%% MUSIC NOTE EXTRACTION with NOISE & FILTERING

clear; clc; close all;

%% STEP 1: Load or Generate Music

musicFile = 'flute-c2.wav'; % <- Change to your own song or note file

if isfile(musicFile)

disp('🎵 Loading music file...');

[x\_clean, fs] = audioread(musicFile);

if size(x\_clean,2) > 1

x\_clean = mean(x\_clean, 2); % Make mono

end

else

% Generate synthetic melody (C5 -> E5 -> G5)

disp('⚠ No music file found, generating a tone melody...');

fs = 44100;

t1 = 0:1/fs:0.8;

t2 = 0:1/fs:0.8;

t3 = 0:1/fs:0.8;

note1 = sin(2\*pi\*523.25\*t1); % C5

note2 = sin(2\*pi\*659.25\*t2); % E5

note3 = sin(2\*pi\*783.99\*t3); % G5

x\_clean = [note1 note2 note3]';

end

x\_clean = x\_clean / max(abs(x\_clean));

%% STEP 2: Play clean music

disp('▶ Playing clean music...');

soundsc(x\_clean, fs);

pause(length(x\_clean)/fs + 0.5);

%% STEP 3: Add noise

SNR\_dB = 5; % SNR in dB (low = more noise)

x\_noisy = awgn(x\_clean, SNR\_dB, 'measured');

disp('▶ Playing noisy music...');

soundsc(x\_noisy, fs);

pause(length(x\_noisy)/fs + 0.5);

%% STEP 4: Filter noise (bandpass for music notes: ~80-2000 Hz)

bpFilt = designfilt('bandpassiir', 'FilterOrder', 8, ...

'HalfPowerFrequency1', 80, ...

'HalfPowerFrequency2', 2000, ...

'SampleRate', fs);

x\_filtered = filtfilt(bpFilt, x\_noisy);

disp('▶ Playing filtered music...');

soundsc(x\_filtered, fs);

pause(length(x\_filtered)/fs + 0.5);

%% STEP 5: Note extraction function

function notesOut = extractNotes(sig, fs)

% Short-time analysis

window\_size = 2048;

noverlap = 1024;

nfft = 4096;

[S, F, T] = spectrogram(sig, window\_size, noverlap, nfft, fs);

S\_mag = abs(S);

[~, peakIdx] = max(S\_mag, [], 1);

peakFreqs = F(peakIdx);

% Frequency to note mapping

noteNames = {'C','C#','D','D#','E','F','F#','G','G#','A','A#','B'};

A4 = 440;

notes = cell(1, length(peakFreqs));

for k = 1:length(peakFreqs)

if peakFreqs(k) > 20

midiNum = round(69 + 12\*log2(peakFreqs(k)/A4));

noteIdx = mod(midiNum, 12) + 1;

octave = floor(midiNum/12) - 1;

notes{k} = sprintf('%s%d', noteNames{noteIdx}, octave);

else

notes{k} = 'N/A';

end

end

% Remove duplicates

validIdx = ~strcmp(notes, 'N/A');

noteSeq = notes(validIdx);

timeSeq = T(validIdx);

changes = [true, ~strcmp(noteSeq(2:end), noteSeq(1:end-1))];

notesOut = unique(noteSeq(changes));

% Display detected notes

for i = 1:length(noteSeq)

if changes(i)

fprintf('Time %.2fs: %s\n', timeSeq(i), noteSeq{i});

end

end

end

%% STEP 6: Extract notes from each version

disp('🎼 Notes in CLEAN music:');

cleanNotes = extractNotes(x\_clean, fs);

disp('🎼 Notes in NOISY music:');

noisyNotes = extractNotes(x\_noisy, fs);

disp('🎼 Notes in FILTERED music:');

filteredNotes = extractNotes(x\_filtered, fs);

%% STEP 7: Plot results for comparison

t = (0:length(x\_clean)-1)/fs;

figure;

subplot(3,2,1); plot(t, x\_clean); title('Clean Music'); xlabel('Time'); ylabel('Amp');

subplot(3,2,2); spectrogram(x\_clean, 1024, 512, 2048, fs, 'yaxis'); title('Spectrogram - Clean');

subplot(3,2,3); plot(t, x\_noisy); title('Noisy Music'); xlabel('Time'); ylabel('Amp');

subplot(3,2,4); spectrogram(x\_noisy, 1024, 512, 2048, fs, 'yaxis'); title('Spectrogram - Noisy');

subplot(3,2,5); plot(t, x\_filtered); title('Filtered Music'); xlabel('Time'); ylabel('Amp');

subplot(3,2,6); spectrogram(x\_filtered, 1024, 512, 2048, fs, 'yaxis'); title('Spectrogram - Filtered');

sgtitle('Music Signal Before/After Noise & Filtering');