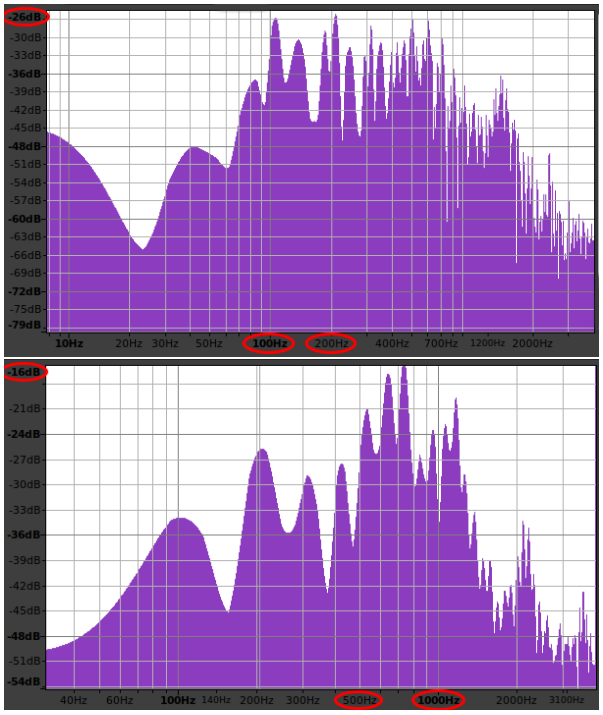
Report finale progetto per algoritmo VAD

La realizzazione dell'algoritmo nasce da una analisi attenta dei file di input tramite il software Audacity.

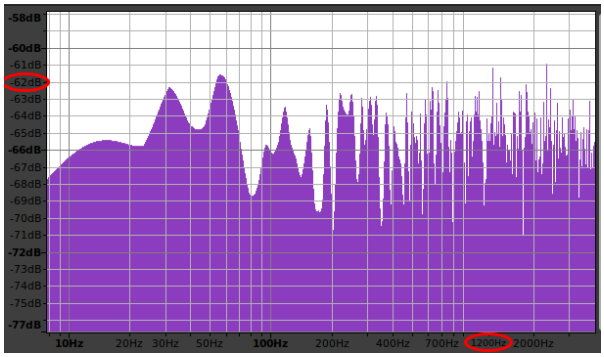
In particolare, grazie alla funzione "Mostra spettro", ho potuto analizzare pacchetti nel dominio delle frequenze.

Ho notato che l'intervallo delle frequenze non era ben delineato. Infatti, la voce ha delle frequenze che variano dai 100 Hz ai 1200 Hz in base al timbro vocale della persona (quindi cambia a seconda del file di input), ma anche in alcuni intervalli in cui è presente solo il rumore di sottofondo sono queste le frequenze dominanti.

Alcuni esempi:



Esempi di due pacchetti di voce in fileinput diversi

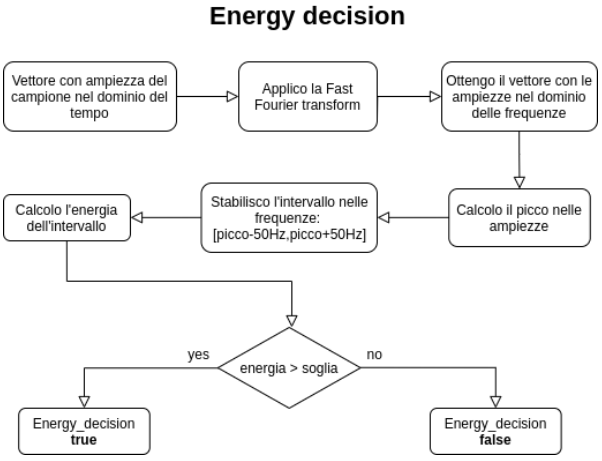


Esempio di un pacchetto di silenzio in un fileinput

Ho quindi notato che la principale differenza tra un pacchetto con voce e uno con silenzio risiedeva nella potenza del segnale.

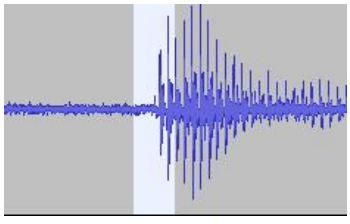
Una prima parte del mio algoritmo ha come obbiettivo quello di:

1. Trasformare il pacchetto in considerazione del segnale nel dominio delle frequenze.
2. Trovarne il picco.
3. Stabilire un intervallo di frequenze: [picco-50Hz, picco+50Hz].
4. Calcolare l'energia nell'intervallo.
5. Valutare se l'energia è superiore ad una soglia stabilita a priori oppure no.



Ho deciso di mantenere una soglia alta ($6*10^7$) per essere sicuro di valutare correttamente un pacchetto di voce e non confonderlo con un alto rumore di sottofondo.

Questo primo algoritmo ho notato non fosse però abbastanza efficiente in quanto in alcune situazioni (in particolare all'inizio del parlato e alla fine) venivano scartati pacchetti perché non avevano energia sufficiente, come in questo esempio:

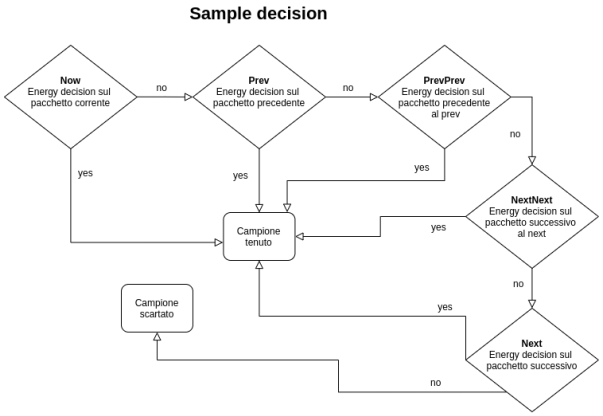


La voce risultava quindi troppo tagliata e in alcuni casi non comprensibile.

Ho deciso di integrare quindi questo primo algoritmo di decisione con un algoritmo che fosse legato all'energia dei campioni successivi e precedenti.

Dato che il ritardo complessivo di codifica deve rimanere al di sotto di 50 ms, ed ogni pacchetto di 160 campioni corrisponde a 20ms, è possibile valutare l'energia dei soli due campioni successivi.

Sostanzialmente anche se l'energia di un pacchetto non è sufficiente, ma lo è uno tra i due precedenti oppure tra i due futuri, il pacchetto viene mantenuto. Questo meccanismo permette di allungare i tempi del parlato e di evitare l'effetto di clipping.



I risultati ottenuti sono stati soddisfacenti.

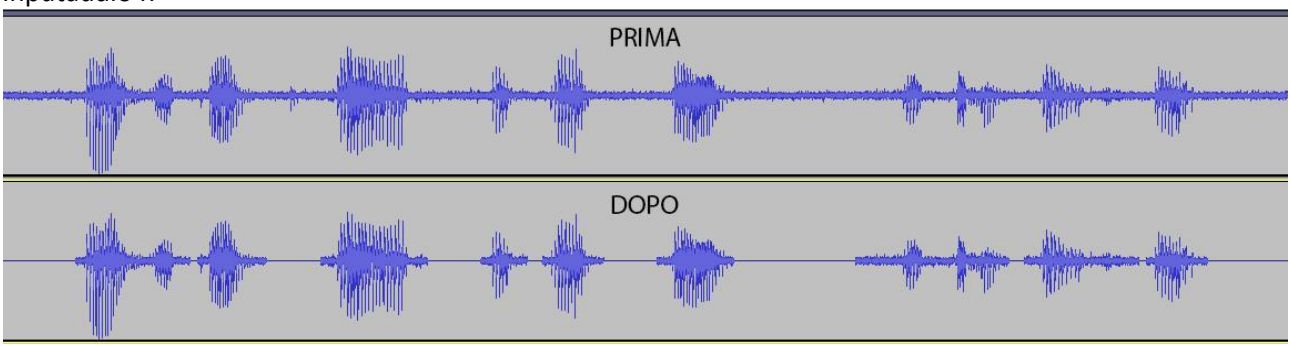
L'algoritmo non gestisce l'ultimo pacchetto (con meno di 160 campioni) se i campioni di ingressi non sono multipli di 160.

Per poter utilizzare la funzione FFT in C++, sono state trovate solo versioni che funzionano se i campioni di ingresso sono potenze di 2. Per questo motivo l'energy decision viene effettuato con anche 96 campioni del pacchetto precedente (96+160 =256). Utilizzando il pacchetto precedente e non quello successivo non si rischia di sforare i 50ms con la valutazione dell'energia nei campioni next e nextNext.

Solo nel caso del primo pacchetto vengono considerati campioni di quello successivo.

Nella scelta del picco, viene scartata la frequenza 0 Hz che spesso è predominante e non permette di individuare la natura del pacchetto.

Inputaudio4:



Inputaudio5: