在 Android 项目中通过 JNI 导入和使用 KissFFT 实现频谱图计算，可以按照以下步骤完成。KissFFT 是一个轻量级的 FFT 库，非常适合嵌入式和移动平台。

步骤概述

1. 下载并集成 KissFFT 到 Android 项目。
2. 配置 CMakeLists.txt 构建系统。
3. 使用 KissFFT 进行 FFT 计算。
4. 使用 JNI 将 C++ 功能暴露给 Java 层。

1. 下载和集成 KissFFT

从 KissFFT GitHub 仓库 下载源代码并添加到你的项目中。

```
git clone https://github.com/mborgerding/kissfft.git
```

将 kissfft 文件夹放入你的 Android 项目的 app/src/main/cpp/ 目录中。

2. 配置 CMakeLists.txt

在 Android 项目的 CMakeLists.txt 中添加 KissFFT 支持。

示例 CMakeLists.txt：

```
cmake_minimum_required(VERSION 3.10.2)

# 设置项目名称
project(audio_processor)

# 设置支持的架构
set(CMAKE_TOOLCHAIN_FILE ${ANDROID_NDK}/build/cmake/android.toolchain.cmake)
set(ANDROID_ABI "arm64-v8a")

# 包含 KissFFT 头文件
include_directories(src/main/cpp/kissfft)

# 添加源文件
add_library(audio_processor SHARED
    src/main/cpp/AudioProcessor.cpp
    src/main/cpp/kissfft/kiss_fft.c
    src/main/cpp/kissfft/tools/kiss_fftr.c)

# 链接 Android log 库
target_link_libraries(audio_processor
```

```
    log)
```

3. FFT 功能实现

在 AudioProcessor.cpp 中实现 FFT 和频谱图计算。

示例代码：

```
#include <jni.h>
#include <vector>
#include <cmath>
#include <cstring>
#include "kissfft/kiss_fft.h"
#include "kissfft/tools/kiss_fftr.h"

#define FFT_SIZE 1024  // FFT 窗口大小

extern "C" {

// 计算频谱图
JNIEXPORT jfloatArray JNICALL
Java_com_example_audio_AudioProcessor_
calculateSpectrogram(JNIEnv *env,
jobject thiz, jfloatArray audioData) {
    // 获取输入数据
    jsize length = env-
>GetArrayLength(audioData);
```

```
    std::vector<float>
inputData(length);
    env-
>GetFloatArrayRegion(audioData, 0,
length, inputData.data());

    // 初始化 KissFFT 配置
    kiss_fftr_cfg cfg =
kiss_fftr_alloc(FFT_SIZE, 0, nullptr,
nullptr);

    // 定义输入和输出
    std::vector<float>
windowedInput(FFT_SIZE, 0.0f);  // 窗
口化的输入
    std::vector<kiss_fft_cpx>
spectrum(FFT_SIZE / 2 + 1);  // FFT 结
果

    // Hann 窗口
    for (int i = 0; i < FFT_SIZE; ++i)
{
        float hann = 0.5f * (1 -
std::cos(2 * M_PI * i / (FFT_SIZE -
1)));
        windowedInput[i] =
inputData[i] * hann;
    }
```

```
    // 执行 FFT
    kiss_fftr(cfg,
windowedInput.data(),
spectrum.data());

    // 计算频谱幅度
    std::vector<float>
magnitudes(FFT_SIZE / 2);
    for (int i = 0; i < FFT_SIZE / 2;
++i) {
        magnitudes[i] =
std::sqrt(spectrum[i].r *
spectrum[i].r + spectrum[i].i *
spectrum[i].i);
    }

    // 释放资源
    free(cfg);

    // 返回结果到 Java 层
    jfloatArray result = env-
>NewFloatArray(magnitudes.size());
    env->SetFloatArrayRegion(result,
0, magnitudes.size(),
magnitudes.data());
    return result;
}
}
```

4. Java 层 JNI 调用

在 Java 层调用 JNI 方法来计算频谱图。

Java 代码：

```
package com.example.audio;

public class AudioProcessor {
    static {

System.loadLibrary("audio_processor");
    }

    // 定义原生方法
    public native float[]
calculateSpectrogram(float[]
audioData);

    // 使用示例
    public void processAudio(float[]
audioData) {
        float[] spectrogram =
calculateSpectrogram(audioData);

System.out.println("Spectrogram
calculated with length: " +
spectrogram.length);
    }
```

```
}
```

5. Gradle 配置

确保在 build.gradle 中启用了 CMake 支持。

示例 build.gradle：

```
android {
    compileSdkVersion 30

    defaultConfig {
        applicationId
"com.example.audio"
        minSdkVersion 21
        targetSdkVersion 30

        externalNativeBuild {
            cmake {
                path "CMakeLists.txt"
            }
        }
    }

    externalNativeBuild {
        cmake {
            path "CMakeLists.txt"
        }
    }
}
```

运行效果

1. 使用 calculateSpectrogram 方法，传入音频数据（如 PCM 浮点数组）。
2. JNI 层完成 FFT 计算，并返回频谱幅度数组。
3. Java 层可以