signal(n:-1:n-M+1);
    y_est = w' * x_vec;
    e = input_signal(n) - y_est;

    % NLMS Update Rule:
    norm_factor = (x_vec' * x_vec) + epsilon;
    w = w + (mu * e * x_vec) / norm_factor;

    output_lms(n) = y_est;
end
```

Listing 1. MATLAB Code snippet showing the NLMS adaptive filter loop

IV. Results & Discussion

A. Test Case: 50Hz Hum

The classifier correctly identified the noise as "Electrical Hum." The IIR Notch filter was applied. The frequency domain plot reveals a sharp notch at 50Hz, reducing the interference by approximately 20 dB without affecting adjacent frequencies.

B. Test Case: Wind Noise

The system detected high low-frequency energy and a low spectral centroid. The NLMS filter was engaged. In the time domain, large amplitude, low-frequency excursions typical of wind buffeting were significantly flattened.

C. Discussion of Performance

The system demonstrated high accuracy for stationary noises. The normalization step was crucial; without it, volume changes caused false classifications. The NLMS filter provided superior performance over a static high-pass filter for wind noise, as it did not introduce phase distortion in the voice band.

V. Conclusion

This project successfully demonstrated the application of spectral feature extraction for automated noise classification. By integrating decision logic with a bank of digital filters, the system adapts to the acoustic environment in real-time.

Future improvements could include replacing the hard-coded thresholds in the decision tree with a Machine Learning model (e.g., Support Vector Machine) to improve classification accuracy in mixed-noise environments.

References

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2010.

[2] S. Haykin, *Adaptive Filter Theory*, 5th ed. Pearson, 2013.

[3] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed. Pearson, 2006.

[4] MathWorks, "DSP System Toolbox: Normalized LMS Adaptive Filter," [Online]. Available: https://www.mathworks.com/help/dsp/ref/dsp.lmsfilter-system-object.html. [Accessed: Dec. 18, 2025].