GENERAL

applied speech and audio processing with matlab examples

Published: September 19, 2026 | Index ID: #-

Applied Speech and Audio Processing with MATLAB Examples

In the digital age, speech and audio have become ubiquitous mediums for communication, entertainment, and data exchange. From voice assistants and music streaming services to forensic investigations and biomedical diagnostics, the ability to analyze, manipulate, and synthesize audio signals is essential. MATLAB, with its extensive signal processing capabilities and user friendly environment, is a go to platform for researchers, engineers, and students alike. This article offers a deep dive into applied speech and audio processing using MATLAB, complete with practical examples that illustrate how to transform raw audio into actionable insights.

Why MATLAB is the Preferred Tool for Audio Engineers

MATLAB excels in three core areas that make it ideal for audio work:

- Rich Signal Processing Toolbox: Pre built functions for Fourier transforms, filtering, spectral analysis, and more.
- Visualization Capabilities: Immediate plotting of time domain waveforms, spectrograms, and frequency responses.
- Rapid Prototyping: Easy to write, test, and iterate code, which is invaluable for exploratory research.

Moreover, MATLAB's integration with other toolboxes—such as the Audio Toolbox, Signal Processing Toolbox, and Machine Learning Toolbox—enables end to end pipelines from data acquisition to model deployment.

Fundamental Concepts in Speech and Audio Signal Processing

Before diving into examples, it's helpful to understand the building blocks that underpin most audio algorithms:

1. Sampling: Converting a continuous signal into discrete samples at a specific rate (e.g., 44.1 /kHz).
2. Windowing: Applying a window function (Hamming, Hanning, Blackman) to reduce spectral leakage during Fourier analysis.
3. Fast Fourier Transform (FFT): Efficient computation of the discrete Fourier transform, revealing the signal's frequency content.
4. Filter Design: Creating low pass, high pass, band pass, or notch filters to isolate or remove frequency components.
5. Feature Extraction: Calculating descriptors such as Mel Frequency Cepstral Coefficients (MFCCs), spectral centroid, and zero crossing rate.
6. Voice Activity Detection (VAD): Determining when speech is present versus silence or noise.
7. Noise Reduction: Techniques like spectral subtraction or Wiener filtering to clean audio.

These concepts form the backbone of the practical examples that follow.

Getting Started: Loading and Visualizing Audio

Let's begin with a simple script that reads an audio file, plots its waveform, and displays a spectrogram. MATLAB's audioread function handles most common formats (WAV, MP3, FLAC). The plot and spectrogram functions provide immediate visual feedback.

fs = 44100; % Sampling frequency in Hz\n[x, fs] = audioread('speech.wav'); % Load audio file\nt = (0:length(x)-1)/fs; % Time vector\n\nfigure;\nsubplot(2,1,1);\nplot(t, x);\ntitle('Time Domain Waveform'); \nxlabel('Time (s)');\nylabel('Amplitude');\n\nsubplot(2,1,2);\nspectrogram(x, 256, 250, 256, fs, 'yaxis'); \ntitle('Spectrogram');\ncolorbar;\n\n\nRunning this script yields a clear view of the audio's temporal structure and its spectral evolution.

Example 1: Spectral Analysis Using the FFT

Understanding the frequency content is crucial for many applications, such as pitch detection or equalization. The following example demonstrates how to compute and plot the magnitude spectrum of a speech signal.

% Read audio\n[x, fs] = audioread('speech.wav');\nx = x(:,1); % Use single channel\n\n% Compute FFT\nN = length(x);\nX = fft(x);\nX_mag = abs(X(1:floor(N/2)));\n\n% Frequency axis\nf = (0:floor(N/2)-1)*(fs/N);\n\n% Plot\nfigure;\nplot(f, X_mag);\ntitle('Magnitude Spectrum of Speech');\nxlabel('Frequency (Hz)'); \nylabel('Magnitude');\nxlim([0 8000]); % Focus on audible range\n\n\nNotice how the spectral peaks align with formants—key features for speech intelligibility and speaker identification.

Example 2: Designing and Applying a Band Pass Filter

Speech signals typically occupy 300 Hz to 3400 Hz. To isolate this band, we can design a Butterworth band pass filter.

% Filter specifications\nlowcut = 300;\nhighcut = 3400;\n[b,a] = butter(6, [lowcut highcut]/(fs/2), 'bandpass');\n\n% Apply filter\ny = filter(b, a, x);\n\n% Plot filtered signal\nfigure;\nplot(t, y);\ntitle('Band Pass Filtered Speech'); \nxlabel('Time (s)');\nylabel('Amplitude');\n\n\nThe filtered output retains intelligible speech while suppressing low frequency hum and high frequency noise.

Example 3: Voice Activity Detection (VAD) with Energy Thresholding

Detecting speech segments is a prerequisite for many downstream tasks. A simple yet effective VAD method relies on short term energy.

% Frame parameters\nframeLen = round(0.025*fs); % 25 ms\nframeStep = round(0.010*fs); % 10 ms\n\n% Compute short term energy\nnumFrames = floor((length(x)-frameLen)/frameStep) + 1;\nenergy = zeros(numFrames,1);\nfor k = 1:numFrames\n idx = (k-1)*frameStep + (1:frameLen);\n energy(k) = sum(x(idx).^2);\nend\n\n% Thresholding\nthreshold = 0.02 * max(energy);\nspeechFrames = energy > threshold; \n\n% Visualize\ntimeFrames = ((0:numFrames-1)*frameStep + frameLen/2)/fs;\nfigure;\nplot(timeFrames, energy); \nhold on;\nplot(timeFrames, threshold*ones(size(timeFrames)), 'r--');\ntitle('Short Term Energy with VAD Threshold');\nxlabel('Time (s)');\nylabel('Energy');\nlegend('Energy','Threshold');\n\n\nBy marking the frames where energy exceeds the threshold, we can isolate speech from silence or background noise.

Example 4: Extracting Mel Frequency Cepstral Coefficients (MFCCs)

MFCCs are the cornerstone of many speech recognition systems. MATLAB's Audio Toolbox offers a convenient function, but the underlying algorithm remains the same.

% Parameters\nnumCoeffs = 13;\nframeLength = 0.025; % 25 ms\nframeShift = 0.010; % 10 ms\n\n% Compute MFCCs\ncoeffs = mfcc(x, fs, ...\n 'LogEnergy', 'Ignore', ...\n 'NumCoeffs', numCoeffs, ...\n 'Window', 'hamming', ...\n 'OverlapLength', round(frameShift*fs));\n\n% Display first few coefficient vectors\nfigure;

\nimagesc(coeffs);\ntitle('MFCC Coefficients');\nxlabel('Frame Index');\nylabel('Coefficient Index');\ncolorbar; \n\n\nThe resulting matrix can feed into classifiers or neural networks for tasks such as speaker verification or phoneme classification.

Example 5: Real Time Audio Processing with MATLAB’s dsp.AudioPlayer and dsp.AudioRecorder

For interactive applications, real time processing is essential. MATLAB’s DSP System Toolbox provides objects that handle streaming audio.

recObj = dsp.AudioRecorder('SampleRate',fs,'NumChannels',1);\nplayObj = dsp.AudioPlayer('SampleRate',fs); \n\n% Simple echo effect\ndelaySamples = round(0.3*fs); % 300 ms\ndelayLine = zeros(delaySamples,1);\n\nwhile true\n audioIn = recObj();\n % Add echo\n audioOut = audioIn + 0.6*delayLine(1);\n delayLine = [audioIn; delayLine(1:end-1)];\n playObj(audioOut);\nend\n\n\nPress Ctrl+C to stop the loop. This live echo demonstrates how to manipulate audio streams on the fly.

Example 6: Noise Reduction Using Spectral Subtraction

Background noise can severely degrade speech quality. Spectral subtraction estimates the noise spectrum during silence and subtracts it from the noisy signal.

% Estimate noise from first 0.5 seconds\nnoiseSample = x(1:round(0.5*fs));\nnoiseFFT = abs(fft(noiseSample)); \n\n% Compute magnitude spectrum of the entire signal\nX = fft(x);\nX_mag = abs(X);\n\n% Subtract noise spectrum\nX_clean = max(X_mag - noiseFFT, 0);\n\n% Reconstruct complex spectrum\nX_clean_complex = X_clean .* exp(1j*angle(X));\n\n% Inverse FFT\ny_clean = real(ifft(X_clean_complex));\n\n% Plot\nfigure; \nsubplot(2,1,1);\nplot(t, x);\ntitle('Original Noisy Speech');\nxlabel('Time (s)');\n\nsubplot(2,1,2);\nplot(t, y_clean); \ntitle('Denoised Speech');\nxlabel('Time (s)');\n\n\nWhile spectral subtraction is a basic technique, it can produce significant improvements for moderate noise levels.

Example 7: Speaker Identification Using Linear Discriminant Analysis (LDA)

Speaker identification involves distinguishing between different voice sources. Combining MFCC extraction with LDA provides a powerful yet straightforward pipeline.

% Load dataset (placeholder)\n% Assume X_train (N_samples x N_features), y_train (labels)\n% and X_test, y_test for evaluation\n\n% Train LDA\nMdl = fitcdiscr(X_train, y_train);\n\n% Predict\ny_pred = predict(Mdl, X_test); \n\n% Accuracy\naccuracy = sum(y_pred == y_test)/length(y_test);\nfprintf('Speaker Identification Accuracy: %.2f% %\\n', accuracy*100);\n\n\nIn practice, you would gather multiple recordings per speaker, extract MFCCs, and train the classifier on a balanced dataset.

Example 8: Implementing a Simple Pitch Tracker

Pitch estimation is a classic problem in speech analysis. A straightforward approach uses autocorrelation to find the fundamental period.

Baca Juga (Artikel Terkait)

- **advanced engineering electromagnetics solution manual edition hardcover by balanis constantine a published by wiley**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-solution-manual-edition-hardcover-by-balanis-constantine-a-published-by-wiley.pdf
- **advanced engineering electromagnetics solutions manual**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-solutions-manual.pdf
- **advanced engineering electromagnetics wiley 1989 grading**
  URL: http://djjsearch.com/advanced-engineering-electromagnetics-wiley-1989-grading.pdf
- **advanced engineering fluid mechanics by biswas**
  URL: http://djjsearch.com/advanced-engineering-fluid-mechanics-by-biswas.pdf