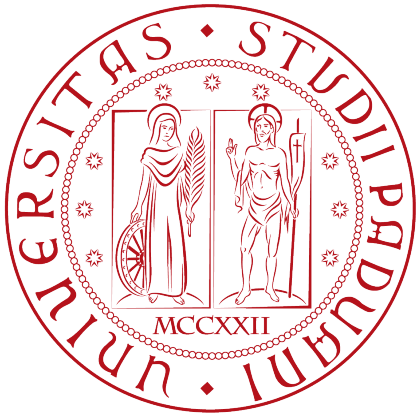

Voice Activity Detection su segnali audio rumorosi mediante analisi Wavelet

Andrea Cracco
Matricola: 586111
Relatore: Prof. Federico Avanzini
Correlatore: Ing. Enrico Marchetto
Università degli Studi di Padova
Facoltà di Ingegneria
Laurea Specialistica in Ingegneria Informatica

21 aprile 2010

Dedicato a tutta la mia famiglia. Un ringraziamento speciale a mamma Bruna per l'aiuto e il sostenimento ricevuti in questi anni.

Indice

Capitolo 1

Introduzione

Negli ultimi anni si è assistito ad una costante crescita della capacità elaborativa dei sistemi e della capacità dei mezzi di archiviazione, unita ad un costante abbassamento dei loro costi. Questo, complice l'aumento di canali televisivi e radiofonici (basti pensare al Dvb-T) o l'uso di registrazioni ambientali a scopo di indagine, ha portato a una sistematica digitalizzazione dell'informazione, generando un forte aumento di archivi di informazione audio. Questa informazione si può definire come una collezione di documenti parlati o documenti audio, cioè delle registrazioni audio, a singolo o doppio canale, contenenti molteplici sorgenti come differenti parlatori, musica, differenti rumori di fondo, differenti tipi di parlato (spontaneo come in conversazioni telefoniche o non spontaneo come in registrazioni di notiziari), differenti tipi di canale su cui poggia il parlato (canale telefonico tradizionale o cellulare, registrazioni microfoniche) e così via. Un interessante problema relativo alla gestione di questi grandi volumi di audio archiviato è il *reperimento dell'informazione* contenuta negli stessi, cioè l'insieme delle tecniche utilizzate per estrarre informazioni utili da tali archivi. Estrarre informazione da un documento audio, anche intuitivamente, risulta molto più difficile rispetto allo stesso processo effettuato su documenti testuali. Si intuisce dunque il crescente bisogno di tecnologie automatiche di estrazione di informazione da tali documenti allo scopo di accesso, ricerca e indicizzazione. Un primo esempio di estrazione di informazione da documenti audio può essere la trascrizione di parlato registrato. Estrarre le parole da una registrazione può essere un processo basilare relativamente semplice, per produrre una trascrizione che però risulta difficile da leggere e non cattura tutta l'informazione presente: 'chi sta parlando?', 'si tratta di una domanda o una affermazione?', 'qual è lo stato d'animo del parlatore?'. Con questo semplice esempio ci si rende conto di quanto siano necessarie altre tecnologie utili ad estrarre i meta-dati presenti in un parlato. I *meta-dati*, come l'informazione su dove avvengono i cambi di parlatore oppure l'informazione sull'identità di chi sta parlando, permettono una più ricca e significativa

trascrizione dei dati e, più in generale, costituiscono una importante fonte di informazione utile. Basti pensare ad un esempio pratico come l'ascolto di una lunga registrazione ambientale in cui il parlatore di interesse parla solo per poco tempo rispetto ad un altro parlatore non di interesse che parla per molto più tempo. Tali informazioni permettono di identificare velocemente la regione di parlato utile.
Risulta dunque interessante, ai fini dell'estrazione dei metadati, il processo di *Voice Activity Detection (VAD)*, cioè il compito di discriminare le regioni audio in parlato e non parlato, dove per non parlato si intende solitamente la musica, il silenzio, il rumore di fondo e così via. Infatti in molte applicazioni di interesse l'audio non-parlato non porta informazioni interessanti da categorizzare.
In questo lavoro di tesi, si è voluto utilizzare un approccio basato sulle Wavelets, in quanto possiedono alcuni importanti vantaggi applicativi rispetto alla trasformata di Fourier classica [6]. Sono stati implementati tre algoritmi di Voice Activity Detection, nei quali viene utilizzato l'approccio basato sulle Wavelets e il Teager Energy Operator [9]:

- Un algoritmo basato sulla *Wavelet Packet Decomposition* e sulla *Voice Activity Shape* [7]
- Un algoritmo basato sull'*entropia spettrale* [11]
- Un algoritmo basato sulla *trasformata Wavelet discreta* e sulla funzione *SAE* [14]

Essi sono stati confrontati con lo standard di mercato $G.729B$, sviluppato dalla *ITU-T* [10]. Allo scopo di test, volendo misurarsi con situazioni reali, si è utilizzato un database audio prodotto dal NIST, un'agenzia del governo degli Stati Uniti d'America che si occupa di sviluppo, promozione e misurazione di standard e tecnologie. Questo ente indice periodicamente dei concorsi relativi alle tecnologie applicate al linguaggio umano. La scelta è ricaduta sul database NIST-RT03, utilizzato per l'omonimo concorso, dato il suo largo impiego in molti lavori pubblicati e la presenza delle trascrizioni delle tempistiche inerenti alle regioni di parlato e non parlato.
Questo lavoro di tesi è organizzato come segue: nel capitolo 2 verranno descritti i processi di Speech Recognition e Voice Activity Detection ed i relativi campi di applicazione; nel capitolo 3 verrà discusso lo standard di mercato $G.729$ per la compressione e codifica del parlato, in modo particolare l'Annex B per il Voice Activity Detection; nel capitolo 4 verrà fatta una trattazione teorico-matematica delle Wavelets e dell'operatore Teager; nel capitolo 5 verranno descritti degli algoritmi di Voice Activity Detection al fine di confronto con l'algoritmo $G.729B$; nel capitolo 6 verranno presentati i test eseguiti su macchina; infine nel capitolo 7 verranno tratte alcune conclusioni sul lavoro di tesi e sui test effettuati.

Capitolo 2

Speech Recognition e VAD

2.1 Speech Recognition

Il termine *speech recognition*(anche conosciuto come riconoscimento automatico del parlato) si riferisce alla conversione di parole in testo. Il termine *voice recognition* viene talvolta utilizzato per riferirsi allo speech recognition quando il sistema di riconoscimento viene impostato per un parlatore specifico, come nel caso della maggior parte dei software di riconoscimento *desktop*, dove c'è un elemento di riconoscimento del parlato che cerca di identificare la persona che parla, per meglio riconoscere cosa sta dicendo. Esso è un termine generico, nel senso che si può riconoscere il parlato di quasi qualsiasi persona, ad esempio come un sistema di call center progettato per riconoscere molteplici voci. Il sistema di voice recognition è un sistema impostato per un utente particolare, che riconosce il parlato basandosi sul suono vocale univoco.

Lo speech recognition trova applicazione nel voice dialing, nel routing delle chiamate, nel domotic appliance control e nella ricerca dell'audio parlato basata sul contenuto; inoltre si pu applicare pure al semplice data entry, alla preparazione di documenti strutturati, al processamento parlato-testo e al direct voice input negli abitacoli degli aeroplani.

2.1.1 Storia

Il primo riconoscitore di parlato fece la sua apparsa nel 1952 e consisteva in un dispositivo per il riconoscimento delle singole cifre parlate. Un altro dispositivo remoto fu il Shoebox della IBM, esposto nel 1964 alla Fiera Mondiale di New York.

Una della applicazioni commerciali del riconoscimento del parlato, negli Stati Uniti, fu il controllo sanitario e in particolare il lavoro dei trascrizionisti medici. In accordo con le competenze delle industrie, al loro albore, si vendevano applicazioni di riconoscimento del parlato allo scopo di eliminare completamente le trascrizioni anziché rendere il processo di trascrizione delle

stesse più efficiente, per cui non vennero accettate. Tuttavia, al tempo il riconoscimento del parlato era tecnicamente arretrato. Inoltre, per poter essere usato in maniera efficiente, richiedeva cambiamenti radicali nel modo di lavorare dei medici, molti dei quali erano restii ad effettuare degli incontri clinici documentati. La più grande limitazione della trascrizione automatica nello speech recognition viene considerata nel software e nel grande spreco di tempo richiesto dall'utente o dal fornitore del sistema per 'addestrare' il software.

Spesso, nello speech recognition automatico, si fa distinzione tra sistemi *artificial syntax*,i quali sono specifici di un certo dominio, e il *processamento del linguaggio naturale*, il quale è specifico di un certo linguaggio. Questi tipi di applicazione si distinguono per i loro obiettivi e contesti.

2.1.2 Applicazioni

Controllo sanitario

Nel controllo sanitario, anche alla nascita di tecnologie di SR più avanzate, le trascrizioni mediche non sono ancora diventate obsolete. Molti esperti nel campo sostengono che con l'uso più ampio di tecnologie di SR, i servizi forniti possono essere redistribuiti anzich''e sostituiti. Lo SR viene utilizzato per aiutare le persone sorde a capire il parlato attraverso una conversione da parlato a testo, che è molto utile. Lo SR può essere implementato in front-end o back-end al processo di documentazione medica. Il SR front-end si ha quando il fornitore detta in una macchina apposita, sulla quale vengono visualizzate le parole dopo essere state riconosciute, e il dettatore è responsabile della modifica e della firma del documento. Non si fa mai utilizzo di un editor di trascrizioni mediche. Il SR back-end o SR Deferred si ha quando il fornitore detta in un sistema di dettatura digitale, e la voce viene data in in put ad una macchina di SR e l'abbozzo di documento riconosciuto viene processato insieme al file voce dall'editor di trascrizioni mediche, che modifica l'abbozzo e finalizza il report. Il SR Deferred viene molto usato nell'industria attualmente. Molte applicazioni di EMR (Electronic Medical Records) potrebbero essere più efficaci ed eseguite più facilmente quando sviluppate insieme ad un motore di SR. Le ricerche, le query e il riempimento di form sarebbero molto più veloci da eseguire via voce che via tastiera.

Settore militare: Aircraft da combattimento

Nell'ultima decade ci sono stati sostanziali miglioramenti ai test e alla valutazione dello speech recognition negli aircraft da combattimento. Particolare nota va al programma di speech recognition sviluppati dagli Stati Uniti per la tecnologia *Advanced Fighter Technology Integration* (AFTI)/F-16 VISTA, il programma in Francia per installare sistemi di SR negli aircraft Mirage,

e i programmi del Regno Unito relativi alla varietà di piattaforme per aircraft. In questi programmi, i riconoscitori del parlato hanno agito in maniera corretta negli aircraft da combattimento con applicazioni riguardanti: impostazione delle frequenze radio, comando del sistema di pilota automatico, impostazione delle coordinate e parametri per il rilascio delle armi, e il controllo dei display di volo. Generalmente, sono stati usati dei vocabolari molto limitati e vincolati, e lo sforzo maggiore è stato compiuto nell'integrazione del riconoscitore del parlato col sistema avionico.

Furono tratte alcune conclusioni da questo lavoro:

- Lo SR ha del potenziale definito per ridurre il carico di lavoro del pilota, ma questo potenziale non fu realizzato in maniera consistente.
- L'ottenimento di una precisione molto alta nel riconoscimento (circa 95%) fu il fattore critico per realizzare dei sistemi di SR utili, in quanto con un fattore di riconoscimento basso, i piloti non userebbero il sistema stesso.
- Sarebbe utile utilizzare un vocabolario e una grammatica più naturali, nonché avere dei tempi di training più bassi, ma solo se venissero mantenuti degli alti rapporti di riconoscimento.

Le ricerche nei laboratori per un SR robusto in ambienti militari hanno prodotto dei risultati promettenti i quali, se fossero estendibili alla cabina di pilotaggio, aumenterebbero l'utilità dello SR in aerei ad alte prestazioni. Lavorando coi piloti svedesi della cabina del JAS-39 Gripen, Englund scoprì che il riconoscimento diminuisce all'aumentare dei carichi di lavoro. Concluse anche che l'adattamento migliorò parecchio i risultati in tutti i casi e dimostrò che interodurre dei modelli di respiro incrementò i punteggi di riconoscimento in maniera significativa. Contrariamente a quello che potrebbe essere previsto, non furono trovati effetti di inglese incompleto nei parlatori. Era evidente che il parlato spontaneo causò problemi al riconoscitore, come ci si aspettava. Ci si aspetta anche che un vocabolario ristretto, e soprattutto una sintassi corretta, portino ad un aumento sostanziale della precisione di riconoscimento. Il Typhoon Eurofighter, attualmente in servizio col RAF del Regno Unito adotta un sistema dipendente dal parlatore, ossia richiede che ogni pilota crei un template. Il sistema non viene usato per ogni compito critico di sicurezza o delle armi, come il rilascio dell'arma o l'abbassamento del carrello d'atterraggio, ma viene usato per un gran numero di altre funzioni della cabina di pilotaggio. I comandi vocali vengono confermati per via visuale e/o feedback auricolare. Il sistema è visto come una caratteristica principale di design nella riduzione del carico di lavoro del pilota, e permette inoltre al pilota di assegnarsi gli obittivi con due semplici comandi vocali o ad ognuno dei suoi gregari con soli cinque comandi.

Settore militare: Elicotteri

I problemi di ottenere un'alta precisione di riconoscimento in condizioni di stress e rumore persistono fortemente sia nell'ambiente dell'elicottero sia nell'ambiente di combattimento. Il problema del rumore acustico è attualmente più grave negli elicotteri, non solamente per gli alti livelli di rumore ma anche perché il pilota generalmente non indossa una maschera per il viso, che ridurrebbe il rumore acustico nel microfono. Sono stati sviluppati sostanziali test e programmi di valutazione nella decade passata nelle applicazioni di SR negli elicotteri, soprattutto da parte della U.S. Army Avionics Research and Development Activity (AVRADA) e del Royal Aerospace Establishment (RAE) nel Regno Unito. In Francia è stato incluso lo SR nell'elicottero Puma. Ci fu molto lavoro utile anche in Canada. I risultati furono incoraggianti, e le applicazioni vocali inclusero:

- il controllo delle radio per le comunicazioni
- l'impostazione dei sistemi di navigazione
- il controllo di un sistema automatizzato di handover

Come nelle applicazioni per il combattimento, l'aspetto su cui passar sopra per la voce negli elicotteri è l'impatto sull'efficacia del pilota. Ci furono risultati incoraggianti segnalati dai test dell'AVRADA, sebbene questi rappresentino solamente una dimostrazione di fattibilità in un ambiente di test. C'è ancora molto da fare sia nel SR e nella tecnologia di SR nel complesso, con l'obiettivo di raggiungere miglioramenti di prestazioni nelle impostazioni operative.

Settore militare: Gestione del combattimento

I centri di controllo della gestione del combattimento generalmente richiedono un accesso rapido e un controllo di database contenenti informazioni che cambiano rapidamente. I comandanti e gli operatori di sistema operano delle query su questi database, che devono essere il più possibile convenienti, in un ambiente nel quale la maggior parte delle informazioni è rappresentata in maniera visuale. L'interazione uomo-macchina via voce ha un potenziale molto utile per questi ambienti. Molti sforzi furono compiuti per interfacciare i riconoscitori di parole singole disponibili in commercio negli ambienti di gestione del combattimento. In uno studio di fattibilità, furono testate delle apparecchiature di SR in contemporanea con un sistema di informazione integrata via display per applicazioni di gestione del combattimento navale. Gli utenti furono molto ottimisti sul potenziale del sistema, sebbene le capacità fossero limitate. I programmi di comprensione del parlato, sponsorizzati dalla Defense Advanced Research Projects Agency (DARPA) negli Stati Uniti, si concentrarono sul problema dell'interfaccia del parlato naturale. Gli sforzi

nello SR si concentrarono su un database di SR continuo (CSR), parlato con ampio vocabolario progettato per rappresentare il compito di gestione delle risorse navali. Sono stati ottenuti dei miglioramenti allo stato dell'arte nel CSR, e le ricerche attualmente sono concentrate nell'integrazione dello SR e il processamento del linguaggio naturale per permettere un'interazione tra il linguaggio parlato e il sistema di gestione delle risorse navali.

Settore militare: Training degli ATC

Il training degli Air Traffic Controllers (ATC) militari (o civili) rapprensenta un'eccellente applicazione per i sistemi di SR. Molti sistemi di training degli ATC attualmente richiedono una persona che funga da pseudopilota, instaurando un dialogo vocale col sistema di training, che simula il dialogo che il controllore dovrebbe condurre coi piloti in una situazione reale degli ATC. Lo SR e le tecniche di sintesi offrono il potenziale per eliminare il bisogno di una persona che agisca da pseudopilota, riducendo così il personale per il training e il supporto. I compiti degli air controller sono caratterizzati da parlato altamente strutturato come output primario del controller, riducendo così la difficoltà del compito di SR. Lo U.S. Naval Training Equipment Center ha sponsorizzato lo sviluppo di prototipi di trainer ATC utilizzando lo SR. Generalmente, la precisione di riconoscimento cala lentamente al provvedere di una buona interazione tra il trainer e il sistema. Tuttavia, i prototipi di sistemi di training hanno dimostrato un potenziale significativo per l'interazione vocale in questi sistemi, e in altre applicazioni di training. La marina degli Stati Uniti ha investito in ricerche di larga scala sui sistemi di training ATC, nei quali un'unità commerciale di SR fu integrata con un sistema di training complesso che include dei display e la creazione di scenari. Sebbene il riconoscitore era vincolato nel vocabolario, uno degli obiettivi dei programmi di training era insegnare ai controller a parlare un linguaggio vincolato, utilizzando delle parole specifiche per il compito dell'ATC. Alcune ricerche in Francia si concentrarono sull'applicazione dello SR nei sistemi di training ATC, virate su aspetti riguardanti sia lo SR sia l'applicazione di vincoli grammaticali nel contesto dei compiti degli ATC. La marina degli Stati Uniti, lo USAF, USMC e FAA utilizzano attualmente dei simulatori ATC con SR acquistati da diversi venditori, come UFA, Inc e ASI. Questo programma utilizza lo SR e il parlato sintetico per permettere al trainer di controllare i veicoli aerei e di terra nella simulazione senza l'esigenza di pseudopiloti. Un altro approccio alla simulazione ATC con SR è stato creato da Supremis. Tale sistema non è vincolato da una grammatica rigida imposta dalle già citate limitazioni delle altre strategie di riconoscimento.

Telefonia ed altri contesti

Lo SR automatico è comunemente utilizzato nel campo della telefonia e nel campo del computer gaming, dove la simulazione è sempre più diffusa. Sebbene ci sia un alto livello di integrazione col processamento delle parole nell'ambito generale dell'utilizzo del personal computer, l'ASR nel campo della produzione di documenti non ha riscontrato gli aumenti di utilizzo previsti. Il miglioramento delle velocità dei processori mobili ha permesso la fattibilità dei sistemi operativi Symbian e degli Smartphone con Windows Mobile. Il parlato viene prevalentemente utilizzato come parte dell'interfaccia utente, per creare comandi vocali predefiniti o personalizzati. I venditori leader in questo campo sono: Microsoft, Nuance Communications, Vito Technology, Speereo Software e SVOX.

Aiuto alle persone disabili

Le persone disabili possono trarre beneficio dai programmi di SR, soprattutto per le persone con difficoltà ad utilizzare le mani, da leggere ferite a disabilità complicate che precludono l'utilizzo dei dispositivi di input. Infatti, le persone che utilizzarono molto la tastiera e svilupparono RSI diventarono rapidamente dei clienti bisognosi di tali applicazioni. Lo SR viene utilizzato per la telefonia adibita alle persone sorde, come la conversione di posta vocale in testo e servizi di relay. Le persone con problemi di apprendimento che hanno problemi nella comunicazione che va dal pensiero alla trascrizione di esso su carta possono beneficiare di tale software.

Altre applicazioni dello SR sono:

- Traduzione automatica
- Automotive speech recognition (es. Ford Sync)
- Telematica (es. sistemi di navigazione dei veicoli)
- Court reporting (Realtime Voice Writing)
- Hands-free computing: interfaccia utente automatizzata per il riconoscimento dei comandi vocali
- Automazione casalinga
- Interactive voice response
- Telefonia mobile (es. email mobile)
- Interazione multimodale
- Valutazione della pronuncia nelle applicazioni di apprendimento di una lingua facilitate da calcolatori

- Robotica
- Videogiochi
- Trascrizione parlato-testo

2.2 Voice Activity Detection

Il *Voice Activity Detection*, o VAD, anche conosciuto come Speech Activity Detection o Speech Detection, è una tecnica usata nel processamento del parlato nel quale viene rilevata la presenza o l'assenza del parlato umano. Le principali applicazioni del VAD vanno nella codifica del parlato e nello SR. Esso può facilitare il processamento del parlato, e può anche essere utilizzato per disattivare alcuni processi durante le sezioni di non parlato in una sessione audio: esso può evitare di effettuare inutili codifiche o trasmissioni di pacchetti audio di silenzio nelle applicazioni di VoIP, ottenendo così un guadagno in tempi di calcolo e larghezza di banda nella rete. Il VAD è una tecnologia importante per lo sviluppo di applicazioni basate sul parlato. Molti algoritmi sono stati sviluppati per fornire svariate feature e compromessi tra latenza, sensitività, precisione e costo computazionale. Alcuni algoritmi VAD forniscono anche un'ulteriore analisi, ad esempio il rilevamento del parlato voiced, unvoiced e sustained. Il VAD è solitamente indipendente dal linguaggio. Il VAD inizialmente è stato studiato per essere utilizzato in sistemi *time assignment speech interpolation* (TASI).

2.2.1 Descrizione dell'algoritmo

Il tipico algoritmo VAD funziona nella maniera seguente:

- Inizialmente c'è una fase di riduzione del rumore, ad esempio per sottrazione spettrale.
- Successivamente vengono calcolate alcune feature o quantità da una sezione del segnale d'ingresso.
- Viene applicata una regola di classificazione per valutare la sezione di parlato come speech o non speech. Spesso questa classificazione si basa su uno o più valori di soglia calcolati.

Potrebbe esserci qualche feedback in questo algoritmo, nel quale la decisione VAD viene utilizzata per migliorare la stima del rumore nella fase di riduzione del rumore, o per variare in maniera adattativa la/e soglia/e. Queste operazioni di feedback permettono di incrementare le prestazioni del VAD quando si ha a che fare con rumore non stazionario (ossia quando esso presenta molte variazioni). Alcuni metodi VAD pubblicati formulano la regola decisione su una base frame by frame utilizzando misure istantanee

sulla distanza tra parlato e rumore. Tali misure includono la pendenza spettrale, i coefficienti di correlazione, il rapporto logaritmico di somiglianza, e le misure di distanza cepstrali, cepstrali pesate e modificate. Indipendentemente dalla scelta degli algoritmi VAD, bisogna fare un compromesso tra l'avere la voce rilevata come rumore e il rumore rilevato come voce (ossia tra avere falsi positivi e falsi negativi). Un algoritmo di VAD operante in un telefono mobile deve essere in grado di rilevare il parlato in presenza di una vasta varietà di rumori acustici di sottofondo. In queste condizioni di difficile rilevamento è spesso preferibile che il VAD esegua un fail safe, ossia che indichi che c'è parlato quando la decisione è di dubbuio, in modo tale da ridurre le possibilità di perdere segmenti di parlato. La difficoltà più grande nel rilevamento del parlato in questa situazione sono i rapporti segnale/rumore (SNR) molto bassi con cui si ha a che fare. Potrebbe essere persino impossibile distinguere tra parlato e rumore utilizzando delle tecniche di semplice level detection quando alcune espressioni del parlato vengono coperte da rumore.

2.2.2 Applicazioni del VAD

IL VAD trova svariate applicazioni:

- Esso è parte integrante di diversi sistemi di speech communication come le conferenze audio, cancellazione degli echi, SR, codifica del parlato e telefonia hands free.
- Nel campo della sorveglianza automatica, il VAD permette di rilevare automaticamente gli istanti dove c'è parlato, in modo tale da identificare l'identità della voce del parlatore, se utilizzato insieme a metodi di speech recognition e speaker diarization.
- Nel campo delle applicazioni multimediali, il VAD permette di ottenere delle applicazioni voce e dati simultaneamente.
- Similarmente, nei sistemi Universal Mobile Telecommunications (UMTS), esso controlla e riduce il bitrate medio e permette un aumento della qualità della codifica del parlato.
- Nei sistemi radio cellulari (ad esempio GSM e CDMA) basati su una modalità di trasmissione discontinua (DTX), il VAD viene utilizzato essenzialmente per aumentare la capacità del sistema riducendo l'interferenza tra i canali e il consumo di potenza nei dispositivi digitali portatili.

Per una gran varietà di applicazioni come la radio digitale mobile, Digital Simultaneous Voice and Data (DSVD) e la memorizzazione del parlato, è desiderabile fornire una trasmissione discontinua dei parametri di

codifica del parlato. Alcuni vantaggi includono il basso consumo medio di potenza nei microtelefoni mobili, un alto bitrate medio per servizi simultanei come la trasmissione dei dati, o una più elevata capacità nei chip di memorizzazione. Tuttavia, il miglioramento dipende principalmente dalla percentuale di pause durante il parlato e l'affidabilità del VAD utilizzato per rilevare questi intervalli. Dall'altro lato, è vantaggioso avere una bassa percentuale di attività del parlato. Inoltre si dovrebbe minimizzare il "taglio" del parlato per mantenere una buona qualità. Questo è il problema cruciale per un algoritmo VAD in condizioni di rumore molto potente.

Utilizzo nel telemarketing

Un'applicazione controversa del VAD si ha in congiunzione coi dialer predittivi utilizzati nelle aziende di telemarketing. Per massimizzare la produttività degli operatori, tali aziende predispongono tali dialer per chiamare molteplici numeri in un numero maggiore rispetto al numero di operatori, sapendo che molte chiamate terminano con l'espressione "Squillo - Nessuna risposta" o con la segreteria telefonica. Quando una persona risponde, tipicamente essa risponde brevemente (con espressioni quali "Pronto","Buonasera", ecc.) e si ha la presenza di un breve periodo di silenzio. I messaggi di segreteria telefonica tipicamente contengono dai 3 ai 15 secondi di parlato continuo. Impostando i parametri VAD correttamente, i dialer possono determinare se una persona risponde o se c'è la segreteria telefonica, e se c'è una persona che risponde, viene trasferita la chiamata ad un operatore disponibile. Se invece rileva la segreteria, l'operatore riattacca. Spesso, il sistema rileva correttamente se una persona risponde alla chiamata e allo stesso tempo non è disponibile alcun operatore, in modo tale da evitare che la persona stia al telefono inutilmente.

2.2.3 Valutazione delle prestazioni

Per valutare le prestazioni del VAD, si confronta l'output utilizzando delle registrazioni di test con quelle di un VAD ideale - creato annotando a mano la presenza/assenza della voce nelle registrazioni. Le prestazioni del VAD vengono comunemente valutate sulla base dei seguenti parametri:

- FEC (Front End Clipping): taglio introdotto nel passaggio da rumore ad attività del parlato
- MSC (Mid Speech Clipping): taglio dovuto alla malclassificazione del parlato come rumore
- OVER: rumore interpretato come parlato dovuto ad una condizione VAD che resta attiva passando da parlato a rumore

- NDS (Noise Detected as Speech): rumore interpretato come parlato in un periodo di silenzio

Sebbene il metodo sopra descritto fornisca delle informazioni sugli obiettivi utili al riguardo delle prestazioni del VAD, è soltanto una misura approssimativa dell'effetto soggettivo. Ad esempio, gli effetti di taglio del segnale audio possono talvolta essere nascosti dalla presenza di rumore di background, dipendentemente dal modello scelto per la sintesi del comfort noise, in modo tale che alcuni tagli misurati con dei test oggettivi non siano udibili. È quindi importante effettuare dei test oggettivi sul VAD, il cui obiettivo principale è assicurare che il taglio percepito sia accettabile. Il tipo di test richiede un certo numero di ascoltatori per giudicare le registrazioni contenenti i risultati ottenuti dall'algoritmo di VAD testato. Gli ascoltatori devono dare un punteggio alle seguenti feature:

- Qualità
- Difficoltà di comprensione
- Udibilità del taglio

Questi punteggi, ottenuti ascoltando diverse sequenze di parlato, vengono utilizzati per calcolare dei risultati medi per ogni feature, ottenendo così una stima globale del comportamento del VAD. Per concludere, dove dei metodi oggettivi sono molto utili in una fase iniziale per valutare la qualità del VAD, i metodi soggettivi sono più significativi. Sebbene siano più costosi (in quanto richiedono la partecipazione di un certo numero di persone per qualche giorno), vengono generalmente utilizzati quando una proposta dev'essere standardizzata.

2.2.4 Implementazioni

Ci sono state svariate implementazioni di algoritmi di VAD:

- Un primo standard di VAD è stato sviluppato dalla British Telecom per l'utilizzo nel servizio di telefonia cellulare digitale Pan europeo nel 1991.
- Lo standard G.729 calcola le seguenti feature per la sua decisione VAD: frequenze delle linee spettrali, energia a banda piena, energia a basse frequenze (sotto 1 kHz), e lo zero-crossing rate. Esso applica una classificazione semplice utilizzando un limite fisso di decisione nello spazio definito da queste feature, e successivamente applica lo smoothing e la correzione adattativa per aumentare la precisione della stima.
- Lo standard GSM include due opzioni VAD sviluppata dalla ETSI. La prima opzione calcola il rapporto SNR in nove bande e applica

una soglia a tali valori. La seconda opzione calcola diversi parametri: potenza del canale, metriche vocali e potenza del rumore. Essa inoltre applica una soglia alle metriche vocali in base allo SNR stimato.

- Il formato Speex di compressione audio utilizza una procedura chiamata Improved Minima Controlled Recursive Averaging, che utilizza una rappresentazione smoothed della potenza spettrale e valuta i minimi nella funzione smoothed del periodogramma.

Capitolo 3

Lo standard G.729

Il G.729 è un algoritmo di compressione audio per la voce che comprime la voce digitale in pacchetti da 10 millisecondi di durata, e rappresenta attualmente lo standard del mercato. Esso viene descritto come una codifica del parlato a 8 kbit al secondo che utilizza una predizione lineare a struttura coniugata basata su codici algebrici (CS-ACELP). Dati i requisiti poco onerosi in termini di larghezza di banda, il G.729 è molto utilizzato nelle applicazioni Voice over Internet Protocol (VoIP) (ad es. Skype) dove ci dev'essere una preservazione della banda. Lo standard opera ad un bitrate di 8 kbit al secondo, ma ci sono alcune estensioni che forniscono rate di 6.4 kbit al secondo (Annex D,F,H,I,C+) e 11.8 kbit al secondo (Annex E,G,H,I,C+) per una qualità del parlato leggermente peggiore o migliore rispettivamente. Il G.729 è stato esteso con varie feature, comunemente denominate come G.729a e G.729b. I toni *Dual Tone Multi Frequency* (DTMF), le trasmissioni via Fax e l'audio di alta qualità non possono essere trasportati in maniera affidabile utilizzando questo codec. Il DTMF richiede l'uso di un carico di lavoro RTP per le cifre DTMF, i toni telefonici e i segnali telefonici, come specificato nel RFC 2833.

3.1 G.729 Annex A

Il G.729a è un'estensione compatibile del G.729, ma richiede meno potenza computazionale. Questa complessità ridotta, tuttavia, bilancia il costo di una qualità leggermente ridotta del parlato. Il G.729a è stato sviluppato da un consorzio di organizzazioni: France Telecom, Mitsubishi Electric Corporation, Nippon Telegraph and Telephone Corporation (NTT), e Université de Sherbrooke. Le caratteristiche del G.729a sono:

- Frequenza di campionamento 8 Khz/16 bit (80 campioni per frame di 10 millisecondi)
- Bitrate fisso (8 kbit al secondo per frame di 10 millisecondi)

- Dimensione dei frame fissa (10 byte per frame di 10 millisecondi)
- Il ritardo dell'algoritmo è di 15 millisecondi per frame, con un ritardo di look-ahead di 5 millisecondi
- È un codificatore del parlato ibrido che utilizza ACELP
- La complessità dell'algoritmo viene quantificata a 15, utilizzando una scala relativa dove il G.711 è dato a 1 e G.723.1 è dato a 25
- Il test PSQM in condizioni ideali fornisce un punteggio medio opinionale di 4.04 al G.729a, confrontato al punteggio di 4.45 per il G.711 (legge mu)
- Il test PSQM in condizioni di stress della rete fornisce un punteggio medio opinionale di 3.51 al G.729a, confrontato al punteggio di 4.13 per il G.711

3.2 G.729 Annex B

Il G.729 è stato esteso nel cosiddetto Annex B (G.729b) [10], il quale fornisce un metodo di compressione del silenzio, caratteristico del modulo di VAD utilizzato. Esso è utilizzato per rilevare l'attività della voce nel segnale. Esso include anche un modulo di trasmissione discontinua (DTX) che opera delle decisioni sull'aggiornamento dei parametri del rumore di background per i frame rumorosi, ossia di non parlato. Viene utilizzato un frame di 2 byte chiamato Silence Insertion Descriptor (SID), trasmesso per permettere una generazione di comfort noise iniziale (CNG). Gli algoritmi di VAD,DTX e CNG vengono utilizzati per ridurre il rate di trasmissione dei dati durante i periodi silenziosi dei segnali parlati. Essi sono progettati e ottimizzati per operare congiuntamente alla *Recommendation V.70*. Quest'ultima si affida nell'utilizzo dei metodi di codifica del parlato del G.729 Annex A. Tuttavia, quando lo si richiede, la versione completa del G.729 può essere utilizzata anche per aumentare la qualità del parlato. Gli algoritmi vengono adattati per operare con entrambe le versioni complete della Recommendation G.729 e l'Annex A del G.729. Nella figura 3.1 viene mostrato lo schema a blocchi del sistema di comunicazione adottante la compressione del silenzio.

3.2.1 Descrizione dell'algoritmo di VAD

Nella figura 3.2 è rappresentato il flowchart dell'algoritmo di VAD utilizzato. Esso agisce su dei frame di parlato digitalizzato. I frame vengono processati in ordine temporale e vengono numerati in maniera consecutiva dall'inizio di ogni conversazione o registrazione. Inizialmente, vengono estratte quattro feature parametriche dal segnale d'ingresso. L'estrazione dei parametri viene

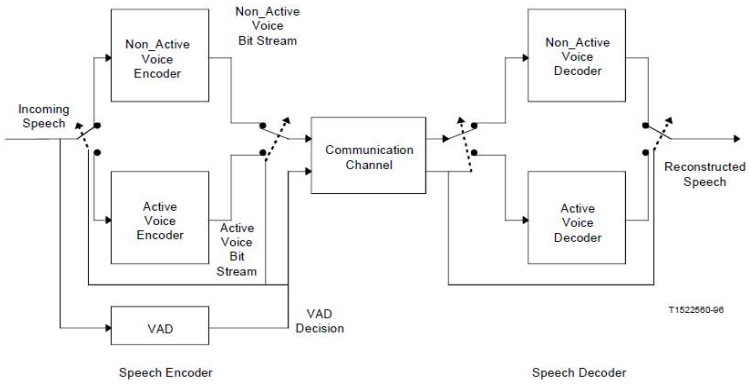

Figura 3.1: Schema di compressione del silenzio nel G.729 Annex B

condivisa coi muduli di codifica per la voce attiva e non attiva, per ragioni di efficienza di calcolo. I parametri sono le energie nei frame a banda piena e a banda bassa, l'insieme delle *Line Spectral Frequencies (LSF)* e lo *Zero Crossing Rate* dei frame. Se il numero del frame è minore di un certo N_i, si ha una fase di inizializzazione delle medie a lungo termine, e la decisione di VAD viene forzata ad assumere il valore 1 se l'energia del frame dall'analisi LPC risulta sopra i 15 dB (si veda l'equazione 3.1). Altrimenti, la decisione VAD viene forzata ad assumere il valore 0. Se il numero del frame è uguale a N_i, si ha una fase di inizializzazione per le energie caratteristiche del rumore di background. Nella fase successiva, viene calcolato un insieme di *parametri di differenza*. Tale insieme viene generato come una misura di differenza tra i parametri del frame corrente e le medie mobili delle caratteristiche del rumore di background. Vengono calcolate quattro misure di differenza:

- Distorsione spettrale
- Differenza di energia
- Differenza di energia a banda bassa
- Differenza Zero-crossing

La decisione VAD iniziale viene eseguita al passo successivo, utilizzando delle regioni di decisione basate su più limiti nello spazio delle quattro misure di differenza sopra citate. La decisione per la voce attiva viene data dall'unione delle regioni di decisione, mentre la decisione per la voce non attiva è data dal logico complementare della prima. Successivamente si utilizzano alcune considerazioni sull'energia e delle decisioni sui frame vicini precedenti per effettuare un raffinamento della decisione. Le medie mobili vengono aggiornate solo in presenza di rumore di background e non in presenza di parlato. Viene testata una soglia adattativa, e l'aggiornamento si ha quando il criterio di sogliatura viene rispettato.

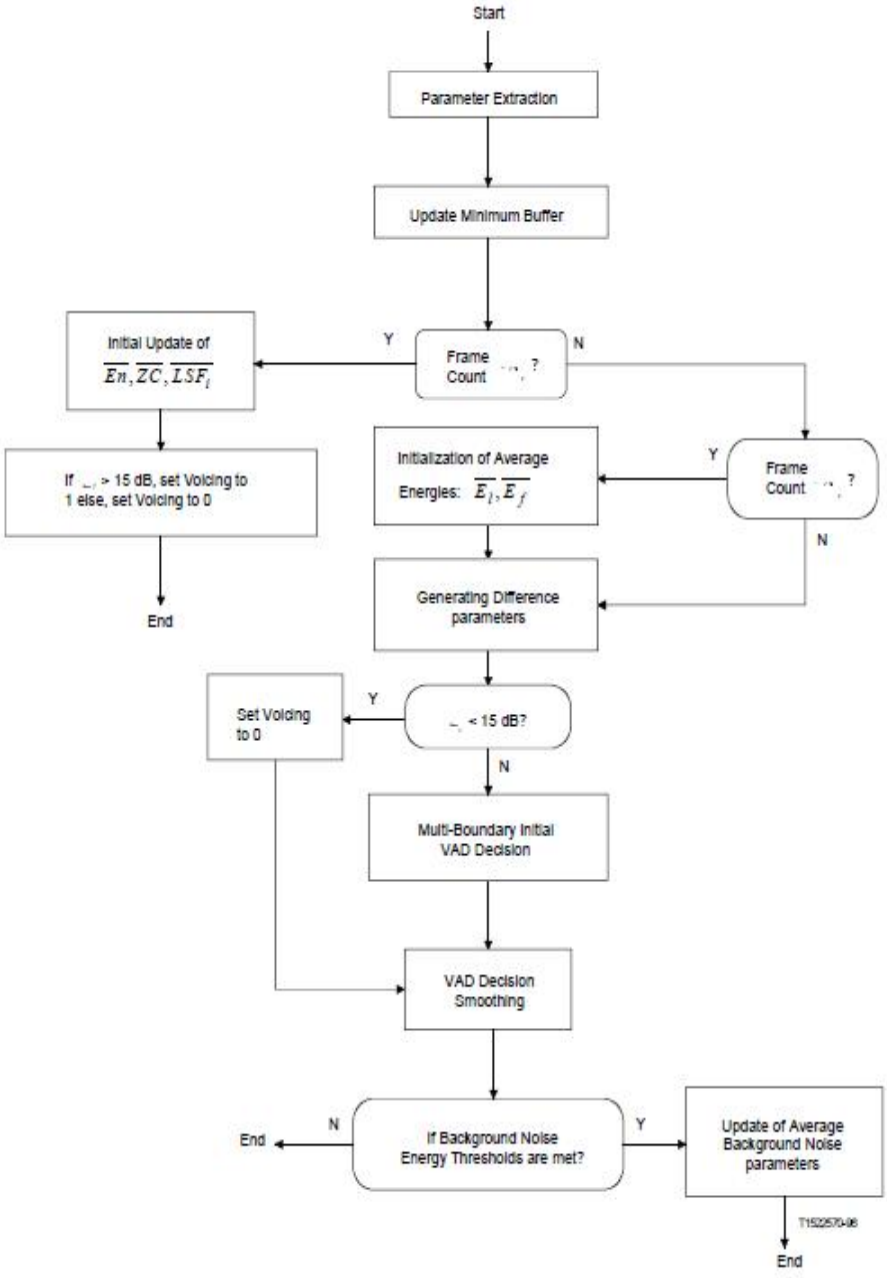

Figura 3.2: Flowchart dell'algoritmo di VAD utilizzato nel G.729 Annex B

Estrazione dei parametri

Per ogni frame viene estratto un insieme di parametri dal segnale parlato. Il modulo di estrazione parametri può essere condiviso tra il VAD, il codificatore di voce attiva e il codificatore di voce non attiva. L'insieme basilare dei parametri è l'insieme dei coefficienti di autocorrelazione. Si denoti tale insieme con:

$$\{R(i)\}_{i=0}^{q} \ \ con\ q = 12$$

Tali parametri sono:

- *Line Spectral Frequencies*: si ottiene dall'autocorrelazione un insieme di coefficienti lineari di predizione, e successivamente si ottiene un insieme $\{LSF_i\}_{i=1}^{p}$, con $p = 10$, derivato da tali coefficienti.

- *Energia a banda piena*: l'energia a banda piena E_f è il logaritmo del primo coefficiente di autocorrelazione normalizzato $R(0)$:

$$E_f = 10 \log_{10}[\frac{1}{N} R(0)] \tag{3.1}$$

  dove $N = 240$ è la dimensione della finestra di analisi LPC sui campioni del parlato.

- *Energia a banda bassa*: L'energia a banda bassa E_l, misurata nell'intervallo che va da 0 a F_l Hz, è così calcolata:

$$E_l = 10 \log_{10}[\frac{1}{N} \mathbf{h^T R h}] \tag{3.2}$$

  dove $\mathbf{h}$ è la risposta impulsiva di un filtro FIR con frequenza di taglio a F_l Hz, e $\mathbf{R}$ è la matrice di autocorrelazione di Toeplitz coi coefficienti di autocorrelazione su ogni diagonale.

- *Zero-crossing rate*: Lo zero-crossing rate normalizzato di ogni frame si calcola con la seguente relazione:

$$ZC = \frac{1}{2M} \sum_{i=0}^{M-1} [|sgn(x(i)) - sgn(x(i-1))|] \tag{3.3}$$

  dove $\{x(i)\}$ è il segnale d'ingresso pre-calcolato, e $M = 80$.

Inizializzazione delle medie mobili basate sul rumore

Per i primi N_i frame, i parametri spettrali del rumore di background, denotati con $\{\overline{LSF_i}\}_{i=1}^{p}$, vengono inizializzati come media dei parametri $\{LSF_i\}_{i=1}^{p}$ dei frame. La media degli attraversamenti sullo zero del rumore di background, denotata con $\overline{ZC}$, viene inizializzata come media del zero-crossing rate ZC del frame. Le medie mobili dell'energia del rumore di background,

denotate con $\overline{E_f}$, e l'energia a banda bassa del rumore di background, denotata con $\overline{E_l}$, vengono inizializzate come segue. Inizialmente, la procedura di inizializzazione utilizza il parametro $\overline{E_n}$, definita come la media dell'energia dei frame E_f sui primi N_i frame. Queste tre medie ($\overline{E_n}$,$\overline{ZC}$ e $\{\overline{LSF_i}\}_{i=1}^{p}$) includono solamente i drame che possiedono un'energia E superiore a 15 dB. Successivamente, la procedura continua come segue:

se $\overline{E_n} \leq T_1$ allora
  $\overline{E_f} = \overline{E_n} + K_0$
  $\overline{E_l} = \overline{E_n} + K_1$
altrimenti se $T_1 < \overline{E_n} < T_2$ allora
  $\overline{E_f} = \overline{E_n} + K_2$
  $\overline{E_l} = \overline{E_n} + K_3$
altrimenti
  $\overline{E_f} = \overline{E_n} + K_4$
  $\overline{E_l} = \overline{E_n} + K_5$

Si veda la figura 3.3 per i valori delle costanti.

Generazione dell'energia minima a lungo termine

Si calcola la *minima energia a lungo termine* E_{min} come il minimo di E_f su N_0 frame precedenti. Siccome N_0 è relativamente grande, si calcola E_{min} utilizzando i valori memorizzati del minimo di E_f su dei segmenti corti di parlato precedenti.

Generazione dei parametri di differenza

Vengono calcolate quattro misure di differenze dai parametri del frame corrente e le medie mobili del rumore di background.

Distorsione spettrale

La misura di *distorsione spettrale* viene generata come somma dei quadrati della differenza tra il vettore del frame corrente $\{LSF_i\}_{i=1}^{p}$ e le medie mobili del rumore di background $\{\overline{LSF_i}\}_{i=1}^{p}$:

$$\Delta S = \sum_{i=1}^{p} (LSF_i - \overline{LSF_i})^2 \tag{3.4}$$

Differenza di energia a banda piena

La misura di *differenza di energia a banda piena* viene generata come differenza tra l'energia del frame corrente, E_f, e la media mobile dell'energia del rumore di background, $\overline{E_f}$:

$$\Delta E_f = \overline{E_f} - E_f \tag{3.5}$$

Differenza di energia a banda bassa

La misura di *differenza di energia a banda bassa* viene generata come differenza tra l'energia a banda bassa del frame corrente, E_l, e la media mobile dell'energia a banda bassa del rumore di background, $\overline{E_l}$:

$$\Delta E_l = \overline{E_l} - E_l \tag{3.6}$$

Differenza zero-crossing

La misura di *differenza zero-crossing* viene generata come differenza tra lo zero-crosssing rate del frame corrente, ZC, e la media mobile dello zero-crossing rate del rumore di background, $\overline{ZC}$:

$$\Delta ZC = \overline{ZC} - ZC \tag{3.7}$$

Decisione iniziale multi-limite sulla voice activity

La decisione iniziale di voice activity si denota con I_{VD}, e viene impostata a 0 (falso) se il vettore dei parametri di differenza sta nella regione di voce non attiva. Altrimenti, la decisione iniziale viene impostata a 1 (vero). Le 14 decisioni sui limiti nello spazio quadridimensionale vengono cosí definite:

1. se $\Delta S > a_1 \Delta ZC + b_1$ allora $I_{VD} = 1$
2. se $\Delta S > a_2 \Delta ZC + b_2$ allora $I_{VD} = 1$
3. se $\Delta E_f < a_3 \Delta ZC + b_3$ allora $I_{VD} = 1$
4. se $\Delta E_f < a_4 \Delta ZC + b_4$ allora $I_{VD} = 1$
5. se $\Delta E_f < b_5$ allora $I_{VD} = 1$
6. se $\Delta E_f < a_6 \Delta S + b_6$ allora $I_{VD} = 1$
7. se $\Delta S > b_7$ allora $I_{VD} = 1$
8. se $\Delta E_l < a_8 \Delta ZC + b_8$ allora $I_{VD} = 1$
9. se $\Delta E_l < a_9 \Delta ZC + b_9$ allora $I_{VD} = 1$
10. se $\Delta E_l < b_{10}$ allora $I_{VD} = 1$
11. se $\Delta E_l < a_{11} \Delta S + b_{11}$ allora $I_{VD} = 1$
12. se $\Delta E_l > a_{12} \Delta E_f + b_{12}$ allora $I_{VD} = 1$
13. se $\Delta E_l < a_{13} \Delta E_f + b_{13}$ allora $I_{VD} = 1$
14. se $\Delta E_l < a_{14} \Delta E_f + b_{14}$ allora $I_{VD} = 1$

Se nessuna di queste condizioni viene rispettata, allora $I_{VD} = 0$. Si veda la figura 3.3 per i valori delle costanti.

Name	Constant	Name	Constant
N_i	32	N_1	4
N_0	128	N_2	10
K_0	0	T_1	671088640
K_1	-53687091	T_2	738197504
K_2	-67108864	T_3	26843546
K_3	-93952410	T_4	40265318
K_4	-134217728	T_5	40265318
K_5	-161061274	T_6	40265318
a_1	23488	b_1	28521
a_2	-30504	b_2	19446
a_3	-32768	b_3	-32768
a_4	26214	b_4	-19661
a_5	0	b_5	-30802
a_6	28160	b_6	-19661
a_7	0	b_7	30199
a_8	16384	b_8	-22938
a_9	-19065	b_9	-31576
a_{10}	0	b_{10}	-17367
a_{11}	22400	b_{11}	-27034
a_{12}	30427	b_{12}	29959
a_{13}	-24576	b_{13}	-29491
a_{14}	23406	b_{14}	-28087

Figura 3.3: Tabella delle costanti

Raffinamento della decisione di voice activity

La decisione iniziale di voice activity viene raffinata (hangover) per risaltare la natura di stazionarietà a lungo termine del segnale parlato. Il raffinamento viene eseguito attraverso una procedura a quattro passi:

1. Si definisce una variabile di flag che indica se c'è stato hangover, denominata v_{flag}. Viene sempre impostata a 0 prima che venga eseguito il raffinamento della decisione di voice activity. Si denotino la decisione raffinata della voice activity del frame attuale, del frame precedente e del frame prima del frame precedente come S^0_{VD},S^{-1}_{VD} e S^{-2}_{VD}, rispettivamente. S^{-1}_{VD} e S^{-2}_{VD} vengono inizializzati a 1. Inizialmente si inizializza $S^0_{VD} = I_{VD}$. Il primo passo di raffinamento consiste in tale istruzione:

   se $(I_{VD} = 0)$ e $(S^{-1}_{VD} = 1)$ e $(E > \overline{E_f} + T_3)$ allora $S^0_{VD} = 1$
   e $v_{flag} = 1$

2. Per il secondo passo di raffinamento si definisce un parametro booleano F^{-1}_{VD} e un contatore di raffinamento C_e. F^{-1}_{VD} viene inizializzato a 1 e C_e viene inizializzato a 0. Si denoti l'energia del frame precedente con E_{-1}. Il secondo passo di raffinamento consiste in tali istruzioni:

   se $(F^{-1}_{VD} = 1)$ e $(I_{VD} = 0)$ e $(S^{-1}_{VD} = 1)$ e $(S^{-2}_{VD} = 1)$ e
   $(|E_f - E_{-1}| \leq T_4)$ allora {
   $S^0_{VD} = 1$
   $v_{flag} = 1$
   $C_e = C_e + 1$
   se $(C_e \leq N_1)$ {
   $F^{-1}_{VD} = 1$
   }
   altrimenti {
   $F^{-1}_{VD} = 0$
   $C_e = 0$
   }
   }
   altrimenti $F^{-1}_{VD} = 1$

3. Per il terzo passo di raffinamento si definisce un contatore di continuità del rumore C_s, inizializzato a 0. Se $S^0_{VD} = 0$ si incrementa C_s. Il terzo passo di raffinamento consiste nelle seguenti istruzioni:

   se $(S^0_{VD} = 1)$ e $(C_s > N_2)$ e $(E_f - E_{-1} \leq T_5)$ {
   $S^0_{VD} = 0$
   $C_s = 0$
   }
   se $(S^0_{VD} = 1)$ allora $C_s = 0$

4. Nel quarto passo, si compie una decisione di voice activity se viene soddisfatta la seguente condizione:

   se $((E_f < \overline{E_f} + T_6)$ e $(frm_{count} > N_0)$ e $(v_{flag} = 0))$ allora
   $S_{VD}^0 = 0$

Aggiornamento delle medie mobili delle caratteristiche del rumore di background

Le medie mobili delle caratteristiche del rumore di background vengono aggiornate all'ultimo passo del modulo di VAD. A questo stadio, si verifica la seguente condizione e si esegue l'aggiornamento se essa è rispettata:

se $(E_f < \overline{E_f} + T_6)$ allora aggiorna

Le medie mobili vengono aggiornate utilizzando uno *schema autoregressivo (AR) di primo ordine*. Vengono utilizzati diversi coefficienti AR per parametri diversi, e si utilizzano insiemi diversi di coefficienti all'inizio della conversazione/registrazione o quando vengono rilevate molte variazioni nelle caratteristiche del rumore. Sia β_{E_f} il coefficiente AR per l'aggiornamento di $\overline{E_f}$, β_{E_l} il coefficiente AR per l'aggiornamento di $\overline{E_l}$, β_{ZC} il coefficiente AR per l'aggiornamento di $\overline{ZC}$ e β_{LSF} il coefficiente AR per l'aggiornamento di $\{LSF_i\}_{i=1}^p$. Il numero totale di frame per il quale la condizione di aggiornamento è soddisfatta si denota con C_n. In base al valore di tale contatore, si ottengono insiemi differenti di tali coefficienti AR. L'aggiornamento AR si effettua attraverso tali relazioni:

$$\overline{E_f} = \beta_{E_f}\overline{E_f} + (1 - \beta_{E_f})E_f \tag{3.8}$$

$$\overline{E_l} = \beta_{E_l}\overline{E_l} + (1 - \beta_{E_l})E_l \tag{3.9}$$

$$\overline{ZC} = \beta_{ZC}\overline{ZC} + (1 - \beta_{ZC})ZC \tag{3.10}$$

$$\overline{LSF}_i = \beta_{LSF}\overline{LSF}_i + (1 - \beta_{LSF})LSF_i \quad i = 1, \ldots, p \tag{3.11}$$

Vengono aggiornati anche i parametri $\overline{E}_f$ e C_n in base alle seguenti istruzioni:

se (frame count $> N_0$) e $(\overline{E}_f < E_{min})$ {
$\overline{E}_f = E_{min}$
$C_n = 0$
}

Capitolo 4

Wavelets e Teager Energy Operator

4.1 Le wavelets

Le *wavelets* sono delle funzioni matematiche utilizzate per decomporre una data funzione in diverse componenti in frequenza, per poi studiare ciascuna di tali componenti con una risoluzione pari alla sua scala. Le wavelets sono copie *traslate e scalate* (dette anche *wavelets figlie*) di una forma d'onda oscillante di *lunghezza finita* o che *decade velocemente* (*wavelet madre*). Si può rappresentare un segnale mediante le wavelets, attraverso le *serie di wavelets* e le *trasformate wavelet*. Tali trasformate hanno alcuni vantaggi rispetto alle tradizionali trasformate di Fourier, per rappresentare funzioni che presentano discontinuità o picchi. La parola *wavelet* fu coniata da Morlet e Grossmann nei primi anni '80. Prima si usava la parola francese *ondelette*, che significa *piccola onda*. Prima di procedere con l'analisi delle wavelets, verranno definite delle basi ortonormali di wavelets ([2],[4]). Infine si vedranno in dettaglio le trasformate wavelet con relative applicazioni.

4.2 Analisi multirisoluzione (MRA) e wavelets

L'*analisi multirisoluzione (MRA) o approssimazione multiscala (MSA)* ([5]) è un metodo di progetto introdotto nel 1988-89 da Stephane Mallat e Yves Mayer. Tale metodo ha dei predecessori nell'analisi microlocale nella teoria delle equazioni differenziali (*ironing method*) e i *metodi a piramide* per il processamento delle immagini, introdotti nel 1981-83 da Burt, Adelson e Crowley.

4.2.1 Definizione

Sia $\{V_j\}_{j\in\mathbb{Z}}$ un insieme di sottospazi di $\mathbb{L}^2(\mathbb{R})$ che soddisfano le seguenti proprietà:

$$V_j \subset V_{j+1} \tag{4.1}$$

$$\bigcap_{j\in\mathbb{Z}} V_j = \{0\} \tag{4.2}$$

$$chiusura(\bigcup_{j\in\mathbb{Z}} V_j) = \mathbb{L}^2(\mathbb{R}) \tag{4.3}$$

Ad ogni V_j venga associato l'operatore $P_{V_j} : \mathbb{L}^2(\mathbb{R}) \to V_j$, che mappa ciascuna funzione di $\mathbb{L}^2(\mathbb{R})$ nella sua proiezione ortogonale in V_j. Per un'arbitraria $f \in \mathbb{L}^2(\mathbb{R})$, la (4.3) assicura che $\lim_{j\to+\infty} P_{V_j} f = f$, e dunque $\{V_j\}$ gode di un'immediata rappresentazione in termini di *successione approssimante* di $\mathbb{L}^2(\mathbb{R})$. Siano verificati i requisiti:

$$f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1} \quad \forall j \tag{4.4}$$

$$f(t) \in V_0 \Rightarrow f(t-k) \in V_0 \quad \forall k \in \mathbb{Z} \tag{4.5}$$

$$\exists \phi(t) \in V_0 : \{\phi(t-k) : k \in \mathbb{Z}\} \text{ è una base ortonormale di } V_0 \tag{4.6}$$

La proprietà (4.4) dota la struttura che stiamo descrivendo della caratteristica di *multirisoluzione*, nel senso che gli spazi $\{V_j\}$ risultano scalati secondo potenze di 2. La (4.5), con l'ausilio di (4.4), si può generalizzare a:

$$f(t) \in V_j \Rightarrow f(t - 2^{-j}k) \in V_j \quad \forall k \in \mathbb{Z}$$

Ossia lo spazio V_0 (risp. V_j) è *invariante* rispetto a traslazioni intere (risp. multiple intere di 2^{-j}). Inoltre, la (4.6) richiede l'esistenza di una base ortonormale di V_0 costituita da traslazioni di una singola funzione $\phi(t)$, detta *funzione di scala* o *father wavelet*. Da (4.4) e (4.5), si ottiene dunque la seguente *base ortonormale per* V_j:

$$\left\{\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k) : k \in \mathbb{Z}\right\} \tag{4.7}$$

Una collezione $\{V_j, j \in \mathbb{Z}\}$ che soddisfi i requisiti (4.1)÷(4.6) verrà chiamata, unitamente alla funzione $\phi(t)$ associata, un'*analisi multirisoluzione* di $\mathbb{L}^2(\mathbb{R})$. Si osserva che $\phi(t)$ caratterizza esaurientemente V_0 e dunque, da (4.1)÷(4.5), l'intera MRA. Si può quindi pensare di definire $\phi(t)$, e da questa costruire V_0 come:

$$V_0 = span\,\{\phi_{0,k}(t)\}$$

e i vari V_j attraverso (4.4). Una scelta appropriata di $\phi(t)$ garantirà che tutti i requisiti siano soddisfatti. Fissato j, la proiezione $P_{V_j} f$, $f \in \mathbb{L}^2(\mathbb{R})$ può quindi essere rappresentata come:

$$P_{V_j} f(t) = \sum_k f_j(k)\phi_{j,k}(t)$$

con $f_j(k) = \langle f, \phi_{j,k} \rangle$. Inoltre, poiche $V_j \subset V_{j+1}$, ciascuna delle $\phi_{j,k}(t)$ può essere rappresentata nella base di V_{j+1}:

$$\phi_{j,k}(t) = \sum_l \langle \phi_{j,k}, \phi_{j+1,l} \rangle \phi_{j+1,l}(t)$$

con:

$$\begin{aligned} \langle \phi_{j,k}, \phi_{j+1,l} \rangle &= \int_{-\infty}^{+\infty} 2^{\frac{j}{2}} \phi(2^j \tau - k) 2^{\frac{j+1}{2}} \phi(2^{j+1}\tau - l) d\tau & (4.8) \\ &= \sqrt{2} \int_{-\infty}^{+\infty} \phi(\tau)\phi(2\tau + 2k - l) d\tau & (4.9) \\ &= h(-(2k-l)) & (4.10) \end{aligned}$$

avendo posto $h(n) = \langle \phi, \phi_{1,n} \rangle \ \forall n \in \mathbb{Z}$. Pertanto:

$$\begin{aligned} f_j(k) &= \langle f, \phi_{j,k} \rangle & (4.11) \\ &= \sum_l h(-(2k-l)) \langle f, \phi_{j+1,l} \rangle & (4.12) \\ &= \sum_l h(-(2k-l)) f_{j+1}(l) & (4.13) \end{aligned}$$

e si conclude che la successione $\{f_j(k)\}_{k \in \mathbb{Z}}$ è ottenibile per *campionamento* dalla convoluzione $h(-k) * f_{j+1}(k)$, cioè considerandone solamente i campioni di indice pari. È possibile caratterizzare le funzioni $\phi(t)$ in termini di $h(k)$ o, più precisamente, dalla sua trasformata di Fourier. In particolare, vale la relazione:

$$|H(\jmath\omega)|^2 + |H(\jmath(\omega + \pi))|^2 = 1 \tag{4.14}$$

dove $H(\jmath\omega)$ è la trasformata di Fourier di $h(k)$. Si consideri ora, $\forall j$, lo spazio:

$$O_j = V_j^{\perp} \cap V_{j+1} \tag{4.15}$$

Ossia, l'insieme delle funzioni di V_{j+1} che sono *ortogonali* a V_j. Vale innanzitutto:

$$V_{j+1} = V_j \oplus O_j \tag{4.16}$$

Si dimostra poi quanto segue. Definite la successione $\{g(n)\}_{n \in \mathbb{Z}}$ e la funzione $\psi(t)$, detta comunemente *mother wavelet*, attraverso le relazioni:

$$\begin{aligned} G(\jmath\omega) &= e^{-\jmath\omega} (H(\jmath(\omega + \pi)))^* & (4.17) \\ \Psi(\jmath\omega) &= G(\jmath\frac{\omega}{2}) \Phi(\jmath\frac{\omega}{2}) & (4.18) \end{aligned}$$

dove Φ è la trasformata di Fourier della father wavelet $\phi(t)$. Vale la seguente

Proposizione 4.2.1:

Data una MRA $\{V_j\}_{j\in\mathbb{Z}}$*, la mother wavelet* $\psi(t)$ *appartiene ad* O_0*, con* O_0 *definito dalla (5.15), ed è tale che:*

$$\{\psi(t-k):\ k\in\mathbb{Z}\} \tag{4.19}$$

costituiscono una base ortonormale di O_0*. Di conseguenza,* $\forall j$ *l'insieme:*

$$\left\{\psi_{j,k}(t)=2^{j/2}\psi(2^j t-k):\ j,k\in\mathbb{Z}\right\} \tag{4.20}$$

costituisce una base ortonormale per O_j*.*

Vale quindi:

$$P_{O_j}f=\sum_k f_j^{\perp}(k)\psi_{j,k}(t) \tag{4.21}$$

dove $f_j^{\perp}(k)=\langle f,\psi_{j,k}\rangle$. Giacché $O_j\subset V_{j+1}$, ciascuna $\psi_{j,k}(t)$ è esprimibile nella base di V_{j+1}. Da (4.13) ottengo:

$$f_j^{\perp}(k)=\sum_l\langle\psi_{j,k},\phi_{j+1,l}\rangle f_{j+1}(l)=\sum_l\overline{g}(-(2k-l))f_{j+1}(l) \tag{4.22}$$

con $\overline{g}(n)=\langle\psi,\phi_{1,n}\rangle,\forall n\in\mathbb{Z}$. Inoltre, $\overline{g}(n)$ coincide con $g(n)$ e dunque, dalla definizione di quest'ultima:

$$\overline{g}(n)=g(n)=(-1)^{1-n}h(1-n) \tag{4.23}$$

Le espressioni (4.13) e (4.22) permettono di ottenere sia $\{f_j(n)\}_{n\in\mathbb{Z}}$ sia $\left\{f_j^{\perp}(n)\right\}_{n\in\mathbb{Z}}$ da $\{f_{j+1}(n)\}_{n\in\mathbb{Z}}$ mediante *filtraggio e campionamento*. Si può anche verificare che tale procedimento è *reversibile*, cioe che $\{f_{j+1}(n)\}_{n\in\mathbb{Z}}$ è calcolabile attraverso *interpolazione e filtraggio* di $\{f_j(n)\}_{n\in\mathbb{Z}}$ e $\left\{f_j^{\perp}(n)\right\}_{n\in\mathbb{Z}}$. Si consideri ora la (5.16). Applicando iterativamente la relazione per una sequenza decrescente di indici $j=\tilde{J}...j_0+1,\tilde{J}\geq j_0+1$ arbitrari si trova:

$$V_{\tilde{J}}=V_{j_0}\oplus O_{j_0}\oplus...\oplus O_{\tilde{J}-1} \tag{4.24}$$

ed essendo (4.7) e (4.20) basi rispettivamente di V_j e O_j, una base per $V_{\tilde{J}}$ e:

$$\{\phi_{j_0,k}(t),\psi_{j_0,k}(t),\psi_{j_0+1,k},...,\forall k\} \tag{4.25}$$

Ricordando (4.1) e (4.3), per $\tilde{J}\to+\infty$ la (4.25) porge la *base ortonormale non omogenea di* $\mathbb{L}^2(\mathbb{R})$. Mentre, facendo tendere j_0 a $-\infty$, si ottiene come *base ortonormale omogenea di* $\mathbb{L}^2(\mathbb{R})$:

$$\{\psi_{j,k}(t),\forall j,k\} \tag{4.26}$$

L'esistenza delle basi ortonormali di $\mathbb{L}^2(\mathbb{R})$ è riassunta nella seguente:

Proposizione 4.2.2:

Sia $\phi(t)$ una father wavelet ammissibile, e sia $\psi(t)$ la corrispondente mother wavelet. Per j_0 arbitrario sono basi di $\mathbb{L}^2(\mathbb{R})$ le collezioni:

$$\{\phi_{j_0,k}(t), \psi_{j_0,k}(t), \psi_{j_0+1,k}, ..., \forall k\} \tag{4.27}$$

$$\{\psi_{j,k}(t), \forall j, k\} \tag{4.28}$$

dette rispettivamente base non omogenea ed omogenea. Ogni $f \in \mathbb{L}^2(\mathbb{R})$ ammette quindi espansioni del tipo:

$$f(t) = \sum_k f_{j_0}(k)\phi_{j_0,k}(t) + \sum_{j \geq j_0} \sum_k f_j^{\perp}(k)\psi_{j,k}(t)$$

$$f(t) = \sum_j \sum_k f_j^{\perp}(k)\psi_{j,k}(t)$$

a seconda della base utilizzata, con:

$$f_{j_0}(k) = \langle f, \phi_{j_0,k} \rangle$$

$$f_j^{\perp}(k) = \langle f, \psi_{j,k} \rangle$$

Tale decomposizione in wavelets di funzioni è detta wavelet decomposition (WD) e l'indice j è detto livello di dettaglio o valore di scala.

4.3 Wavelets a supporto compatto

Sia $f(t)$ una funzione della variabile t. Tale funzione si dice a *supporto compatto* se vale 0 al di fuori di un certo intervallo di interesse. Si consideri ora la funzione $\phi(t)$ di una MRA. Dall'espressione $g(n) = \langle \psi, \phi_{1,n} \rangle$ e sapendo che $\psi(t) \in V_1$, si deduce che:

$$\psi(t) = \sum_k \langle \psi, \phi_{1,k} \rangle \phi_{1,k}(t) \tag{4.29}$$

$$= \sum_k g(k)\phi_{1,k}(t) \tag{4.30}$$

$$= \sum_k (-1)^{1-k} h(1-k)\phi_{1-k}(t) \tag{4.31}$$

Se $\phi(t)$ avesse supporto compatto, la successione $h(n) = \langle \phi, \phi_{1,n} \rangle$ ammetterebbe *un numero finito di elementi non nulli*. Allora, per la (4.31), $\psi(t)$ sarebbe a sua volta a supporto compatto, e tali sarebbero $\psi_{j,k}(t)$ e $\phi_{j,k}(t)$ $\forall j, k$.
Si consideri il seguente esempio:
Sia la *wavelet di Haar* così definita:

$$\phi^H(t) = \begin{cases} 1 & 0 \leq t \leq 1 \\ 0 & altrimenti \end{cases} \tag{4.32}$$

Ovviamente $\phi^H(t)$ è a supporto compatto e si verifica che $\left\{\phi^H_{0,k}, \forall k\right\}$ è ortonormale. Posto:

$$\phi^H_{j,k}(t) = 2^{\frac{j}{2}}\phi^H(2^j t - k)$$

costruisco gli spazi V_j come:

$$V_j = span\left\{\phi^H_{j,k}(t), \forall k\right\}$$

Per costruzione, i requisiti della MRA sono soddisfatti. Per cui $\phi^H(t)$ è ammissibile. Infine, si osserva che alla $\phi^H(t)$ si accompagna la funzione:

$$\psi^H(t) = \begin{cases} 1 & 0 \leq t \leq \frac{1}{2} \\ -1 & \frac{1}{2} \leq t \leq 1 \\ 0 & altrimenti \end{cases} \tag{4.33}$$

La (4.32) ha il difetto di generare wavelets senza alcun grado di regolarità, per il fatto che la wavelet di Haar è *discontinua* e quindi *non differenziabile*. Per questo motivo, la ricerca di funzioni dalle proprietà più accattivanti ha portato all'individuazione di svariate wavelets a supporto compatto. Ad esempio, le mother wavelet di *Morlet*, di *Meyer* e la wavelet *a cappello messicano*, mostrate nelle seguenti figure.

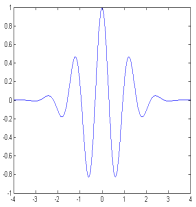

Figura 4.1: La mother wavelet di Morlet

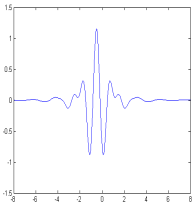

Figura 4.2: La mother wavelet di Meyer

Si abbia ora la seguente definizione:
Una funzione f(t) possiede N momenti nulli se:

$$\int_{-\infty}^{+\infty} \tau^l f(\tau) d\tau = 0, \quad l = 0, ..., N-1$$

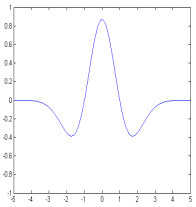

Figura 4.3: La mother wavelet a cappello messicano

È possibile costruire una MRA con $\phi(t)$ e $\psi(t)$ a supporto compatto, dove:

$$\phi(t), \psi(t) \in \mathbf{C}^m(\mathbb{R})$$

per ogni m naturale, ossia $\phi(t)$ e $\psi(t)$ sono *continue e differenziabili m volte.* L'enunciato che segue lega l'annullarsi dei momenti delle wavelets con la loro regolarità.

Proposizione:

Sia $\{\psi_{j,k}(t), \forall j,k\}$ *una base ortonormale di* $\mathbb{L}^2(\mathbb{R})$ *e sia, per qualche* $n \geq 0$*:*

$$|\psi(t)| \leq C(1+|t|)^{-(n+1+\epsilon)} \quad \epsilon > 0$$

$$\psi(t) \in \mathbf{C}^n(\mathbb{R})$$

con $\psi^{(l)}(t)$ *derivata l-esima limitata per ogni l=0,...,n. Allora:*

$$\int_{-\infty}^{+\infty} \tau^l \psi(\tau) d\tau = 0 \quad l = 0, ..., n \tag{4.34}$$

Le wavelets $\mathbf{C}^m(\mathbb{R})$ *a supporto compatto soddisfano tutte le ipotesi del teorema, e quindi presentano m+1 momenti nulli. In formule:*

$$\psi(t) \in \mathbf{C}^m(\mathbb{R}) \Rightarrow \int_{-\infty}^{+\infty} \tau^l \psi(\tau) d\tau = 0 \quad l = 0, ..., m \tag{4.35}$$

4.4 Le serie di wavelets e le trasformate wavelet

Una *serie di wavelets* [5] è una rappresentazione di una funzione (a valori reali o complessi) *quadrato-integrabile* rispetto ad una certa serie ortonormale generata da una wavelet.

4.4.1 Serie di wavelets

Si consideri la (4.28), cioè la *base ortonormale omogenea di* $\mathbb{L}^2(\mathbb{R})$, dove:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \quad j,k \in \mathbb{Z}$$

Tale base permette ad una qualsiasi funzione $f \in \mathbb{L}^2(\mathbb{R})$ di essere espansa come:

$$f(t) = \sum_{j,k=-\infty}^{+\infty} c_{jk}\psi_{jk}(t) \tag{4.36}$$

Tale rappresentazione di una funzione f si chiama *serie di wavelet*. I coefficienti c_{jk} sono chiamati *coefficienti wavelet*.

4.4.2 Trasformata wavelet

La *trasformata integrale wavelet* è una trasformata integrale definita come:

$$[W_\psi f](a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{+\infty} \overline{\psi(\frac{t-b}{a})} f(t)dt \tag{4.37}$$

Dove i *coefficienti wavelet* c_{jk} sono dati da:

$$c_{jk} = [W_\psi f](2^{-j}, k2^{-j})$$

Si ha che $a = 2^{-j}$ è il *fattore di scala* e $b = k2^{-j}$ è il *fattore di traslazione*. A differenza della trasformata di Fourier, che è una trasformata integrale in entrambe le direzioni, la serie wavelet è una trasformata integrale in una direzione, mentre è una serie nell'altra direzione, così come le serie di Fourier.

4.4.3 Trasformata wavelet continua (CWT)

Nella CWT(*continuous wavelet transform*), viene proiettato un dato segnale ad energia finita su una famiglia continua di bande di frequenza (o sottospazi di $\mathbb{L}^2(\mathbb{R})$), ad esempio su una banda di frequenze del tipo $[f, 2f]$ per tutte le frequenze positive $f > 0$. Per cui, il segnale originale può essere ricostruito tramite un'integrazione su tutte le componenti di frequenza risultanti.

Definizione matematica

La CWT di una funzione f è la trasformata wavelet definita dalla relazione:

$$\gamma(\tau, s) = \int_{-\infty}^{+\infty} f(t) \frac{1}{|s|} \overline{\psi(\frac{t-\tau}{s})} dt \tag{4.38}$$

dove τ è il fattore di traslazione, s il fattore di scala e ψ la mother wavelet. La funzione f può essere ricostruita con la *trasformata inversa*:

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \gamma(\tau, s) \frac{1}{\sqrt{|s|}} \psi(\frac{t-\tau}{s}) d\tau \frac{ds}{s^2} \tag{4.39}$$

dove:

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\zeta)|^2}{|\zeta|} d\zeta \tag{4.40}$$

si chiama *costante di ammissibilità* e $\hat{\psi}$ è la trasformata di Fourier di ψ. Per ottenere con successo la trasformata inversa, la costante di ammissibilità deve soddisfare la condizione:

$$0 < C_\psi < +\infty$$

Si può anche mostrare che $\hat{\psi}(0) = 0$, cioè che tale wavelet si può integrare in 0. La funzione ψ serve come prototipo per le wavelets figlie, che infatti sono delle copie traslate e scalate della mother wavelet:

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{|s|}} \psi(\frac{t-\tau}{s}) \tag{4.41}$$

Alcune applicazioni della CWT

- Determinazione della dimensione frattale:
  Guardando agli estremi della CWT, in base alle traslazioni, si può quantificare la dimensione frattale di una funzione.

- Analisi tempo-frequenza:
  Guardando agli estremi della CWT, in base al fattore di scala, si può decomporre un segnale in termini sia temporali sia di frequenza simultaneamente. Tale analisi tempo-frequenza basata sulla CWT ha alcuni vantaggi rispetto ad altri metodi di analisi tempo-frequenza. Inoltre ha molte applicazioni in materie quali la fisica, la chimica, la biologia, l'ingegneria e la finanza.

4.4.4 Trasformata wavelet discreta (DWT)

La *trasformata wavelet discreta (DWT)* è una trasformata wavelet in cui le wavelets sono *campionate e discretizzate*. La prima DWT fu inventata dal matematico ungherese Alfred Haar. Per una sequenza d'ingresso di 2^n numeri, la trasformata wavelet di Haar può essere considerata un campionamento dei valori d'ingresso, il quale memorizza la differenza e itera la somma. Questo processo viene ripetuto ricorsivamente, accoppiando le somme per ottenere la prossima scala. Infine si ottengono $2^n - 1$ differenze e una somma. Questa semplice DWT illustra le proprietà che si desiderano dalle wavelets in generale. Per prima cosa, può essere calcolata in un numero di operazioni pari a $O(n)$; inoltre, non analizza soltanto le informazioni in frequenza dell'ingresso esaminandole a diverse scale, ma anche le informazioni temporali, ad esempio quante volte si presentano le informazioni in frequenza. Queste proprietà combinate stanno alla base della *Fast Wavelet Transform (FWT)*,

un alternativa alla classica Fast Fourier Transform. L'insieme di DWT più utilizzato fu formulato dalla matematica belga Ingrid Daubechies nel 1988. Questa formulazione si basa sull'uso delle relazioni di ricorrenza per generare progressivamente campioni discreti dettagliati di una funzione mother wavelet implicita, in cui ogni risoluzione è il *doppio* della scala precedente. Nel suo trattato, Daubechies ricavò una famiglia di wavelets, la prima delle quali fu la *wavelet di Haar*. L'interesse in questo campo esplose da allora, e furono sviluppate molte alternative alle wavelets originarie di Daubechies.

Definizione

La DWT di un segnale viene calcolata facendolo passare attraverso una *serie di filtri*. Per prima cosa, i campioni $x(n)$ vengono passati ad un filtro *passa-basso* con risposta impulsiva $g(n)$, per cui ottengo come risultato la convoluzione:

$$y(n) = (x * g)(n) = \sum_{k=-\infty}^{+\infty} x(k)g(n-k)$$

Il segnale viene simultaneamente decomposto utilizzando un filtro *passa-alto* con risposta impulsiva $h(n)$. Le uscite danno rispettivamente i *coefficienti di dettaglio* (dal filtro passa-alto) e i *coefficienti di approssimazione* (dal filtro passa-basso). È importante che i due filtri siano connessi uno all'altro, e tali filtri prendono il nome di filtri *quadrature mirror*. Tuttavia, siccome metà delle frequenze del segnale sono state rimosse, metà dei campioni può essere scartata. Le uscite dei filtri poi vengono *sottocampionate* di un fattore 2:

$$y_{low}(n) = \sum_{k=-\infty}^{+\infty} x(k)g(2n-k) \tag{4.42}$$

$$y_{high}(n) = \sum_{k=-\infty}^{+\infty} x(k)h(2n-k) \tag{4.43}$$

Questa decomposizione ha dimezzato la risoluzione temporale dato che solo metà delle uscite dei filtri caratterizzano il segnale d'ingresso. Però, ogni uscita viene rappresentata solo su metà della banda di frequenza del segnale d'ingresso, per cui la risoluzione in frequenza viene raddoppiata. Definendo

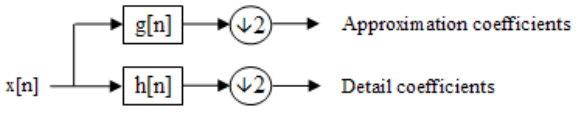

Figura 4.4: Diagramma a blocchi dell'analisi dei filtri

l'operatore di sottocampionamento $\downarrow$:

$$(y \downarrow k)(n) = y(kn) \tag{4.44}$$

Si possono scrivere in tale maniera le relazioni (5.42) e (5.43):

$$y_{low} = (x * g) \downarrow 2 \tag{4.45}$$
$$y_{high} = (x * h) \downarrow 2 \tag{4.46}$$

Tuttavia, calcolare la convoluzione $x * g$ con un sottocampionamento successivo sprecherebbe molto tempo. Esistono a tal proposito delle ottimizzazioni, ad esempio il *Lifting Scheme* [13], dove tali calcoli vengono eseguiti contemporaneamente.

Banchi di filtri in cascata

Questa decomposizione viene ripetuta per incrementare la risoluzione in frequenza e i coefficienti di approssimazione decomposti con filtri passa-alto e passa-basso e quindi sottocampionati. Questo procedimento si rappresenta con un albero binario che rappresenta un sottospazio con una localizzazione tempo-frequenza diversa. Tale albero si chiama *banco di filtri* [12]. Ad esempio si consideri un banco di filtri a 3 livelli: Ad ogni livello il segnale viene

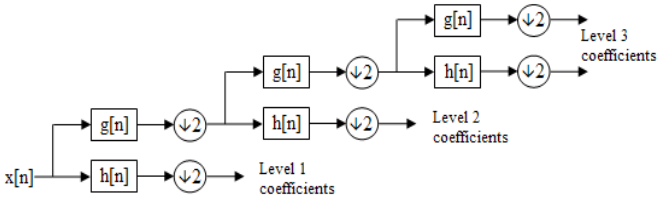

Figura 4.5: Un banco di filtri a 3 livelli

decomposto in basse e alte frequenze. Il segnale d'ingresso deve avere un numero di campioni multiplo di 2^n dove n è il numero di livelli. Ad esempio, con un segnale con 32 campioni, intervallo di frequenza $[0, f_n]$ e 3 livelli di decomposizione, vengono prodotte 4 uscite:

Livelli	*Frequenze*	*Campioni*
3	$[0, \frac{f_n}{8}]$	4
3	$[\frac{f_n}{8}, \frac{f_n}{4}]$	4
2	$[\frac{f_n}{4}, \frac{f_n}{2}]$	8
1	$[\frac{f_n}{2}, f_n]$	16

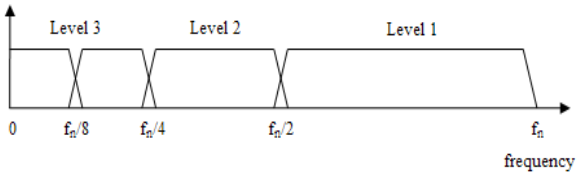

Figura 4.6: Rappresentazione nel dominio della frequenza della DWT

Un esempio in codice Java

Viene qui riportato un esempio in codice Java della DWT utilizzando la wavelet di Haar:

```
public static int[ ] invoke(int[ ] input){
  //Si assume che la lunghezza input.length sia una potenza di
   2
  int[ ] output = new int[input.length];
  for(int length = input.length >> 1; ; length >>= 1){
  //length=2^n, con n decrescente
    for(int i = 0; i < length; i++) {
      int sum = input[i*2]+input[i*2+1];
      int difference = input[i*2]-input[i*2+1];
      output[i] = sum;
      output[length+i] = difference;
    }
  if (length == 1)
    return output;
  //Scambia gli array per eseguire la prossima iterazione
  System.arraycopy(output, 0, input, 0, length<<1);
  }
}
```

Alcune applicazioni della DWT

La DWT ha un vasto numero di applicazioni nella scienza, nell'ingegneria e nella matematica. È anche utilizzata nella codifica dei segnali, per rappresentare un segnale discreto in una forma più ridondante, spesso come precondizione per la compressione dei dati. Ovviamente tali applicazioni sono strettamente legate alla disponibilità di *algoritmi efficienti* che implementino fedelmente la DWT. Tali algoritmi prendono il nome di *FWT (Fast Wavelet Transform).*

4.5 Le wavelet di Daubechies

Chiamate così in onore di *Ingrid Daubechies*, le wavelet di Daubechies [2] sono una famiglia di wavelet ortogonali che definiscono una trasformata wavelet discreta e sono caratterizzate da un numero massimale di momenti nulli per un certo supporto. Per ogni tipo di wavelet di questa classe, c'è una funzione di scala (*father wavelet*) che genera un'analisi multirisoluzione ortogonale.

4.5.1 Proprietà

In generale, le wavelet di Daubechies si scelgono in modo tale che abbiano il più alto numero A di momenti nulli (anche se ciò non implica la miglior smoothness), per un dato supporto di larghezza $N = 2A$, e tra le $2A - 1$ possibili soluzioni si sceglie quella il cui filtro di scala ha la fase massima. La trasformata wavelet è quindi facilmente implementabile attraverso l'algoritmo di FWT. Le wavelet di Daubechies sono molto utilizzate per risolvere un gran numero di problemi, ad esempio le proprietà di autosomiglianza di un segnale, problemi frattali, discontinuità di segnali, ecc. Le wavelet di Daubechies non sono definite in termini di funzioni di scala e di wavelet; infatti, non è possibile scriverle in forma chiusa. Tuttavia, è possibile utilizzare il *cascade algorithm* [3] per calcolare iterativamente una stima dei valori delle funzioni di scala e di wavelet semplicemente facendo la trasformata inversa un numero appropriato di volte. Normalmente vengono utilizzate delle wavelet ortogonali di Daubechies con un numero di punti che va da 2 a 20 (soltanto numeri pari vengono utilizzati),e si indicano come $D2, D4, ..., D20$. L'indice si riferisce al numero N di coefficienti. Ogni wavelet ha un numero di momenti nulli pari alla metà del numero di coefficienti. Ad esempio, la $D2$ (*wavelet di Haar*) ha un momento nullo, la $D4$ ne ha due, ecc. Un momento nullo limita il fatto che le wavelet rappresentano il comportamento polinomiale dell'informazione in un segnale. Ad esempio, la $D2$, con un momento nullo, codifica facilmente i polinomi ad un coefficiente o i componenti costanti un segnale. La $D4$ codifica i polinomi con due coefficienti, nonché i componenti costanti e lineari di un segnale; la $D6$ codifica i polinomi di terzo grado, nonché i componenti costanti, lineari e quadratici dei segnali. Questa abilità di codificare i segnali è comunque soggetta al fenomeno della dispersione di scala, e alla mancanza di invarianza dell'operazione di shifting, che emergono dall'operazione discreta di shifting durante l'applicazione della trasformata. Le sottosequenze che rappresentano i componenti lineari e quadratici vengono trattati in maniera differenta dalla trasformata, dipendentemente dall'allineamento dei punti con le locazioni pari o dispari nella sequenza. La mancanza della proprietà importante di invarianza dello shifting ha condotto allo sviluppo di diverse versioni dell'algoritmo di trasformata wavelet discreta shift-invariante.

4.5.2 Costruzione

Sia la sequenza per la scalatura (filtro passa-basso) sia la sequenza wavelet (filtro passa-banda) saranno qui normalizzate per avere somma pari a 2 e la somma dei quadrati pari a 2. In alcune applicazioni, esse vengono normalizzate per avere somma pari a $\sqrt{2}$, in modo tale che entrambe le sequenze e tutti gli shift di esse con un numero pari di coefficienti siano ortonormali l'una all'altra. Utilizzando la rappresentazione generale per una sequenza di scalatura di una trasformata discreta wavelet ortogonale con un ordine di approssimazione pari ad A si ottiene:

$$a(Z) = 2^{1-A}(1+Z)^A p(Z) \tag{4.47}$$

dove $N = 2A$, p è a coefficienti reali, $p(1) = 1$ e ha grado $grado(p) = A - 1$. Si può scrivere la condizione di ortogonalità come:

$$a(Z)a(Z^{-1}) + a(-Z)a(-Z^{-1}) = 4 \tag{4.48}$$

oppure come:

$$(2-X)^A P(X) + X^A P(2-X) = 2^A \tag{4.49}$$

dove il polinomio di Laurent $X := \frac{1}{2} \cdot (2 - Z - Z^{-1})$ genera tutte le sequenze simmetriche, con $X(-Z) = 2 - X(Z)$. Inoltre, $P(X)$ è il polinomio simmetrico di Laurent $P(X(Z)) = p(Z)p(Z^{-1})$. Date le relazioni:

$$X(e^{\jmath\omega}) = 1 - cos(\omega)$$

$$p(e^{\jmath\omega})p(e^{-\jmath\omega}) = |p(e^{\jmath\omega})|^2$$

il polinomio P assume valori non negativi nel segmento $[0, 2]$. L'equazione (4.49) ammette una soluzione minima $\forall\, A$, che può essere ottenuta per divisione e troncamento di serie di potenze in X:

$$P_A(X) = \sum_{k=0}^{A-1} \binom{A+k-1}{A-1} 2^{-k} X^k. \tag{4.50}$$

Ovviamente, essa assume valori positivi nel segmento $[0, 2]$. L'equazione omogenea per (4.49) è antisimmetrica su $X = 1$ e ammette la soluzione generale $X^A(X-1)R((X-1)^2)$, dove R è un polinomio a coefficienti reali. Per cui si ha la condizione:

$$P(X) = P_A(X) + X^A(X-1)R((X-1)^2) \;\geq 0 \;\; in\; [0,2] \tag{4.51}$$

si traduce in un insieme di restrizioni lineari sui coefficienti di R. I valori di P nell'intervallo $[0, 2]$ vengono limitati da una quantità pari a $4A - r$, dove massimizzando r si ottiene un sistema lineare con un numero infinito di disequazioni. Per risolvere $P(X(Z)) = p(Z)p(Z^{-1})$ in p si utilizza una

tecnica chiamata *fattorizzazione spettrale*, oppure *algoritmo di Fejer-Reisz*. Il polinomio $P(X)$ viene diviso in fattori lineari:

$$P(X) = (X - \mu_1) \dots (X - \mu_N) \tag{4.52}$$

con $N = A + 1 + 2grado(R)$. Ogni fattore lineare rappresenta un polinomio di Laurent:

$$(X(Z) - \mu) = -\frac{1}{2}Z + 1 - \mu - \frac{1}{2}Z^{-1} \tag{4.53}$$

che può essere fattorizzato in due fattori lineari. Si può assegnare uno qualsiasi dei due fattori lineari a $p(Z)$, ottenendo così $2N$ possibili soluzioni. Per ottenere la massima fase si sceglie quella che ha le radici complesse di $p(Z)$ dentro la circonferenza goniometrica di raggio unitario, ottenendo così una fase reale.

4.6 La Fast Wavelet Transform (FWT)

La *Fast Wavelet Transform* è un modo per processare i dati. In particolare, dati campionati da qualche sorgente. È piu veloce della FFT: infatti ha una complessità pari a $O(n)$, in confronto a quella della FFT $O(nlog_2 n)$. Inoltre, essa è più strettamente correlata con l'analisi dei segnali, perché l'idea dell'algoritmo deriva direttamente dalla teoria del filtraggio attraverso filtri passa-basso e passa-alto.

4.6.1 Descrizione

Le trasformate wavelet sono basate sull'applicazione di filtri passa-alto e passa-basso con delle speciali proprietà. In particolare, tali filtri devono essere dei *quadrature mirror filters:*
Una coppia di filtri QMF è un insieme di operatori lineari che soddisfano la condizione:

$$H^T H + L^T L = I \tag{4.54}$$

dove H e L sono le *funzioni di trasferimento* rispettivamente dei filtri passa-alto e passa-basso e I è la *matrice identità*. Siccome spesso viene persa dell'informazione nel processo di filtraggio, usare dei filtri QMF assicura che, dai risultati provenienti dai filtri passa-alto e passa-basso, possiamo ricostruire il segnale originale in modo esatto. Infatti, dati v vettore dei coefficienti del segnale originale, H, L, Hv *e* Lv, possiamo scrivere:

$$H^T Hv + L^T Lv = Iv = v \tag{4.55}$$

Un caso particolare di filtri QMF sono i *filtri di Haar:*
I filtri di Haar sono definiti da H e L, dati da:

$$L = \frac{1}{\sqrt{2}}[(1)(1)] \tag{4.56}$$

$$H = \frac{1}{\sqrt{2}}[(1)(-1)] \tag{4.57}$$

Il filtro L di Haar prende la *media* di due elementi contigui nel segnale, mentre il filtro H prende la *differenza* di tali elementi. Nella pratica i filtri di Haar non vengono quasi mai utilizzati, ma sono comodi come esempio, perché sono abbastanza semplici da analizzare.

4.6.2 Pseudocodice FWT

Viene qui presentato lo pseudocodice dell'algoritmo di FWT. Dato un segnale f definito nel dominio D_f, un filtro passa-alto H e un filtro passa-basso L:
FWT(f):

1. if length(f)==2 then
2. return [Hf,Lf]
3. else
4. Applica H a f e ritorna un nuovo segnale f_H con dominio di cardinalità dimezzata rispetto a D_f
5. Applica L a f e ritorna un altro nuovo segnale f_L con dominio di cardinalità dimezzata rispetto a D_f
6. return (f_H concatenato con FWT(f_L))

Utilizzando tale pseudocodice, si può implementare facilmente l'algoritmo di FWT. Da notare che se l'algoritmo deve mantenere una complessità lineare, la riduzione e il prolungamento utilizzati devono essere calcolati con una complessità lineare.

4.7 Confronto tra trasformata wavelet e trasformata di Fourier

Vengono qui confrontate la trasformata wavelet e la trasformata di Fourier [1].

4.7.1 Somiglianze

La FFT e la DWT sono entrambi operatori lineari che generano una struttura dati contenente $log_2 n$ segmenti di lunghezza variabile, che solitamente vengono riempiti e trasformati in un vettore di dati di lunghezza 2^n. Le proprietà matematiche delle matrici coinvolte nelle trasformate sono anch'esse simili. La matrice inversa di trasformazione, sia per la FFT sia per la DWT, è

la trasposta dell'originale. Quindi, entrambe le trasformate si possono vedere come una *rotazione nello spazio* delle funzioni su un dominio diverso. Per la FFT, questo nuovo dominio contiene delle funzioni base come seno e coseno. Per la DWT, questo nuovo dominio contiene delle funzioni base più complicate, cioè le *mother wavelet*. Entrambe le trasformate hanno un'altra somiglianza. Le funzioni base sono localizzate in frequenza, per cui si possono calcolare facilmente le *distribuzioni di potenza* attraverso strumenti matematici come lo *spettro di potenza* e gli *scaleogrammi*.

4.7.2 Differenze

La differenza più interessante fra queste trasformate è che le funzioni singole wavelet sono localizzate nello spazio, mentre le armoniche di Fourier non lo sono. Questa caratteristica di localizzazione, insieme alla localizzazione in frequenza delle wavelets, rende molte funzioni e operatori *sparsi* quando vengono trasformati nel dominio delle wavelet. Questa sparsità risulta utile in molte applicazioni come la compressione dei dati, il rilevamento di caratteristiche nelle immagini e la rimozione del rumore da sorgenti rumorose. Un modo di vedere le differenze in risoluzione tempo-frequenza tra la trasformata di Fourier e la trasformata wavelet è di guardare le *coperture* delle funzioni base nel piano tempo-frequenza. La figura 4.7 mostra una finestra della trasformata di Fourier, dove la finestra è semplicemente un'onda quadra. La finestra tronca la funzione seno (o coseno) per riempire una finestra di una larghezza particolare. Dato che viene usata una singola finestra per tutte le frequenze, la risoluzione dell'analisi è la stessa in tutte le locazioni del piano tempo-frequenza.

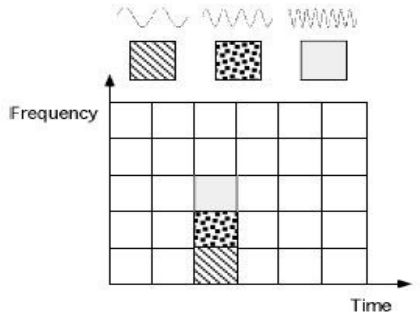

Figura 4.7: Funzioni base di Fourier, riempimenti tempo-frequenza e copertura del piano tempo-frequenza.

Un vantaggio delle trasformate wavelet è che la finestra varia. Per isolare le discontinuità del segnale, bisognerebbe avere delle funzioni base molto *brevi*. Allo stesso tempo, per ottenere un'analisi in frequenza dettagliata,

bisognerebbe avere delle funzioni base molto lunghe. Un modo per ottenere tale risultato è di avere delle funzioni base corte alle alte frequenze e lunghe alle basse frequenze. Questo espediente è esattamente quello che si ottiene con le trasformate wavelet. La figura 4.8 mostra la copertura del piano tempo-frequenza con la wavelet di Daubechies. Le trasformate wavelet hanno un insieme infinito di possibili funzioni base, al contrario delle trasformate di Fourier che hanno soltanto funzioni del tipo seno e coseno. Quindi da cui si può notare come le trasformate wavelet permettano di coprire efficientemente il piano tempo-frequenza, facendo variare la larghezza delle finestre, al contrario delle trasformate di Fourier che non permettono variazioni della larghezza delle finestre. Per cui l'analisi wavelet dà accesso immediato all'informazione, la quale può essere oscurata da altri metodi tempo-frequenza, come l'analisi di Fourier. Tuttavia, le funzioni base delle wavelet sono di più difficile trattazione matematica e di più difficile analisi rispetto alle funzioni base di Fourier, ossia semplici funzioni trigonometriche.

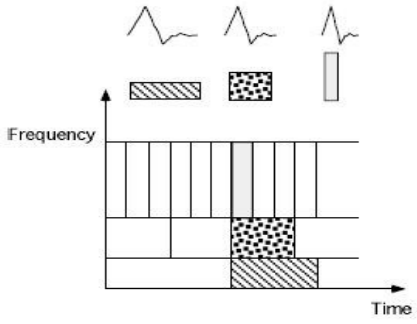

Figura 4.8: Wavelet di Daubechies,riempimenti tempo-frequenza e copertura del piano tempo-frequenza.

4.7.3 Un esempio grafico di confronto

Nella figura 4.9 viene rappresentata l'approssimazione di un segnale con la trasformata di Fourier utilizzando 80 coefficienti. Si nota come tale approssimazione non sia efficiente a causa delle discontinuità del segnale. Nella figura 4.10 viene rappresentata l'approssimazione dello stesso segnale con la trasformata wavelet, sempre utilizzando 80 coefficienti. In questo caso, l'approssimazione risulta più efficiente. Tuttavia, si possono notare delle *oscillazioni* vicino ai punti di picco in quanto non vengono utilizzati abbastanza coefficienti tali da permettere la ricostruzione perfetta del segnale.

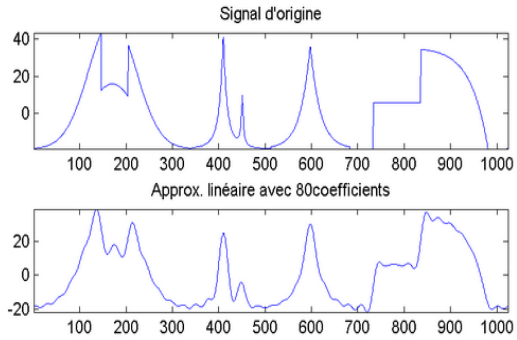

Figura 4.9: Approssimazione con la trasformata di Fourier utilizzando 80 coefficienti.

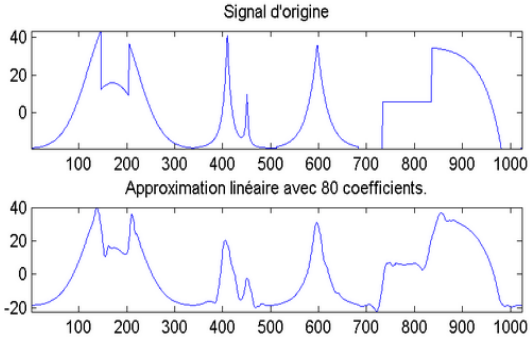

Figura 4.10: Approssimazione con la trasformata wavelet utilizzando 80 coefficienti.

4.7.4 Vantaggi applicativi

Come descritto da Kačur, Frank e Rozinaj [6], la trasformata di Fourier classica funziona bene quando si ha a che fare con *segnali stazionari in senso ampio*, ma non è molto utile coi segnali non stazionari, a causa della localizzazione dell'informazione nascosta. Questo problema fu corretto con la *Short Time Fourier Transform (STFT)*, la quale però dall'altro lato provoca una distorsione spettrale dovuta alle proprietà delle sue finestre temporali. Tuttavia, attraverso la STFT si ottiene soltanto una distribuzione uniforme tempo-frequenza e delle funzioni a base armonica. La trasformata Wavelet può eliminare questi difetti. I loro vantaggi, molto importanti ai fini di questo lavoro di tesi, sono:

- Esse forniscono una localizzazione esplicita degli eventi nel segnale, se le funzioni di scala e le funzioni wavelet hanno supporto compatto.

- Possono essere ottenute svariate distribuzioni tempo-frequenza (solitamente ad andamento logaritmico).
- Teoricamente, esiste un numero infinito di funzioni di scala e di funzioni wavelet.
- La DWT può essere facilmente implementata da banchi di filtri [12] o dal *Lifting Scheme* [13].

4.8 Applicazioni delle wavelets

Vengono qui presentate un paio di possibili applicazioni delle wavelets [1].

4.8.1 Computer e vista umana

Nei primi anni 80, David Marr cominciò a lavorare al Laboratorio di Intelligenza Artificiale del MIT sulla vista artificiale per i robot. Egli è un esperto nel sistema visivo umano e il suo obiettivo era di capire perché i primi tentativi di costruire un robot capace di comprendere l'ambiente che lo circonda non ebbero successo. Marr credeva che fosse importante fondare delle istituzioni scientifiche con lo scopo di eseguire ricerche sulla vista, e allo stesso tempo, si doveva limitare lo scopo della ricerca escludendo tutto quello che dipende dall'allenamento, cultura ecc. e concentrarsi soltanto sugli aspetti meccanici e involontari della vista. Questa vista a basso livello è la parte che ci permette di ricreare l'organizzazione tridimensionale del mondo fisico attorno a noi dalle eccitazioni che stimolano la retina. Marr pose queste questioni:

- Com'è possibile definire i contorni degli oggetti dalle variazioni dell'intensità della luce?
- Com'è possibile sentire la profondità?
- Com'è possibile percepire il movimento?

Egli sviluppò alcune soluzioni algoritmiche per rispondere a queste domande. La teoria di Marr si basa sul fatto che il processamento delle immagini nella vista umana ha una complicata struttura gerarchica che coinvolge diversi livelli di analisi. Ad ogni livello, il sistema retinale possiede una rappresentazione visuale che scala progressivamente in maniera geometrica. Il suo argomento dipendeva dalle *rilevazioni dei cambi di intensità*. Egli affermò che i cambi di intensità avvengono a scale diverse in un immagine, così la loro rilevazione richiede l'uso di operatori a dimensioni diverse. Inoltre disse che spesso i cambi di intensità producono dei picchi nella derivata prima dell'immagine. Queste due ipotesi richiedono che il filtro visivo abbia due caratteristiche: dev'essere un operatore differenziale e dev'essere capace di

essere regolato per operare su tutte le scale. L'operatore di Marr era una wavelet che oggi viene chiamata *wavelet di Marr*.

4.8.2 Compressione delle impronte digitali dell'FBI

Dal 1924 a oggi, l'ente FBI degli Stati Uniti ha memorizzato circa 30 milioni di impronte digitali. L'archivio consiste principalmente di impronte a inchiostro su fogli di carta. Le scansioni facsimile delle impronte vengono distribuite alle agenzie delle forze dell'ordine, ma la qualità della digitalizzazione è spesso bassa. Dato che un numero di giurisdizioni stanno sperimentando la memorizzazione digitale delle impronte, spesso si ha il problema di *incompatibilità tra i formati dei dati*. Questo problema portò la richiesta alla comunità per la giustizia criminale per la *digitalizzazione* e per uno standard di compressione.Nel 1993, la divisione Criminal Justice Information Services dell'FBI sviluppò degli standard per la digitalizzazione delle impronte e la compressione, in cooperazione con il National Institute of Standards and Technology, con il Los Alamos National Laboratory, commerciali e con le comunità per la giustizia criminale. Si ponga tale problema della memorizzazione dei dati in prospettiva. Le immagini delle impronte sono digitalizzate a una risoluzione di 500 pixel per pollice con 256 livelli di scala di grigio di informazione per pixel. Una singola impronta ha circa 700.000 pixel e richiede circa 0.6 Mbyte per essere memorizzata. Un paio di mani, quindi, richiede circa 6 Mbyte per la memorizzazione. Per cui, digitalizzare l'archivio attuale dell'FBI richiederebbe circa 200 Terabyte e richiederebbe un costo di circa 200 milioni di dollari. Ovviamente, una *compressione dei dati efficiente* è importante per ridurre tali numeri.

Figura 4.11: Un'impronta digitale del pollice sinistro. L'immagine a sinistra è originale, mentre quella a destra è la ricostruzione di una compressione 26:1

4.9 Il Teager Energy Operator

Il *Teager Energy Operator* (TEO) [9] è un operatore non-lineare molto potente, e permette di tener traccia dell'energia di modulazione, nonché identificare l'ampiezza e la frequenza istantanee. È stato sperimentalmente dimostrato che il TEO può aumentare la discriminabilità tra il parlato e il rumore e può sopprimere le componenti rumorose. Rispetto alla soppressione del rumore tradizionale basata sul dominio delle frequenze, la soppressione del rumore basata sul TEO è facilmente implementabile nel dominio del tempo. Nello spettrogramma della frase corrotta da rumore, in figura 4.12, è possibile vedere che il rumore corrotto viene quasi soppresso e la rappresentazione in frequenza dei formanti viene migliorata (caratteristica molto utile per il calcolo delle funzioni di BSE e VAS, presentate nel capitolo successivo).

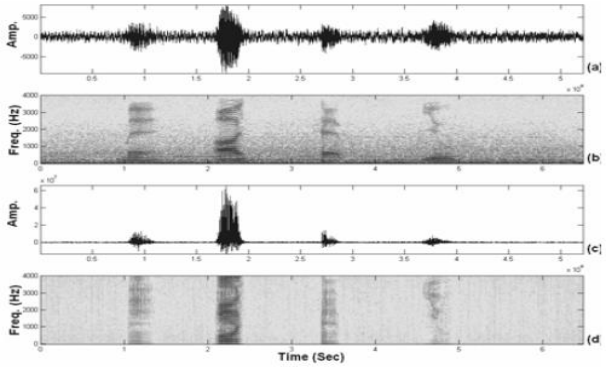

Figura 4.12: Illustrazione della soppressione del rumore tramite TEO in una frase corrotta da rumore: (a) Forma d'onda della frase rumorosa 'May I help you?' (b) Spettrogramma della frase (c) Forma d'onda del TEO (d) Spettrogramma del TEO

Nel tempo continuo, si può definire il TEO come:

$$\Psi_c[s(t)] = [\dot{s}(t)]^2 - s(t)\ddot{s}(t) \tag{4.58}$$

dove $s(t)$ è un segnale a tempo continuo e $\dot{s} = \frac{ds}{dt}$. Nel tempo discreto invece, è possibile approssimare l'operatore come:

$$\Psi_d[s(n)] = s(n)^2 - s(n-1)s(n+1) \tag{4.59}$$

dove $s(n)$ è un segnale a tempo discreto. Si consideri un segnale contenente parlato $s(n)$, corrotto da rumore additivo incorrelato $u(n)$. Il segnale risultante è quindi:

$$y(n) = s(n) + u(n) \tag{4.60}$$

Sia $\Psi_d[y(n)]$ il TEO del segnale rumoroso. Esso si definisce come:

$$\Psi_d[y(n)] = \Psi_d[s(n)] + \Psi_d[u(n)] + 2\widetilde{\Psi}_d[s(n), u(n)] \tag{4.61}$$

dove $\Psi_d[s(n)]$ e $\Psi_d[u(n)]$ sono il TEO del segnale parlato e del rumore additivo, rispettivamente. Sia $\widetilde{\Psi}_d[s(n), u(n)]$ l'energia Ψ_d-mutua tra $s(n)$ e $u(n)$, tale che:

$$\widetilde{\Psi}_d[s(n), u(n)] = s(n)u(n) - 0.5s(n-1)\cdot u(n+1) + 0.5s(n+1)\cdot u(n-1) \tag{4.62}$$

dove il simbolo $\cdot$ rappresenta l'operatore di *prodotto interno*. Dato che i segnali $s(n)$ e $u(n)$ hanno media nulla e sono indipendenti, il valore atteso dell'energia mutua Ψ_d , $\Psi_d[s(n), u(n)]$, è pari a zero. Per cui, è possibile ricavare la seguente equazione:

$$E\{\Psi_d[y(n)]\} = E\{\Psi_d[s(n)]\} + E\{\Psi_d[u(n)]\} \tag{4.63}$$

Infatti, il TEO del parlato è significativamente maggiore rispetto al TEO del rumore. Per cui, confrontato con $E\{\Psi_d[y(n)]\}$, il valore atteso $E\{\Psi_d[u(n)]\}$ è quindi trascurabile. Si ottiene così la relazione:

$$E\{\Psi_d[y(n)]\} \approx E\{\Psi_d[s(n)]\} \tag{4.64}$$

Capitolo 5

Algoritmi di VAD implementati

In questo capitolo verranno descritte le implementazioni relative agli algoritmi di Voice Activity Detection sviluppati. Essi verranno poi confrontati con l'algoritmo G.729B, essendo lo standard di mercato, e verranno poi analizzati i risultati nel capitolo relativo ai test.

5.1 Algoritmo basato sulla Wavelet Packet Transform e Voice Activity Shape

Il primo algoritmo di VAD implementato è quello di Chiodi e Massicotte [7]. Esso è basato sulla WPT e può essere diviso in quattro fasi, come visualizzato in figura:

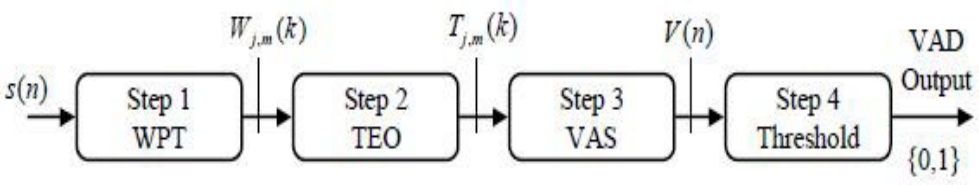

Figura 5.1: Schema dell'algoritmo di VAD basato su WPT

5.1.1 Fase 1: Decomposizione tramite Wavelet

Il segnale parlato $s(n)$ viene decomposto in frame di 256 campioni ciascuno. La scelta della dimensione dipende dalla scelta del range di frequenza che si vuole analizzare e dalla frequenza di campionamento. Se si vuole avere più informazione a basse frequenze il frame dev'essere più grande, mentre se si vuole analizzare il segnale ad alte frequenze il frame dev'essere più

piccolo. Per gli scopi di questa tesi, 256 è un buon valore, considerando che la frequenza di campionamento è di 8 KHz. Per decomporre il segnale, vengono usati dei banchi di filtri corrispondenti a tale mother wavelet, per ottenere i coefficienti di approssimazione e di dettaglio, dati dalle seguenti relazioni:

$$a(k) = \sum_{n=1}^{N} g(n-2k)s(n) \tag{5.1}$$

$$d(k) = \sum_{n=1}^{N} h(n-2k)s(n) \tag{5.2}$$

dove $g(n)$ e $h(n)$ stanno ad indicare i coefficienti del filtro passa basso e del filtro passa alto, rispettivamente. In questa implementazione dell'algoritmo, si utilizzano dei filtri con coefficienti dati dalle Wavelet di Daubechies, in quanto permettono di conservare la selettività delle frequenze all'aumentare del livello di decomposizione delle wavelet. Si implementa la DWT tramite cascata di questi filtri, e si ottiene l'albero di decomposizione corrispondente alla Wavelet Packet Transform. Questo approccio a filtri rende la DWT adattabile alle applicazioni real time. Utilizzando la WPT, ogni frame viene decomposto in S segnali di sottobanda (S=16). Nell'implementazione effettuata, si utilizza un albero di decomposizione bilanciato a 4 livelli, e la scelta della mother wavelet è ricaduta sulla *Wavelet di Daubechies a 10 punti*. Si ottengono 16 segnali di dimensione diversa (dipendentemente dal livello di decomposizione), denominati $W_{j,m}(k)$, dove j è il livello di decomposizione (nella scala delle frequenze), e $j = 1, 2, ..., 2^J$ ($J = 8$), m è l'indice del segnale di sottobanda ($1 \leq m \leq S$), e k è l'indice dei coefficienti $k = 1, 2, ..., 2^j$. Il livello di decomposizione j rappresenta i range di frequenza di interesse per rilevare i frame di parlato e non parlato.

5.1.2 Fase 2: applicazione del TEO

L'obiettivo dell'operatore TEO è quello di determinare la periodicità di ogni segnale di sottobanda. Esso viene calcolato per ogni sottobanda tramite l'equazione:

$$T_{j,m}(k) = \Psi[W_{j,m}(k)] \tag{5.3}$$

Tale operazione permette di rilevare la forma della periodicità e il decadimento dei momenti non transitori e dei segnali aperiodici, nonché permette di sopprimere il rumore.

5.1.3 Fase 3: Estrazione della Voice Activity Shape

Dopo l'applicazione del TEO, l'algoritmo calcolare la varianza di ogni segnale TEO, $T_{j,m}(k)$, e calcola la seguente sommatoria:

$$V(n) = \sum_{m=1}^{S} var(T_{j,m}(k)) \; k = 1, 2, ..., 2^j \tag{5.4}$$

dove $n = 1, 2, ..., N$ e $var(\bullet)$ è l'operatore di varianza. Ad ogni frame si assegna un valore $V(n)$. Il risultato che si ottiene corrisponde ad una curva di *Voice Activity Shape (VAS)* che caratterizza l'evoluzione del parlato e del non parlato nel segnale osservato $s(n)$. Il valore di $V(n)$ è elevato nei periodi di voce e basso nei periodi di non voce.

5.1.4 Fase 4: decisione basata su sogliatura

L'algoritmo di Chiodi e Massicotte si basa su una sogliatura fissa sui valori della VAS, calcolata tenendo conto dei primi 10 frames, valutati come rumorosi. Per ragioni di efficienza, nell'implementazione dell'algoritmo non viene utilizzata tale sogliatura fissa, ma una *sogliatura adattativa pesata (AWT)* descritta nell'algoritmo di Chen, Wu, Ruan e Truong [8]. Essa viene calcolata da un algoritmo iterativo, seguendo i passi qui sotto descritti:

1. Si ponga l'indice $k = 1$ e si definisca $V^{(1)}(n) = V(n)$, dove $V(n)$ è data dall'equazione (5.4)

2. Sia $V^{(k+1)}(n)$ definito dalla seguente relazione:

$$V^{(k+1)}(n) = \begin{cases} V^{(k)}(n) & se\ V^{(k)}(n) < E[V^{(k)}(n)] \\ E[V^{(k)}(n)] & altrimenti \end{cases} \tag{5.5}$$

   dove $E[V^{(k)}(n)]$ è la media di $V^{(k)}(n)$.

3. Si ripeta il passo 2, così da ottenere la misura denominata *Second Derivative Round Mean (SDRM)*, ossia $E[V^{(2)}(n)]$.

4. Si determini il *voiced rate* del segnale parlato:

$$p = \frac{Lv}{L} \tag{5.6}$$

   dove Lv è la lunghezza delle regioni di $V^{(2)}(n)$ in cui $V^{(2)}(n) = V^{(1)}(n)$ e L è la lunghezza del segnale d'ingresso.

5. Il valore di soglia della VAD sarà contenuto nel seguente intervallo:

$$\left[\frac{E[V^{(2)}(n)] + E[V^{(3)}(n)]}{2} \; , \; p\, \frac{E[V^{(2)}(n)] + E[V^{(3)}(n)]}{2} \right]$$

6. Infine, il valore di soglia adattativa di ogni frame può essere calcolato con l'ausilio della seguente equazione:

$$AWT(i) = \begin{cases} Max(Frame(i)) + 0.1 & se\ Max(Frame(i)) < Noise\_dis(n) \\ E[V^{(2)}(n)] + E[V^{(3)}(n)] & altrimenti \end{cases} \tag{5.7}$$

dove $AWT(i)$ è il valore di soglia adattativa di ogni frame i, mentre $Frame(i)$ e $Noise\_dis(n)$ sono definiti dalle seguenti relazioni:

$$Frame(i) = [V((i-1) * Num + 1), V(i * Num)] \tag{5.8}$$

$$Noise\_dis(n) = p\,\frac{E[V^{(2)}(n)] + E[V^{(3)}(n)]}{2} \tag{5.9}$$

dove $Num = 5$ nell'implementazione utilizzata.

5.2 Algoritmo basato sull'entropia spettrale

Il secondo algoritmo di VAD implementato è quello di Wang e Tasi [11]. Esso si basa sul calcolo dell'*entropia spettrale*, e fu originalmente implementato attraverso l'analisi di Fourier. In questo lavoro di tesi, si è scelto di implementare tale algoritmo sia attraverso l'analisi di Fourier sia attraverso l'analisi Wavelet. Prima di descrivere l'algoritmo, è giusto dare una definizione di *entropia* nel dominio delle energie spettrali:

$$H(l) = \sum_{\omega} P(|X(\omega, l)|^2) log[\frac{1}{P(|X(\omega, l)|^2)}] \tag{5.10}$$

dove $P(|X(\omega, l)|^2) = |X(\omega, l)|^2 / \sum_{\omega} |X(\omega, l)|^2$ è la probabilità del ω-esimo componente in frequenza nel magnitudo spettrale per il l-esimo frame.

5.2.1 Entropia spettrale e BSE

Per caratterizzare il segnale parlato utilizzando la misura di entropia sopra definita, si utilizza una decomposizione in bande per individuare i componenti in frequenza delle formanti. In tale algoritmo, l'analisi spettrale viene derivata dall'operatore TEO sul parlato rumoroso tramite la seguente relazione:

$$X_T(\omega, l) = |\sum_{n=0}^{N-1} H(n)T(n, l) \exp(-\jmath 2\pi\omega n/N)|\ con\ 0 \leq \omega \leq N-1 \tag{5.11}$$

dove $X_T(\omega, l)$ rappresenta il *magnitudo spettrale* dell'operatore TEO sul parlato in ingresso nella ω-esima componente in frequenza del l-esimo frame. N è il numero totale di componenti in frequenza in ogni frame ($N = 256$ in questo caso), mentre $H(n)$ è una *finestra di Hamming*. Successivamente,

ogni frame viene decomposto in 32 sottobande uniformi. L'energia spettrale del parlato nella m-esima banda del l-esimo frame si ottiene dalla seguente relazione:

$$E_S(m,l) = \sum_{\omega=(m-1)\frac{N/2}{Nb}}^{(m-1)\frac{N/2}{Nb}+(\frac{N/2}{Nb}-1)} X_T^2(\omega,l) \; con \; 1 \leq m \leq Nb \tag{5.12}$$

dove Nb è la dimensione totale della banda per ogni frame (Nb=32 nell'implementazione). Si può stimare la percentuale di energia spettrale in ogni banda attraverso l'equazione:

$$P_{E_S}(m,l) = \frac{E_S(m,l)}{\sum_{m=1}^{Nb} E_S(m,l)} \; con \; 1 \leq m \leq Nb \tag{5.13}$$

Tuttavia, l'entropia non può indicare in maniera efficiente una distribuzione (in termini spaziali) della sequenza di dati. Infatti, non viene sufficientemente discriminata la differenza tra parlato e non parlato, dato che l'ordine di sottobanda non viene considerato. Per ridurre tale problema, viene definito un insieme di *fattori di peso*, così definito:

$$W(m,l) = \frac{1}{3}\sum_{i=m-1}^{m+1}(e_i - \mu)^2 \tag{5.14}$$

$$\mu = \frac{1}{3}\sum_{i=m-1}^{m+1} e_i \;\; , \;\; e_{m-1} = \frac{min[P_{E_s}(l)]}{P_{E_s}(m-1,l)} \tag{5.15}$$

$$e_m = \frac{min[P_{E_s}(l)]}{P_{E_s}(m,l)} \;\; , \;\; e_{m+1} = \frac{min[P_{E_s}(l)]}{P_{E_s}(m+1,l)} \tag{5.16}$$

dove $W(m,l)$ rappresenta la varianza tra l'energia parziale delle tre bande, e $min[P_{E_s}(l)]$ è l'energia spettrale minima tra tutte le 32 bande per il frame l-esimo, e si può considerare come un fattore normalizzato. Da questa analisi risulta che, se $W(m,l)$ della banda m-esima è grande, questo implica che il formante potrebbe essere localizzato attorno a queste sottobande, ossia dalla $(m-1)$-esima alla $(m+1)$-esima. Sommando i pesi di ogni sottobanda, basandosi sulla variazione dell'energia tra le sottobande, la *misura BSE*, denominata $H(l)$, si può definire come:

$$H(l) = \log(\sum_{m=1}^{Nb} W(m,l) P_{E_s}(m,l) \log[\frac{1}{P_{E_s}(m,l)}]) \tag{5.17}$$

Nell'articolo di Wang e Tasi, viene utilizzata una soglia fissa per la misura BSE misurata sul rumore di background noto a priori. Detta T_h tale soglia, si definisce una *voiced sound flag* tramite la seguente relazione:

$$f_h(l) = \begin{cases} 1 & se \; H(l) > T_h \\ 0 & altrimenti \end{cases} \tag{5.18}$$

Nell'implementazione utilizzata, invece, si utilizza una sogliatura adattativa più efficiente, descritta nel paragrafo 5.3.6.

5.2.2 Rapporto tra le energie low-band e full-band

A differenza dei suoni vocali, i suoni non vocali non possiedono componenti nella frequenza dei formanti. Misurando il livello di energia, il souno non vocale è difficilmente discriminato dal rumore di background. La maggior parte di tali suoni, tuttavia, mostrano una concentrazione spettrale negli intervalli di alta frequenza, mentre il rumore di background mostra una distribuzione spettrale uniforme. È possibile distinguere tra rumore di background e rumore *speech-active* esaminando la distribuzione dell'energia lungo le frequenze. L'*energia low-band*, misurata nell'intervallo di frequenza sotto i 1000 Hz, viene così calcolata:

$$E_{low}(l) = 10\log(\sum_{\omega=0}^{1/\pi} |X_T(\omega, l)|^2) \tag{5.19}$$

dove X_T denota l'analisi spettrale basata sull'operatore TEO applicato al segnale d'ingresso. In maniera simile si misura l'*energia full-band*, nell'intervallo $(0 - 4000$ Hz), come:

$$E_{full}(l) = 10\log(\sum_{\omega=0}^{\pi} |X_T(\omega, l)|^2) \tag{5.20}$$

Per cui, il *rapporto tra le energie low-band e full-band*, denominato $R_{lf}(l)$, è dato dalla relazione:

$$R_{lf}(l) = \frac{E_{low}(l)}{E_{full}(l)} \tag{5.21}$$

L'articolo di Wang e Tasi imposta una flag su tale rapporto, denominata f_{lf}, basata sulla relazione:

$$f_{lf} = \begin{cases} 0 & se\ T_{lf1} \leq R_{lf}(l) \leq T_{lf2} \\ 1 & altrimenti \end{cases} \tag{5.22}$$

dove T_{lf1} e T_{lf2} sono soglie costanti per il parlato vocale e non vocale, rispettivamente. Nell'implementazione utilizzata, viene utilizzata la sogliatura adattativa descritta nel paragrafo (5.1.4), in quanto permette una discriminazione più efficiente del parlato.

5.2.3 Decisione VAD

La decisione di VAD è data dalla semplice relazione di *or* logico tra le decisioni di flag sopra descritte:

$$VAD(l) = f_h(l) \cup f_{lf}(l) \tag{5.23}$$

I valori risultanti, ossia 0 e 1, indicano rispettivamente i frame vocali e non vocali.

5.3 Algoritmo basato su trasformata Wavelet Discreta e TEO

Il terzo algoritmo implementato è quello di Wu e Wang [14]. Esso è basato sull'utilizzo della trasformata Wavelet discreta sul Teager Energy Operator. Di seguito vengono descritti i passi dell'algoritmo.

5.3.1 Trasformata Wavelet discreta

La trasformata Wavelet si basa sull'analisi del segnale in tempo ed in frequenza. Tale analisi adotta una tecnica di finestratura con delle regioni a dimensione variabile, e permette infatti l'utilizzo di intervalli lunghi di tempo dove si vuole ottenere delle informazioni precise a bassa frequenza, nonché l'utilizzo di regioni più corte dove si vuole avere informazioni ad alta frequenza. I segnali contenenti parlato contengono molte componenti transitorie e godono della proprietà di non stazionarietà. Quando si utilizza la proprietà di analisi multirisoluzione della trasformata Wavelet, è necessario avere una miglior risoluzione temporale nell'intervallo ad alta frequenza per rilevare le componenti transitorie che variano rapidamente,mentre è richiesta una miglior risoluzione in frequenza nell'intervallo a bassa frequenza per tener traccia in maniera precisa dei formanti che variano lentamente nel tempo. Attraverso l'analisi multirisoluzione, si può ottenere una buona classificazione del parlato in vocale, non vocale o componenti transitorie. Nel 1988, Mallat sviluppò un'implementazione efficiente della trasformata Wavelet discreta utilizzando dei banchi di filtri [12]. In tale algoritmo, i coefficienti di approssimazione A_j e dettaglio D_j, al livello j-esimo, del segnale d'ingresso vengono determinati utilizzando dei *filtri quadrature mirror (QMF)*. I segnali di sottobanda A e D sono i coefficienti di approssimazione e di dettaglio e sono ottenuti utilizzando dei filtri passa-basso e passa-alto rispettivamente, implementati utilizzando le Mother Wavelet di Daubechies. Nell'implementazione utilizzata, si utilizzano le Wavelet di Daubechies a 4 punti. Utilizzando la trasformata Wavelet discreta, si può dividere il segnale parlato in quattro sottobande non uniformi. Nella figura seguente si utilizza una decomposizione Wavelet a tre livelli.

La struttura di decomposizione Wavelet può essere utilizzata per ottenere la periodicità più significativa nelle sottobande.

5.3.2 Teager Energy Operator

Come detto in precedenza, l'operatore TEO permette una miglior discriminabilità tra parlato e rumore e sopprime ulteriormente le componenti

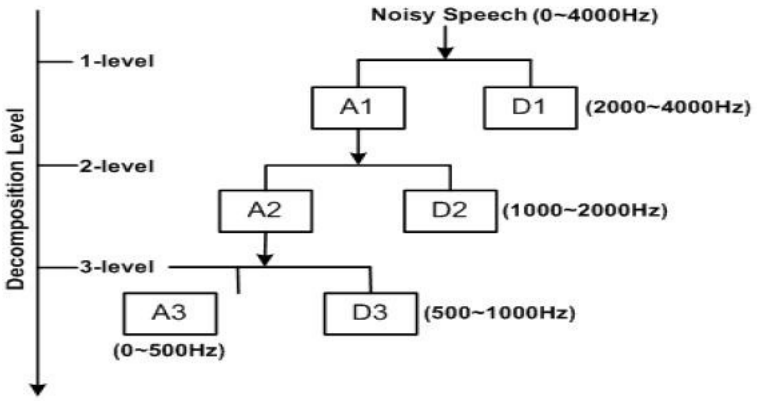

Figura 5.2: Decomposizione a 3 livelli tramite Wavelet

rumorose dai segnali parlati rumorosi [15]. Inoltre, il metodo di soppressione del rumore basato sul TEO può essere implementato molto più facilmente nel dominio del tempo rispetto all'approccio tradizionale basato sul dominio delle frequenze.

5.3.3 Calcolo della SSACF

La *funzione di auto-correlazione (ACF)* utilizzata per misurare la periodicità delle sequenze dei segnali di sottobanda è così definita:

$$R(k) = \sum_{n=0}^{p-k} s(n)s(n+k) \ , \ k = 0, 1, \ldots p \tag{5.24}$$

dove p è la lunghezza della ACF e k sta ad indicare il numero di shift dei campioni. Tale funzione sarà qui definita nel dominio delle sottobande e sarà chiamata *funzione di autocorrelazione per i segnali di sottobanda (SSACF)*. Essa può essere ricavata dai coefficienti Wavelet di ogni sottobanda dopo aver applicato il Teager Energy Operator. Si può notare che la SSACF del parlato vocale ha più picchi rispetto al parlato non vocale e al rumore bianco. Inoltre, per il parlato vocale, la ACF ha una periodicità più elevata del rumore bianco, soprattutto nella sottobanda $A3$.

5.3.4 Calcolo della DSSACF e della MDSSACF

Per valutare la periodicità dei segnali di sottobanda, viene utilizzato un metodo *Mean-Delta* per ogni SSACF [16]. Inizialmente, si utilizza una misura simile alla valutazione delta cepstrum per stimare la periodicità della SSACF, ossia la *funzione di autocorrelazione Delta per i segnali di sottobanda (DSSACF)*,

così calcolata:

$$\dot{R}_M(k) = \frac{\sum_{m=-M}^{M} mR(k+m)}{\sum_{m=-M}^{M} m^2} \tag{5.25}$$

dove $\dot{R}_M(k)$ indica la DSSACF su un insieme di M campioni vicini. Per un frame particolare, essa viene calcolata utilizzando soltanto la SSACF del frame in questione (processamento intra-frame), mentre il cepstro delta viene calcolato utilizzando solo i coefficienti cepstrali dai frame vicini (processamento inter-frame). Si osserva che il valore della DSSACF è quasi simile alle variazioni locali della SSACF. Successivamente, il delta della SSACF viene mediato su un insieme di M campioni vicini $\overline{R}_M$, dove la *media dei valori assoluti della DSSACF (MDSSACF)* si ottiene dalla relazione:

$$\overline{R}_M = \frac{1}{N_b} \sum_{k=0}^{N_b-1} |\dot{R}_M(k)| \tag{5.26}$$

dove N_b indica la lunghezza del segnale di sottobanda. Il parametro *SAE* finale si ottiene sommando i quattro valori di MDSSACF dei segnali di sottobanda. Infatti, ognuna di esse fornisce informazione per estrarre in maniera precisa gli istanti in cui c'è voice activity.

5.3.5 Schema a blocchi dell'algoritmo VAD

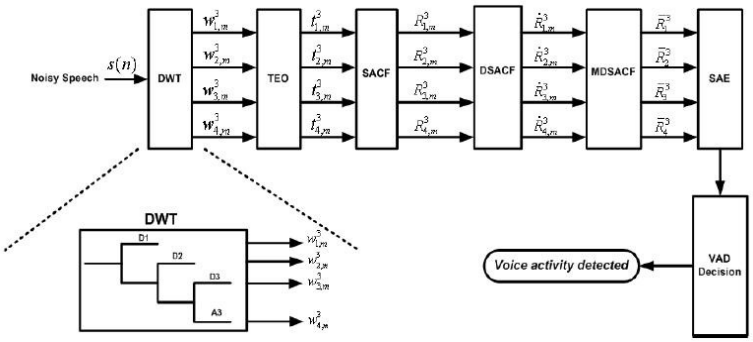

Figura 5.3: Schema a blocchi dell'algoritmo VAD

Nella figura soprastante si può vedere lo schema a blocchi dell'algoritmo di VAD. Per un dato livello di decomposizione j, la trasformata Wavelet decompone il segnale parlato rumoroso in $j+1$ sottobande corrispondenti agli insiemi di coefficienti Wavelet, denominati $w^j_{k,m}$. In questo caso, per il livello $j = 3$:

$$w^3_{k,m} = DWT\{s(n), 3\}\ ,\ n = 1, \ldots, N\ ,\ k = 1, \ldots, 4 \tag{5.27}$$

dove $w^3_{k,m}$ indica il m-esimo coefficiente della k-esima sottobanda, mentre N corrisponde alla lunghezza della finestra. La lunghezza di ogni sottobanda è

$N/2^k$. Ad esempio, se $k = 1$, $w^3_{1,m}$ corrisponde al segnale di sottobanda $D1$. Applicando l'operatore TEO, si ottiene:

$$t^3_{k,m} = \Psi_d[w^3_{k,m}] \ , \ k = 1, \ldots, 4 \tag{5.28}$$

La SSACF si ottiene calcolando l'energia del segnale $t^3_{k,m}$ nella maniera seguente:

$$R^3_{k,m} = R[t^3_{k,m}] \tag{5.29}$$

dove $R[\cdot]$ denota l'operazione di auto-correlazione. Successivamente, si calcola la DSSACF tramite la relazione:

$$\dot{R}^3_{k,m} = \Delta[R^3_{k,m}] \tag{5.30}$$

dove $\Delta[\cdot]$ denota l'operatore Delta. Si ottiene la MDSSACF tramite la relazione:

$$\overline{R}^3_k = E[\dot{R}^3_{k,m}] \tag{5.31}$$

dove $E[\cdot]$ denota l'operatore di media. Infine, il parametro SAE si ottiene dalla relazione:

$$SAE = \sum_{k=1}^{4} \overline{R}^3_k \tag{5.32}$$

5.3.6 Decisione VAD basata su sogliatura adattativa

Per determinare in maniera accurata il limite per la voice activity, la decisione VAD viene solitamente operata via sogliatura. Per stimare con precisione le caratteristiche del rumore che variano nel tempo, si utilizza un valore di soglia adattativo derivato dalle statistiche del parametro SAE durante i frame rumorosi, e il processo di decisione VAD aggiorna ricorsivamente la soglia utilizzando la media e la varianza dei valori della SAE. Inizialmente, si calcolano la media e la varianza iniziali del rumore dei primi cinque frame, assumendo che tali frame contengano soltanto rumore. Vengono poi calcolate le soglie per il parlato e per il rumore attraverso tali relazioni:

$$T_s = \mu_n + \alpha_s \cdot \sigma_n \tag{5.33}$$

$$T_n = \mu_n + \beta_n \cdot \sigma_n \tag{5.34}$$

dove T_s e T_n indicano la soglia del parlato e la soglia del rumore, rispettivamente. Similmente, μ_n e σ_n indicano la media e la varianza dei valori della funzione SAE, rispettivamente. La regola di decisione VAD è così definita:

se $(SAE(t) > T_s)$ allora $VAD(t) = 1$
altrimenti se $(SAE(t) < T_n)$ allora $VAD(t) = 0$
altrimenti $VAD(t) = VAD(t-1)$

Se il risultato del rilevamento trova un periodo rumoroso, la media e la varianza dei valori della SAE vengono così aggiornati:

$$\mu_n(t) = \gamma \cdot \mu_n(t-1) + (1-\gamma) \cdot SAE(t) \tag{5.35}$$

$$\sigma_n(t) = \sqrt{[SAE^2_{buffer}]_{mean} - [\mu_n(t)]^2} \tag{5.36}$$

$$[SAE^2_{buffer}]_{mean}(t) = \gamma \cdot [SAE^2_{buffer}]_{mean}(t-1) + (1-\gamma) \cdot SAE(t)^2 \tag{5.37}$$

Dove $[SAE^2_{buffer}]_{mean}(t-1)$ è la media in memoria del valore della SAE in un frame contenente soltanto rumore. Vengono successivamente aggiornate le soglie utilizzando la media e la varianza aggiornate dei valori della SAE. Le due soglie vengono aggiornate soltanto nei periodi di inattività vocale, e non durante i periodi di voice activity.

5.3.7 Impostazione dei parametri VAD

I parametri dell'algoritmo, utilizzati nell'implementazione di questo lavoro di tesi, assumono i seguenti valori:

- dimensione dei frame=256 campioni per frame
- $M = 8$
- $\alpha_s = 5$
- $\beta_n = -1$
- $\gamma = 0.95$

Capitolo 6

Test

In questo capitolo verranno descritti gli aspetti riguardanti i test effettuati sul database e successivamente verranno presentati i risultati ottenuti. Tali test sono stati eseguiti su un notebook dotato di processore *Intel Mobile Core 2 Duo P8400 @ 2.26 GHz*, 4 GB di memoria RAM DDR2 @ 400 MHz e disco fisso da 5400 rpm.

6.1 Linguaggio di programmazione e descrizione del database utilizzato

La scelta del linguaggio di programmazione è ricaduta su *Matlab*. Principalmente, la ragione che ha guidato la scelta è stata la grande disponibilità di funzioni, spesso evolute ed ottimizzate, utili allo scopo di questa tesi. Infatti tutto il sistema si basa sull'elaborazione di matrici, punto forte di questo linguaggio. Durante la fase iniziale del lavoro di tesi, i test sulle funzionalità realizzate nel sistema venivano fatti su piccoli file audio contenenti qualche secondo di parlato. Tuttavia, per avere un'indicazione delle prestazioni del sistema, si è optato per l'utilizzo di un database largamente utilizzato in letteratura, il *database NIST-RT03*. NIST (National Institute of Standard and Technology) è un'agenzia del governo degli Stati Uniti d'America che si occupa di sviluppo, promozione e misurazione di standard e tecnologie, ed è attiva nel campo delle tecnologie relative al linguaggio umano. L'agenzia indice periodicamente dei concorsi nell'ambito del *Rich Transcription Evaluation Project* (RT), un progetto nato con lobiettivo di sviluppare tecnologie evolute di trascrizione automatica del testo parlato, finalizzate a una maggiore leggibilità da parte dell'umano e una maggiore utilità del loro utilizzo tramite macchine automatiche. Due sono i task definiti all'interno di questo progetto: trascrizioni Speech-to-Text(STT) e Metadata Extraction (MDE). Nella maggior parte degli articoli studiati la valutazione delle prestazioni dei sistemi o dei nuovi algoritmi sviluppati viene fatta su database provenienti dal progetto RT o, comunque, prodotti direttamente dal NIST. Quindi, a

seguito di una ricerca tra i vari database utilizzati nei concorsi degli ultimi anni, la scelta è ricaduta su RT03 il quale possedeva tutte le caratteristiche necessarie agli scopi di questo lavoro di tesi.

6.1.1 Contenuto del database NIST-RT03

Non tutto l'audio contenuto nel database è utilizzato nel processo di Voice Activity Detection in RT03. Questo porta alla mancanza, per alcuni file audio, delle trascrizioni necessarie a valutare l'accuratezza del processo di VAD operato sui file stessi. Risulta quindi utile dare una spiegazione delle parti del database utilizzate in questo lavoro di tesi. Nella sottocartella $data/audio/eval03$ risiedono tutti i file audio, in formato *.sph*, suddivisi per lingua e successivamente per tipologia. Questo lavoro di tesi si concentra sui file CTS contenuti nella sottocartella $data/audio/eval03/english/cts$, i quali sono file audio in formato *.sph* a due canali, dove ogni canale è relativo ad uno dei due parlatori coinvolti nella conversazione. Tali file sono campionati alla frequenza di campionamento $F_s = 8000$ Hz, e hanno una durata di circa 5 minuti ognuna. Nella sottocartella $data/indices$ risiedono dei file con estensione *.uem*, i quali hanno il compito di delimitare le regioni dei file audio che devono essere analizzate dai processi nellambito di RT03. Solo per tali regioni dei file audio viene data la trascrizione e quindi è ragionevole fare una analisi. Nella sottocartella $data/references/eval03$ sono memorizzati i file di trascrizioni, sempre suddivisi prima per lingua e poi per tipologia. La sottocartella dedicata alla CTS in inglese è $data/references/eval03/english/cts$, la quale contiene svariati file di trascrizioni con diversi formati. Quelli utili allo scopo del VAD sono i file con estensione:

- *.mdtm*: contengono le informazioni sui cambi di turno nel formato previsto dal NIST; questo formato permette la valutazione dei sistemi di speaker diarization tramite strumenti sviluppati appositamente dal NIST stesso.

- *.uem*: questi file si differenziano da quelli contenuti in $data/indices$; in questo caso delimitano piccole regioni di parlato sovrapposto. Sono utili soprattutto per le registrazioni di notiziari, le quali presentano degli stacchi pubblicitari che non vengono trascritti. Si noti come non tutti i file audio di conversazioni telefoniche in inglese siano trascritti.

6.2 Procedura di test utilizzata e risultati

L'approccio utilizzato per i test è il seguente:

- Inizialmente, si è voluto testare l'efficacia degli algoritmi di VAD. Essi sono stati testati su piccoli file audio contenenti pochi secondi di

parlato, corrotto da rumore bianco, al fine di ottimizzare i parametri relativi al calcolo delle feature e delle sogliature utilizzate.

- Successivamente, tramite la procedura *LeggiFile.m* e il metodo di *FeatExtraction*, vengono convertiti i file audio dal formato *.sph* al formato *.wav*, allo scopo di un calcolo delle feature più efficace. Inoltre, tramite la lettura delle etichettature contenute nei file *.mdtm* e *.uem*, si costruisce un vettore booleano (ossia contenente soltanto valori 0 e 1) relativo al parlato a scopo di confronto. Da notare che non tutte le conversazioni telefoniche contenute nel database sono dotate di etichettatura, per cui sono state convertite soltanto quelle per le quali erano presenti i file *.uem* e *.mdtm*.

- Si è poi passati a testare i vari algoritmi sul database *NIST-RT03*, sporcando preliminarmente le conversazioni telefoniche con del rumore. Dopo aver calcolato le feature necessarie e la sogliatura sulle conversazioni, vengono costruiti dei vettori booleani del parlato in base a queste ultime. Si noti che, per gli algoritmi implementati basati sulla Voice Activity Shape e sull'entropia spettrale rispettivamente, viene utilizzato un metodo di *unione dei silenzi corti* per i periodi di silenzio minori o uguali a 300 millisecondi. Ossia, vengono considerati tali periodi come periodi di parlato se nell'algoritmo, all'inizio e alla fine di tali periodi, viene rilevato del parlato.

- Infine, si confrontano i vettori booleani ottenuti dagli algoritmi con quelli letti dal database, attraverso dei parametri di confronto sotto definiti.

6.2.1 Confronto preliminare e scrematura

Al fine di ottenere un'indicazione sul miglior algoritmo implementato da confrontare con lo standard di mercato $G.729B$, del quale è stata scaricata un'efficiente implementazione in Matlab su [19]. È stata eseguita un'analisi preliminare su un sottoinsieme del database, composto da 5 conversazioni. Siano definiti:

- S_i $(i = 1, 2, ...)$: gli intervalli temporali dove c'è del parlato
- N_i $(i = 1, 2, ...)$: gli intervalli temporali dove c'è del rumore
- T_i $(i = 1, 2, ...)$: gli intervalli temporali *true* dove si stima la presenza di parlato e c'è veramente parlato (sottoinsieme di S_i)
- R_i $(i = 1, 2, ...)$: gli intervalli temporali dove si stima la presenza di rumore e c'è veramente rumore (sottoinsieme di N_i)

Nome file	SNR	VAS	BSE Wavelet	BSE Fourier	SAE
fsh_60262	5	50.6688%	58.3676%	40.4845%	80.0295%
	0	44.8973%	45.0869%	27.2249%	76.1955%
	-5	55.2607%	34.3865%	24.9921%	77.2699%
fsh_60354	5	74.7774%	69.9321%	38.3376%	72.5827%
	0	66.4970%	64.2494%	34.7116%	72.8796%
	-5	58.1001%	43.7765%	32.8244%	73.6853%
fsh_60416	5	47.1246%	55.5272%	29.5101%	76.1661%
	0	44.5900%	40.1065%	27.5080%	81.5974%
	-5	39.8403%	36.7093%	21.2034%	79.3823%
fsh_60463	5	35.5252%	49.9891%	29.2752%	84.3533%
	0	48.8607%	31.76%	21.8207%	82.9201%
	-5	85.6228%	26.8663%	21.3108%	82.4870%
fsh_60493	5	33.6643%	51.2517%	30.0095%	75.3011%
	0	33.6009%	34.8368%	26.8829%	73.4840%
	-5	66.1033%	27.2420%	23.0062%	77.6252%

Tabella 6.1: Risultati algoritmi di VAD

Si indica con:

$$sim = \sum_i (T_i + R_i)$$

l'*indice di somiglianza dei vettori booleani del parlato*. Utilizzando del rumore bianco e i valori 5,0 e -5 dB per il rapporto SNR, si è ottenuta una stima delle prestazioni medie dei vari algoritmi su tali conversazioni, e in base a tali risultati, si è scelto l'algoritmo da confrontare direttamente con il $G.729B$.

Leggendo la tabella, si nota che l'algoritmo che meglio si confronta al $G.729B$ è quello basato sulla SAE, in quanto fornisce prestazioni altamente superiori rispetto agli algoritmi basati sulla VAS e sulla BSE. Questa superiorità è dovuta molto probabilmente ad un algoritmo di sogliatura adottato più efficace, nonché ad un calcolo di feature più utili dal punto di vista statistico per eseguire la decisione di Voice Activity.

6.2.2 Confronto diretto tra SAE e G.729B

Siano definiti i seguenti valori:

- S_i $(i = 1, 2, ...)$: gli intervalli temporali dove c'è del parlato
- N_i $(i = 1, 2, ...)$: gli intervalli temporali dove c'è del rumore
- T_i $(i = 1, 2, ...)$: gli intervalli temporali *true* dove si stima la presenza di parlato e c'è veramente parlato (sottoinsieme di S_i)
- R_i $(i = 1, 2, ...)$: gli intervalli temporali dove si stima la presenza di rumore e c'è veramente rumore (sottoinsieme di N_i)

- M_i $(i = 1, 2, ...)$: gli intervalli temporali *miss* dove si stima la presenza di rumore e invece c'è del parlato (sottoinsieme di S_i complementare a T_i)
- F_i $(i = 1, 2, ...)$: gli intervalli temporali *falsi positivi* dove si stima la presenza di parlato e invece c'è del rumore (sottoinsieme di N_i)

I parametri di confronto calcolati sono i seguenti:

- *Indice di somiglianza dei vettori booleani del parlato*:

$$sim = \sum_i (T_i + R_i)$$

- *Probabilità di successo*:

$$P_{ok} = \frac{\sum_i T_i}{\sum_i S_i} < 1 \; sempre$$

- *Probabilità di miss*:

$$P_{miss} = \frac{\sum_i M_i}{\sum_i S_i} = 1 - P_{ok} < 1 \; sempre$$

- *Probabilità di false detection*:

$$P_{false} = \frac{\sum_i F_i}{\sum_i T_i} < 1 \; sempre$$

Tali parametri vengono calcolati utilizzando dei rapporti segnale rumore SNR pari a 5,0 e -5 dB ed utilizzando quattro tipi di rumore da aggiungere alle conversazioni:

- *Rumore bianco*: il rumore più semplice utilizzato a scopo di confronto e scrematura preliminare tra i metodi. Esso ha densità spettrale di potenza costante, ed è facilmente generabile tramite una funzione che genera numeri casuali in un intervallo di tempo prefissato.
- *Babble noise*: esso viene percepito come disturbo causato dal dialogo simultaneo tra più persone.
- *Car noise*: esso viene percepito come disturbo causato dagli ambienti interni ad un'automobile.
- *Factory noise*: esso viene percepito come disturbo causato dagli ambienti interni ad una fabbrica.

Per gli ultimi tre tipi di rumore, sono stati scaricati dei piccoli sample di rumore dal *database NOISEX-92* [17][18] e sono stati aggiunti in maniera ciclica alle conversazioni telefoniche. Nelle seguenti tabelle verranno presentati i risultati relativi all'esecuzione degli algoritmi di SAE e G.729B sulle conversazioni telefoniche del database *NIST-RT03*, con i parametri di confronto espressi in percentuale.

Tipo di rumore	$SNR(dB)$	sim	P_{ok}	P_{miss}	P_{false}
White	5	80,1115%	96,5367%	3,4637%	22,0641%
	0	80,5316%	97,1602%	2,8398%	22,0489%
	-5	80,6827%	97,4214%	2,5787%	22,0881%
Babble	5	79,3454%	95,2685%	4,7315%	21,9868%
	0	79,6395%	95,7333%	4,2666%	22,0777%
	-5	80,0054%	96,2662%	3,7338%	21,9929%
Car	5	78,6934%	94,2842%	5,7157%	22,0432%
	0	78,7781%	94,4430%	5,5571%	22,0672%
	-5	78,9649%	94,7544%	5,2456%	22,0658%
Factory	5	78,9806%	94,6406%	5,3595%	21,9858%
	0	79,0860%	94,8407%	5,1593%	22,0043%
	-5	79,1705%	95,0174%	4,9826%	22,0466%

Tabella 6.2: Risultati medi dell'algoritmo basato sulla SAE

Tipo di rumore	$SNR(dB)$	sim	P_{ok}	P_{miss}	P_{false}
White	5	80,0931%	95,0041%	4,9960%	20,7552%
	0	76,3970%	89,7536%	10,2464%	21,1903%
	-5	70,8191%	81,7316%	18,2684%	21,5332%
Babble	5	85,8381%	98,5615%	1,4384%	16,2840%
	0	85,3236%	98,2808%	1,7191%	16,7203%
	-5	84,7049%	97,8215%	2,1785%	17,1394%
Car	5	85,9985%	99,0260%	0,9740%	16,4399%
	0	85,9761%	98,9865%	1,0135%	16,4347%
	-5	85,9244%	98,8597%	1,1404%	16,4075%
Factory	5	86,5493%	97,6482%	2,3517%	14,5737%
	0	86,2269%	96,2408%	3,7593%	13,7229%
	-5	85,2186%	93,7703%	6,2297%	12,7938%

Tabella 6.3: Risultati medi dell'algoritmo $G.729B$

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	80.2402	98.5462	1.4538	23.2350
fsh_60354.wav	73.8338	97.9892	2.0108	33.6127
fsh_60416.wav	80.7029	96.2080	3.7920	20.1864
fsh_60463.wav	84.1363	98.0457	1.9543	16.9298
fsh_60493.wav	77.9632	97.9357	2.0643	26.4440
fsh_60549.wav	84.9989	97.4595	2.5405	15.1546
fsh_60571.wav	84.1590	95.5857	4.4143	14.2857
fsh_60593.wav	76.7129	97.0267	2.9733	27.6075
fsh_60627.wav	65.9661	96.5550	3.4450	48.8603
fsh_60648.wav	77.3063	97.8792	2.1208	27.3176
fsh_60650.wav	82.8400	97.9305	2.0695	18.7582
fsh_60720.wav	81.6238	97.3575	2.6425	19.9308
fsh_60732.wav	78.2590	98.2394	1.7606	26.3027
fsh_60797.wav	82.6013	98.0102	1.9898	19.2087
fsh_60862.wav	73.4033	97.3851	2.6149	33.9333
fsh_60885.wav	84.3635	97.9742	2.0258	16.5676
fsh_61039.wav	84.1357	98.1076	1.8924	17.0558
fsh_61192.wav	76.5068	96.9730	3.0270	27.7313
sw_45097.wav	70.6490	96.4463	3.5537	38.3872
sw_45142.wav	82.8248	94.5826	5.4174	15.2304
sw_45355.wav	69.0886	95.6729	4.3271	40.8609
sw_45454.wav	82.6656	96.1716	3.8284	17.0572
sw_45586.wav	77.8104	95.5573	4.4427	24.1164
sw_45654.wav	86.7480	97.9391	2.0609	13.3105
sw_45713.wav	81.5055	93.6638	6.3362	16.0509
sw_45727.wav	73.3376	96.1603	3.8397	32.8184
sw_45819.wav	70.1312	95.0735	4.9265	37.5758
sw_46140.wav	80.5031	94.7748	5.2252	18.9026
sw_46412.wav	83.3924	94.9512	5.0488	14.6476
sw_46512.wav	89.4348	96.6071	3.3929	8.3768
sw_46615.wav	85.2380	95.2325	4.7675	12.4411
sw_46677.wav	86.9165	95.1472	4.8528	10.0073
sw_46789.wav	82.3554	94.7635	5.2365	16.0682
sw_46868.wav	81.1959	97.3976	2.6024	20.8730
sw_47346.wav	87.1995	96.2506	3.7494	10.9026
sw_47411.wav	83.2667	93.7187	6.2813	13.5576

Tabella 6.4: Risultati sperimentali SAE con rumore bianco e $SNR = 5$ dB

Nome file	*sim*	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	76.6588	93.3448	6.6552	23.8912
fsh_60354.wav	72.4812	97.8908	2.1092	37.0061
fsh_60416.wav	84.0666	96.2993	3.7007	15.2838
fsh_60463.wav	72.2424	81.3228	18.6772	16.6634
fsh_60493.wav	70.4661	84.0963	15.9037	24.3401
fsh_60549.wav	81.7207	96.1299	3.8701	18.4460
fsh_60571.wav	82.9906	95.6381	4.3619	16.1226
fsh_60593.wav	78.2858	97.8850	2.1150	25.8076
fsh_60627.wav	63.0360	82.8368	17.1632	42.5687
fsh_60648.wav	80.5225	98.8089	1.1911	23.1712
fsh_60650.wav	82.1722	94.9512	5.0488	16.6282
fsh_60720.wav	76.8616	90.8399	9.1601	20.3138
fsh_60732.wav	74.2033	92.3878	7.6122	27.2053
fsh_60797.wav	82.7620	95.1404	4.8596	16.1023
fsh_60862.wav	76.7732	96.6813	3.3187	27.1696
fsh_60885.wav	85.2902	94.5490	5.4510	11.6629
fsh_61039.wav	84.6926	96.4086	3.5914	14.4828
fsh_61192.wav	71.3125	92.3070	7.6930	33.1973
sw_45097.wav	75.3918	97.4946	2.5054	30.3786
sw_45142.wav	85.3172	97.5829	2.4171	14.8145
sw_45355.wav	71.9506	96.9486	3.0514	36.7911
sw_45454.wav	84.6625	96.6989	3.3011	14.8280
sw_45586.wav	77.0544	97.2566	2.7434	27.2466
sw_45654.wav	86.8866	96.6833	3.3167	11.6940
sw_45713.wav	83.4666	95.3201	4.6799	14.9800
sw_45727.wav	74.1852	95.4716	4.5284	31.5421
sw_45819.wav	71.3643	93.0933	6.9067	34.2379
sw_46140.wav	82.8011	97.0633	2.9367	18.0172
sw_46412.wav	88.4992	97.8539	2.1461	10.8109
sw_46512.wav	89.9117	98.7960	1.2040	10.0209
sw_46615.wav	87.4292	96.3897	3.6103	10.7272
sw_46677.wav	89.1015	97.9649	2.0351	10.1956
sw_46789.wav	86.0278	97.7901	2.2099	14.0371
sw_46868.wav	79.3821	96.3440	3.6560	22.3665
sw_47346.wav	86.0642	95.2313	4.7687	11.2509
sw_47411.wav	87.3154	98.6447	1.3553	13.1846

Tabella 6.5: Risultati sperimentali G.729B con rumore bianco e $SNR = 5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	80,0080	98,0270	1,9730	23,1200
fsh_60354.wav	73,8760	97,8190	2,1807	33,4390
fsh_60416.wav	81,5760	97,0790	2,9208	19,8210
fsh_60463.wav	84,7440	98,5530	1,4467	16,6370
fsh_60493.wav	77,3080	97,2390	2,7614	26,7710
fsh_60549.wav	85,0210	97,5840	2,4159	15,2370
fsh_60571.wav	84,8610	96,5320	3,4683	14,2960
fsh_60593.wav	77,2040	97,6810	2,3186	27,4500
fsh_60627.wav	66,2660	97,1290	2,8708	48,7030
fsh_60648.wav	77,5810	98,0130	1,9866	27,0610
fsh_60650.wav	83,0760	98,4160	1,5841	18,8730
fsh_60720.wav	82,1640	97,9270	2,0725	19,7350
fsh_60732.wav	78,0460	97,9690	2,0314	26,3750
fsh_60797.wav	83,0060	99,0050	0,9949	19,5310
fsh_60862.wav	73,9180	98,5060	1,4943	33,9850
fsh_60885.wav	84,3630	98,2020	1,7979	16,7610
fsh_61039.wav	83,9240	98,0580	1,9422	17,2680
fsh_61192.wav	76,8250	97,2700	2,7297	27,5350
sw_45097.wav	71,1130	97,2910	2,7090	38,2630
sw_45142.wav	83,7640	95,7400	4,2600	15,1230
sw_45355.wav	70,1080	97,2550	2,7454	40,3500
sw_45454.wav	83,9730	97,6010	2,3988	16,7080
sw_45586.wav	78,0620	96,2070	3,7932	24,3050
sw_45654.wav	87,1270	98,8500	1,1503	13,6730
sw_45713.wav	82,1330	94,4820	5,5179	16,0110
sw_45727.wav	73,5700	96,4410	3,5594	32,6940
sw_45819.wav	70,7660	95,9090	4,0910	37,2180
sw_46140.wav	81,0690	95,8950	4,1055	19,1490
sw_46412.wav	84,5190	96,2850	3,7152	14,4940
sw_46512.wav	89,8140	97,2720	2,7281	8,5788
sw_46615.wav	85,7200	95,8230	4,1775	12,4140
sw_46677.wav	87,7540	96,0440	3,9558	9,8898
sw_46789.wav	82,5010	95,2940	4,7056	16,3590
sw_46868.wav	81,4710	98,4020	1,5975	21,3410
sw_47346.wav	87,7540	96,9360	3,0636	10,8960
sw_47411.wav	84,1510	95,0320	4,9678	13,6970

Tabella 6.6: Risultati sperimentali SAE con rumore bianco e $SNR = 0$ dB

Nome file	*sim*	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	72,1100	86,6000	13,4000	24,4790
fsh_60354.wav	70,7090	95,2550	4,7445	37,8560
fsh_60416.wav	79,1010	89,8710	10,1290	15,6110
fsh_60463.wav	65,4570	71,7950	28,2050	16,5770
fsh_60493.wav	63,3820	73,1640	26,8360	24,9600
fsh_60549.wav	78,2570	91,2400	8,7605	18,5610
fsh_60571.wav	79,8310	90,1870	9,8134	15,1270
fsh_60593.wav	75,0910	93,1510	6,8494	26,3610
fsh_60627.wav	54,6100	68,7620	31,2380	48,1930
fsh_60648.wav	78,1310	95,7700	4,2299	23,8210
fsh_60650.wav	78,2920	88,9820	11,0180	16,1330
fsh_60720.wav	68,7000	79,9440	20,0560	21,6370
fsh_60732.wav	70,0070	85,2370	14,7630	27,3480
fsh_60797.wav	79,1400	89,9260	10,0740	15,9520
fsh_60862.wav	74,0610	92,6500	7,3497	27,7300
fsh_60885.wav	79,7020	88,2630	11,7370	12,4630
fsh_61039.wav	80,9480	92,1480	7,8516	15,1900
fsh_61192.wav	67,9700	85,1190	14,8810	32,8040
sw_45097.wav	73,1890	93,6270	6,3730	30,5730
sw_45142.wav	83,4670	95,4000	4,6005	15,0930
sw_45355.wav	70,1220	94,3780	5,6217	37,7450
sw_45454.wav	81,5390	92,8870	7,1133	15,2000
sw_45586.wav	75,0660	94,0910	5,9086	27,4930
sw_45654.wav	82,5940	91,5600	8,4402	11,9800
sw_45713.wav	80,3400	91,3580	8,6421	15,2170
sw_45727.wav	71,8170	90,9920	9,0080	31,6650
sw_45819.wav	68,7080	87,9290	12,0710	34,4660
sw_46140.wav	78,7400	91,7380	8,2617	18,5160
sw_46412.wav	84,8120	93,3710	6,6294	10,8970
sw_46512.wav	88,3300	97,0050	2,9955	10,1540
sw_46615.wav	83,6120	91,4740	8,5257	10,5600
sw_46677.wav	86,0630	94,5150	5,4848	10,4650
sw_46789.wav	82,7640	94,0260	5,9735	14,5560
sw_46868.wav	77,2860	93,0710	6,9287	22,3900
sw_47346.wav	81,1040	89,2780	10,7220	11,5060
sw_47411.wav	85,2400	96,3660	3,6341	13,5700

Tabella 6.7: Risultati sperimentali G.729B con rumore bianco e $SNR = 0$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	79,8190	97,6640	2,3364	23,0730
fsh_60354.wav	74,0250	97,9330	2,0674	33,3140
fsh_60416.wav	81,9380	98,0020	1,9985	20,1310
fsh_60463.wav	84,0710	97,8930	2,1066	16,8780
fsh_60493.wav	77,7100	97,9360	2,0643	26,7730
fsh_60549.wav	85,2360	97,9080	2,0922	15,2630
fsh_60571.wav	86,2220	98,1570	1,8433	14,1340
fsh_60593.wav	77,3110	98,2270	1,7730	27,7150
fsh_60627.wav	66,3090	97,5440	2,4561	48,8550
fsh_60648.wav	77,2010	97,4500	2,5503	27,1350
fsh_60650.wav	83,0120	98,5440	1,4563	19,0560
fsh_60720.wav	81,6020	97,3580	2,6425	19,9570
fsh_60732.wav	77,9820	97,5350	2,4648	26,1320
fsh_60797.wav	82,2600	97,9850	2,0153	19,6040
fsh_60862.wav	73,5110	97,7010	2,2989	34,0000
fsh_60885.wav	84,4720	98,1260	1,8739	16,5680
fsh_61039.wav	84,0300	98,1820	1,8177	17,2460
fsh_61192.wav	77,0160	97,7570	2,2432	27,6470
sw_45097.wav	70,7960	97,0870	2,9129	38,5840
sw_45142.wav	84,1900	96,3560	3,6444	15,1550
sw_45355.wav	69,7260	97,0460	2,9543	40,7750
sw_45454.wav	83,6410	97,3100	2,6896	16,8580
sw_45586.wav	78,7750	97,1160	2,8839	24,1040
sw_45654.wav	85,9910	97,4360	2,5641	13,7480
sw_45713.wav	83,1850	95,6510	4,3488	15,7660
sw_45727.wav	73,8020	97,1970	2,8027	32,9010
sw_45819.wav	71,6770	97,1480	2,8522	36,7440
sw_46140.wav	81,7610	96,5660	3,4337	18,8610
sw_46412.wav	85,2280	97,0470	2,9531	14,3310
sw_46512.wav	90,7630	98,4640	1,5360	8,6380
sw_46615.wav	86,2440	96,7190	3,2806	12,6160
sw_46677.wav	87,9840	96,2970	3,7029	9,8639
sw_46789.wav	83,2080	96,1150	3,8851	16,2190
sw_46868.wav	81,4280	98,7370	1,2626	21,6600
sw_47346.wav	87,8570	97,3710	2,6292	11,1760
sw_47411.wav	84,5930	95,6050	4,3946	13,6900

Tabella 6.8: Risultati sperimentali SAE con rumore bianco e $SNR = -5$ dB

Nome file	*sim*	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	66,4590	76,4130	23,5870	23,5850
fsh_60354.wav	67,1110	88,7760	11,2240	38,9650
fsh_60416.wav	71,2410	79,4150	20,5850	15,9420
fsh_60463.wav	60,0150	64,1100	35,8900	16,4330
fsh_60493.wav	58,6620	64,5010	35,4990	23,8940
fsh_60549.wav	69,1680	78,9520	21,0480	19,4910
fsh_60571.wav	75,0940	83,8930	16,1070	15,3270
fsh_60593.wav	67,8170	82,6640	17,3360	28,1130
fsh_60627.wav	53,3470	62,5230	37,4770	45,8880
fsh_60648.wav	74,0670	90,4670	9,5326	24,9110
fsh_60650.wav	73,3690	82,3940	17,6060	16,4100
fsh_60720.wav	59,3740	66,8750	33,1250	22,9630
fsh_60732.wav	63,5100	74,2270	25,7730	27,6820
fsh_60797.wav	73,8100	82,6290	17,3710	16,0810
fsh_60862.wav	68,9770	84,4180	15,5820	28,3530
fsh_60885.wav	71,5560	77,8090	22,1910	12,4300
fsh_61039.wav	73,0700	81,8190	18,1810	15,5230
fsh_61192.wav	61,4040	74,6030	25,3970	35,0940
sw_45097.wav	69,4230	87,6450	12,3550	31,4440
sw_45142.wav	77,4070	87,4930	12,5070	15,3800
sw_45355.wav	67,4230	89,2650	10,7350	38,3510
sw_45454.wav	78,2310	88,3210	11,6790	15,1240
sw_45586.wav	70,5570	87,7570	12,2430	28,8070
sw_45654.wav	76,0800	83,0130	16,9870	11,6680
sw_45713.wav	74,8360	83,9410	16,0590	15,2440
sw_45727.wav	67,0930	84,1260	15,8740	33,6230
sw_45819.wav	65,2980	80,8700	19,1300	34,4560
sw_46140.wav	73,4930	84,1340	15,8660	18,5580
sw_46412.wav	78,3450	85,5330	14,4670	11,0990
sw_46512.wav	82,4390	89,9550	10,0450	10,3200
sw_46615.wav	75,4550	81,4980	18,5020	10,7200
sw_46677.wav	79,3900	86,1980	13,8020	10,3650
sw_46789.wav	77,1850	87,0520	12,9480	15,0190
sw_46868.wav	73,2420	86,7040	13,2960	22,3910
sw_47346.wav	74,2990	80,7860	19,2140	11,5630
sw_47411.wav	81,2410	91,5590	8,4410	13,9770

Tabella 6.9: Risultati sperimentali G.729B con rumore bianco e $SNR = -5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	79,4610	96,7550	3,2451	22,8070
fsh_60354.wav	73,1340	95,9500	4,0498	33,1760
fsh_60416.wav	78,5730	93,3130	6,6872	20,4560
fsh_60463.wav	83,4850	96,8020	3,1980	16,6490
fsh_60493.wav	77,4140	97,9090	2,0912	27,1360
fsh_60549.wav	84,2650	96,2890	3,7111	15,0030
fsh_60571.wav	82,7130	94,0330	5,9665	14,6250
fsh_60593.wav	75,5820	94,7900	5,2100	27,4240
fsh_60627.wav	66,3740	97,8950	2,1053	48,9410
fsh_60648.wav	75,7440	94,4430	5,5570	26,7770
fsh_60650.wav	81,9390	96,7300	3,2703	18,8590
fsh_60720.wav	81,4080	96,7100	3,2902	19,6620
fsh_60732.wav	78,0460	97,9410	2,0585	26,3550
fsh_60797.wav	82,1320	96,9640	3,0357	18,9160
fsh_60862.wav	72,5680	95,8050	4,1954	34,0130
fsh_60885.wav	82,9320	96,1760	3,8238	16,7460
fsh_61039.wav	83,4780	97,3610	2,6394	17,2120
fsh_61192.wav	75,2970	95,0270	4,9730	27,8730
sw_45097.wav	71,1760	95,4560	4,5441	36,9850
sw_45142.wav	82,1210	93,3760	6,6240	15,0050
sw_45355.wav	68,3030	93,6740	6,3265	40,7770
sw_45454.wav	80,4030	93,0460	6,9542	17,1090
sw_45586.wav	76,4050	93,5050	6,4952	24,3120
sw_45654.wav	85,9490	96,7890	3,2111	13,2210
sw_45713.wav	81,4040	93,5470	6,4531	16,0710
sw_45727.wav	72,7680	95,0670	4,9327	32,8420
sw_45819.wav	70,0890	94,5260	5,4739	37,2750
sw_46140.wav	79,4970	93,3570	6,6434	18,9500
sw_46412.wav	83,1420	94,5460	5,4537	14,5840
sw_46512.wav	89,1400	95,8960	4,1036	8,0325
sw_46615.wav	85,3010	95,4920	4,5079	12,6050
sw_46677.wav	85,6600	93,7670	6,2328	10,1550
sw_46789.wav	81,3980	93,6050	6,3948	16,2160
sw_46868.wav	79,8230	95,2850	4,7153	20,8760
sw_47346.wav	87,0760	95,8160	4,1838	10,6420
sw_47411.wav	82,2350	92,0230	7,9771	13,2360

Tabella 6.10: Risultati sperimentali SAE con babble noise e $SNR = 5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	84,1030	97,9830	2,0170	18,0680
fsh_60354.wav	82,2220	98,0620	1,9384	23,2790
fsh_60416.wav	87,2210	98,8570	1,1425	13,7870
fsh_60463.wav	86,9280	98,2400	1,7598	13,6580
fsh_60493.wav	84,0200	98,5450	1,4554	18,4940
fsh_60549.wav	86,4420	99,0830	0,9167	15,2450
fsh_60571.wav	89,0400	97,5660	2,4340	10,5690
fsh_60593.wav	83,7320	98,3350	1,6651	19,1640
fsh_60627.wav	77,3050	96,3360	3,6636	29,6090
fsh_60648.wav	84,8530	98,6500	1,3501	17,6180
fsh_60650.wav	89,1790	99,0830	0,9165	11,8390
fsh_60720.wav	85,5260	99,5050	0,4953	16,8620
fsh_60732.wav	84,2530	97,7910	2,2088	18,1820
fsh_60797.wav	86,2540	99,3760	0,6240	15,5650
fsh_60862.wav	81,1970	97,9180	2,0816	22,3350
fsh_60885.wav	89,6480	98,9330	1,0667	10,6430
fsh_61039.wav	88,9610	98,8790	1,1211	11,6690
fsh_61192.wav	79,4070	95,7410	4,2586	24,2950
sw_45097.wav	83,0980	98,9510	1,0486	21,2370
sw_45142.wav	88,0140	99,6550	0,3448	13,4760
sw_45355.wav	79,5380	97,5670	2,4329	26,4570
sw_45454.wav	86,5500	97,1950	2,8050	13,0290
sw_45586.wav	82,9330	98,7850	1,2151	20,7890
sw_45654.wav	90,3150	99,3840	0,6157	10,2480
sw_45713.wav	87,9590	98,8830	1,1165	12,8350
sw_45727.wav	82,2070	97,4860	2,5143	21,9130
sw_45819.wav	77,9390	98,6610	1,3392	28,9240
sw_46140.wav	85,6440	99,1680	0,8324	16,3520
sw_46412.wav	90,5590	99,7870	0,2128	10,2540
sw_46512.wav	91,1570	99,7900	0,2103	9,5432
sw_46615.wav	90,6790	99,2580	0,7417	9,6736
sw_46677.wav	90,3870	97,7090	2,2913	8,5088
sw_46789.wav	88,5800	99,6250	0,3745	12,6990
sw_46868.wav	84,1060	98,8780	1,1218	18,5160
sw_47346.wav	90,0800	99,0330	0,9666	10,1530
sw_47411.wav	90,1370	99,5160	0,4841	10,7340

Tabella 6.11: Risultati sperimentali G.729B con babble noise e $SNR = 5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	79,4190	96,8070	3,1931	22,9020
fsh_60354.wav	73,2190	96,4030	3,5967	33,3730
fsh_60416.wav	79,8510	94,8500	5,1499	20,1240
fsh_60463.wav	83,8320	97,1320	2,8680	16,5140
fsh_60493.wav	77,6670	98,1230	1,8767	26,9670
fsh_60549.wav	84,3300	96,4630	3,5367	15,0790
fsh_60571.wav	83,4150	94,5670	5,4329	14,2600
fsh_60593.wav	76,3290	96,2080	3,7916	27,5020
fsh_60627.wav	66,3740	98,0220	1,9777	49,0070
fsh_60648.wav	75,7020	94,8190	5,1812	27,1230
fsh_60650.wav	82,2820	97,2150	2,7849	18,8440
fsh_60720.wav	81,6670	97,1500	2,8497	19,7070
fsh_60732.wav	78,0030	98,2390	1,7606	26,6340
fsh_60797.wav	82,1750	97,1940	2,8061	19,0550
fsh_60862.wav	72,8030	96,2930	3,7069	34,0200
fsh_60885.wav	83,1920	96,5810	3,4186	16,7800
fsh_61039.wav	83,4150	97,1860	2,8137	17,1410
fsh_61192.wav	75,8060	95,8920	4,1081	27,8470
sw_45097.wav	70,9440	95,7180	4,2820	37,4920
sw_45142.wav	82,6750	94,0660	5,9345	14,9480
sw_45355.wav	67,8990	93,5240	6,4757	41,2890
sw_45454.wav	80,5480	93,5550	6,4454	17,3790
sw_45586.wav	77,1180	94,4400	5,5599	24,1270
sw_45654.wav	86,3270	97,2680	2,7318	13,2050
sw_45713.wav	81,5660	93,6870	6,3128	15,9970
sw_45727.wav	72,9360	95,2910	4,7085	32,7650
sw_45819.wav	70,3850	95,1600	4,8401	37,2690
sw_46140.wav	80,2310	94,1280	5,8721	18,6890
sw_46412.wav	83,4760	94,9750	5,0250	14,5690
sw_46512.wav	89,4980	96,3090	3,6910	8,0219
sw_46615.wav	85,1540	95,3270	4,6731	12,6270
sw_46677.wav	86,3720	94,6410	5,3588	10,1580
sw_46789.wav	82,2100	94,4500	5,5502	15,9680
sw_46868.wav	80,0130	95,8000	4,1999	21,0600
sw_47346.wav	87,4260	96,2960	3,7037	10,6840
sw_47411.wav	82,7620	92,6200	7,3800	13,1510

Tabella 6.12: Risultati sperimentali SAE con babble noise e $SNR = 0$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	82,9940	97,8730	2,1272	19,3820
fsh_60354.wav	80,9630	97,9970	2,0030	25,0180
fsh_60416.wav	87,3910	98,9270	1,0733	13,6490
fsh_60463.wav	86,0470	97,1570	2,8425	13,7490
fsh_60493.wav	83,4620	97,7600	2,2400	18,5430
fsh_60549.wav	86,0510	98,8560	1,1439	15,5180
fsh_60571.wav	88,4690	97,2300	2,7702	10,9430
fsh_60593.wav	82,8810	98,4230	1,5768	20,3260
fsh_60627.wav	75,3380	96,3270	3,6731	32,4980
fsh_60648.wav	83,6360	98,3320	1,6683	18,8820
fsh_60650.wav	88,1910	98,8170	1,1829	12,7710
fsh_60720.wav	85,0640	99,1260	0,8737	17,1010
fsh_60732.wav	81,0830	97,4060	2,5938	21,9900
fsh_60797.wav	86,4770	98,9200	1,0803	14,9110
fsh_60862.wav	81,3210	97,4020	2,5978	21,7610
fsh_60885.wav	89,5260	98,8880	1,1122	10,7400
fsh_61039.wav	88,8150	98,4870	1,5126	11,4880
fsh_61192.wav	78,5280	95,7810	4,2187	25,5530
sw_45097.wav	82,3220	98,8050	1,1947	22,1470
sw_45142.wav	88,0670	99,3370	0,6627	13,1380
sw_45355.wav	79,9330	97,2880	2,7124	25,6850
sw_45454.wav	85,9140	96,3030	3,6965	12,9830
sw_45586.wav	82,3300	98,6140	1,3863	21,4300
sw_45654.wav	90,3570	99,1900	0,8099	10,0240
sw_45713.wav	88,1390	98,3400	1,6603	12,1420
sw_45727.wav	80,3870	97,3090	2,6913	24,2820
sw_45819.wav	76,6070	98,3290	1,6707	30,5200
sw_46140.wav	85,5100	98,6850	1,3147	16,1050
sw_46412.wav	90,7160	99,7040	0,2958	10,0060
sw_46512.wav	91,2760	99,7210	0,2792	9,3498
sw_46615.wav	90,9580	99,2870	0,7126	9,3888
sw_46677.wav	90,1970	97,4490	2,5511	8,4796
sw_46789.wav	88,5110	99,3290	0,6711	12,5170
sw_46868.wav	84,0760	98,8780	1,1218	18,5530
sw_47346.wav	90,0870	98,4770	1,5230	9,6378
sw_47411.wav	90,0250	99,3560	0,6442	10,7180

Tabella 6.13: Risultati sperimentali G.729B con babble noise e $SNR = 0$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	79,7770	97,3000	2,6999	22,8390
fsh_60354.wav	73,9820	97,4230	2,5772	33,0230
fsh_60416.wav	81,1290	96,1820	3,8176	19,6320
fsh_60463.wav	84,1580	97,7160	2,2843	16,6230
fsh_60493.wav	77,7100	98,1230	1,8767	26,9130
fsh_60549.wav	84,3510	96,5130	3,4869	15,0970
fsh_60571.wav	84,8820	96,0710	3,9292	13,8600
fsh_60593.wav	76,4780	96,6720	3,3279	27,6520
fsh_60627.wav	66,2660	97,7670	2,2329	49,0380
fsh_60648.wav	76,1030	95,6780	4,3221	27,2450
fsh_60650.wav	82,5610	97,5470	2,4527	18,7790
fsh_60720.wav	81,4510	97,0980	2,9016	19,9310
fsh_60732.wav	78,0460	98,6190	1,3814	26,8610
fsh_60797.wav	82,2810	97,6280	2,3724	19,2840
fsh_60862.wav	72,9100	96,4660	3,5345	33,9890
fsh_60885.wav	83,3440	96,8350	3,1654	16,8150
fsh_61039.wav	83,8810	97,9330	2,0667	17,2130
fsh_61192.wav	75,9760	96,3240	3,6757	27,9460
sw_45097.wav	70,5440	95,5140	4,4859	37,9380
sw_45142.wav	83,1450	94,8290	5,1711	15,0610
sw_45355.wav	68,3240	94,1810	5,8192	41,0650
sw_45454.wav	81,5030	94,4270	5,5731	16,9620
sw_45586.wav	78,0200	95,4010	4,5986	23,7200
sw_45654.wav	86,7480	97,8430	2,1567	13,2260
sw_45713.wav	81,6670	93,9210	6,0790	16,0820
sw_45727.wav	73,1480	95,4880	4,5123	32,6090
sw_45819.wav	71,0200	95,9380	4,0622	36,8770
sw_46140.wav	80,4400	94,4760	5,5238	18,7250
sw_46412.wav	83,6640	95,2610	4,7392	14,6000
sw_46512.wav	89,3080	96,4470	3,5534	8,3670
sw_46615.wav	85,7830	96,0820	3,9179	12,5770
sw_46677.wav	86,9370	95,2620	4,7378	10,0920
sw_46789.wav	82,3970	95,0050	4,9952	16,2310
sw_46868.wav	80,9850	97,4230	2,5767	21,1580
sw_47346.wav	87,7340	96,6850	3,3150	10,6880
sw_47411.wav	83,5400	93,5040	6,4963	13,0270

Tabella 6.14: Risultati sperimentali SAE con babble noise e $SNR = -5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	82,4870	97,7500	2,2497	19,9240
fsh_60354.wav	78,7560	98,1220	1,8784	28,2460
fsh_60416.wav	86,9150	98,4040	1,5965	13,7490
fsh_60463.wav	84,9070	95,8540	4,1457	13,9570
fsh_60493.wav	82,5810	96,4220	3,5775	18,5390
fsh_60549.wav	86,0410	98,7880	1,2116	15,4720
fsh_60571.wav	87,2430	96,4800	3,5197	11,7290
fsh_60593.wav	81,9310	98,2930	1,7071	21,4400
fsh_60627.wav	73,9280	95,9520	4,0485	34,3190
fsh_60648.wav	82,6010	98,4260	1,5745	20,2610
fsh_60650.wav	87,6480	98,4800	1,5197	13,1170
fsh_60720.wav	84,3250	98,2810	1,7193	17,2850
fsh_60732.wav	79,2370	95,6080	4,3922	22,9740
fsh_60797.wav	86,2840	98,4980	1,5016	14,7770
fsh_60862.wav	81,5380	97,0990	2,9007	21,2320
fsh_60885.wav	89,5630	98,6600	1,3400	10,4920
fsh_61039.wav	88,5300	97,6210	2,3791	11,0370
fsh_61192.wav	77,0960	95,6260	4,3738	27,4340
sw_45097.wav	81,7720	98,4660	1,5342	22,6070
sw_45142.wav	88,8340	98,6630	1,3369	11,6510
sw_45355.wav	78,9440	97,3200	2,6803	27,1130
sw_45454.wav	85,5030	95,7890	4,2113	13,0100
sw_45586.wav	81,7730	97,9750	2,0252	21,6430
sw_45654.wav	89,9790	98,6260	1,3743	9,9365
sw_45713.wav	87,9020	97,7960	2,2040	11,9320
sw_45727.wav	78,8820	97,3130	2,6869	26,3600
sw_45819.wav	74,8400	97,5590	2,4411	32,4230
sw_46140.wav	84,7630	98,2730	1,7270	16,6550
sw_46412.wav	90,7030	99,4010	0,5987	9,7464
sw_46512.wav	91,1010	99,4630	0,5367	9,3083
sw_46615.wav	90,8990	98,8550	1,1452	9,0585
sw_46677.wav	90,1580	97,6290	2,3707	8,6931
sw_46789.wav	88,6320	98,8760	1,1235	11,9780
sw_46868.wav	83,2510	98,4220	1,5778	19,2000
sw_47346.wav	90,0770	97,7570	2,2435	8,9828
sw_47411.wav	89,7520	99,0280	0,9718	10,7350

Tabella 6.15: Risultati sperimentali G.729B con babble noise e $SNR = -5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	78,5550	95,5350	4,4652	22,9890
fsh_60354.wav	72,3070	94,5910	5,4092	33,3830
fsh_60416.wav	77,9340	92,4930	7,5070	20,5820
fsh_60463.wav	81,8360	95,2280	4,7716	17,2970
fsh_60493.wav	77,5830	97,0510	2,9491	26,2710
fsh_60549.wav	82,9920	94,5950	5,4047	15,0340
fsh_60571.wav	82,4790	93,6450	6,3546	14,5560
fsh_60593.wav	74,6210	93,4530	6,5466	27,7000
fsh_60627.wav	65,7950	96,1720	3,8278	48,9220
fsh_60648.wav	74,7310	93,2620	6,7383	27,2310
fsh_60650.wav	80,3730	94,7110	5,2887	19,0990
fsh_60720.wav	80,8030	95,6480	4,3523	19,5290
fsh_60732.wav	76,8720	96,2080	3,7920	26,5770
fsh_60797.wav	81,4930	96,0710	3,9286	18,9590
fsh_60862.wav	71,9250	95,0000	5,0000	34,3620
fsh_60885.wav	82,7590	95,7960	4,2036	16,6270
fsh_61039.wav	83,0120	96,8130	3,1873	17,3100
fsh_61192.wav	74,6390	93,5950	6,4054	27,6640
sw_45097.wav	70,7120	95,1350	4,8645	37,4460
sw_45142.wav	82,0990	93,3510	6,6486	15,0090
sw_45355.wav	67,9410	92,6590	7,3411	40,6760
sw_45454.wav	79,8210	92,4400	7,5600	17,3000
sw_45586.wav	75,4400	92,2060	7,7942	24,5420
sw_45654.wav	85,4020	96,3340	3,6664	13,4580
sw_45713.wav	80,7160	92,7520	7,2481	16,2090
sw_45727.wav	71,7540	92,7690	7,2309	32,6280
sw_45819.wav	69,9200	93,3160	6,6840	36,7090
sw_46140.wav	77,5050	90,8930	9,1067	19,3540
sw_46412.wav	82,9330	94,3080	5,6918	14,6210
sw_46512.wav	88,5070	95,2090	4,7914	8,0905
sw_46615.wav	84,8400	94,7840	5,2160	12,5000
sw_46677.wav	85,6190	93,5600	6,4397	10,0050
sw_46789.wav	81,2530	93,3640	6,6361	16,1800
sw_46868.wav	79,5900	94,7950	5,2048	20,7670
sw_47346.wav	86,9120	95,5420	4,4582	10,5770
sw_47411.wav	81,2880	90,9480	9,0518	13,3930

Tabella 6.16: Risultati sperimentali SAE con car noise e $SNR = 5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	83,0930	98,2320	1,7679	19,5520
fsh_60354.wav	82,6260	98,3520	1,6477	22,9330
fsh_60416.wav	86,9550	99,2810	0,7194	14,4650
fsh_60463.wav	87,6270	99,2640	0,7362	13,7310
fsh_60493.wav	84,9870	98,7150	1,2847	17,4290
fsh_60549.wav	86,4420	99,2550	0,7453	15,3920
fsh_60571.wav	89,1300	98,6940	1,3059	11,4860
fsh_60593.wav	83,4580	98,8650	1,1353	19,9460
fsh_60627.wav	81,6440	96,5930	3,4070	23,4260
fsh_60648.wav	-2,0000	-2,0000	-2,0000	-2,0000
fsh_60650.wav	88,1370	99,6750	0,3251	13,5850
fsh_60720.wav	85,0060	99,6010	0,3986	17,5650
fsh_60732.wav	88,2470	97,9310	2,0691	13,1220
fsh_60797.wav	85,8870	99,7500	0,2496	16,3120
fsh_60862.wav	79,7840	98,9290	1,0707	24,9480
fsh_60885.wav	89,5530	99,2830	0,7174	11,0650
fsh_61039.wav	87,9830	99,5520	0,4485	13,3920
fsh_61192.wav	80,5740	97,2080	2,7918	23,8330
sw_45097.wav	84,8660	99,0330	0,9669	18,9720
sw_45142.wav	87,9000	99,8390	0,1609	13,7660
sw_45355.wav	75,1970	99,4410	0,5590	33,8690
sw_45454.wav	87,3970	99,2650	0,7348	13,8620
sw_45586.wav	82,8340	99,1020	0,8978	21,1690
sw_45654.wav	90,3410	98,7830	1,2167	9,6720
sw_45713.wav	87,4370	99,7720	0,2284	14,2100
sw_45727.wav	82,1150	97,5570	2,4435	22,0970
sw_45819.wav	79,3850	98,8980	1,1019	27,1150
sw_46140.wav	84,9500	99,7360	0,2645	17,6550
sw_46412.wav	90,5790	99,7840	0,2164	10,2300
sw_46512.wav	91,3450	99,6880	0,3119	9,2437
sw_46615.wav	90,8560	99,4550	0,5454	9,6545
sw_46677.wav	91,4530	97,9070	2,0929	7,4927
sw_46789.wav	88,0500	99,8290	0,1706	13,4830
sw_46868.wav	84,0960	99,2010	0,7990	18,7930
sw_47346.wav	90,5040	99,6610	0,3388	10,2460
sw_47411.wav	89,5090	99,7800	0,2197	11,6840

Tabella 6.17: Risultati sperimentali G.729B con car noise e $SNR = 5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	78,6600	95,7680	4,2316	23,0410
fsh_60354.wav	72,0740	94,3930	5,6075	33,5730
fsh_60416.wav	78,3170	93,0050	6,9946	20,5230
fsh_60463.wav	82,2700	95,7360	4,2640	17,2060
fsh_60493.wav	77,7100	97,3460	2,6542	26,3290
fsh_60549.wav	83,0130	94,7200	5,2802	15,1200
fsh_60571.wav	82,6710	93,8150	6,1848	14,4780
fsh_60593.wav	74,9630	93,8630	6,1375	27,5500
fsh_60627.wav	65,8370	96,5870	3,4131	49,0750
fsh_60648.wav	74,7310	93,4770	6,5235	27,3980
fsh_60650.wav	80,7160	95,1970	4,8033	19,0820
fsh_60720.wav	80,6740	95,4920	4,5078	19,5600
fsh_60732.wav	77,3630	96,7230	3,2774	26,3230
fsh_60797.wav	81,6630	96,2760	3,7245	18,9190
fsh_60862.wav	71,8390	94,8850	5,1149	34,4030
fsh_60885.wav	82,6280	95,7460	4,2542	16,7420
fsh_61039.wav	82,9270	96,7130	3,2869	17,3270
fsh_61192.wav	74,6600	93,8110	6,1892	27,8020
sw_45097.wav	70,7120	95,2520	4,7480	37,5230
sw_45142.wav	82,1210	93,3760	6,6240	15,0050
sw_45355.wav	67,7710	92,5690	7,4306	40,8770
sw_45454.wav	80,0710	92,7310	7,2692	17,2460
sw_45586.wav	76,0070	93,0110	6,9888	24,4410
sw_45654.wav	85,1910	96,1900	3,8102	13,5770
sw_45713.wav	80,6350	92,6580	7,3416	16,2250
sw_45727.wav	71,8390	92,9370	7,0628	32,6300
sw_45819.wav	69,4540	93,1140	6,8856	37,2520
sw_46140.wav	77,8830	91,3410	8,6589	19,2590
sw_46412.wav	82,9130	94,2840	5,7156	14,6250
sw_46512.wav	88,5280	95,2320	4,7685	8,0886
sw_46615.wav	84,4410	94,3360	5,6644	12,5590
sw_46677.wav	85,9740	93,9510	6,0488	9,9633
sw_46789.wav	81,3980	93,5330	6,4672	16,1510
sw_46868.wav	79,8860	95,0790	4,9214	20,6230
sw_47346.wav	86,9320	95,5650	4,4353	10,5740
sw_47411.wav	81,5410	91,2350	8,7652	13,3510

Tabella 6.18: Risultati sperimentali SAE con car noise e $SNR = 0$ dB

Nome file	*sim*	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	83,6850	98,4280	1,5719	18,9650
fsh_60354.wav	82,3340	98,5600	1,4400	23,5070
fsh_60416.wav	87,0080	99,1420	0,8579	14,2830
fsh_60463.wav	87,9620	99,1460	0,8543	13,2350
fsh_60493.wav	84,8880	98,5120	1,4879	17,3820
fsh_60549.wav	86,0920	99,3780	0,6218	15,9140
fsh_60571.wav	89,9370	98,3930	1,6072	10,2600
fsh_60593.wav	84,0320	99,0460	0,9545	19,3620
fsh_60627.wav	81,3760	96,5740	3,4260	23,8040
fsh_60648.wav	-2,0000	-2,0000	-2,0000	-2,0000
fsh_60650.wav	88,3450	99,3420	0,6580	13,0510
fsh_60720.wav	84,9420	99,6580	0,3423	17,6880
fsh_60732.wav	87,8700	97,6940	2,3061	13,4010
fsh_60797.wav	85,6340	99,7110	0,2886	16,5770
fsh_60862.wav	79,9080	98,9170	1,0835	24,7790
fsh_60885.wav	89,2280	99,2670	0,7326	11,4180
fsh_61039.wav	87,7420	99,5210	0,4789	13,6440
fsh_61192.wav	80,0170	97,1510	2,8494	24,5540
sw_45097.wav	84,1820	98,9990	1,0013	19,8460
sw_45142.wav	87,7640	99,8310	0,1685	13,9170
sw_45355.wav	76,2320	99,2760	0,7239	32,3200
sw_45454.wav	87,2800	99,1870	0,8132	13,9300
sw_45586.wav	82,3950	98,9850	1,0147	21,6410
sw_45654.wav	89,9400	98,9260	1,0738	10,2550
sw_45713.wav	87,5040	99,6770	0,3226	14,0520
sw_45727.wav	82,6230	97,4590	2,5408	21,3200
sw_45819.wav	78,5880	99,0010	0,9988	28,2810
sw_46140.wav	85,4540	99,6230	0,3773	16,9610
sw_46412.wav	90,5660	99,7580	0,2417	10,2210
sw_46512.wav	91,2390	99,7100	0,2901	9,3799
sw_46615.wav	90,8950	99,3420	0,6581	9,5081
sw_46677.wav	91,3520	98,0190	1,9810	7,7124
sw_46789.wav	87,9560	99,8000	0,2002	13,5650
sw_46868.wav	84,1680	99,2740	0,7264	18,7630
sw_47346.wav	90,4980	99,5220	0,4779	10,1280
sw_47411.wav	89,5290	99,6980	0,3016	11,5890

Tabella 6.19: Risultati sperimentali G.729B con car noise e $SNR = 0$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	79,1030	96,2880	3,7124	22,8900
fsh_60354.wav	72,4130	95,0160	4,9844	33,5320
fsh_60416.wav	78,3390	93,0310	6,9690	20,5180
fsh_60463.wav	82,7690	96,3200	3,6802	17,1010
fsh_60493.wav	77,7100	97,5600	2,4397	26,4910
fsh_60549.wav	83,4230	95,1930	4,8070	15,0440
fsh_60571.wav	83,0320	94,2760	5,7240	14,4580
fsh_60593.wav	75,4540	94,5720	5,4283	27,4300
fsh_60627.wav	66,5670	97,4800	2,5199	48,4290
fsh_60648.wav	75,1530	94,0130	5,9866	27,2420
fsh_60650.wav	81,1670	95,7330	4,2667	18,9750
fsh_60720.wav	80,5870	95,3890	4,6114	19,5820
fsh_60732.wav	77,4060	97,0750	2,9252	26,5350
fsh_60797.wav	82,0470	96,8880	3,1122	18,9570
fsh_60862.wav	72,0960	95,0000	5,0000	34,1200
fsh_60885.wav	82,4980	95,6700	4,3302	16,8340
fsh_61039.wav	83,0540	96,9620	3,0378	17,3860
fsh_61192.wav	74,6600	94,4050	5,5946	28,2570
sw_45097.wav	70,6280	95,5140	4,4859	37,8160
sw_45142.wav	82,0350	93,2780	6,7225	15,0210
sw_45355.wav	68,1960	93,2560	6,7443	40,6720
sw_45454.wav	80,2370	92,9250	7,0754	17,2100
sw_45586.wav	75,7970	92,8550	7,1447	24,5940
sw_45654.wav	85,1070	96,1900	3,8102	13,6770
sw_45713.wav	80,7570	92,7750	7,2247	16,1790
sw_45727.wav	71,8390	93,1890	6,8105	32,8120
sw_45819.wav	69,3900	93,4890	6,5111	37,5960
sw_46140.wav	78,6160	92,2120	7,7880	19,0770
sw_46412.wav	82,8080	94,1650	5,8347	14,6430
sw_46512.wav	88,5910	95,3000	4,6997	8,0828
sw_46615.wav	84,8610	94,9020	5,0979	12,5840
sw_46677.wav	86,0160	93,9970	6,0028	9,9584
sw_46789.wav	81,8140	94,0150	5,9846	16,0680
sw_46868.wav	80,0130	95,3100	4,6895	20,6540
sw_47346.wav	87,0760	95,6560	4,3439	10,4920
sw_47411.wav	81,4780	91,2590	8,7413	13,4520

Tabella 6.20: Risultati sperimentali SAE con car noise e $SNR = -5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	84,0050	98,4690	1,5311	18,5970
fsh_60354.wav	82,2090	98,5230	1,4769	23,6570
fsh_60416.wav	87,1950	99,1270	0,8732	14,0520
fsh_60463.wav	87,7390	98,5870	1,4134	13,0070
fsh_60493.wav	84,4090	98,3860	1,6140	17,8750
fsh_60549.wav	86,3010	99,2270	0,7732	15,5370
fsh_60571.wav	89,8710	98,2460	1,7540	10,2050
fsh_60593.wav	83,8220	98,7090	1,2908	19,3560
fsh_60627.wav	79,8580	96,9970	3,0031	26,3560
fsh_60648.wav	-2,0000	-2,0000	-2,0000	-2,0000
fsh_60650.wav	89,1020	99,3810	0,6189	12,1940
fsh_60720.wav	85,1610	99,4560	0,5436	17,2580
fsh_60732.wav	86,3630	97,8120	2,1876	15,4610
fsh_60797.wav	85,5110	99,5830	0,4173	16,6140
fsh_60862.wav	79,8410	98,8480	1,1517	24,8130
fsh_60885.wav	89,1260	99,1500	0,8503	11,4280
fsh_61039.wav	87,7380	99,2890	0,7107	13,4460
fsh_61192.wav	79,9770	96,6630	3,3369	24,2290
sw_45097.wav	83,3120	99,0760	0,9239	21,0540
sw_45142.wav	87,6900	99,6630	0,3371	13,8560
sw_45355.wav	77,5100	98,6300	1,3699	30,0880
sw_45454.wav	87,0690	98,8140	1,1862	13,8500
sw_45586.wav	82,0770	98,7390	1,2611	21,8560
sw_45654.wav	90,0290	99,2600	0,7403	10,4560
sw_45713.wav	87,8510	99,4490	0,5510	13,4540
sw_45727.wav	82,9860	97,3180	2,6825	20,7050
sw_45819.wav	78,2700	98,8260	1,1735	28,5900
sw_46140.wav	85,5750	99,5220	0,4784	16,7320
sw_46412.wav	90,8170	99,7800	0,2200	9,9624
sw_46512.wav	91,2100	99,8220	0,1777	9,5146
sw_46615.wav	90,9250	99,3240	0,6762	9,4586
sw_46677.wav	91,2500	97,7920	2,2083	7,6123
sw_46789.wav	88,2480	99,7780	0,2225	13,2110
sw_46868.wav	84,1920	99,0320	0,9685	18,5360
sw_47346.wav	90,1960	99,3510	0,6491	10,3110
sw_47411.wav	89,9200	99,4600	0,5399	10,9310

Tabella 6.21: Risultati sperimentali G.729B con car noise e $SNR = -5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	78,1970	95,4830	4,5171	23,4090
fsh_60354.wav	72,3490	94,6190	5,3809	33,3430
fsh_60416.wav	78,3390	92,9800	7,0202	20,4740
fsh_60463.wav	83,3770	96,5990	3,4010	16,6050
fsh_60493.wav	77,3720	97,0780	2,9223	26,5670
fsh_60549.wav	83,4880	95,2930	4,7073	15,0550
fsh_60571.wav	82,9680	94,1550	5,8453	14,4260
fsh_60593.wav	74,9630	93,7260	6,2739	27,4450
fsh_60627.wav	65,9450	96,5230	3,4769	48,8760
fsh_60648.wav	74,9630	93,4500	6,5503	27,0610
fsh_60650.wav	80,2450	94,7880	5,2121	19,3260
fsh_60720.wav	81,1060	96,2950	3,7047	19,6930
fsh_60732.wav	76,9150	96,3430	3,6566	26,6240
fsh_60797.wav	81,4710	96,1480	3,8520	19,0500
fsh_60862.wav	72,8680	95,6320	4,3678	33,4740
fsh_60885.wav	82,8240	95,7960	4,2036	16,5480
fsh_61039.wav	82,5030	96,2650	3,7351	17,4600
fsh_61192.wav	74,5120	94,0000	6,0000	28,1480
sw_45097.wav	70,1010	94,0580	5,9423	37,6280
sw_45142.wav	82,2700	93,5980	6,4024	15,0220
sw_45355.wav	67,6860	92,1810	7,8186	40,7580
sw_45454.wav	80,4030	93,0700	6,9300	17,1310
sw_45586.wav	76,0280	92,7250	7,2746	24,1800
sw_45654.wav	85,8860	96,8850	3,1153	13,3810
sw_45713.wav	80,9790	93,0560	6,9441	16,1560
sw_45727.wav	72,5990	93,9460	6,0538	32,2790
sw_45819.wav	70,4060	93,8350	6,1654	36,3520
sw_46140.wav	79,0780	92,8590	7,1411	19,0510
sw_46412.wav	83,1210	94,4510	5,5489	14,5230
sw_46512.wav	89,0130	95,8050	4,1953	8,0881
sw_46615.wav	84,4620	94,3830	5,6172	12,5780
sw_46677.wav	85,9330	93,9050	6,0948	9,9682
sw_46789.wav	81,3360	93,5330	6,4672	16,2280
sw_46868.wav	80,2660	95,5940	4,4061	20,5660
sw_47346.wav	87,1790	95,8850	4,1152	10,5870
sw_47411.wav	82,1510	92,1180	7,8815	13,4300

Tabella 6.22: Risultati sperimentali SAE con factory noise e $SNR = 5$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	84,5050	97,2600	2,7397	16,9470
fsh_60354.wav	83,1730	97,6320	2,3677	21,5850
fsh_60416.wav	88,0500	97,7340	2,2658	11,8160
fsh_60463.wav	86,9960	95,0430	4,9567	10,6710
fsh_60493.wav	85,8980	94,9470	5,0533	12,9690
fsh_60549.wav	87,1710	98,2660	1,7338	13,6650
fsh_60571.wav	89,8410	97,1290	2,8706	9,2084
fsh_60593.wav	86,1260	98,0280	1,9720	15,8320
fsh_60627.wav	79,6470	94,2550	5,7448	24,5310
fsh_60648.wav	86,0340	98,1440	1,8559	15,7060
fsh_60650.wav	89,7990	98,4920	1,5080	10,5740
fsh_60720.wav	86,6560	98,1080	1,8924	14,3030
fsh_60732.wav	87,0630	95,8830	4,1171	12,8330
fsh_60797.wav	86,1770	98,4670	1,5328	14,8770
fsh_60862.wav	81,6690	97,4410	2,5594	21,3370
fsh_60885.wav	89,0650	98,5230	1,4766	10,9340
fsh_61039.wav	88,2290	98,2900	1,7102	11,9940
fsh_61192.wav	80,8890	94,8060	5,1937	21,4590
sw_45097.wav	83,9410	98,4310	1,5685	19,7030
sw_45142.wav	88,2970	99,3260	0,6742	12,8620
sw_45355.wav	79,9360	97,7730	2,2267	26,0500
sw_45454.wav	86,9940	96,8850	3,1146	12,2240
sw_45586.wav	83,0640	97,8410	2,1588	19,8540
sw_45654.wav	90,8800	98,3400	1,6602	8,6535
sw_45713.wav	88,4270	98,6330	1,3666	12,0700
sw_45727.wav	83,6980	95,6970	4,3026	18,3630
sw_45819.wav	78,4920	97,8590	2,1410	27,5770
sw_46140.wav	85,9260	98,2960	1,7037	15,2700
sw_46412.wav	90,3440	99,1920	0,8079	9,9560
sw_46512.wav	91,6150	98,9520	1,0481	8,2680
sw_46615.wav	90,7120	98,3350	1,6652	8,7884
sw_46677.wav	90,7300	96,9540	3,0455	7,4063
sw_46789.wav	88,3130	99,0510	0,9492	12,5000
sw_46868.wav	85,6080	97,9860	2,0136	15,9010
sw_47346.wav	90,2700	98,8340	1,1663	9,7582
sw_47411.wav	91,5380	98,5030	1,4968	8,2064

Tabella 6.23: Risultati sperimentali G.729B con factory noise e $SNR = 5$ dBB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	78,5760	95,8460	4,1537	23,2120
fsh_60354.wav	72,2860	94,7610	5,2393	33,5330
fsh_60416.wav	78,5090	93,2870	6,7128	20,5160
fsh_60463.wav	83,2030	96,5990	3,4010	16,8160
fsh_60493.wav	77,4980	97,0510	2,9491	26,3810
fsh_60549.wav	83,3590	95,2680	4,7323	15,1900
fsh_60571.wav	83,0320	94,3000	5,6997	14,4800
fsh_60593.wav	75,0270	93,9440	6,0556	27,5260
fsh_60627.wav	66,1160	96,6830	3,3174	48,6970
fsh_60648.wav	74,5200	93,3150	6,6846	27,5600
fsh_60650.wav	80,2020	94,7880	5,2121	19,3800
fsh_60720.wav	81,1490	96,3470	3,6528	19,6830
fsh_60732.wav	77,3200	96,7770	3,2232	26,4200
fsh_60797.wav	81,0020	95,8160	4,1837	19,3560
fsh_60862.wav	73,1890	96,0920	3,9080	33,3430
fsh_60885.wav	82,7800	95,8470	4,1530	16,6450
fsh_61039.wav	82,7150	96,3150	3,6853	17,2440
fsh_61192.wav	74,7880	94,4860	5,5135	28,1460
sw_45097.wav	69,6170	94,1450	5,8549	38,3970
sw_45142.wav	82,5900	93,9670	6,0330	14,9630
sw_45355.wav	67,9410	92,6290	7,3709	40,6570
sw_45454.wav	80,7560	93,5550	6,4454	17,1200
sw_45586.wav	76,5730	93,2970	6,7030	23,9210
sw_45654.wav	85,8220	97,0050	2,9954	13,5620
sw_45713.wav	81,1210	93,1730	6,8272	16,0850
sw_45727.wav	73,0630	94,5350	5,4652	32,0490
sw_45819.wav	70,6390	94,3820	5,6180	36,3860
sw_46140.wav	79,1400	93,1330	6,8674	19,2090
sw_46412.wav	83,2670	94,6890	5,3108	14,5620
sw_46512.wav	89,1610	96,0110	3,9890	8,1184
sw_46615.wav	84,4620	94,2410	5,7588	12,4470
sw_46677.wav	86,1210	94,0200	5,9798	9,8581
sw_46789.wav	81,6480	93,8710	6,1293	16,1440
sw_46868.wav	80,8160	96,2120	3,7877	20,3800
sw_47346.wav	86,7470	95,4960	4,5039	10,7250
sw_47411.wav	82,3410	92,3810	7,6188	13,4440

Tabella 6.24: Risultati sperimentali SAE con factory noise e $SNR = 0$ dB

Nome file	*sim*	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	84,9160	96,4760	3,5236	15,7430
fsh_60354.wav	83,2460	97,0790	2,9215	21,0330
fsh_60416.wav	88,3030	96,1610	3,8392	10,0690
fsh_60463.wav	83,8090	89,9130	10,0870	9,6900
fsh_60493.wav	83,5280	90,6290	9,3707	12,0440
fsh_60549.wav	87,3090	97,0980	2,9016	12,4580
fsh_60571.wav	89,1930	95,5260	4,4740	8,4732
fsh_60593.wav	86,4400	97,2460	2,7541	14,7480
fsh_60627.wav	78,3770	91,5990	8,4010	24,3090
fsh_60648.wav	85,3550	97,0510	2,9491	15,6220
fsh_60650.wav	89,8500	97,2820	2,7183	9,4013
fsh_60720.wav	86,3730	96,6220	3,3782	13,3350
fsh_60732.wav	85,2700	92,6670	7,3330	12,2650
fsh_60797.wav	86,1200	97,3750	2,6248	13,9900
fsh_60862.wav	82,1340	96,4170	3,5832	19,8870
fsh_60885.wav	88,7600	97,3620	2,6382	10,2230
fsh_61039.wav	88,3980	96,8680	3,1315	10,5030
fsh_61192.wav	79,8710	91,8020	8,1982	20,3710
sw_45097.wav	84,6950	97,6190	2,3807	18,0270
sw_45142.wav	88,1700	98,5410	1,4594	12,3150
sw_45355.wav	81,1380	96,8840	3,1156	23,6590
sw_45454.wav	86,3260	95,6060	4,3940	11,8530
sw_45586.wav	83,5160	96,9520	3,0483	18,5240
sw_45654.wav	90,4760	97,0130	2,9869	7,8690
sw_45713.wav	87,6960	97,6150	2,3853	12,0100
sw_45727.wav	83,6950	94,9090	5,0905	17,6900
sw_45819.wav	78,1410	95,9330	4,0670	26,6180
sw_46140.wav	85,4670	97,4060	2,5944	15,0550
sw_46412.wav	90,3150	98,6080	1,3922	9,4554
sw_46512.wav	91,6510	98,2230	1,7770	7,5466
sw_46615.wav	90,3610	97,1860	2,8140	8,1104
sw_46677.wav	89,9780	95,8610	4,1389	7,2160
sw_46789.wav	89,1060	98,8510	1,1495	11,4070
sw_46868.wav	85,5090	97,3930	2,6068	15,5130
sw_47346.wav	89,7620	97,3140	2,6857	8,9283
sw_47411.wav	90,9130	97,5800	2,4202	8,0627

Tabella 6.25: Risultati sperimentali G.729B con factory noise e $SNR = 0$ dB

Nome file	sim	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	78,8290	96,1060	3,8941	23,0960
fsh_60354.wav	72,1590	94,6760	5,3243	33,6520
fsh_60416.wav	78,8500	93,6200	6,3797	20,3610
fsh_60463.wav	83,4640	96,6500	3,3503	16,5440
fsh_60493.wav	77,6040	96,9710	3,0295	26,1820
fsh_60549.wav	83,3150	95,1930	4,8070	15,1750
fsh_60571.wav	83,2450	94,6640	5,3359	14,5530
fsh_60593.wav	75,4110	94,6540	5,3464	27,5500
fsh_60627.wav	65,6230	96,2680	3,7321	49,2380
fsh_60648.wav	74,7100	93,8520	6,1477	27,7170
fsh_60650.wav	80,6090	95,1710	4,8288	19,1950
fsh_60720.wav	81,1270	96,2690	3,7306	19,6450
fsh_60732.wav	77,1500	96,5600	3,4399	26,4800
fsh_60797.wav	80,5760	95,3570	4,6429	19,5020
fsh_60862.wav	73,1460	96,3220	3,6782	33,5620
fsh_60885.wav	83,1490	96,2270	3,7731	16,5260
fsh_61039.wav	83,0750	96,7380	3,2620	17,1690
fsh_61192.wav	75,3180	95,2700	4,7297	28,0280
sw_45097.wav	70,2910	94,9020	5,0976	37,9070
sw_45142.wav	82,4190	93,7950	6,2054	15,0170
sw_45355.wav	67,8990	92,7190	7,2814	40,7790
sw_45454.wav	81,3160	94,1120	5,8881	16,9160
sw_45586.wav	76,5520	93,6090	6,3913	24,2020
sw_45654.wav	85,7800	97,1240	2,8756	13,7180
sw_45713.wav	80,6960	92,7050	7,2948	16,1920
sw_45727.wav	72,9790	94,7870	5,2130	32,3480
sw_45819.wav	70,4700	94,3820	5,6180	36,6300
sw_46140.wav	79,3080	93,5060	6,4942	19,3190
sw_46412.wav	83,3090	94,7370	5,2632	14,5550
sw_46512.wav	88,8650	96,0110	3,9890	8,4527
sw_46615.wav	84,1900	94,2410	5,7588	12,7720
sw_46677.wav	86,3720	94,3650	5,6348	9,8952
sw_46789.wav	81,8560	94,2080	5,7915	16,1890
sw_46868.wav	80,6040	96,2380	3,7619	20,6690
sw_47346.wav	87,4050	96,0450	3,9552	10,4740
sw_47411.wav	82,4670	92,5720	7,4278	13,4670

Tabella 6.26: Risultati sperimentali SAE con factory noise e $SNR = -5$ dB

Nome file	*sim*	P_{ok}	P_{miss}	P_{false}
fsh_60262.wav	83,7020	93,7450	6,2551	14,8950
fsh_60354.wav	84,1300	96,0170	3,9830	18,8760
fsh_60416.wav	87,0620	93,0410	6,9590	8,5959
fsh_60463.wav	78,1320	82,6220	17,3780	9,6969
fsh_60493.wav	78,3720	83,6120	16,3880	12,2570
fsh_60549.wav	86,7120	95,2770	4,7230	11,5250
fsh_60571.wav	86,5050	92,2650	7,7348	8,6261
fsh_60593.wav	86,5530	96,0140	3,9860	13,5060
fsh_60627.wav	74,5680	84,5380	15,4620	24,3770
fsh_60648.wav	84,7580	95,4230	4,5766	14,9570
fsh_60650.wav	88,7940	95,3120	4,6884	8,8230
fsh_60720.wav	84,8610	93,6830	6,3174	12,5410
fsh_60732.wav	81,3800	87,5220	12,4780	12,7490
fsh_60797.wav	85,8100	95,1760	4,8245	12,3840
fsh_60862.wav	82,4360	94,6470	5,3534	17,9830
fsh_60885.wav	87,6660	95,4300	4,5703	9,6897
fsh_61039.wav	88,2190	94,7780	5,2218	8,7453
fsh_61192.wav	77,0630	87,2820	12,7180	20,5470
sw_45097.wav	84,6950	96,3040	3,6957	16,9080
sw_45142.wav	88,9700	97,2920	2,7082	10,2440
sw_45355.wav	81,4030	95,5420	4,4580	22,2030
sw_45454.wav	84,7240	93,2150	6,7850	11,5690
sw_45586.wav	83,7290	95,6160	4,3845	17,1020
sw_45654.wav	89,8640	95,6090	4,3905	7,2294
sw_45713.wav	87,3080	95,3380	4,6618	10,3760
sw_45727.wav	83,2730	93,7010	6,2990	17,2340
sw_45819.wav	77,3810	92,3720	7,6279	24,9040
sw_46140.wav	85,0250	95,4840	4,5159	13,8950
sw_46412.wav	89,9990	97,9620	2,0379	9,2158
sw_46512.wav	91,1930	97,2220	2,7779	7,1133
sw_46615.wav	89,0540	95,0950	4,9046	7,6158
sw_46677.wav	89,5720	94,8400	5,1600	6,6887
sw_46789.wav	90,0000	97,8900	2,1098	9,4962
sw_46868.wav	85,0660	94,9440	5,0563	13,9030
sw_47346.wav	89,2420	94,8430	5,1575	7,1641
sw_47411.wav	90,6470	96,0790	3,9208	6,9408

Tabella 6.27: Risultati sperimentali G.729B con factory noise e $SNR = -5$ dB

Capitolo 7

Conclusioni

In questo lavoro di tesi, si è voluto mettere a confronto alcuni algoritmi di Voice Activity Detection con lo standard di mercato $G.729B$, al fine di ottenere una buona discriminazione tra parlato e non parlato nelle conversazioni telefoniche sporcate da rumore. Dopo aver eseguito una scrematura preliminare, si è scelto di mettere a confronto l'algoritmo SAE con l'algoritmo $G.729B$. Dai risultati ottenuti dal confronto, si possono notare i seguenti aspetti:

- L'algoritmo basato sulla SAE si comporta, mediamente, meglio dell'algoritmo $G.729B$ per quanto riguarda il parlato sporcato da rumore bianco, soprattutto quando si ha un SNR basso.
- L'algoritmo basato sulla SAE si comporta, mediamente, peggio dell'algoritmo $G.729B$ per quanto riguarda il parlato sporcato dai rumori babble, car e factory. Questo può essere dovuto alla distribuzione statistica più complessa di tali tipi di rumore rispetto al rumore bianco, nonché al tipo di sogliatura utilizzata dall'algoritmo basato sulla SAE, che probabilmente meglio si adatta alla natura statistica di quest'ultimo.
- Si nota, tuttavia, che l'algoritmo basato sulla SAE, a differenza del $G.729B$, tende a non degradare le proprie prestazioni al diminuire del rapporto SNR. Questo aspetto sta ad indicare la *robustezza al rumore* dell'algoritmo stesso.

Sicuramente, l'algoritmo basato sulla SAE può essere una buona alternativa allo standard $G.729B$ in presenza di rumore bianco, ma soffre di un difetto prestazionale quando si ha a che fare con rumori di natura statistica più complessa. È sicuramente auspicabile ottenere dei futuri miglioramenti all'algoritmo, attraverso un calcolo delle features più efficace e/o attraverso un miglioramento dell'algoritmo di sogliatura adattativa.

Capitolo 8

Appendice

Codice Matlab.

Funzione di lettura dei file:

```
function LeggiFile;
%funzione che legge i file SPH, li converte in WAV e ricava un array
%booleano sul parlato etichettato nei file del database di riferimento
close all; clc;
%legge tutti i nomi dei file SPH dalla directory delle conversazioni
audiofiles=dir('F:\Documenti\Uni\Database\data\audio\eval03\english\cts');
%legge la cartella dei file MDTM per marcare gli intervalli
MDTMfiles=dir('F:\Documenti\Uni\Database\data\references\eval03\english\cts');
%file UEM di riferimento per le etichette dei file
UEMref='F:\Documenti\Uni\Database\data\indices\
        expt_03_stt1x_eval03_eng_cts_spch_expt_1.uem';
%aggiunge il path per la procedura di lettura degli SPH
path(path,'F:\Documenti\Uni\Tesi\FeatExtraction2');

for indref=3:length(audiofiles)
    fname=char(audiofiles(indref).name);
    SPHname=['F:\Documenti\Uni\Database\data\audio\eval03\english\cts\',fname];
    %legge 10 campioni del file SPH saltando 100 campioni, e ricava la
    %frequenza di campionamento
    [S1,FS,ffx]=readsph(SPHname,'srponl',10,100);
    fprintf('Nome file: %s \nFrequenza di campionamento: %g Hz \n',fname,FS);
    %nome del file utile per la lettura del file UEM e MDTM
    nomefile=strrep(fname,'.sph','');

    MDTMname=['F:\Documenti\Uni\Database\data\references\eval03\
              english\cts\',nomefile,'.spkreval.mdtm'];
    %se esiste il file MDTM legge il file e lo converte
```

```
if exist(MDTMname)>0
    %legge il file SPH
    [S,ls]=sig_load(SPHname,'');
    %lettura file UEM
    [nome,canale,inizio,fine]=textread(UEMref,'%s %d %f %f','headerlines',14);
    i=1;
    flag=0;
    while flag==0
        k=strfind(char(nome(i)),nomefile);
        if isempty(k)==0
            a=inizio(i);  %intervallo etichettato
            b=fine(i);
            flag=1;
        end
        i=i+1;
    end

    %numero di campioni del vettore di logic
    N=ceil(b*FS)-ceil(a*FS)+1;
    V=zeros(N,1);                %vettore per il file SPH
    [nome,canale,inizio,durata,z,x,c,m]=textread(MDTMname,
                '%s %d %f %f %s %s %s %s','headerlines',1);
    NI=length(inizio);
    for i=1:NI
%intervalli da marcare nel vettore
        start=ceil((inizio(i)-a)*FS);
        if start==0
            start=1;
        end
        finish=ceil((inizio(i)+durata(i)-a)*FS);
        for index=start:finish
            V(index)=1;
        end
    end

    x=linspace(a,b,N);
    S=S(ceil(a*FS):ceil(b*FS));    %porzione di segnale da convertire
    S=normalizza(S);     %normalizza il segnale audio

    wavename=[nomefile,'.wav'];
    %scrive la porzione di file etichettata in un file .WAV
    wavwrite(S,FS,char(wavename));
    fprintf('File .WAV generato: %s \n\n',wavename);
    save([nomefile,'.mat'],'a','b','V','S');
```

```
        %salva le variabili di inizio,fine etichettatura,il segnale e il
        %vettore booleano
    else
        fprintf('Il file MDTM per il file %s non esiste \n\n',fname);
    end
end

fname=strrep(fname,'_','-');
figure(1);
plot(x,S);
title(['Segnale ricavato dal file ',fname,'
      nell''intervallo etichettato']);
xlabel('t');
ylabel('S(t)');
figure(2);
plot(x,V);
title(['Vettore booleano del parlato per il file ',fname]);
xlabel('t');
ylabel('V(t)');
```

Funzione di VAD utilizzando VAS e BSE:

```
function [simV,sim,simF] = MyVAD(nomefile,SNR,ntype)
%funzione che esegue il VAD di un segnale audio corrotto da rumore
%attraverso i metodi di VAS ed entropia spettrale BSE
%ntype=tipo di rumore: 1 (bianco),2 (babble), 3 (factory), 4 (car)
close all;clc;

if nargin<=2
    ntype=1;
end

carica=strrep(nomefile,'.wav','.mat');
flag=0;
if exist(carica)>0
    load(carica);    %carica le variabili a,b,V,S dal file .mat
    flag=1;
else
    fprintf('Non  disponibile il file .mat per tale file\n');
end

[S,fs]=wavread(nomefile); %recupera il segnale e la frequenza di sampling
fprintf('Frequenza di campionamento: %g Hz \n',fs);

if flag==1
```

```
    if nargin==1
        %immissione de l rapporto SNR desiderato
        SNR=input('Inserire il rapporto SNR desiderato (in dB): \n');
    end
    %ricava l'ampiezza del rumore da immettere in base al SNR
    ampR=noiseamp(S,SNR,V);
    fprintf('Ampiezza del rumore da immettere sul segnale: %g \n',ampR);
    %corrompe il segnale con rumore desiderato
    [S,noise]=corrompi(S,ampR,ntype);
else
    S=normalizza(S);
    fprintf('Non sono disponibili informazioni relative
             all''entit\'a del rumore \n');
end

N=length(S);    %numero di campioni del segnale
if flag==1
    x=linspace(a,b,N);
else
    x=linspace(0,N/fs,N);
end
figure(1);
plot(x,S);
xlabel('tempo');
ylabel('S(t)');
title('Segnale d''ingresso');

Num=256;    %granularit\'a dei frame
%calcola la VAS sul segnale audio
VoiceActShape=log10(VAS(S,Num));
%calcola la soglia adattativa sulla VAS
%e costruisce il vettore booleano
threshold=soglia(VoiceActShape);
VVAS=costruisci(VoiceActShape,threshold,Num,fs);
%numero di campioni del segnale VAS
N1=length(VoiceActShape);
if flag==1
    x1=linspace(a,b,N1);
else
    x1=linspace(0,N/fs,N1);
end
figure(2);
plot(x1,VoiceActShape,x1,threshold);
legend('VAS','AWT');
```

```
xlabel('tempo');
ylabel('VAS(t)');
title('Voice Activity Shape del segnale d''ingresso');

%calcola BSE e RLF
[entropia,Rlf,entropiaFFT,RlfFFT]=entropy(S);

entropia=1./entropia;
%filtro FIR per visualizzare l'inviluppo del segnale
m=3;
Sfilt=filtfilt(ones(1,m)./m,1,entropia);
%ricava la soglia per la BSE utilizzando quella per la SAE
[Ts,Tn,VBSE]=sogliaSAE(Sfilt);

figure(3);
plot(x1,Sfilt,x1,Ts,x1,Tn);
legend('BSE','Ts','Tn');
xlabel('Tempo');
ylabel('BSE');
title('Misura BSE del segnale d''ingresso');
Rlf=1./Rlf;
%filtro FIR per inviluppo
Rlffilt=filtfilt(ones(1,m)./m,1,Rlf);
%ricava la soglia per la RLF utilizzando quella per la VAS
threshold=soglia(Rlffilt);
VRLF=costruisci(Rlffilt,threshold,Num,fs);
figure(4);
plot(x1,Rlffilt,x1,threshold);
legend('RLF','AWT');
xlabel('Tempo');
ylabel('Rlf');
title('Misura Rlf del segnale d''ingresso');

for ind=1:length(entropiaFFT)
    if entropiaFFT(ind)~=0
        entropiaFFT(ind)=1./entropiaFFT(ind);
    end
end

%filtro FIR per inviluppo
Sfilt1=filtfilt(ones(1,m)./m,1,entropiaFFT);
%ricava la soglia per la BSE di Fourier
%utilizzando quella per la SAE
[TsF,TnF,VBSEFFT]=sogliaSAE(Sfilt1);
```

```
figure(5);
plot(x1,Sfilt1,x1,TsF,x1,TnF);
legend('BSE FFT','Ts','Tn');
xlabel('Tempo');
ylabel('BSE FFT');
title('Misura BSE del segnale d''ingresso utilizzando la FFT');
%filtro FIR per inviluppo
Rlffilt1=filtfilt(ones(1,m)./m,1,RlfFFT);
threshold=soglia(Rlffilt1);
VRLFFFT=costruisci(Rlffilt1,threshold,Num,fs);
figure(6);
plot(x1,Rlffilt1,x1,threshold);
legend('RLF FFT','AWT');
xlabel('Tempo');
ylabel('Rlf FFT');
title('Misura Rlf del segnale d''ingresso utilizzando la FFT');

figure(7);
plot(x1,VBSE);
xlabel('Tempo');
ylabel('VBSE(t)');
title('Vettore booleano del parlato per la BSE');
if flag==1
    figure(8);
    plot(x,V);
    xlabel('Tempo');
    ylabel('V(t)');
    title('Vettore booleano del parlato originale etichettato');
    %calcola i fattori di somiglianza per la VAS e la BSE
    [simVAS,PokVAS,PmissVAS,PfalseVAS]=
                            confronta(VVAS,V,Num);
    fprintf('\nFattore di somiglianza VAS: %g %% \n',simVAS);
    fprintf('P(ok) VAS   : %g %% \n',PokVAS);
    fprintf('P(miss) VAS : %g %% \n',PmissVAS);
    fprintf('P(false) VAS: %g %% \n',PfalseVAS);

    VBSE=VBSE | VRLF;
    [simBSE,PokBSE,PmissBSE,PfalseBSE]=
                            confronta(VBSE,V,Num);
    fprintf('\nFattore di somiglianza BSE: %g %% \n',simBSE);
    fprintf('P(ok) BSE   : %g %% \n',PokBSE);
    fprintf('P(miss) BSE : %g %% \n',PmissBSE);
    fprintf('P(false) BSE: %g %% \n',PfalseBSE);
```

```
    VBSEFFT=VBSEFFT | VRLFFFT;
    [simBSEFFT,PokBSEFFT,PmissBSEFFT,PfalseBSEFFT]=
                          confronta(VBSEFFT,V,Num);
    fprintf('\nFattore di somiglianza
             BSEFFT: %g %% \n',simBSEFFT);
    fprintf('P(ok) BSEFFT   : %g %% \n',PokBSEFFT);
    fprintf('P(miss) BSEFFT : %g %% \n',PmissBSEFFT);
    fprintf('P(false) BSEFFT: %g %% \n',PfalseBSEFFT);

else
    simVAS=0;
    PokVAS=0;
    PmissVAS=0;
    PfalseVAS=0;

    simBSE=0;
    PokBSE=0;
    PmissBSE=0;
    PfalseBSE=0;

    simBSEFFT=0;
    PokBSEFFT=0;
    PmissBSEFFT=0;
    PfalseBSEFFT=0;
end
```

Funzione di VAD utilizzando la SAE:

```
function [sim,Pok,Pmiss,Pfalse]=MyVAD2(nomefile,SNR,ntype);
%funzione che esegue il VAD di un segnale utilizzando
%la funzione SAE
%ntype=tipo di rumore:1(bianco),2(babble),3(factory),4(car)
close all;clc;

if nargin<=2
    ntype=1;
end

carica=strrep(nomefile,'.wav','.mat');
flag=0;
if exist(carica)>0
%carica le variabili a,b,V,S dal file .mat
    load(carica);
    flag=1;
```

```
else
    fprintf('Non  disponibile il file
             .mat per tale file\n');
end
%recupera il segnale e la frequenza di sampling
[S,fs]=wavread(nomefile);
fprintf('Frequenza di campionamento del
         file %s: %g Hz \n',nomefile,fs);

if flag==1
    if nargin==1
        %immissione del rapporto SNR desiderato
        SNR=input('Inserire il rapporto SNR
                   desiderato (in dB): \n');
    end
    %ricava l'ampiezza del rumore da immettere
    ampR=noiseamp(S,SNR,V);
    fprintf('Ampiezza del rumore da immettere
             sul segnale: %g \n',ampR);
    %corrompe il segnale con rumore desiderato
    [S,noise]=corrompi(S,ampR,ntype);
else
    S=normalizza(S);
    fprintf('Non sono disponibili informazioni relative
             all''entit\'a del rumore \n');
end

N=length(S);
if flag==1
    x=linspace(a,b,N);
else
    x=linspace(0,N/fs,N);
end
figure(1);
plot(x,S);
xlabel('tempo');
ylabel('S(t)');
title('Segnale d''ingresso');

%coefficienti di approssimazione e dettaglio wavelet
[CA,D1]=dwt(S,'db10');
[CA,D2]=dwt(CA,'db10');
[A3,D3]=dwt(CA,'db10');
```

```
%calcola le Subband Signal ACF
%autocorrelation function dei 4 segnali
Num=256;
ACFA3=SSACF(A3,Num);
ACFD1=SSACF(D1,Num);
ACFD2=SSACF(D2,Num);
ACFD3=SSACF(D3,Num);

%normalizza i segnali ACF sulla loro media
ACFA3=normalizza(ACFA3,2);
ACFD1=normalizza(ACFD1,2);
ACFD2=normalizza(ACFD2,2);
ACFD3=normalizza(ACFD3,2);

if flag==1
    xa=linspace(a,b,length(ACFA3));
else
    xa=linspace(0,N/fs,length(ACFA3));
end
figure(2);
plot(xa,ACFA3);
title('SSACF del segnale A3');

DSSACFA3=DSSACF(ACFA3);      %definizione DSSACF
DSSACFD1=DSSACF(ACFD1);
DSSACFD2=DSSACF(ACFD2);
DSSACFD3=DSSACF(ACFD3);

%inizializzazione MDSSACF
MDSSACFA3=MDSSACF(DSSACFA3,Num);
MDSSACFD1=MDSSACF(DSSACFD1,Num);
MDSSACFD2=MDSSACF(DSSACFD2,Num);
MDSSACFD3=MDSSACF(DSSACFD3,Num);

if flag==1
    x1=linspace(a,b,length(MDSSACFD1));
    x2=linspace(a,b,length(MDSSACFD2));
    x3=linspace(a,b,length(MDSSACFD3));
else
    x1=linspace(0,N/fs,length(MDSSACFD1));
    x2=linspace(0,N/fs,length(MDSSACFD2));
    x3=linspace(0,N/fs,length(MDSSACFD3));
end
figure(3);
```

```
plot(x3,MDSSACFA3);
title('MDSSACF per il segnale A3');
legend('MDSSACF A3');
figure(4);
plot(x1,MDSSACFD1);
title('MDSSACF per il segnale D1');
legend('MDSSACF D1');
figure(5);
plot(x2,MDSSACFD2);
title('MDSSACF per il segnale D2');
legend('MDSSACF D2');
figure(6);
plot(x3,MDSSACFD3);
title('MDSSACF per il segnale D3')
legend('MDSSACF D3');

%definizione della funzione SAE
SAEfun=SAE(MDSSACFD1,MDSSACFD2,MDSSACFD3,MDSSACFA3);

N1=length(SAEfun);
if flag==1
    x1=linspace(a,b,N1);
else
    x1=linspace(0,N/fs,N1);
end
[Ts,Tn,VAD]=sogliaSAE(SAEfun);
figure(7);
plot(x1,SAEfun,x1,Ts,x1,Tn);
xlabel('t');
ylabel('SAE(t)');
title('SAE del segnale');
legend('SAE','Ts','Tn');
figure(8);
plot(x1,VAD);
title('Vettore booleano ricavato del parlato');
if flag==1
    %calcola i fattori di somiglianza della SAE
    [sim,Pok,Pmiss,Pfalse]=confronta(VAD,V,Num);
    fprintf('\nFattore di somiglianza SAE: %g %% \n',sim);
    fprintf('P(ok)   : %g %% \n',Pok);
    fprintf('P(miss) : %g %% \n',Pmiss);
    fprintf('P(false): %g %% \n',Pfalse);
else
    sim=-1;
```

```
    Pok=-1;
    Pmiss=-1;
    Pfalse=-1;
end
```

Funzione che restituisce l'ampiezza del rumore da immettere sul segnale:

```
function ris=noiseamp(S,SNR,V);
%funzione che restituisce l'ampiezza da dare al rumore,
%dato un SNR di input (in dB)
N=length(S);
if nargin==2
    V=ones(1,N);
end
ampS=0;
%variabile di supporto per la lunghezza
ns=0;
for i=1:N
    if V(i)==1
    %aggiorna la RMS del segnale e del rumore
    %se c'\'e parlato
        ampS=ampS+S(i)^2;
        ns=ns+1;
    end
end
ampS=sqrt(ampS/ns);
ris=ampS/(10^(SNR/20));
```

Funzione di corruzione del segnale con del rumore desiderato:

```
function [ris,noise]=corrompi(S,amp,ntype);
%funzione che corrompe con rumore il segnale S
%d'ingresso con ampiezza parametrica e restituisce
%il segnale corrotto e la forma d'onda del rumore
%ntype: 1 (bianco),2 (babble), 3 (factory), 4 (car)
if nargin==1
    amp=0.02;
end
if nargin==2
    ntype=1;
end
N=length(S);
if (ntype==1)
    %funzione di generazione di rumore bianco
    noise=randn(1,N);
```

```
    noise=amp*noise;
    %corrompe il segnale con rumore bianco
    ris=S+noise';
else
    if (ntype==2)
        noise=wavread('babblenoise.wav');
    elseif (ntype==3)
        noise=wavread('factorynoise1.wav');
    else
        noise=wavread('carnoise.wav');
    end
    noise=normalizza(noise);
    noise=amp*noise;
    nr=length(noise);
    ris=S;
    %corrompe il segnale con rumore desiderato
    if N>nr
        for i=0:floor(N/nr)-1
            inizio=i*nr+1;
            fine=min((i+1)*nr,N);
            ris(inizio:fine)=S(inizio:fine)+noise(1:nr);
        end
    else
        ris(1:N)=S(1:N)+noise(1:N);
    end
end
```

Funzione di normalizzazione del segnale:

```
function normalizzato=normalizza(S,flag);
%funzione che normalizza le ampiezze del segnale tra
%-1 e 1 se flag=1, mentre normalizza con picco 1 e
%ampiezza 0 se flag=2. Se invece flag=3, normalizza
%banalmente sul massimo valore del segnale

if nargin==1 | (nargin==2 & flag==1)
    mn=min(S);
    mx=max(S);
    normalizzato=(S-mn)/(0.5*(mx-mn))-1;
elseif flag==2
    normalizzato=(S-mean(S))/max(abs((S-mean(S))));
elseif flag==3
    massimo=max(S);
    normalizzato=S/massimo;
end
```

Funzione che calcola il SNR di un segnale:

```
function SNR=calcolaSNR(S,noise,V);
%funzione che calcola il rapporto SNR di un segnale audio
%S: segnale audio, V: vettore booleano del parlato
%noise: rumore
N=length(S);
if nargin==2
    V=ones(1,N);
end
ampS=0;
ns=0;         %variabile di supporto per la lunghezza
ampR=0;
for i=1:N
    if V(i)==1
        %aggiorna la RMS del segnale e del rumore se
        %c'\'e parlato
        ampS=ampS+S(i)^2;
        ampR=ampR+noise(i)^2;
        ns=ns+1;
    end
end
ampS=sqrt(ampS/ns)
ampR=sqrt(ampR/ns)
SNR=20*log10(ampS/ampR);    %uscita SNR
```

Funzione di calcolo del TEO:

```
function ris = TEO(y);
%funzione che calcola il Teager Energy Operator di un
%segnale discreto y(n)
N=length(y);
ris=zeros(1,N);
%definizione TEO
ris(1)=(y(1)^2)-(y(1)*y(2));
for i=2:N-1
   ris(i)=(y(i)^2)-(y(i-1)*y(i+1));
end
ris(N)=(y(N)^2)-(y(N-1)*y(N));
```

Funzione di calcolo della VAS:

```
function ris = VAS (y,Num);
%funzione che suddivide il segnale in frame di Num campioni
%e ne calcola la VAS attraverso la varianza
%y: segnale d'ingresso, Num: granularit\'a dei frame
```

```
N=length(y);
if N>Num
    ris=zeros(1,floor(N/Num)+1);     %inizializzazione VAS
    %finestratura del segnale con Num campioni alla volta
    for i=0:floor(N/Num)
        inizio=i*Num+1;
        if (i+1)*Num<=N
            fine=(i+1)*Num;
        else
            fine=N;
        end
        samples=y(inizio:fine);
        %calcola la WPD a 4 livelli con mother wavelet di
        %Daubechies a 10 punti
        T=wpdec(samples,4,'db10');
        %ottengo gli indici delle foglie dell'albero
        %in ordine crescente
        [foglie,j]=leaves(T,'sort');
        for k=1:length(foglie)
            %leggo i coefficienti e calcolo TEO e VAS
            cfs=read(T,'data',foglie(k));
            ris(i+1)=ris(i+1)+var(TEO(cfs));
        end
    end
end
```

Funzione di sogliatura per la VAS:

```
function AWT = soglia(y);
%funzione di sogliatura adattiva pesata del VAS (AWT)
%y: segnale VAS
yold=y;          %segnale attuale Vk(n)
N=length(y);     %lunghezza segnale
k=1;             %indice algoritmo
Lv=0;            %numero dei campioni di V(2)(n)==V(1)(n)
while k<=2
    ynew=zeros(1,N);    %definizione nuovo segnale
    thr=mean(yold);     %media del vecchio segnale
    for i=1:N
        if yold(i)<thr
            ynew(i)=yold(i);
            if k==1
                Lv=Lv+1;
            end
        else
```

```
            ynew(i)=thr;
        end
    end
    if k==1      %segnali utili per ricavare le soglie
        y2=ynew;
    else
        y3=ynew;
    end
    k=k+1;
    yold=ynew;
end
SDRM=mean(y2);  %second derivative round mean
TDRM=mean(y3);  %media del segnale V(3)(n)
p=Lv/N;         %voiced rate
noisedis=p*(SDRM+TDRM)/2;   %Noise_dis(p)

%divide il vettore della VAS in 5 campioni alla volta
%e calcola la soglia adattativa
Num=5;
AWT=zeros(1,N);      %definizione segnale di soglia
if N>Num
    for i=0:floor(N/Num)
        inizio=i*Num+1;
        if (i+1)*Num<=N
            fine=(i+1)*Num;
        else          %ultima porzione di segnale
            fine=N;
        end
        maxframe=max(y(inizio:fine));
        if maxframe<noisedis
            AWT(inizio:fine)=maxframe+0.1;
        else
            AWT(inizio:fine)=SDRM+TDRM;
        end
    end
end
```

Funzione di costruzione del vettore booleano del parlato:

```
function V=costruisci(f,thr,Num,FS);
%funzione che costruisce il vettore booleano del parlato
%e unisce i silenzi corti
N=length(f);    %lunghezza
V=zeros(1,N);   %inizializzazione
for i=1:N
```

```
    if thr(i)<=f(i)    %definizione vettore del parlato
        V(i)=1;
    end
end
%variabile di supporto per l'unione dei silenzi corti
next=ceil(0.3*FS/Num);
for i=1:N-next-1
    if V(i)==1
        j=1;
        while (V(i+j)==0) & (j<next)
            j=j+1;
        end
        if (j>1) & (j<next)  %unione silenzi corti
            V(i:i+j)=1;
        elseif j==next
            if V(i+j+1)==1
                V(i:i+j)=1;
            end
        end
    end
end
```

Funzione che calcola l'entropia spettrale di un segnale:

```
function [entr,Rlf,entr1,Rlf1]=entropy(S);
%funzione che calcola l'entropia spettrale
%S: segnale d'ingresso
N=length(S);
Num=256;
sovr=1;    %flag sui frame sovrapposti
levels=4;  %livelli di decomposizione Wavelet
if N>Num
    M=floor(N/Num)+1;    %numero di frame del segnale
    Nb=2.^levels;        %band size di ogni frame
    Nb1=32;              %band size per la FFT
    %inizializzazione spectral energy e energie in percentuale
    Es=zeros(Nb,M);
    Es1=zeros(Nb1,M);    %SE per la FFT
    PEs=Es;
    PEs1=Es1;
    S=Hamming(N).*S;  %finestratura di Hamming
    Rlf=zeros(1,M);   %rapporto energie Low-band e Full-band
    Rlf1=Rlf;     %rapporto energie per la FFT
    %finestratura del segnale Num campioni alla volta
    for i=0:M-1
```

```
if (sovr==0 & (i+1)*Num<=N) |
(sovr==1 & (i+1)*Num-Num/2<=N)
    if i==0 | sovr==0
        samples=S(i*Num+1:(i+1)*Num);
    else
    %frame sovrapposti di Num/2 campioni
        samples=S(i*Num+1-Num/2:(i+1)*Num-Num/2);
    end
else
    if sovr==0
        samples=S(i*Num+1:N);
    else
        samples=S(i*Num+1-Num/2:N);
    end
end
%calcola la WPD a 4 livelli con mother wavelet
%di Daubechies a 10 punti
T=wpdec(samples,levels,'db10');
samples=TEO(samples);
%calcola il modulo della FFT dei campioni "teagerati"
F=abs(fft(samples));
LF=length(F);
%ottengo gli indici delle foglie
%dell'albero in ordine crescente
[foglie,j]=leaves(T,'sort');
%variabile per il calcolo delle energie in percentuale
somma=0;
for k=1:length(foglie)
    %leggo i coefficienti e calcolo TEO
    cfs=read(T,'data',foglie(k));
    teager=TEO(cfs);
    Es(k,i+1)=energy(teager);    %calcola l'energia
    somma=somma+Es(k,i+1);
end
%riempio le colonne della matrice delle
%energie percentuali
for k=1:length(foglie)
    PEs(k,i+1)=Es(k,i+1)/somma;
end
%rapporto Low-band e Full-band per l'i+1-esimo frame
Rlf(i+1)=PEs(2,i+1);

%numero campioni per sottobanda per la FFT
nf=floor(Num/Nb1);
```

```
        somma1=0;
        for k=1:Nb1
            if k*nf<=LF
                signal=F((k-1)*nf+1:k*nf);
            else
                signal=F((k-1)*nf+1:LF);
            end
            Es1(k,i+1)=energy(signal);
            somma1=somma1+Es1(k,i+1);
        end
        for k=1:Nb1
            PEs1(k,i+1)=Es1(k,i+1)/somma1;
        end
        Rlf1(i+1)=PEs1(4,i+1);
    end

    %definizione matrice delle varianze delle energie
    W=zeros(Nb,M);
    %matrice varianze FFT
    W1=W;
    for i=1:M
        input=PEs(1:Nb,i);
        %riempimento matrice delle varianze
        output=define(input);
        W(1:Nb,i)=output;

        %riempimento matrice FFT varianze
        input=PEs1(1:Nb1,i);
        output=define(input);
        W1(1:Nb1,i)=output;
    end

    %definizione del segnale di misura BSE
    H=zeros(1,M);
    %BSE FFT
    H1=H;
    for i=1:M
        sum=0;    %coefficienti BSE
        for k=1:Nb
            sum=sum+W(k,i)*PEs(k,i)*log(1/PEs(k,i));
        end
        H(i)=log(sum);
        sum=0;
        for k=1:Nb1
```

```
            if PEs1(k,i)~=0
                sum=sum+W1(k,i)*PEs1(k,i)*log(1/PEs1(k,i));
            end
        end
        if sum~=0
            H1(i)=log(sum);
        end
    end
    entr=H;     %output
    entr1=H1;
end
```

Funzioni di supporto per il calcolo dell'entropia spettrale:

```
function W=define(y);
%funzione che definisce i parametri per la varianza
%dell'energia spettrale e riempie le colonne della
%matrice suddetta
%y: vettore colonna d'ingresso
N=length(y);
minimo= min(y);
e=zeros(N,1);
for i=1:N    %definizione coefficienti
    if y(i)~=0
        e(i)=minimo/y(i);
    end
end
mu=zeros(N,1);  %definizione medie
mu(1)=0.5*(e(1)+e(2));
mu(N)=0.5*(e(N-1)+e(N));
for i=2:N-1
    mu(i)=(1/3)*(e(i-1)+e(i)+e(i+1));
end
W=zeros(N,1);   %definizione vettore d'uscita
W(1)=0.5*((e(1)-mu(1)).^2+(e(2)-mu(2)).^2);
W(N)=0.5*((e(N-1)-mu(N-1)).^2+(e(N)-mu(N)).^2);
for i=2:N-1
    W(i)=(1/3)*((e(i-1)-mu(i-1)).^2+(e(i)-mu(i)).^2+
         (e(i+1)-mu(i+1)).^2);
end

function ener=energy(y);
%funzione che calcola l'energia di un segnale
%y: segnale d'ingresso
N=length(y);
```

```
ener=0;
for i=1:N
    ener=ener+y(i).^2;   %sommo i quadrati dei coefficienti
end
ener=abs(ener);
```

Funzioni per il calcolo della SAE:

```
function ris=SSACF(y,Num);
%funzione che calcola la SSACF di un segnale shiftando
%i campioni Num alla volta
N=length(y);
if N>Num
    ris=zeros(1,N);    %inizializzazione
    y1=TEO(y);      %teager del segnale
    %finestratura del segnale Num campioni alla volta
    for i=0:floor(N/Num)
        inizio=i*Num+1;
        if (i+1)*Num<=N
            fine=(i+1)*Num;
        else
            fine=N;
        end
        samples=y1(inizio:fine);
        p=length(samples);   %indice di shift ACF
        somma=zeros(1,p);    %valore ACF
        for k=1:p
            for m=1:p-k
                somma(k)=somma(k)+(samples(m)*samples(m+k));
            end
        end
        ris(inizio:fine)=somma;
    end
end

function ris=DSSACF(y);
%funzione che calcola la Delta Subband Signal
%ACF di un segnale y

N=length(y);
M=8;   %neighbourhood per DSSACF
ris=y;      %inizializzazione DSSACF
quoziente=0;
for ind=-M:M
    quoziente=quoziente+ind.^2;
```

```
end
for k=1:N
    somma=0;
    for m=-M:M
        if k+m>=1 & k+m<=N
            somma=somma+(m*y(k+m));
        end
    end
    ris(k)=somma/quoziente;   %definizione DSSACF
end

function ris=MDSSACF(y,Num);
%funzione che calcola la Mean Delta SS ACF di un segnale y,
%suddividendolo in Num campioni per frame

N=length(y);
ris=zeros(1,floor(N/Num)+1);    %inizializzazione MDSSACF
%finestratura del segnale per Num campioni alla volta
for i=0:floor(N/Num)
    inizio=i*Num+1;
    if (i+1)*Num<=N
        fine=(i+1)*Num;
    else
        fine=N;
    end
    somma=0;
    for index=inizio:fine
        somma=somma+abs(y(index));
    end
    ris(i+1)=(1/Num)*somma;     %definizione della MDSSACF
end

function ris=SAE(D1,D2,D3,A3);
%funzione che calcola SAE a partire dalle funzioni
%MDSSACF ai livelli 1,2,3

N1=length(D1);
N2=length(D2);
N3=length(D3);
ris=zeros(1,N1);       %inizializza
ratio13=ceil(N1/N3);
ratio12=ceil(N1/N2);   %rapporto lunghezze
for i=1:N1
    i2=ceil(i/ratio12);
```

```
    i3=ceil(i/ratio13);
    ris(i)=D1(i)+D2(i2)+D3(i3)+A3(i3);
end
```

Funzione di sogliatura adattativa per la SAE:

```
function [Ts,Tn,VAD]=sogliaSAE(y);
%funzione di sogliatura per la SAE
%y: segnale d'ingresso

N=length(y);   %lunghezza del segnale
Num=5;  %passo di aggiornamento della sogliatura
k=1;    %indice di scorrimento
as=5;   %parametri per la sogliatura fissi
bn=-1;
gamma=0.95;
Ts=zeros(1,N);   %inizializzazione soglie e decisione VAD
Tn=zeros(1,N);
VAD=zeros(1,N);
%media e varianza iniziali del segnale
mu=mean(y(1:5));
sigma=var(y(1:5));
%buffer di memoria per i noise-only frames
mem=[];
while k<=N
    %sogliature iniziali per speech e noise
    Ts(k)=mu+as*sigma;
    Tn(k)=mu+bn*sigma;
    %aggiornamento decisione VAD
    if y(k)>Ts(k)
        VAD(k)=1;
        mem=[];     %memoria noise-only azzerata
    elseif y(k)<Tn(k)
        VAD(k)=0;
        %aggiungo il valore attuale al noise-only frame
        mem=[mem,y(k)];
    else
        if k>1
            VAD(k)=VAD(k-1);
            if VAD(k)==0
            %aggiungo il valore attuale al noise-only frame
                mem=[mem,y(k)];
            end
        end
    end
```

```
    if VAD(k)==0   %noise period: aggiorna parametri
       mu=gamma*mu+(1-gamma)*y(k);
        SAEbufferprev=mean(mem.^2);
        SAEbuffer=gamma*SAEbufferprev+(1-gamma)*(y(k)^2);
        sigma=abs(SAEbuffer-mu^2);
    end
    k=k+1;  %avanzamento frame
end
```

Funzione che ricava i parametri di confronto:

```
function [sim,Pok,Pmiss,Pfalse]=confronta(V,Vorig,Num);

%funzione che confronta il vettore ricavato del parlato
%col vettore originale del database

Si=0;    %intervalli temporali dove c'e' speech
Ni=0;    %intervalli temporali dove c'e' noise
Ti=0;
Mi=0;
Fi=0;
sim=0;   %indice di somiglianza dei vettori
for i=1:length(V)-1
check=round(mean(Vorig((i-1)*Num+1:i*Num)));
    if check==1
        Si=Si+1;
        if V(i)==1
            sim=sim+1;
            Ti=Ti+1;       %intervalli true
        else
            Mi=Mi+1;       %intervalli miss
        end
    else
        Ni=Ni+1;
        if V(i)==1
            Fi=Fi+1;       %intervalli false positive
        else
            sim=sim+1;
        end
    end
end
sim=(sim/i)*100;      %definizione della somiglianza
Pok=(Ti/Si)*100;      %definizione delle probabilit
Pmiss=(Mi/Si)*100;
Pfalse=(Fi/Ti)*100;
```

Bibliografia

[1] Amara Graps, An introduction to wavelets, da *IEEE Computational Sciences and Engineering, Volume 2, Number 2*, 1995, pp 50-61.

[2] Ingrid Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, 1992, ISBN 0-89871-274-2.

[3] P. Schniter, *Computing the Scaling Function: The Cascade Algorithm*, The Connexions Project, 2009

[4] Alfred Haar, *Zur Theorie der orthogonalen Funktionensysteme*, Mathematische Annalen, **69**, pp 331-371, 1910.

[5] Charles K. Chui, *An Introduction to Wavelets*, Academic Press, San Diego, 1992, ISBN 0121745848.

[6] Juraj Kačur, Juraj Frank , Gregor Rozinaj, *Speech detection in the noisy environment using Wavelet Transform*, 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications, 2-5 July 2003.

[7] R. Chiodi and D. Massicotte, *Voice Activity Detection Based on Wavelet Packet Transform in Communication Nonlinear Channel*, 2009 First International Conference on Advances in Satellite and Space Communications

[8] Shi-Huang Chen, Hsin-Te Wu, Chia-Hsiang Chen, Jiun-Ching Ruan, and T K. Truong, *Robust Voice Activity Detection Algorithm Based On The Perceptual Wavelet Packet Transform*, Department of Computer Science & Information Engineering, Shu-Te University, 2005

[9] James F. Kaiser, *On a simple algorithm to calculate the 'energy' of a signal*, Bell Communications Research, in Proc. ICASSP'90, 1990, pp.381-384.

[10] ITU-T Recommendation G.729, *Annex B: A silence compression scheme for G.729 optimized for terminals conforming to ITU-T V. 70*

[11] Kun-Ching Wang and Yi-Hsing Tasi, *Voice Activity Detection Algorithm with Low Signal-to-Noise Ratios Based on Spectrum Entropy*, 2008 Second International Symposium on Universal Communication

[12] S. Mallat, *A theory for multiresolution signal decomposition: the wavelet representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7), 1989, pp.674-693.

[13] I. Daubechies and W. Sweldens, *Factoring Wavelet Transforms into Lifting Steps*, Bell Laboratories, Lucent Technologies, 1996.

[14] Bing-Fei Wu and Kun-Ching Wang, *Voice Activity Detection Based on Auto-Correlation Function Using Wavelet Transform and Teager Energy Operator*, Computational Linguistics and Chinese Language Processing Vol. 11, No. 1, March 2006, pp. 87-100

[15] Jabloun, F., A. E. Cetin, and E. Erzin, *Teager energy based feature parameters for speech recognition in car noise*, IEEE Signal Processing Letters, 6(10), 1999, pp.259-261.

[16] A. Ouzounov, *A Robust Feature for Speech Detection*, Cybernetics and Information Technologies, 4(2), 2004, pp.3-14.

[17] Varga, A., and H. J. M. Steeneken, *Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise onspeech recognition systems*, Speech Communication, 12, 1993, pp.247-251.

[18] *www.speech.cs.cmu.edu/comp.speech/Section1/Data/noisex.html*

[19] *www-mmsp.ece.mcgill.ca/Courses/2007-2008/ECSE412B/ Project/MATLAB/*