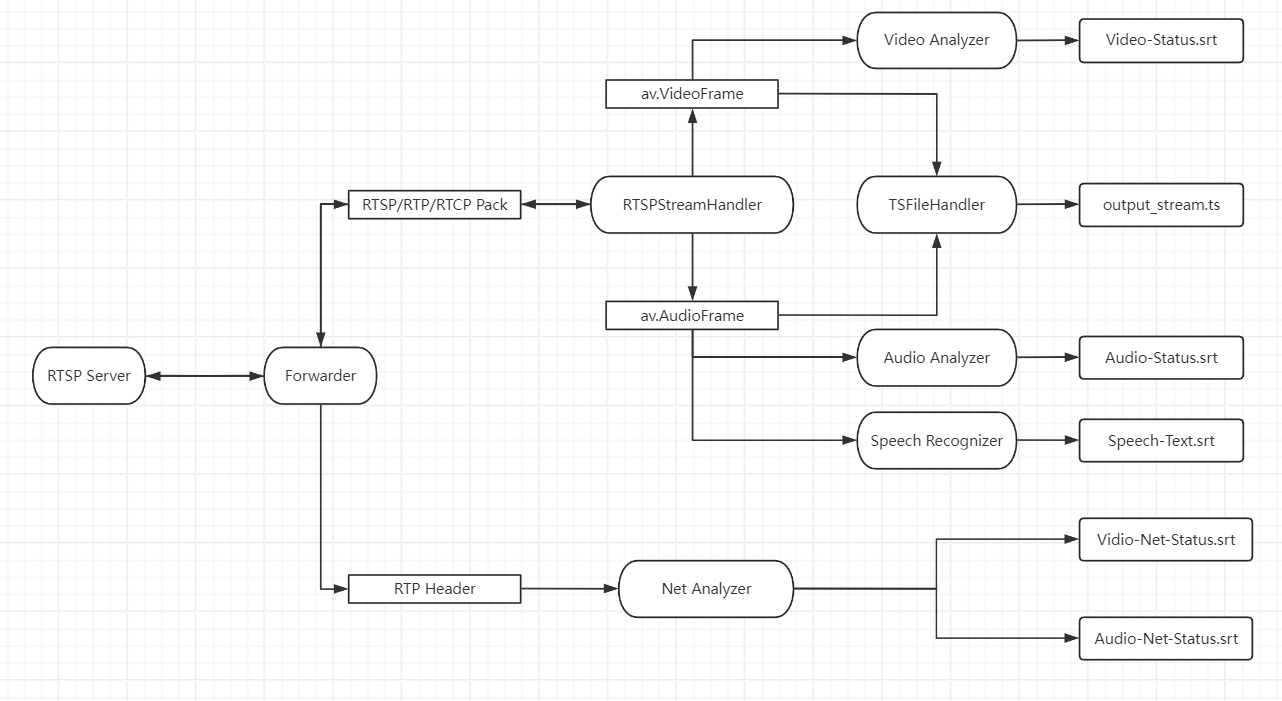
**RTSP流媒体网络中设备B（客户端）的使用方法**

**平台:** amd64/arm64

**系统:** Linux/Windows/Maos

**程序介绍**

框架图：



介绍：

·RTSPStreamHandler 进程使用Pyav库拉取RTSP流，获取VideoFrame和AudioFrame，并将帧数据通过进程队列发送给其他进程

· TSFileHandler 进程使用Pyav库将从进程队列中VideoFrame和AudioFrame同步后写入TS文件

·Video Analyer进程分析从进程队列中获取的VideoFrame，并将分析结果写入 Video-Status.srt文件中

·Audio Analyer进程分析从进程队列中获取的AudioFrame，并将分析结果写入 Audio-Status.srt文件中

·Speech Recognizer 进程从进程队列中获取AudioFrame，并基于Vosk 框架在本地进行语音识别，后将识别结果写入 Audio-Status.srt文件中

·Forwarder 进程负责转发RTSP/RTP/RTCP包，以及拷贝服务端发送的RTP包头到指定的进程队列

·Net Analyzer 进程负责从进程队列中获取RTP包头，利用RTP包头信息计算抖动和丢包，利用ping方法计算延迟。

**测试**

main.0.1.0.py：

RTSP Server <---> RTSPStreamHandler ---> TSFileHandler ---> output\_stream.ts

仅在拉取RTSP流后保存为TS文件

main.0.1.1.py：

...RTSPStreamHandler ---> TS...

...RTSPStreamHandler ---> Video analyzer ---> Video-Status.srt

添加Video Analyzer 进程，分析视频轨道

main.0.1.2.py：

...RTSPStreamHandler ---> TS...

...RTSPStreamHandler ---> Video...

...RTSPStreamHandler ---> Audio analyzer ---> Audio-Status.srt

添加Audio Analyzer 进程，分析音频轨道

main.0.2.0.py：

RTSP Server <---> Forwarder <---> RTSPStreamHandler...

...RTSPStreamHandler ---> TS...

...RTSPStreamHandler ---> Video...

...RTSPStreamHandler ---> Audio...

...Forwarder ---> Net Analyzer --->Video-Net-Status.srt/Audio-Net-Status.srt

添加转发器和网络分析器， 分析网络延迟、抖动、丢包

main.0.3.0.py

...RTSPStreamHandler ---> TS...

...RTSPStreamHandler ---> Video...

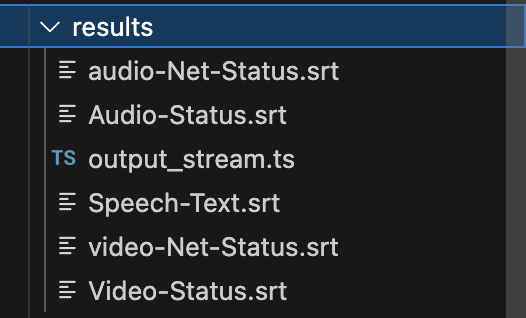
...RTSPStreamHandler ---> Audio...

..RTSPStreamHandler ---> Speech Recognizer ---> Speech-Text.srt

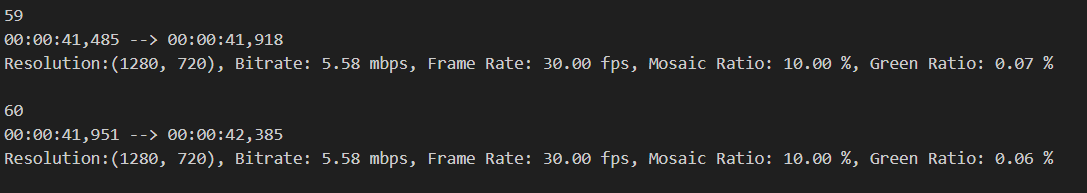
添加Speech Recognizer 进程，识别音频（基于Vosk框架本地识别）

**结果**

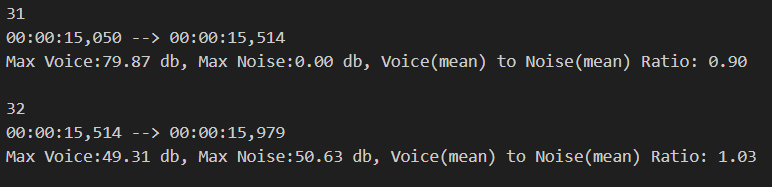
1.results 文件夹如期出现各个文件



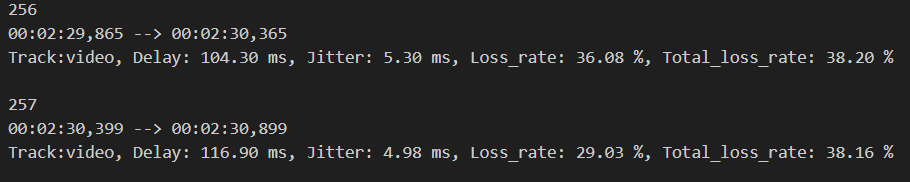
2.成功解析视频的分辨率（1280，720），比特率（5.58 mbps），帧率（30 fps），马赛克比例（10.00%），绿屏比例（0.07%）



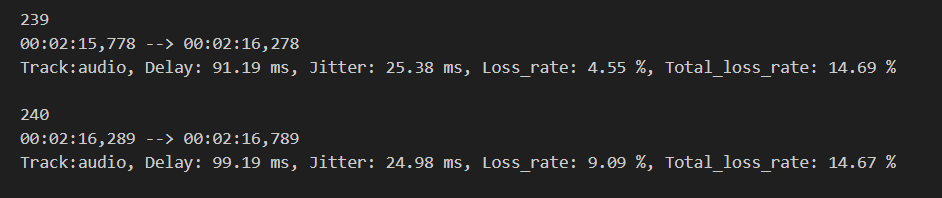
3.成功解析音频中人声和噪音的最高分贝值， 人声平均分贝值与噪音平均分贝值的比值



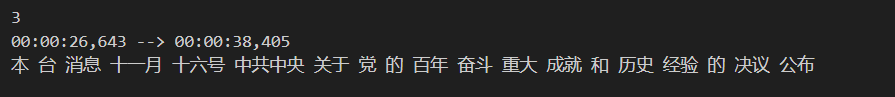
4.成功解析传输视频时的延迟、抖动、丢包率（每500ms）和总丢包率



5.成功解析传输音频时的延迟、抖动、丢包率（每500ms）和总丢包率

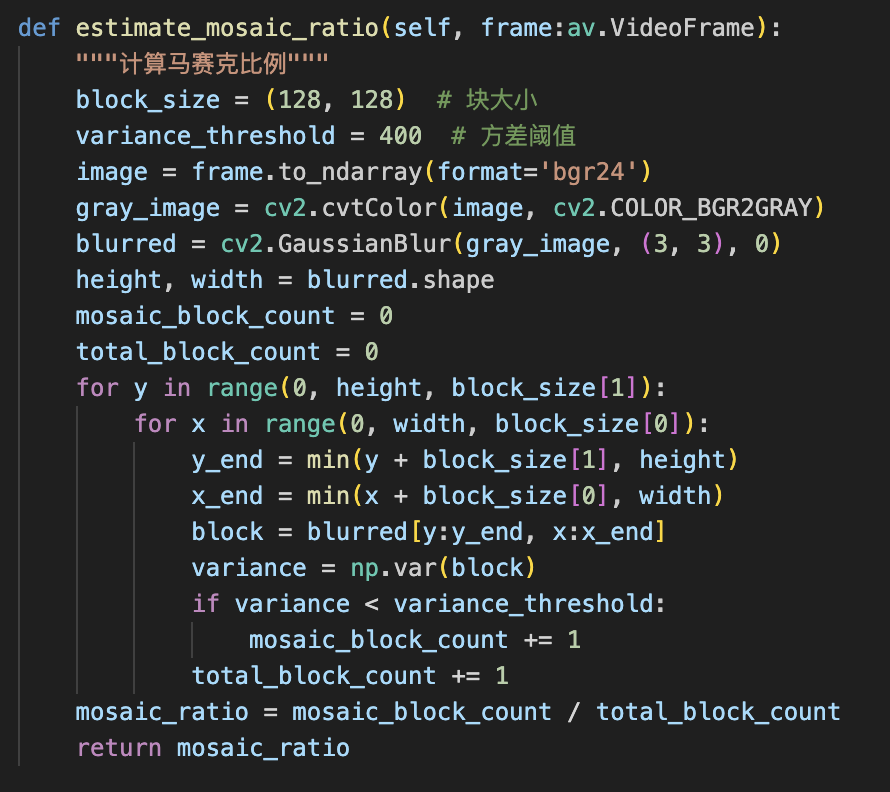


6.成功识别中文语音



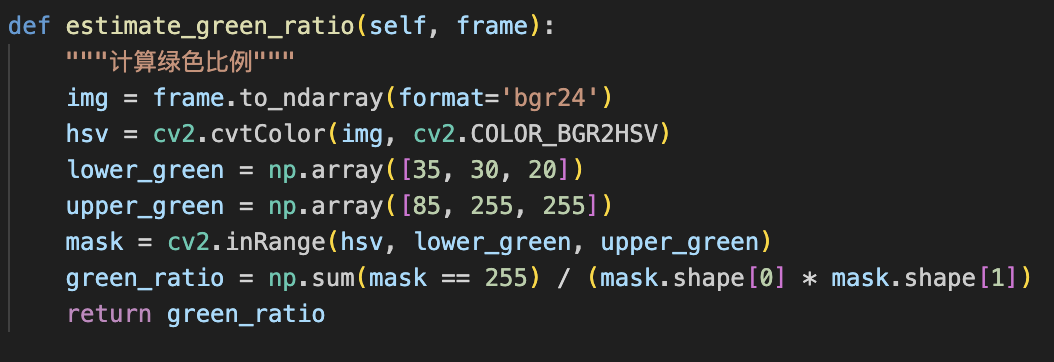
**不足以及改进方向**

1.马赛克计算方法：通过分析图像块的方差来确定马赛克



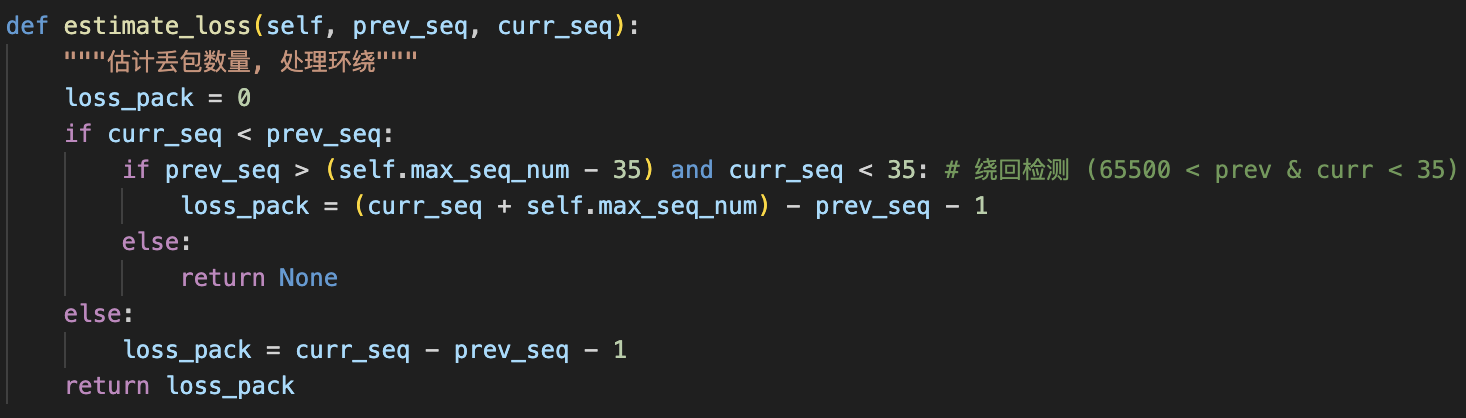
1. 块的大小、方差阈值、高斯模糊核心等参数仍需调教  
    （2）可尝试利用GPU加速处理过程

2.绿屏计算方法：直接计算绿色像素的占比



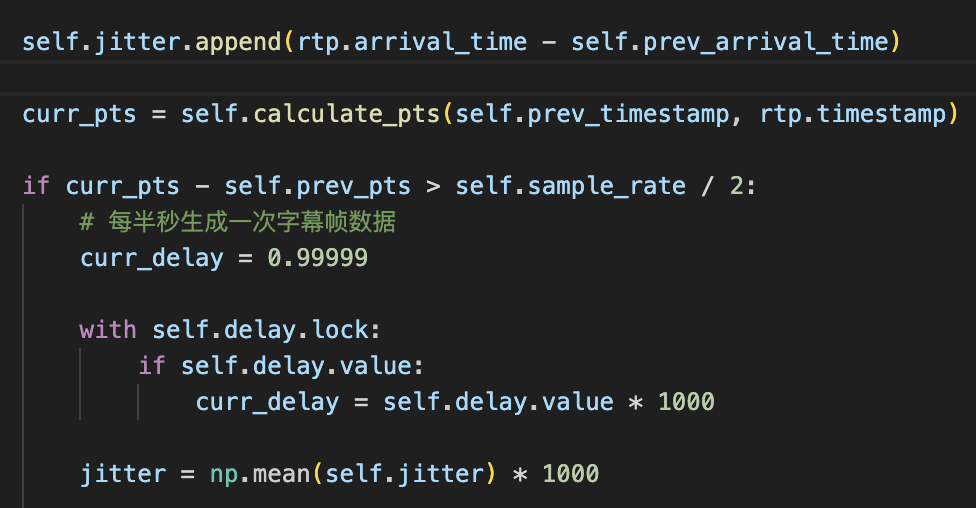
1. 流媒体中的绿屏可能是帧损坏造成的，方法仍需改进
2. 可尝试利用GPU加速处理过程

3.丢包率计算：计算前一个RTP包头的Seq和当前RTP包头的Seq差值



1. FFmpeg或Pyav在拉取RTSP流时对RTP包有一套缓存流程，延迟到达包不一定会直接丢弃。但是在本方案中将延迟到达的包直接算作丢包，故丢包率可能比实际高。

4.抖动计算：500ms内每两个连续的RTP包到达时间的差的均值



1. 可能无法体现发送端的抖动情况，应同RTP丢包率计算方法一起改进

5.卡顿率的计算（预想）：RTSPStreamHandler在获取帧数据时计算帧与帧之间的间隔，若超过一定时限（例如100ms）算一次卡顿，记录卡顿的时间。卡顿率 = 卡顿时长/（最后一帧.play\_time - 最初一帧.play\_time）

1. Forwarder 类可以尝试用C/C++实现，提高效率。