

**语音信号处理**

**实验报告**

**实验名称： 语音信号的时域分析**

**学生姓名： 赵展文 学号： 2024360809**

**授课教师： 马 英 职称： 教授**

**院 系： 智能科学与工程学院**

**专业班级： 2024级集成电路工程**

1. **短时能量及短时平均幅度分析**

程序代码

%% 清除缓存

clear;

close all;

%% 读取音频文件

[file, path] = uigetfile('\*.wav', '选择WAV文件');

filename = fullfile(path, file);

[y, fs] = audioread(filename);

y = y(:, 1); % 取单通道音频

%% 计算短时能量

% 设定帧参数

frame\_length = 200; % 帧长

frame\_shift = 80; % 帧移

% 分帧处理

fn = fix((length(y) - frame\_length) / frame\_shift) + 1; % 计算帧数

frames = zeros(frame\_length, fn); % 预分配存储空间

for i = 1:fn

start\_idx = (i - 1) \* frame\_shift + 1; % 计算起始索引

end\_idx = start\_idx + frame\_length - 1; % 计算结束索引

frames(:, i) = y(start\_idx:end\_idx); % 逐帧提取

end

% 计算 Hamming 窗

hamming\_win = hamming(frame\_length); % 计算 Hamming 窗

% Hamming 加窗

for i = 1:fn

frames(:, i) = frames(:, i) .\* hamming\_win; % 加窗

end

En = zeros(1, fn);

for i = 1:fn

frame = frames(:, i);

En(i) = sum(frame .^ 2); % 能量计算

end

%% 绘制波形

figure;

% 原始语音

subplot(4,1,1);

plot(y);

title('原始语音信号');

xlabel('时间/s');

ylabel('幅值');

grid on;

% 短时能量

subplot(4,1,2);

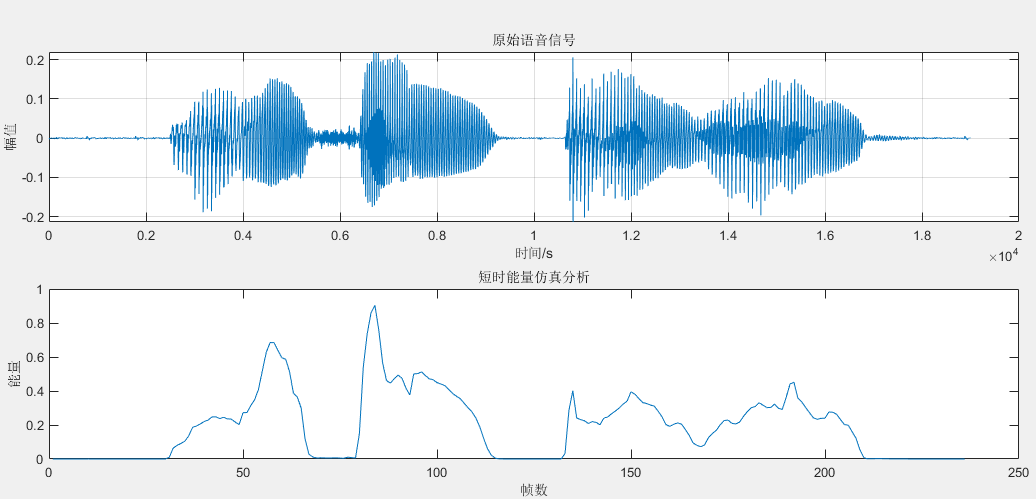
plot(En);

title('短时能量仿真分析');

xlabel('帧数');

ylabel('能量');

波形图



1. **短时相关分析**

程序代码

%% 清除缓存

clear;

close all;

%% 读取音频文件

[file, path] = uigetfile('\*.wav', '选择WAV文件');

filename = fullfile(path, file);

[y, fs] = audioread(filename);

y = y(:, 1); % 取单通道音频

%% 计算短时自相关函数

% 设定帧参数

frame\_length = 512; % 帧长

frame\_shift = 64; % 帧移

% 分帧处理

fn = fix((length(y) - frame\_length) / frame\_shift) + 1; % 计算帧数

frames = zeros(frame\_length, fn); % 预分配存储空间

for i = 1:fn

start\_idx = (i - 1) \* frame\_shift + 1; % 计算起始索引

end\_idx = start\_idx + frame\_length - 1; % 计算结束索引

frames(:, i) = y(start\_idx:end\_idx); % 逐帧提取

end

% 计算 Hamming 窗

hamming\_win = hamming(frame\_length); % 计算 Hamming 窗

% Hamming 加窗

for i = 1:fn

frames(:, i) = frames(:, i) .\* hamming\_win; % 加窗

end

frame\_idx = 105; % 选择帧进行自相关分析

if frame\_idx > fn

frame\_idx = fn; % 避免索引超出帧数

end

frame = frames(:, frame\_idx); % 取出目标帧

y\_corr = xcorr(frame, 'biased'); % 归一化计算自相关

y\_corr = y\_corr(frame\_length:end); % 仅取正值部分

self\_relate\_lags = (0:length(y\_corr)-1) / fs; % 计算滞后时间（秒）

%% 绘制波形

figure;

% 原始语音

subplot(4,1,1);

plot(y);

title('原始语音信号');

xlabel('时间/s');

ylabel('幅值');

grid on;

% 短时自相关

subplot(4,1,2);

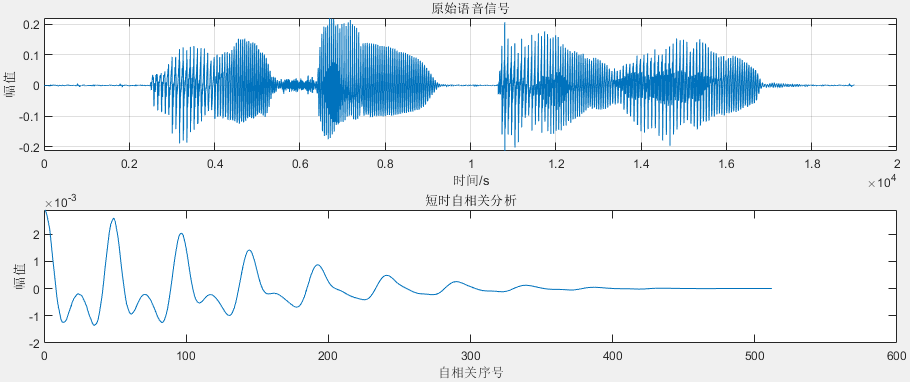
plot(y\_corr);

title('短时自相关分析');

xlabel('自相关序号');

ylabel('幅值');

波形图



1. **AMDF算法分析**

程序代码

%% 清除缓存

clear;

close all;

%% 读取音频文件

[file, path] = uigetfile('\*.wav', '选择WAV文件');

filename = fullfile(path, file);

[y, fs] = audioread(filename);

y = y(:, 1); % 取单通道音频

% 设定帧参数

frame\_length = 512; % 帧长

frame\_shift = 64; % 帧移

% 分帧处理

fn = fix((length(y) - frame\_length) / frame\_shift) + 1; % 计算帧数

frames = zeros(frame\_length, fn); % 预分配存储空间

for i = 1:fn

start\_idx = (i - 1) \* frame\_shift + 1; % 计算起始索引

end\_idx = start\_idx + frame\_length - 1; % 计算结束索引

frames(:, i) = y(start\_idx:end\_idx); % 逐帧提取

end

% 计算 Hamming 窗

hamming\_win = hamming(frame\_length); % 计算 Hamming 窗

% Hamming 加窗

for i = 1:fn

frames(:, i) = frames(:, i) .\* hamming\_win; % 加窗

end

frame\_idx = 105; % 选择帧

if frame\_idx > fn

frame\_idx = fn; % 避免索引超出帧数

end

frame = frames(:, frame\_idx); % 取出目标帧

%% AMDF算法分析

amdf\_max\_lag = 320; % 设置最大滞后范围

AMDF = zeros(amdf\_max\_lag, 1);

for k = 1:amdf\_max\_lag

diff\_sum = sum(abs(frame(1:end-k) - frame(1+k:end)));

AMDF(k) = diff\_sum / (frame\_length - k);

end

amdf\_lags = (1:amdf\_max\_lag) / fs \* 1000; % 转换为毫秒单位

%% 绘制波形

figure;

% 原始语音

subplot(4,1,1);

plot(y);

title('原始语音信号');

xlabel('时间/s');

ylabel('幅值');

grid on;

% AMDF算法

subplot(4,1,2);

plot(AMDF);

title('AMDF算法分析');

xlabel('滞后时间 (ms)');

ylabel('AMDF 值');

波形图

