UAV Detecting

Jialiang LU

Thesis submitted to

Zhejiang University

in partial fulfilment for the course

Signals & Systems

Information Science

College of Information Science & Electronic Engineering
https://github.com/Riverlu233

2025 年 6 月 21 日

目录

浙江大学项目报告

专业：信息工程
姓名：卢嘉良
学号：3230104505
日期：2025 年 6 月 21 日
地点：None

课程名称：信号与系统　　指导老师：虞露，廖依伊
实验名称：基于声音信号的无人机检测　　报告类型：总结报告　仿真软件：MATLAB

一、 研究任务

使用 MATLAB 实现无人机声音检测系统，要求：

(1) 从声音信号中提取 MFCC 特征

(2) 使用 SVM 分类器区分无人机和非无人机声音

(3) 系统准确率达到 85% 以上

(4) 对比分析无人机与非无人机声音的频谱特征

二、 研究流程

说明：所有代码可以在附录 A 当中进行查阅，由于代码长度并不长，读者可以借助于附录 A 的注释迅速找到各部分所对应代码。

1. 数据获取

研究所使用的数据集是指导老师已经给出的两个数据集，分别记录的是 UAV 有和无状态下的音频采样结果，每个数据集中包含 10 个子数据集，已经转化为 MATLAB 数据存储的标准格式.mat。采样频率以及在机器学习过程中的划分方法在下方给出。

- **数据集：**包含无人机声音和非无人机声音的音频文件
- **采样频率：**5120 Hz
- **数据划分：**75% 训练集，25% 测试集

2. 数据加载与预处理

脚本首先通过 dir 和 load 命令加载指定路径下的.mat 文件。dir 函数返回的是一个保存有文件信息的结构体数组 droneFiles，利用其中的文件名以及存放数据集的文件夹的路径就可以利用 filepath 函数来生成所有需要使用的数据集的路径。接下来逐个利用 load 命令来加载这些文件，考虑到 load 产生的结果为一个 struct，并且在命令行中利用 load 和 fieldnames 可以查看出所给数据就是一个 $1 \times n$ 的数组，因此只需要读取出 load 后的结构体中的唯一一个数组变量的值即可。

此外，为便于区分，无人机声音标记为 1，非无人机声音标记为 0。因为有无人机声音和无无人机的声音是分开读入的，因此只需要在循环中给标签数组 allLabels 赋值即可。为便于处理，将所有子数据集当中的数据全部放在一个 cell 数组 allAudioData 当中，最终它应该是一个包含 20 个数组（即 20 个子数据集）的 cell 数组，与之配套的 allLabels 数组存储标签来对两类数据进行区分。

3. 特征提取

考虑原始的语音序列通常数据量很大，直接使用原始数据进行训练会导致模型参数量很大，模型很难收敛，因此一般来说不会将语音信号序列直接交给模型进行训练。因为原始的语音序列通常数据量很大，直接使用原始数据进行训练会导致模型参数量很大，模型很难收敛。所以我们希望能够从语音序列中提取出一些低维的特征向量，且这些特征向量能够很好表示原信号的特性。这里我们使用 Mel 倒谱系数特征（MFCC）。Mel 倒谱系数特征模拟人耳的听觉模型，用从低频到高频由密到疏且带宽逐渐增大的一组带通滤波器，对输入信号进行滤波，提取包络，作为信号的基本特征。这种音频特征模拟了人耳听觉模型，对信号没有任何假设和限制，符合人耳的听觉特性，当信噪比低时也有较好的识别性能。

(1) 预加重

由于人耳对高频信号更不敏感，因此先利用一个高通滤波器对信号的高频部分进行小幅度的增强。高通滤波器的传输函数为：

$$H(z) = 1 - \mu z^{-1}$$

其中 $\mu = 0.9375$。考虑到实际上预加重对于每一组数据都是必要的，我们在分帧之前先进行预加重操作。其中的预加重系数 μ 在代码开头进行了定义。代码中的 signal 实际上指的就是原数据集当中的一个子数据集的所有数据。执行预加重操作时不需要详细定义其步骤，因为 MATLAB 给出了数字滤波器函数 filter，直接指定参数并调用即可。

Listing 1: Pre-emphasis

```
PRE_EMPH_COEFF = 0.9375; % Pre-emphasis coefficient

signal_preemph = filter([1 -PRE_EMPH_COEFF], 1, signal);
```

(2) 分帧和加窗处理

考虑到语音信号在短时域（10-30ms）内可被认为是稳态的，并且对于无人机分类问题来说，10-30ms 的语音信号就足够检测出是否有无人机出现，所以可以考虑对语音信号进行分帧。此外，进行分帧处理同时还能够使训练数据集变得很大，能够训练出更好的模型。在本实验中帧长帧长设为 256，步长设为 128，它们在代码开头被定义。

考虑到每次 FFT 变换只能对有限长度的时域数据进行变换，因此，需要对时域信号进行信号截断。即使是周期信号，如果截断的时间长度不是周期的整数倍（周期截断），那么，截取后的信号将会存在泄漏。为了将这个泄漏误差减少到最小程度，需要使用窗函数。加窗主要是为了使时域信号似乎更好地满足 FFT 处理的周期性要求，减少泄漏。此处所加的窗函数的类型为汉明窗，即 frames 是分帧后信号乘上窗函数之后的结果。

代码实现时，每次取 allAudioData 中的一个子数据集对其进行分帧操作，分帧时使用的是 enframe 函数，需要向其传递帧长度和帧间隔参数。需要注意的是，enframe 函数的返回值是一个二维的矩阵。执行完分帧后需要对每一帧进行加窗，实际上就是将每一帧的向量点乘上窗函数。窗函数也是由 MATLAB

的代码直接生成的 Hamming 窗，只需向其传递需加窗的序列的长度，即帧长度即可。这一步骤涉及的核心代码如下：

Listing 2: Windowing & Enframing

```
FRAME_LENGTH_SAMPLES = 256; % Frame length
FRAME_HOP_SAMPLES = 128; % Frame hop (step size)

window = hamming(FRAME_LENGTH_SAMPLES);

 % (2) Framing
frames_raw = enframe(signal_preemph, FRAME_LENGTH_SAMPLES, FRAME_HOP_SAMPLES);
frames=frames_raw;
numFrames = size(frames, 1);

% (3) Windowing
for j = 1:numFrames
  frames(j, : )=frames(j, : )'.*window;
end
```

(3)　快速傅里叶变换

要进行快速傅里叶变换非常简单，直接调用 fft 函数求得每一帧信号的频谱即可，求功率谱时同理，直接对信号的绝对值的平方调用 fft 即可, 它表示信号在各频率分量上的能量大小。由于这一功率谱需要经过 Mel 滤波器组，所以需要对其长度做限制，即得到单边功率谱，这点将在 Mel 滤波器映射中说明。

Listing 3: FFT

```
% (3) FFT
frameFFT = fft(frameCur, FRAME_LENGTH_SAMPLES);

% (4) Power Spectrum
powerSpec = abs(frameFFT(1:(FRAME_LENGTH_SAMPLES/2 + 1))).^2;
```

(4)　Mel 滤波器组映射

由于人类对频率的感知并不是线性的，并且对低频信号的感知要比高频信号敏感，因此 Mel 标度 (the Mel Scale) 被提出，它是 Hz 的非线性变换，对于以 Mel Scale 为单位的信号，可以做到人们对于相同频率差别的信号的感知能力几乎相同。要得到一个 Mel 频谱，实际上就是要将原功率谱和若干个 Mel 滤波器做点乘。值得注意的是，Mel 滤波器组是对于正的频率的一个非线性标度，因此当功率谱和 Mel 滤波器做点乘时我们只需要功率谱的正单边谱，因此在前文做 FFT 时控制了生成的频率谱的长度。

本项目中的一组 24 个滤波器是通过 voicebox 中的 melbankm 函数来生成的，需要给出的参数包括梅尔滤波器组的数目，采样频率和帧长度。生成该滤波器组的代码在参考文档中已经给定。考虑到经过的映射是对数映射，为避开 $\log 0$ 这样的结果，需要给 0 值处补充一个足够小的值。在此之后将 Mel 滤波器组和功率谱做点乘，就将功率谱的能量映射到了 24 个 Mel 频带上，即得到 Mel 能量谱 melPowerCur。后面还对该能量谱执行了一次取对数操作，得到了对数能量谱，这也是为了模拟人耳对声音强度的感知过程，并且可以压缩数值的动态范围，使得特征对音量的变化不那么敏感。

Listing 4: MelBank

```
melBank = melbankm(NUM_MEL_FILTERS, FRAME_LENGTH_SAMPLES, FS, 0, 0.5, 'm');
```

```
melBank = full(melBank);
melBank = melBank / max(melBank(:)); % Normalize

melPowerCur = melBank * powerSpec';
melPowerCur_log = log(max(melPowerCur, 1e-6));
```

(5) DCT 与倒谱分析

经过 Mel 滤波后得到的是一个一个 24 维的对数 Mel 谱，但这个谱的各维度之间仍然存在较高的相关性，若一个频带能量高，它相邻的频带能量也很可能比较高。为了使特征更适合用于 SVM 的训练，我们考虑进行一步解相关操作，也就是倒谱分析。如果我们对对数频谱再做 FFT, 就得到了倒谱。查阅资料可以得到，频谱中缓慢变化的部分（谱包络）通常对应于发声体的物理特性（如无人机马达的结构、人的声道形状），这是我们想要的可以区分二者的重要特征；而频谱中快速变化的部分（谱细节）通常对应于激励信号的谐波结构（如无人机螺旋桨的转速、人的声带振动基频），这部分信息变化快且不稳定，不利于进行分析。在倒谱中，谱包络信息集中在低倒频系数上，而谱细节信息则分布在高倒频系数上。

在此处我们根据文档使用离散余弦变换 (DCT) 来代替 FFT。由于 DCT 对于高度相关的信号具有非常好的能量集中特性，绝大部分信号能量（即谱包络信息）在经过 DCT 后都会被集中到前几个系数上。因此我们考虑通过只保留前几个系数来高效地压缩数据，丢弃后面的高倒频系数，进而平滑频谱并减少噪音影响。此外，DCT 是一个正交变换，它能够有效地解除输入特征之间的相关性，利于提供给 SVM 进行训练。

在倒谱分析这一步过程中，DCT 的系数，DCT 的实现以及实现 DCT 后的加窗谱线增强的代码均已经在参考文档中给出, 实际上就是生成了一个 DCT 系数构成的矩阵，通过一个线性变换将 24 维的对数 Mel 谱转换为了 12 维的 MFCC 系数。为了提升代码的可读性，此处窗函数被封装为 lifter_w。最后每一帧的静态 MFCC 系数数组 mfcc_frame_liftered 被存入矩阵 mfccs_static_current_file 当中一行一行存储。

Listing 5: DCT

```
% DCT matrix for MFCC
dctcoef = zeros(NUM_MFCC_COEFFS, NUM_MEL_FILTERS);
for k = 1:NUM_MFCC_COEFFS
   n_dct = 0:(NUM_MEL_FILTERS-1);
   dctcoef(k,:) = cos((2*n_dct+1)*k*pi/(2*NUM_MEL_FILTERS));
end
% Cepstral liftering weights
N_lifter = 1:NUM_MFCC_COEFFS;
lifter_w = 1 + 6 * sin(pi * N_lifter' / NUM_MFCC_COEFFS);
lifter_w = lifter_w / max(lifter_w); % Normalize

mfcc_frame_raw = dctcoef * melPowerCur_log;
mfcc_frame_liftered = mfcc_frame_raw .* lifter_w;
mfccs_static_current_file(j, :) = mfcc_frame_liftered';
```

(6) 动态特征表示

MFCC 只反映了语音参数的静态特性，实际上声音是连续的，帧与帧之间是有联系的，因而需要增加特征来表示这种动态特性。实验也表明，把动、静态特征结合起来才能有效提高系统的识别性能。这通常通过计算每一帧的 12 个倒谱特征的一阶差分甚至二阶差分来实现。给出如下的一阶差分公式：

$$dm(i) = \frac{2m(i+2) + m(i+1) - m(i-1) - 2m(i-2)}{3}$$

上述公式说明对于每一帧，它的 12 个差分 MFCC 系数是它的后两帧的静态 MFCC 系数减去前两帧 MFCC 系数的差值的三分之一，这就导致差分 MFCC 系数要比静态 MFCC 系数少 4 帧，因为头 2 个和末 2 个位置上的帧不存在前、后两帧。同时，取其它相邻数据集的前 2 帧和后 2 帧的 MFCC 系数来运算也是不可取的，因为我们无法确定选取数据集时是同等条件下连续选取的。

代码实现时，由于差分 MFCC 系数是利用相邻帧来计算的，可以考虑先按照 MFCC 系数的顺序，再按照帧的顺序来求，即一次循环求出每一帧的第 i 个差分 MFCC 系数。最终求得的差分 MFCC 系数也被存放在矩阵 mfccs_delta_current_file 中按行存储。实现的代码如下：

Listing 6: DELTA MFCC

```
mfccs_delta_current_file = zeros(numFrames - 4, NUM_MFCC_COEFFS);

for k = 1:NUM_MFCC_COEFFS % For each MFCC coefficient
    m_coeff_col = mfccs_static_current_file(:, k);
    denominator_delta = 3;
    for frame_idx = 3:(numFrames - 2)
        mfccs_delta_current_file(frame_idx-2, k) = (2*m_coeff_col(frame_idx+2) + m_coeff_col(frame_idx+1) -
            m_coeff_col(frame_idx-1) - 2*m_coeff_col(frame_idx-2)) / denominator_delta;
    end
end
```

(7)　特征组合

最终要将每一帧的静态和差分 MFCC 系数组合在一起从成为一个 24 维的特征向量。考虑到静态 MFCC 比动态 MFCC 要多 4 帧的数据，我们去掉存储静态 MFCC 数据的矩阵的前两行和后两行，然后将 mfccs_static_current_file 和 mfccs_delta_current_file 拼接，这样就得到了一个子数据集的完全的 MFCC 系数矩阵，每一行对应着一帧的 MFCC 特征向量。再将所有子数据集的每个 MFCC 系数矩阵全部存入一个大矩阵 allFeatures 当中，存入时竖向拼接，即仍然保持每行对应一帧的特性，就得到了存储着总数据集特征的矩阵 allFeatures。

与此同时，由于这是一个二分类问题，训练 SVM 还需要给数据打上标签。最终我们训练时肯定不适用子数据集来进行训练，否则数量太少而且不便于划分，故训练时使用的是每一帧的数据，所以每一帧的数据都需要打上标签。为此采用 repmat 函数来构造一个行数和对应子数据集中帧数相同，只有一列，并且存储每一帧的标签的矩阵，称之为每个子数据集的标签矩阵。将每个子数据集的标签矩阵按照同样的方式和合并进一个矩阵 allFrameLabels 当中，使得最终标签矩阵中每一行只有一个值，表示对应帧有无无人机，将其作为总数据集标签矩阵。这样就得到了符合文档中要求形式的特征矩阵和标签矩阵，它们能够作为训练 SVM 的函数的输入。

Listing 7: Feature-Combination

```
%Feature splicing of frames within a dataset
 mfccs_static_for_combination = mfccs_static_current_file(3:(numFrames-2), :);
mfccs_combined = [mfccs_static_for_combination, mfccs_delta_current_file];

%Splicing features from different datasets
allFeatures = [allFeatures; mfccs_combined];
allFrameLabels = [allFrameLabels; repmat(currentLabel, size(mfccs_combined, 1), 1)];
%Note that they are not in the same loop
```

4. 训练 SVM

支持向量机是一种二分类模型，其基本思想是寻找一个超平面使两类数据间隔最大化。本实验使用径向基函数 (RBF) 核的 SVM，其决策函数为：

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right)$$

其中 $K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2)$ 为 RBF 核函数。作为机器学习的一部分内容，在这里我们不详细讨论 SVM 的工作原理，而直接调用 MATLAB 中集成好的模块来实现。

(1) 训练集与测试集划分

按照参考文档中的要求，原数据集的 75% 划分为训练集，25% 划分为测试集。由于在动态特征表示中我们需要相邻帧之间的关系，因此对于数据的打乱在处理完特征之后进行，同时这样我们只需要对特征矩阵和特征标签作打乱即可，不再需要打乱原数据集。

按照参考文档中的建议，此处先调用 rng 函数来生成随机数种子，再通过 randperm 函数生成打乱后的帧顺序。由于已经是经过打乱的，可以直接取其中前 75% 所对应的帧数据作为训练集，后 25% 作为测试集；选取时既要选取出特征矩阵，也要选取出标签矩阵。

Listing 8: Devision

```
numTotalFrames = size(allFeatures, 1);
rng('default'); % For reproducibility
shuffledIndices = randperm(numTotalFrames); %

numTrainFrames = floor(TRAIN_RATIO * numTotalFrames);
trainIndices = shuffledIndices(1:numTrainFrames);
testIndices = shuffledIndices(numTrainFrames+1:end);

X_train = allFeatures(trainIndices, :);
Y_train = allFrameLabels(trainIndices, :);
X_test = allFeatures(testIndices, :);
Y_test = allFrameLabels(testIndices, :);
```

(2) SVM 训练

根据文档的说明，对于划分好的数据以及他们的标签，直接使用 fitcsvm 函数进行训练。其中参数'Standardize'，true 将数据标准化，加快训练, 'KernelFunction','RBF' 使用高斯核将数据映射到高维空间使数据线性可分，'KernelScale','auto' 自动选择高斯核参数。我们不考虑 fitsvm 返回的数据类型是什么，直接认为返回了一个定义了一些操作的对象，称之为 SVM 分类器。本次实验的训练时间约为 2 分半钟。

Listing 9: SVM_Training & Prediction

```
SVMModel = fitcsvm(X_train, Y_train, ...
   'Standardize', true, ...
   'KernelFunction', 'RBF', ...
   'KernelScale', 'auto');

Y_pred = predict(SVMModel, X_test);
```

(3)　预测

由于报告并不要求对于机器学习进行调参，因此不对训练结果进行评估，直接使用训练好的模型的数据采用对象的 predict 函数对测试集进行预测。在这里返回的结果 Y_pred 是一个格式和 Y_test 完全相同的数组，即每一行对应着对于测试帧的预测结果。代码也在 Listing9 当中列出了。

三、　研究结果

1.　加窗效果

由于每一帧加窗前后都有一定区别，此处仅对处理过程中有无人机和没有无人机的第一个数据集的第一帧进行加窗前后的时域图像和功率谱图像进行绘制，分别如图 1 和图 2 所示。 对于有无人机情

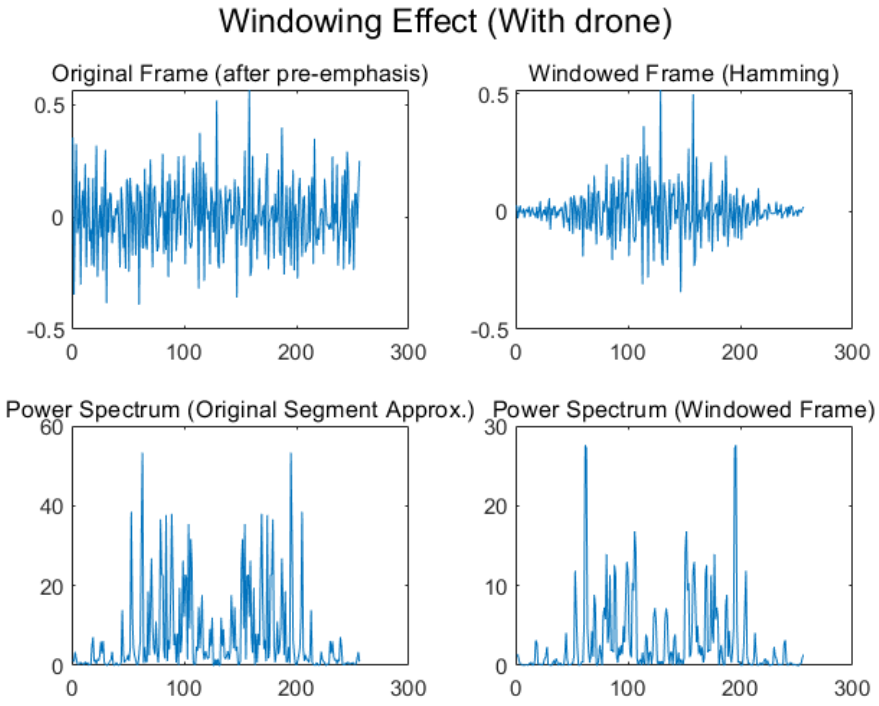

图 1: 有无人机的情形下声音信号的时域图与功率谱

形下的声音信号（图 1），可以看出经过加窗处理之后的信号幅度在大部分时间内有所减弱，并且在帧的两端被平滑地衰减到接近 0，这表明 Hamming 窗有效地减少了信号在帧边界处的不连续性；而中间部分的信号形状与原始帧保持了较为相似的形状。加窗后的功率谱不仅能量峰值有了大约百分之五十的降低，而且谱形状变得更为清晰了，说明 Hamming 窗对信号的整体能量有衰减作用，并且使得原本可能存在的频率泄漏得到了抑制，每个频率成分的能量更集中在离散的频率点上。

对于没有无人机情形下的声音信号（图 2），原始帧的时域信号振幅较小，在经过汉明窗之后的振幅则变得非常小。同样地，加窗处理过后的信号在帧的两端被平滑地衰减。相较于原始帧的功率谱，加窗后的功率谱的整体能量进一步降低，但是功率谱的峰值变得更加尖锐并且数量减少，这也表明 Hamming 窗对于频率泄漏有抑制作用。

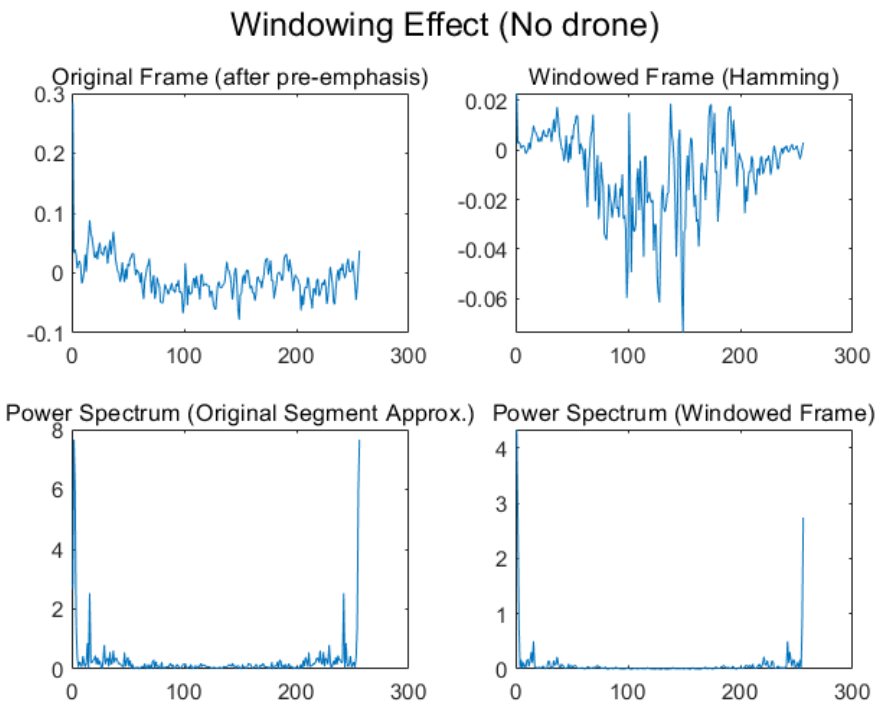

图 2: 没有无人机的情形下声音信号的时域图与功率谱

2. MFCC 特征对比

经查阅资料，要比较两种不同信号的 MFCC 特征，最好的方法是绘制并排热力图，如图 3 和图 4 所示，两张图分别表示静态 MFCC 系数和差分 MFCC 系数的热力图。在热力图中，X 轴表示时间维度上的帧数；Y 轴表示 MFCC 系数的维度，从 C0 到 C11 共 12 个 MFCC 系数，描述了频谱包络的形状；图中的颜色表示 MFCC 系数的数值大小。

对于静态 MFCC 热力图来说，注意到两个热力图的 C0 部分有很大差异。作为第一个非零倒谱系数，C0 主要捕捉的是频谱的倾斜度，当 C0 为负值时表示频谱偏向低频，即低频能量占主导，而为正值时表示频谱偏向高频，即高频能量占主导。观察有无人机情形下的热力图会发现，绝大部分时间内 C0 显示为深蓝色到浅蓝色，这表明无人机声音的频谱严重偏向低频，即其主要能量集中在较低的频率范围内。这可能是由于无人机飞行时螺旋桨按照一定（范围内）的速率旋转，因此产生低频的振动发声，而这一振动被采集了，就表现为明显的低频倾向。在没有无人机声音的右图中，C0 普遍显示为红色或黄色，因此背景噪声在低频区域具有弱得多的能量。

除此之外，可以发现有无人机的声音有着更明显的区分度，在某些行内有明显的变化，这可能是因为无人机行为的突然改变，比如启停，改变速度，完成任务等，在不同行为下发出的声音表现出不同的特征，而对于没有无人机的声音，在一行内的行为总是稳定且连续的，这表明测量时的背景噪声可能比较稳定，没有主要决定此时声音特征的单一因素。

而差分 MFCC 热力图的可解释特征十分不明显，基本上只有认为 SVM 在学习过程中可以揭示其特征。需要说明的是，上面的两个 MFCC 热力图都是针对读入的第一个有无人机/无无人机的子数据集的结果绘制的，而不是根据总数据集的结果绘制的。

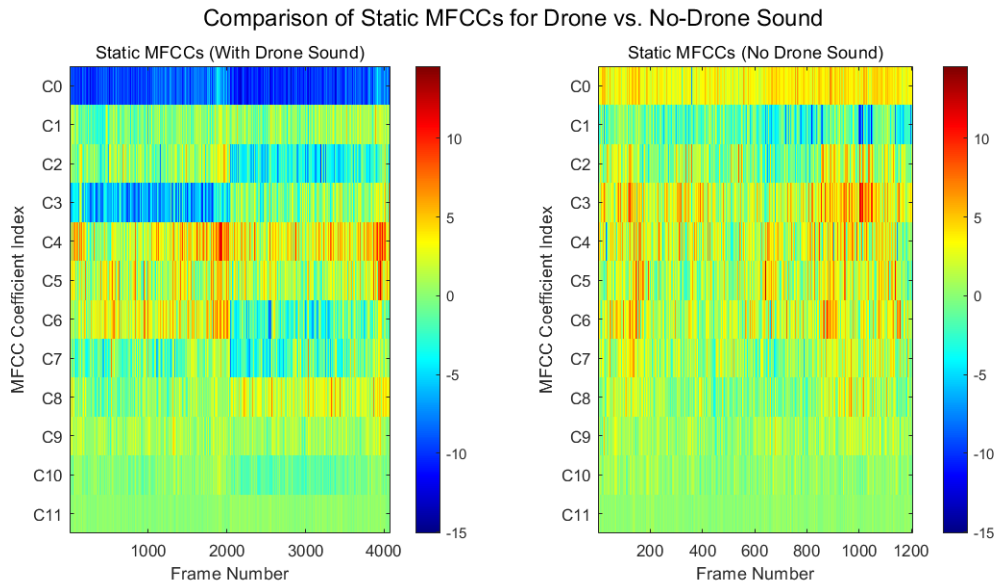

图 3: 静态 MFCC 热力图

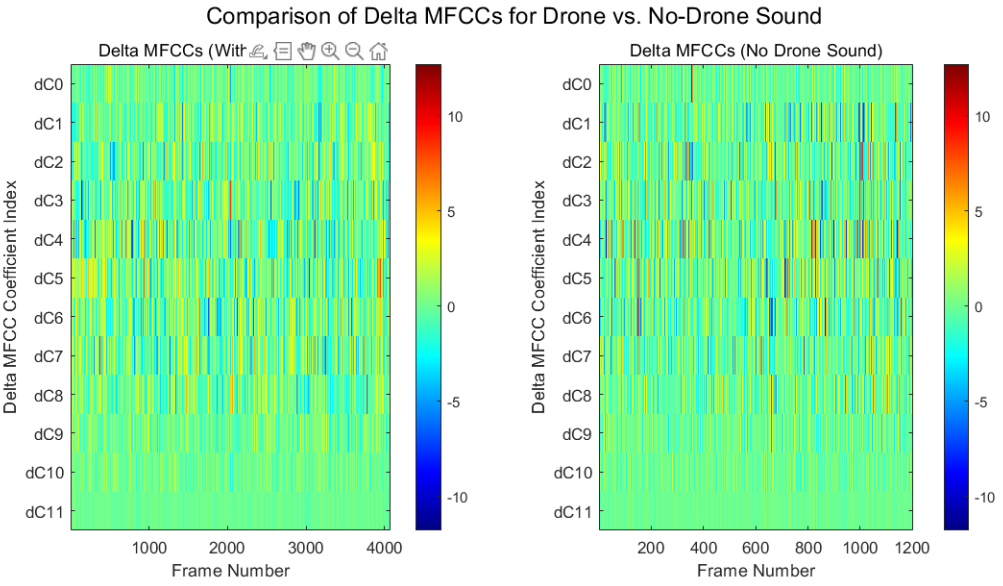

图 4: 差分 MFCC 热力图

3. 频谱特征对比

在频谱特征对比中采用的是 spectrogram 函数，该函数的行为实际上是执行短时傅里叶变换，即对分出的每一帧执行傅里叶变换，将所有帧的功率谱结果按时间顺序排列，形成一个二维矩阵，最后以热力图的形式可视化，得到的结果如图 5 和图 6 所示。图中 X 轴代表帧对应的时间，Y 轴代表频率域，以颜色表示特定时间点的帧在特定频率的功率。

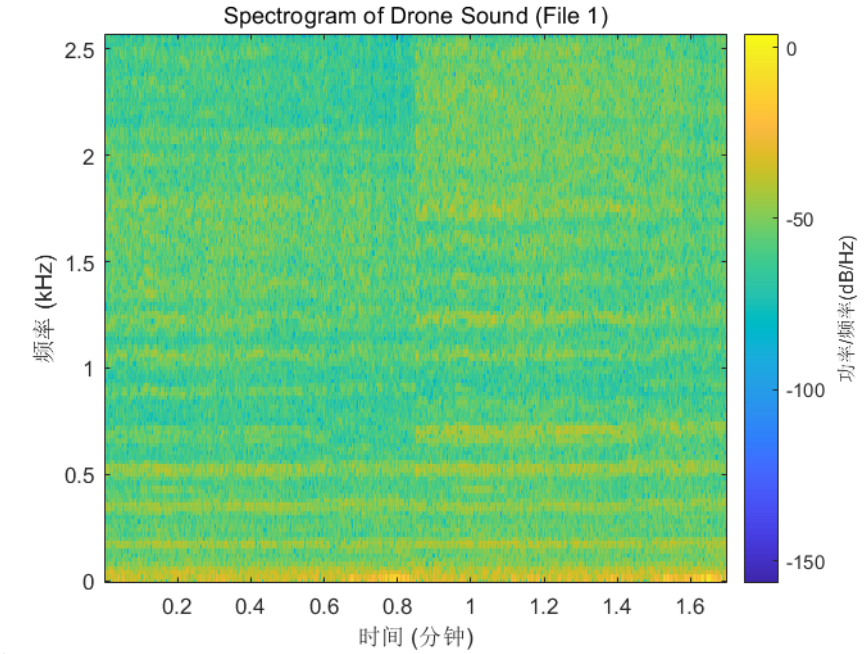

图 5: 有无人机时的频谱图

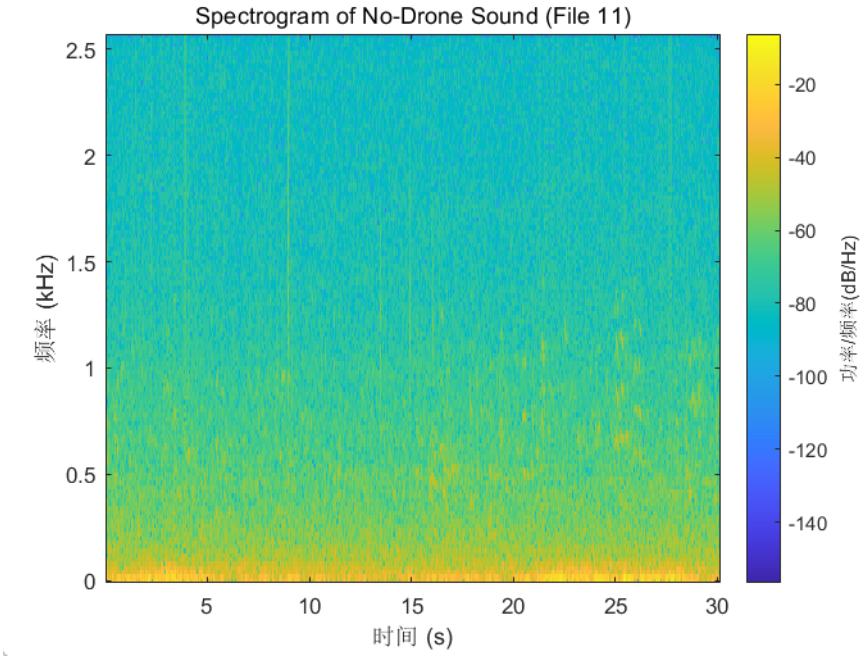

图 6: 没有无人机时的信号频谱图

对比图 5 和图 6 可以发现，在 0.5 kHz 以内的低频区域，有无人机的频谱图有非常显著且持续的黄色高能量条带，这表明无人机的声音信号的能量集中在较低的频率范围内，这和 MFCC 热力图中能够得到的结果是一致的，这应该和无人机螺旋桨和电机的行为有直接联系。

此外，还可以明显看出有无人机情形下的声音信号随时间的热力图变化比较稳定，呈现出特定的规

律与分界，比如持续的黄色水平线和 0.85min 左右的明显突变。按照 MFCC 系数热力图的解释方法，这也是因为此时无人机的行为是决定该声音信号的因素，在无人机行为没有明显变化时会呈现出稳定的特征；同时，当无人机行为发生变化时会有明显的分界。与之相比，没有无人机的背景噪声信号只有一些分立的黄色竖线，这符合噪音信号随机性和暂时性出现的特点，整体表现不出明显规律和规律变化。

4. 分类准确率

最终对于数据集的划分以及预测准确率如表 1 所示：

Total Frames	**Training Samples**	**Testing Samples**
130343	97757	32586
Correct Predictions	**Wrong Predictions**	**Accuracy**
29649	2937	90.9869%

表 1: 数据集的划分及预测准确率

混淆矩阵如下表 2 所示：

Case	P	N
T	17563	12086
F	1549	1388

表 2: 分类结果混淆矩阵

矩阵中各元素的含义为：

- **TP:** 有无人机并且预测正确
- **TN:** 没有无人机并且预测正确
- **TP:** 有无人机但是预测错误
- **TP:** 没有无人机但是预测错误

四、 研究总结

本次实验实现了基于 MFCC 特征和 SVM 的无人机声音检测系统，主要成果如下：

(1) 成功提取了声音信号的 MFCC 特征，有效表征了无人机声音的特性

(2) 通过频谱和 MFCC 特征对比，尝试分析了无人机声音的频谱特征

(3) 构建的 SVM 分类器在测试集上达到 90.9826% 的准确率

附录 A　Source Code

in MATLAB

```
clear; clc; close all;
set(0,'defaultfigurecolor','w') ;

% Declaration of Parameters
BASE_FOLDER_DRONE = 'C:\Users\jiebro\Desktop\ZJU\Second Year Latter\Signals & Systems\Design\Original_Data\Exist_UAV';
BASE_FOLDER_NO_DRONE = 'C:\Users\jiebro\Desktop\ZJU\Second Year Latter\Signals &
     Systems\Design\Original_Data\None_UAV';
FS = 5120;                % Sampling frequency in Hz
FRAME_LENGTH_SAMPLES = 256; % Frame length
FRAME_HOP_SAMPLES = 128;  % step size
NUM_MFCC_COEFFS = 12;     % Number of MFCC coefficients
NUM_MEL_FILTERS = 24;     % Number of Mel filters
PRE_EMPH_COEFF = 0.9375;  % Pre-emphasis coefficient
TRAIN_RATIO = 0.75;       % Percentage of data in the test set

allAudioData = {}; % Save all data
allLabels = []; % 1 for drone, 0 for no-drone

% Load Drone Sounds
droneFiles = dir(fullfile(BASE_FOLDER_DRONE, '*.mat')); % Open the folder
for i = 1:length(droneFiles)
   filePath = fullfile(BASE_FOLDER_DRONE, droneFiles(i).name);
   matData = load(filePath);
   % Assuming the audio data is in a variable named 'data' or the first variable
   fnames = fieldnames(matData);
   audioSignal = matData.(fnames{1});
   allAudioData{end+1} = audioSignal;
   allLabels(end+1) = 1;
end

% Load No-Drone Sounds
noDroneFiles = dir(fullfile(BASE_FOLDER_NO_DRONE, '*.mat'));
for i = 1:length(noDroneFiles)
   filePath = fullfile(BASE_FOLDER_NO_DRONE, noDroneFiles(i).name);
   matData = load(filePath);
   fnames = fieldnames(matData);
   audioSignal = matData.(fnames{1});
   allAudioData{end+1} = audioSignal;
   allLabels(end+1) = 0;
end

% Feature Extraction
allFeatures = [];
allFrameLabels = [];

% Define Hamming window for enframe or manual application
window = hamming(FRAME_LENGTH_SAMPLES); %

% DCT matrix
dctcoef = zeros(NUM_MFCC_COEFFS, NUM_MEL_FILTERS);
for k = 1:NUM_MFCC_COEFFS
   n_dct = 0:(NUM_MEL_FILTERS-1);
   dctcoef(k,:) = cos((2*n_dct+1)*k*pi/(2*NUM_MEL_FILTERS));
end

% Cepstral liftering weights
N_lifter = 1:NUM_MFCC_COEFFS;
lifter_w = 1 + 6 * sin(pi * N_lifter' / NUM_MFCC_COEFFS);
lifter_w = lifter_w / max(lifter_w); % Normalize

% Mel filter bank
melBank = melbankm(NUM_MEL_FILTERS, FRAME_LENGTH_SAMPLES, FS, 0, 0.5, 'm'); % 'm' for Mel scale
melBank = full(melBank);
melBank = melBank / max(melBank(:)); % Normalize
```

```
% Plotting setup for one drone and one no-drone file
droneFileIdx = find(allLabels == 1, 1, 'first');
noDroneFileIdx = find(allLabels == 0, 1, 'first');
plottedWindowEffect_withdrone = false;
plottedWindowEffect_nodrone = false;
plottedSpectrograms = false;

% Plotting setup for heat map
mfccs_drone_plot_static = [];
mfccs_nodrone_plot_static = [];
mfccs_drone_plot_delta = [];
mfccs_nodrone_plot_delta = [];

for i = 1:length(allAudioData)
   signal = allAudioData{i};
   currentLabel = allLabels(i);
   % Pre-emphasis
   signal_preemph = filter([1 -PRE_EMPH_COEFF], 1, signal);

   % Framing
   frames_raw = enframe(signal_preemph, FRAME_LENGTH_SAMPLES, FRAME_HOP_SAMPLES);
   frames=frames_raw;
   numFrames = size(frames, 1);
   mfccs_static_current_file = zeros(numFrames, NUM_MFCC_COEFFS);

   for j = 1:numFrames
      % Windowing
      frames(j, : )=frames(j, : )'.*window;
      frameCur = frames(j, :);

      % Show windowing effect
      if ~plottedWindowEffect_withdrone && currentLabel
         figure;
         subplot(2,2,1); plot(frames_raw(j,:)); title('Original Frame (after pre-emphasis)');
         subplot(2,2,2); plot(frameCur); title('Windowed Frame (Hamming)');
         subplot(2,2,3); plot(abs(fft(frames_raw(j, : ))).^2); title('Power Spectrum (Original Segment Approx.)');
         subplot(2,2,4); plot(abs(fft(frameCur)).^2); title('Power Spectrum (Windowed Frame)');
         sgtitle('Windowing Effect (With drone)');
         plottedWindowEffect_withdrone = true;
      end

      if ~plottedWindowEffect_nodrone && ~currentLabel
         figure;
         subplot(2,2,1); plot(frames_raw(j,:)); title('Original Frame (after pre-emphasis)');
         subplot(2,2,2); plot(frameCur); title('Windowed Frame (Hamming)');
         subplot(2,2,3); plot(abs(fft(frames_raw(j, : ))).^2); title('Power Spectrum (Original Segment Approx.)');
         subplot(2,2,4); plot(abs(fft(frameCur)).^2); title('Power Spectrum (Windowed Frame)');
         sgtitle('Windowing Effect (No drone)');
         plottedWindowEffect_nodrone = true;
      end

      % FFT
      frameFFT = fft(frameCur, FRAME_LENGTH_SAMPLES);
      % Power Spectrum
      powerSpec = abs(frameFFT(1:(FRAME_LENGTH_SAMPLES/2 + 1))).^2;
      % Mel Filterbank Application
      melPowerCur = melBank * powerSpec';
      melPowerCur_log = log(max(melPowerCur, 1e-6)); % Add small constant to avoid log(0)
      % DCT
      mfcc_frame_raw = dctcoef * melPowerCur_log;
      % Cepstral liftering
      mfcc_frame_liftered = mfcc_frame_raw .* lifter_w;
      mfccs_static_current_file(j, :) = mfcc_frame_liftered';
   end

   % Delta MFCCs
   mfccs_delta_current_file = zeros(numFrames - 4, NUM_MFCC_COEFFS);
   denominator_delta = 3;
```

```
    for k = 1:NUM_MFCC_COEFFS
        m_coeff_col = mfccs_static_current_file(:, k);
        for frame_idx = 3:(numFrames - 2)
            mfccs_delta_current_file(frame_idx-2, k) = ...
                (2*m_coeff_col(frame_idx+2) + m_coeff_col(frame_idx+1) - ...
                m_coeff_col(frame_idx-1) - 2*m_coeff_col(frame_idx-2)) / denominator_delta;
        end
    end

    % Storing the data needed to create a heat map
    if i == droneFileIdx
        mfccs_drone_plot_static = mfccs_static_current_file';
        mfccs_drone_plot_delta = mfccs_delta_current_file';
    end
    if i == noDroneFileIdx
        mfccs_nodrone_plot_static = mfccs_static_current_file';
        mfccs_nodrone_plot_delta = mfccs_delta_current_file';
    end

    % Combine static and delta features
    mfccs_static_for_combination = mfccs_static_current_file(3:(numFrames-2), :);
    mfccs_combined = [mfccs_static_for_combination, mfccs_delta_current_file];
    valid_indices = 3:(numFrames-2);

    % Plot spectrogram and average MFCC for with-drone/no-drone file
    if ~plottedSpectrograms && (i == droneFileIdx || i == noDroneFileIdx)
        figure;
        spectrogram(signal, window, FRAME_LENGTH_SAMPLES - FRAME_HOP_SAMPLES, FRAME_LENGTH_SAMPLES, FS, 'yaxis');
        if currentLabel == 1
            title(['Spectrogram of Drone Sound (File ', num2str(i), ')']);
        else
            title(['Spectrogram of No-Drone Sound (File ', num2str(i), ')']);
        end
    end

    allFeatures = [allFeatures; mfccs_combined];
    allFrameLabels = [allFrameLabels; repmat(currentLabel, size(mfccs_combined, 1), 1)];
end

disp(['Total frames for training/testing: ', num2str(size(allFeatures, 1))]);

% Data set segmentation
numTotalFrames = size(allFeatures, 1);
rng('default');
shuffledIndices = randperm(numTotalFrames); %

numTrainFrames = floor(TRAIN_RATIO * numTotalFrames);
trainIndices = shuffledIndices(1:numTrainFrames);
testIndices = shuffledIndices(numTrainFrames+1:end);

X_train = allFeatures(trainIndices, :);
Y_train = allFrameLabels(trainIndices, :);
X_test = allFeatures(testIndices, :);
Y_test = allFrameLabels(testIndices, :);

disp(['Training samples: ', num2str(size(X_train, 1))]);
disp(['Testing samples: ', num2str(size(X_test, 1))]);

% Training
SVMModel = fitcsvm(X_train, Y_train, ...
    'Standardize', true, ...
    'KernelFunction', 'RBF', ...
    'KernelScale', 'auto');
disp('SVM training complete.');

% Prediction and Evaluation
disp('5. Predicting on test set and evaluating...');
Y_pred = predict(SVMModel, X_test);
```

```
% Get the Confusion-Matrix
C = confusionmat(Y_test, Y_pred);
TN = C(1,1); % True Negative
FP = C(1,2); % False Positive
FN = C(2,1); % False Negative
TP = C(2,2); % True Positive

disp(['True Negatives (TN): ', num2str(TN)]);
disp(['False Positives (FP): ', num2str(FP)]);
disp(['False Negatives (FN): ', num2str(FN)]);
disp(['True Positives (TP): ', num2str(TP)]);

% Show the accuracy
correctPredictions = sum(Y_pred == Y_test);
totalTestSamples = length(Y_test);
accuracy = correctPredictions / totalTestSamples;
disp(['Accuracy: ', num2str(accuracy * 100), '%']);

% Heat mapping of static MFCC coefficients
figure('Name', 'Static MFCC Side-by-Side Comparison', 'NumberTitle', 'off', 'Position', [100, 100, 1000, 500]);

min_static_val = min([mfccs_drone_plot_static(:); mfccs_nodrone_plot_static(:)]);
max_static_val = max([mfccs_drone_plot_static(:); mfccs_nodrone_plot_static(:)]);

% With drone
subplot(1, 2, 1);
imagesc(mfccs_drone_plot_static);
colormap('jet');
caxis([min_static_val, max_static_val]);
colorbar;
title('Static MFCCs (With Drone Sound)');
xlabel('Frame Number');
ylabel('MFCC Coefficient Index');
yticks(1:NUM_MFCC_COEFFS);
yticklabels(arrayfun(@(x) sprintf('C%d', x-1), 1:NUM_MFCC_COEFFS, 'UniformOutput', false));

% No drone
subplot(1, 2, 2);
imagesc(mfccs_nodrone_plot_static);
colormap('jet');
caxis([min_static_val, max_static_val]);
colorbar;
title('Static MFCCs (No Drone Sound)');
xlabel('Frame Number');
ylabel('MFCC Coefficient Index');
yticks(1:NUM_MFCC_COEFFS);
yticklabels(arrayfun(@(x) sprintf('C%d', x-1), 1:NUM_MFCC_COEFFS, 'UniformOutput', false));

sgtitle('Comparison of Static MFCCs for Drone vs. No-Drone Sound');

% Heat mapping of static MFCC coefficients
figure('Name', 'Delta MFCC Side-by-Side Comparison', 'NumberTitle', 'off', 'Position', [100, 100, 1000, 500]); % 新图窗

min_delta_val = min([mfccs_drone_plot_delta(:); mfccs_nodrone_plot_delta(:)]);
max_delta_val = max([mfccs_drone_plot_delta(:); mfccs_nodrone_plot_delta(:)]);

% With drone
subplot(1, 2, 1);
imagesc(mfccs_drone_plot_delta);
colormap('jet');
caxis([min_delta_val, max_delta_val]);
colorbar;
title('Delta MFCCs (With Drone Sound)');
xlabel('Frame Number');
ylabel('Delta MFCC Coefficient Index');
yticks(1:NUM_MFCC_COEFFS);
yticklabels(arrayfun(@(x) sprintf('dC%d', x-1), 1:NUM_MFCC_COEFFS, 'UniformOutput', false)); % 标记为 dC0, dC1...

% No drone
```

```
subplot(1, 2, 2);
imagesc(mfccs_nodrone_plot_delta);
colormap('jet');
caxis([min_delta_val, max_delta_val]);
colorbar;
title('Delta MFCCs (No Drone Sound)');
xlabel('Frame Number');
ylabel('Delta MFCC Coefficient Index');
yticks(1:NUM_MFCC_COEFFS);
yticklabels(arrayfun(@(x) sprintf('dC%d', x-1), 1:NUM_MFCC_COEFFS, 'UniformOutput', false));

sgtitle('Comparison of Delta MFCCs for Drone vs. No-Drone Sound');
```